Privacy-preserving cryptographic protocols

Patrick Towa

▶ **To cite this version:**

Patrick Towa. Privacy-preserving cryptographic protocols. Cryptography and Security [cs.CR]. Université Paris sciences et lettres, 2020. English. NNT : 2020UPSLE056 . tel-03601839

HAL Id: tel-03601839
https://theses.hal.science/tel-03601839

Submitted on 8 Mar 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Privacy-Preserving Cryptographic Protocols

Soutenue par
Patrick Towa
Le 30 septembre 2020

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Dario Fiore IMDEA Software Institute	*Rapporteur*
Dennis Hofheinz ETH Zürich	*Rapporteur*
Benoît Libert CNRS, ENS Lyon	*Examinateur*
Gregory Neven DFINITY	*Examinateur*
Duong Hieu Phan Télécom Paris, Institut Polytechnique de Paris	*Examinateur*
David Pointcheval CNRS, ENS Paris	*Président du jury*
Carla Ràfols Universitat Pompeu Fabra	*Examinateur*
Damien Vergnaud Sorbonne Université, CNRS, LIP6, Institut Universitaire de France	*Directeur de thèse*

ENS
ÉCOLE NORMALE SUPÉRIEURE

IBM **Research** | Zurich

To my mother

Abstract

This manuscript proposes new cryptographic protocols that are respectful of users' privacy and which have real-world applications.

In a first part, the focus is on group signatures, a primitive which allows members of a user group to anonymously sign on behalf of the group, and on message confidentiality. To remove the trust on single authorities, group signatures are here defined in a setting with multiple authorities and support both threshold issuance and threshold opening. These group signatures are then used as authentication mechanism for vehicle-to-vehicle communication, and combined with zone encryption, a new primitive whereby vehicles can efficiently encrypt their communication, they provide strong, well-defined privacy guarantees for cooperative intelligent transport systems. Thereafter, public-key encryption is studied in a more general context in which users do not have access to secure storage to protect their secret keys, but can leverage passwords and interaction with servers to obtain comparable security guarantees without renouncing their privacy.

In a second part, the topic of study are general-purpose cryptographic primitives which have privacy-preserving applications. First come zero-knowledge arguments, a type of cryptographic schemes which enable a computationally bounded prover to convince a verifier of a statement without disclosing any information beyond that. More specifically, we study arguments for the satisfiability of Diophantine equations that have logarithmic communication and round complexity, as well as their applications to privacy-preserving cryptography. Then, we tackle the problem of proving that a user algorithm selected and correctly used a truly random seed in the generation of her cryptographic key, a problem of fundamental importance to the security of any public-key cryptographic scheme.

Résumé

Ce manuscrit propose des nouveaux protocoles cryptographiques qui sont respectueux de la vie privée des utilisateurs et qui ont des applications dans la vie réelle.

Dans une première partie, l'accent est mis sur les signatures de groupe, une primitive cryptographique qui permet aux membres d'un groupe d'utilisateurs de signer anonymement au nom du groupe, et sur la confidentialité des messages. Pour éviter de faire confiance à des autorités uniques, les signatures de groupe sont ici définies avec plusieurs autorités et permettent l'émission à seuil de titres de créance ainsi que l'ouverture à seuil. Ces signatures de groupe sont alors utilisées comme mécanisme d'authentification pour la communication entre véhicules, et, combinées au chiffrement par zone, une nouvelle primitive permettant aux véhicules de chiffrer éfficacement leur communication, elles assurent de fortes garanties de sécurité bien définies pour les systèmes de transport coopératifs et intelligents. Par la suite, le chiffrement à clef publique est étudié dans un contexte plus général dans lequel les utilisateurs n'ont pas accès à un support de stockage sécurisé pour leurs clefs secrètes, mais peuvent tirer parti de mots de passe et de l'interaction avec des serveurs pour obtenir des garanties de sécurité comparables tout en préservant leurs vies privées.

Dans une deuxième partie, nous étudions des primitives cryptographiques à portée générale qui ont des applications à la protection de la vie privée. Dans un premier temps, nous étudions les arguments à divulgation nulle, un type de schémas cryptographiques qui permettent à un prouveur avec une puissance de calcul limitée de convaincre un vérifieur d'une assertion sans révéler aucune information supplémentaire. Plus précisement, nous étudions des arguments de satisfiabilité d'équations diophantiennes qui ont une complexité de communication et une complexité de tour logarithmiques, ainsi que leurs applications à la cryptographie qui vise à protéger la vie privée. Ensuite, nous considérons la question de prouver que l'algorithme d'un utilisateur a correctement choisi et utilisé une graine réellement aléatoire pour générer les clefs de l'utilisateur, un problème d'une importance capitale pour la sécurité de tout système cryptographique à clef publique.

Acknowledgments

I am too fond of reading books to care to write them.

— Lord Henry

I knew that writing a PhD thesis would be a difficult exercise, but little did I know that writing acknowledgments would be the most daunting part of it. Being left with no further opportunity to indulge my proclivity for procrastination (after all, "never put off till tomorrow the fun you can have today"[1]), I eventually had to try my best as what follows is most likely the only part that will ever matter to the overwhelming majority of readers.

In hindsight, and despite (or rather thanks to) its inner and outer circumstances, the experience which comes to an end with these words has been most enriching both professionally and personally. I cannot reflect on it without feeling lucky and sincerely thankful towards all those who contributed to its conclusion. I apologize to those I esteem and am indebted to, but forgot to mention.

First and foremost, I would like to express my profound gratitude to my advisor Damien Vergnaud. Not because it is customary to thank the advisor first – you know way too well that decorum could not compel me to act against my will, but because this document would have decidedly never come to life without your intellectual acumen, your mentoring and your unfaltering support in dire times. You never cease to impress me with your likability, your humbleness and your childlike curiosity and enthusiasm about every scientific topic, despite remarkable wits and an already titanic breadth of knowledge. I am grateful for your advice and teachings, I cannot thank you enough for watchfully allowing me to express myself and then separating the intellectual wheat from the chaff. Working with you has been a pleasure and an honor. For what it is worth, you have my utmost respect and thankfulness, I truly wish you further success in the rest of your career and, more importantly, much happiness in your personal life.

I am also grateful to Jan Camenisch for giving me the opportunity to join his group at IBM Research – Zurich and for his supervision during the first part of my PhD. I have learned many valuable lessons during our research discussions.

I cannot stress enough my appreciation to Anja Lehmann for taking up the torch and greatly contributing to the first part of this document. Thank you for your scientific input, for sharing your astounding presentation skills and for your care even after moving on with your career.

I am considerably indebted to Gregory Neven who took the time on multiple occasions to aid me birth ideas which were initially in inextricable forms, to read proofs and to provide precious technical advice and feedback.

[1] Aldous Huxley, *Brave New World.*

Thank you also for all the laughter and good moments shared at the lab. Thank you for accepting to be part of my committee.

I would also like to thank my (other) co-authors from whom I learned a great deal. In particular, I would like to thank Maria, Manu, Céline, Ida and Laura. And of course Olivier: your kindness and limpid explanations made collaboration with you an enjoyable experience.

Now I can only express my gratitude to Dario Fiore and Dennis Hofheinz for accepting the ungrateful task of reviewing this (long, perhaps too long) document, and for both thoroughly doing it. Heartfelt thank you for your feedback and suggestions.

I also thank David Pointcheval not only for accepting to be part of my committee, but also for his tireless assistance with the ENS administration (how I ended up registered in the wrong department still eludes me), his light-speed responsiveness at any hour of the day, and his warm hospitality and kindness during my sporadic visits. The reputation of your intellect precedes you, and the glimpses I have had of it during our helpful discussions suggest it is not exaggerated.

I must also thank Benoît Libert for kindly accepting to be a mid-thesis reviewer and a member of my committee. I am also grateful to Duong Hieu Phan and Carla Ràfols for accepting this latter role as well.

If "experience [...] is merely the name men gave to their mistakes,"[2] I guess I now have a smattering of experience in cryptography. Thank you all for your help throughout this journey.

Un chaleureux merci à Annick Mallet et M. Boglietto pour avoir cru en moi au point de rouvrir prématurément le programme de bourses d'Égide (maintenant Campus France) alors que je n'étais encore que fraîchement bachelier à Yaoundé. Sans votre soutien et votre considération, je n'aurais probablement jamais pu vivre cette aventure, recontrer toutes ces personnes et apprendre autant.

Genuine thanks to the ETH administration for awarding me a scholarship during the second year of my Master's degree program. Merci également à Brice Tsakam et sa femme Martina qui m'ont également aidé durant la même période.

Étonnamment, cette aventure scientifique a commencé grâce à un homme qui m'a insuflé son goût pour les lettres, je veux bien sûr parler de Bertrand Johanet. C'est grâce à vos conseils que j'ai appris l'existence des classes préparatoires scientifiques, mais surtout, c'est à travers vous que j'ai pu pour la première fois témoigner de l'existence de la passion. Dans votre cas, il s'agissait d'une passion pour l'enseignement et les lettres. Vos soudaines envolées lyriques, qui faisaient passer Cyrano pour un amateur, savaient restaurer l'accalmie dans une classe d'adolescents braillards (moi le premier) et susciter l'admiration. Vous pariez avec morale élégance les pointes

[2] Oscar Wilde, *The Picture of Dorian Gray.*

railleuses dont nous espérions vous faire don, et à la fin de chaque envoi, nos esprits d'ignorants fripons vous touchiez.[3]Il m'arrive bien souvent d'en être nostalgique, votre pédagogie était véritablement unique. Là où mon éducation avait jusque là échoué à éveiller ma sensibilité littéraire, vous avez réussi. Ce n'est que bien plus tard que j'ai redécouvert pareil engouement, sans jamais confondre vitesse et précipitation, mais pour une cousine (pas si lointaine) des lettres que sont les mathématiques. Pour vos enseignements, votre passion, votre patience, votre gentillesse, merci.

Bien vite, j'ai fait la connaissance d'une autre personne qui n'a jamais manqué de m'encourager (dans un style entièrement différent mais tout aussi attachant), même après notre séparation et durant le déroulement de ce doctorat. Il s'agit de Stéphane Saint-Jalmes. À coups de bâtons et de carottes, de traits d'esprit mordants, vous avez su me guider et réellement éveiller mon intérêt pour les sciences. Je vous dois beaucoup, merci pour tout.

Lecteur ou lectrice, si je t'ai un jour rabâché les oreilles avec les maths, le seul et unique coupable n'est autre que Jean-Louis Garcin. Le «Grand» Jean-Louis, surclassant Le Petit Robert, n'a vraisemblablement jamais laissé un(e) élève insensible à sa personne ou aux mathématiques. Regorgeant d'enthousiasme, de passion, d'humour, d'attention, de quiétude, de «petites ruses de sioux,» d'histoires sur les ensembles de Mandelbrot, vous m'avez transmis votre goût pour les mathématiques et la logique. Grâce à vous, je me suis tourné avec courage et admiration vers ces joies de la nature (vous ne manquerez certainement pas de noter la référence à Hermite). Si je ne vous avais pas rencontré, cette thèse n'aurait probablement pas vu le jour. Cette fois-ci je peux le dire : merci M. Garcin.

Je ne saurais assez remercier Michel Cognet pour avoir su pousser cet intérêt plus loin. Vos enseignements, vos encouragements et votre bienveillance m'ont aidé à avancer. J'avais toujours hâte de travailler avec vous pour pouvoir étudier les rouages et la finesse de votre pensée singulière. Merci pour votre soutien inconditionnel et surtout pour vos délicieux cocktails car travailler c'est bien, mais profiter de la vie tant qu'on le peut c'est mieux.

Je tiens également à remercier Jean-Aristide Cavaillès pour son appui et sa bienveillante exigence. Sévère mais toujours juste, vous avez su constamment repousser nos limites, et ce, avec un dévouement exemplaire. Je n'oublierai jamais cette veille de vacances de Nöel où votre fatigue devenait apparente après tout le travail que vous aviez fourni pour nous. Je réalisai alors qu'il se fallait de travailler encore plus, ne serait-ce que pour faire honneur à vos sacrifices. Votre goût pour la rigueur et les problèmes pratiques a indéniablement eu une influence sur mon choix de la cryptographie.

Marie-Laure Chable's vitality and merciless constructive criticism have

[3]Paraphrase de la «ballade du duel» tirée de *Cyrano de Bergerac, Acte I, Scène IV,* d'Edmond Rostand.

substantially contributed to this thesis from a stylistic viewpoint (only the positive parts). Though you must have by now cringed several times at the linguistic awkwardness and amateurishness of this text, I know you will appreciate the expression of a forthright thank you. You taught me that "practice makes perfect," you taught me to "be bold." At times, when laziness or uncertainty loom, I hear your voice and then keep going forth.

J'adresse mes sincères remerciements à Catherine Costentin et à Jean-Jacques Dupont (deux nombres me viennent immédiatement à l'esprit : 0,69 et 0,707) pour leurs enseignements et leur bonne humeur.

I wish to express my gratitude to Daniel, Tanja, Michaela and Thorben whose lively and agreeable German lessons punctuated my Monday and Wednesday evenings for almost three years during this thesis. Insincere apologies for making the preparation of your classes a living hell lest I had new convoluted grammar questions; candid thanks for your dedication, the quality of your teaching, the laughter, and for tolerating my mischievousness and sharp tongue. I have very fond memories of all this time spent together.

Though I cannot quote all those who wholeheartedly dedicated their time to contribute to my learning (I can at least mention Gabriel, Viviane, Ms. Ango-Ela, Philippe Bonneau, Ms. Loisel, Ms. Mbarga, Stephen Brown, Antoine Chambert-Loir, Lionel Gabet, Simon Häberli, Martin Hirt, Paul Nelson, Ron Rosenthal, Alain-Sol Sznitman, Vincent Tassion, Stefan Wolf), I want to thank them all. Many teach just for a living or because they are obliged to, they taught with authentic care and will to share. They did not simply teach a subject, they taught how to think, which is crucial in research and in life in general. The effort such people put in it is often not rightly appreciated.

> *One never actually stops being a student.*
>
> — Alain-Sol Sznitman

Warm thank you to Michael Osborne, Alexandra Gorski, the administration of IBM Research – Zurich and the cafeteria team for maintaining a friendly environment conducive to research. Thank you to my colleagues and friends whom I have not mentioned yet. In particular, I want to thank Sepand, Cecilia, Tommaso and Jesus, my long-time officemates with whom I shared bursts of laughter and had countless discussions insignificant to the fate of humankind. Also, thank you Bertram (perhaps one day you will be able to keep a straight face when you lie), Björn, Chris (especially for the telescope!), Christian, David (for your advice and uproarious humor – the anticipation of lunchtime with you had become a motivation to go to work), Guillaume (for your keen interest in art and sports), Gregor, Jonathan-senpai (for your kindness and the discussions on research and life in general – I wish you resounding success in your academic endeavors), Julia (for your cheerfulness), Khanh (this guy *though* – this expresses it all), Luca

"DJ de Feo," Lydia (for your advice and kindness), Magdalena, Matthieu, Nick, Pauline (for the hospitality and the parties), "underground" Patrick my accomplice, Rafael (for the laughter, the discussions) and Romain. Gregor, Khanh, Tommaso, Rafael, our utter lack of progress at billiards despite uncountable hours spent at the table still dazzles me.

Je remercie toute l'équipe du LIP6 (Anand, Charles, Florette, Jean-Claude, Jérôme, Natascha, Thomas) pour leur accueil lors de mes visites et pour les discussions enrichissantes que nous avons eues. Je remercie également les membre de l'équipe CASCADE que j'ai rencontrés : Anca, Geoffroy, Georg, Mélissa, Michel, Michele, Thomas.

I am grateful to my tennis team and coach at the TC Sonnenberg and all the other tennis, badminton and squash sparring partners. The thrill of healthy competition with you helped me keep a balance during these years and above all, experience carefree fun.

In difficult times as well as in times of joy, I could always count on my (other) dear friends with and thanks to whom I grew, laughed, organized pranks, wrestled in the corridor 3 of the Lycée Chaptal after khôlles, shared brunches, dinners and drinks (many at Kasheme), traveled, danced, discovered new books, watched series and movies, listened to music, visited museums, gardens and castles, started playing the piano, played soccer, pétanque, molky and a variety of board games, left the casino before it was too late, hiked, tried skiing, wakeboarding and kitesurfing, learned about tinkering and photography, attended weddings, *lived*. They supported me in the face of adversity and some played an important role in major decisions. Though I cannot quote them all, I would like to thank Pierre and his family, Ice Block Steve, Natacha, Marie-Eve, Stella, Mr. Anderson, Dany the Lumberjack, Elizouz le charmeur de ces dames, Mr. Rouanne, The Scarecrow, Elian and his parents, La Team Muret, Amine Brooklyn, Miss Bonnet, Giorgia, Aida (I am still expecting money for all the orange juice you stole from me), Arthuryash, Aubrite, Chloé, Corentin x2, Damien, poy_boy, Florian, Hugo, Iab, Jean-John-Juan, le joli petit o, Sacha, Sylvie, TV, Hatem, Manu, David, Emile, Julien "D'où," Julien Le Rapide, Jan, Kike, Matthieu, Joren, Mikush, Chris, Tettje, Aaron, Teo, Hantao, Gabriele, Valentin "Le Rouge," Paul, Claude and Petit Louis, Charlotte and Valentin, Sabine, Myriam, Henry and Blanche, Mahe, Val Eris, Em Slim Shady, Frederik, Delilah, Emilien, Der Tonet, Pol, Stefan, Valentine, Nazim, Dan, JP, Bernard, Christelle, Moune, and my dear flatmates Nathalie, Thomas, Tatiana, Kevin, Luca and Yasmin. Sandro, for your teaching and collaboration a long time ago, the now gone tennis games, the advice: thank you.

I particularly want to thank the artists whose music accompanied me during the writing of this thesis. Among others, Bob Dylan, Jimi Hendrix, Burna Boy, Richard Bona, Curtis Harding, Kaleo, Alt-J, Cage The Elephant, Tracy Chapman, Mark Knopfler, Nick Waterhouse, the composers and singers of the Metal Gear Solid franchise, Sam Smith, Jenny Owen,

Ramin Djawadi, Damso, Jonas Straumann, Liszt, Rachmaninoff, Ravel. I also want to thank Maurice Leblanc and Jack London for their characters of Arsène Lupin and Martin Eden who made me laugh and inspired me strength and determination during novelist getaways in between two proofs. Thank you Eiichiro Oda for making every Friday morning more exciting than Christmas for over two decades now.

Enfin et surtout, je tiens à remercier chaleureusement les membres de ma famille pour leur amour, leur soutien inébranlable, leurs conseils, leur tolérance, leur joie de vivre et pour m'avoir toujours poussé dans la bonne direction. Je tiens en particulier à remercier mes frères et soeurs Sorel, Steeve, Line et Olivia, sans oublier mon parrain Joseph et mon grand-père Jacques. Line, plus qu'une soeur, tu as été une seconde mère pour moi depuis dix ans maintenant que je suis en Europe; cette thèse t'est en partie dédiée.

I hope that one day in the future, I will be able to look at each of those I hold dear to my heart and say "you and I have memories longer than the road that stretches out ahead."[4]

Mama-na, ma reconnaissance envers toi est ineffable et éternelle. Du fond de mon être, merci. Cette thèse t'est dédiée.

[4]The Beatles, *Two of Us*.

Contents

Chapter 1

General Introduction

Cryptography has traditionally been concerned with only one question: how can two parties, who have agreed on a *secret*, *confidentially* communicate? With the advent of key exchange protocols put forth by Diffie and Hellman [DH76], the scope of cryptography has vastly expanded. From then on, it was possible for two parties who had never met to privately communicate, and even ensure that the messages they received indeed originated from the claimed sender, i.e., guarantee the *authenticity* of messages.

Nowadays, cryptography is inconspicuously omnipresent in our daily lives e.g., through credit cards, passports, access badges, smartphones, emails and websites using the HTTPS protocol. With this surge of digital communication and the aggregation of personal data by public authorities and private services, a new question has emerged, namely that of *privacy*. The last decade of scandals has taught the general public – though these questions were already of prime concern for the research community – not only that people's data was harvested at large scale in cleartext by service providers, but also that the content of communication is not the only sensitive aspect of communication. Knowing who communicates with whom or who signed which message, or in other words the metadata of communication, is also a critical aspect of privacy. These realizations have lead to legislative endeavors such as the European General Data Protection Regulation to enforce accountability on data aggregators.

Nevertheless, a common misconception is that the usability of digital services *must* come at the price of privacy. Said otherwise, to benefit from technological progress, privacy must be forgone. After all, to perform any computation on data, it seems that one must be able to *see* the data and the potentiality of other alternatives seems paradoxical. Although this argument seems plausible, it is not entirely true: one must indeed have access to information to compute on it, but there is a priori no fundamental reason as to why this information must be in clear. For instance, to grant access to an age-restricted service, a provider need only know that users are above

a certain age, not their actual date of birth. Gentry even showed that it is possible to evaluate any boolean circuit on encrypted data with the first ever fully homomorphic encryption scheme [Gen09]; which implies for instance that any machine-learning task could be done on encrypted data. This, together with the celebrated zero-knowledge proofs, indicates that technological advancement must not necessarily come at the price of privacy with regard to theoretical feasibility.

However, there is also the question of *practical* efficiency. Even if a cryptographic innovation provides all the desired security and privacy guarantees desired for a certain task, it may still fail to be widely adopted if the incurred computational overhead is prohibitive. Therefore, the issue of privacy should also be addressed from the angle of efficiency. It is the approach taken in this thesis: it tackles several privacy challenges raised by real-world technological inventions and attempts to propose practical solutions with well-determined privacy guarantees.

This thesis comprises two parts. The first part studies, in Chapter 4, group signatures in a context with several authorities of whom some may be malicious, and based on its results, addresses in Chapter 5 the question of privacy and confidentiality in the context of vehicle-to-vehicle communication. Chapter 6 also considers the issue of privacy and confidentiality, but in a more general scenario in which no secure storage is assumed to be available. Chapters 4 and 5 are based on joint work with Jan Camenisch, Manu Drijvers, Anja Lehmann and Gregory Neven [CDL+20a, CDL+20b], and Chapter 6 is based on joint work with Olivier Blazy, Laura Brouilhet, Céline Chevalier, Ida Tucker and Damien Vergnaud. The second part of this thesis is concerned with more general-purpose cryptographic primitives which are also indispensable to privacy-related applications. Chapter 9 deals with zero-knowledge arguments for Diophantine satisfiability and their applications in (privacy-preserving) cryptography, and Chapter 10 addresses the fundamental question of randomness in key generation, which is the heart of all cryptographic schemes. Chapter 9 is based on joint work with Damien Vergnaud [TV20], and Chapter 10 on work with Olivier Blazy and Damien Vergnaud [BTV20].

Chapter 2

Preliminaries

This chapter introduces the notation and the elementary notions used throughout this thesis. It covers standard mathematical and algorithmic concepts, and introduces the main concepts of "provable" security. The classical computational assumptions and cryptographic primitives involved in this work are also recalled. The chapter assumes some familiarity with rudimentary mathematics and standard computability theory. The preliminaries specific to each chapter are given at the beginning of the corresponding chapter.

Contents

2.1 Mathematical Notation

Sets, Integers. For any two sets $\mathcal{X}$ and $\mathcal{Y}$, denote by $\mathcal{Y}^{\mathcal{X}}$ the set of functions from $\mathcal{X}$ to $\mathcal{Y}$.

The set of integers is denoted $\mathbb{Z}$, and the set of non-negative integers $\mathbb{N}$. The set of positive integers is denoted $\mathbb{N}^*$. The absolute value of an integer n is denoted $|n|$. For any two integers $a \leq b \in \mathbb{Z}$, $[\![a; b]\!]$ denotes the set $\{a\}$ if $a = b$ and $\{a, a+1, \ldots, b\}$ if $a < b$. For an integer $n \geq 1$, $[\![n]\!]$ stands for the set $[\![1; n]\!]$.

Groups, Rings and Fields. All groups considered herein are Abelian, i.e., commutative. Generic groups are denoted $\mathbb{G}$, and when made explicit, $T_{\mathbb{G}}$ denotes the binary complexity of computing group operations. The neutral element of a group $\mathbb{G}$ is denoted $1_{\mathbb{G}}$, and $\mathbb{G}^*$ stands for $\mathbb{G} \backslash \{1_{\mathbb{G}}\}$. The subgroup generated by an element $g \in \mathbb{G}$ is denoted $\langle g \rangle$. For $h \in \mathbb{G}$, $\sqrt{\langle h^2 \rangle}$ denotes the subgroup $\{g \in \mathbb{G} \colon \exists \alpha \in \mathbb{Z}, g^2 = h^{2\alpha}\}$.

Given a ring $(R, +, \cdot)$ and an integer $\nu \geq 1$, $R[X_1, \ldots, X_\nu]$ denotes the ring of formal polynomials with coefficients in R.

The finite field of cardinality p^k where p is a prime and k a positive integer is denoted $\mathbb{F}_{p^k}$ (recall that all finite fields of the same cardinality are isomorphic).

Modular Arithmetic. Given a positive integer n, $\mathbb{Z}_n$ denotes the set of residue classes modulo n. $(\mathbb{Z}_n, +, \cdot)$ is a ring, and $(\mathbb{Z}_n^*, \cdot)$ stands for the multiplicative group of residue classes modulo n, i.e., residue classes that are invertible modulo n. If n is prime, then $\mathbb{Z}_n^* = \mathbb{Z}_n \setminus \{0\}$ and $(\mathbb{Z}_n, +, \cdot)$ is a field.

Vectors, Matrices and Modules. Vectors and matrices are denoted in bold font. Consider a group $(\mathbb{G}, \cdot)$ and a positive integer n. For $\mathbf{g} \in \mathbb{G}^n$, if n is even, set $\mathbf{g}_1 \coloneqq \begin{bmatrix} g_1 & \cdots & g_{n/2} \end{bmatrix}$ and $\mathbf{g}_2 \coloneqq \begin{bmatrix} g_{n/2+1} & \cdots & g_n \end{bmatrix}$, and if n is odd, set $\mathbf{g}_1 \coloneqq \begin{bmatrix} g_1 & \cdots & g_{\lfloor n/2 \rfloor} & 1_{\mathbb{G}} \end{bmatrix}$ and $\mathbf{g}_2 \coloneqq \begin{bmatrix} g_{\lceil n/2 \rceil} & \cdots & g_n \end{bmatrix}$. For $\mathbf{a} \in \mathbb{Z}^n$, if n is even, set $\mathbf{a}_1 \coloneqq \begin{bmatrix} a_1 & \cdots & a_{n/2} \end{bmatrix}$ and $\mathbf{a}_2 \coloneqq \begin{bmatrix} a_{n/2+1} & \cdots & a_n \end{bmatrix}$, and if n is odd, set $\mathbf{a}_1 \coloneqq \begin{bmatrix} a_1 & \cdots & a_{\lfloor n/2 \rfloor} & 0 \end{bmatrix}$ and $\mathbf{a}_2 \coloneqq \begin{bmatrix} a_{\lceil n/2 \rceil} & \cdots & a_n \end{bmatrix}$.

For $n \in \mathbb{N}^*$, $z \in \mathbb{Z}$ and $\mathbf{g} = \begin{bmatrix} g_1 & \ldots & g_n \end{bmatrix} \in \mathbb{G}^n$, let $\mathbf{g}^z \coloneqq \begin{bmatrix} g_1^z & \cdots & g_n^z \end{bmatrix} \in \mathbb{G}^n$. For $\mathbf{a} = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix} \in \mathbb{Z}^n$, define $\mathbf{g}^{\mathbf{a}} \coloneqq \prod_{i=1}^n g_i^{a_i}$. For $\mathbf{g}$ and $\mathbf{h}$ in $\mathbb{G}^n$, $\mathbf{g} \circ \mathbf{h} \in \mathbb{G}^n$ denotes their component-wise product.

Given a ring $(R, +, \cdot)$, a positive integer n and a vector $\mathbf{a} \in R^n$, $\mathbf{a}X$ denotes the vector $\begin{bmatrix} a_1 X & a_2 X & \cdots & a_n X \end{bmatrix} \in R^n[X]$. For two vectors $\mathbf{a}$ and $\mathbf{b}$ in R^n, $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes their inner product, and $\mathbf{a} \circ \mathbf{b}$ denotes the Hadamard product (i.e., component-wise product) of $\mathbf{a}$ and $\mathbf{b}$, i.e., $\mathbf{a} \circ \mathbf{b} \coloneqq$

$\begin{bmatrix} a_1 b_1 & \cdots & a_n b_n \end{bmatrix}$. At times, when clear from the context, $\mathbf{a} \circ \mathbf{b}$ may be simply denoted $\mathbf{ab}$.

For a matrix $\mathbf{A} \in R^{n \times m}$ with n and m positive integers, $\mathbf{A}^{\mathrm{T}}$ stands for its transpose. For two matrices $\mathbf{A}$ and $\mathbf{B}$ of equal size, $\mathbf{A} \circ \mathbf{B}$ denotes their Hadamard product.

Given a $\mathbb{Z}_n$ module $\mathbb{G}$ and a family of vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{G}^k$, the $\mathbb{Z}_n$ submodule of $\mathbb{G}$ generated by the family is denoted $\mathrm{span}_{\mathbb{Z}_n}(\mathbf{u}_1, \ldots, \mathbf{u}_k)$. When the ring $\mathbb{Z}_n$ is clear from the context, the submodule is simply denoted $\mathrm{span}(\mathbf{u}_1, \ldots, \mathbf{u}_k)$.

Probabilities. Consider a finite set $\mathcal{X}$. Given a random variable X with values in $\mathcal{X}$ and $x \in \mathcal{X}$, $\Pr[X = x]$ denotes the probability that X takes value x. Besides, $X \in_R \mathcal{X}$ and $X \leftarrow_\$ \mathcal{X}$ equivalently denote that X has a uniform distribution over $\mathcal{X}$.

For any two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over $\mathcal{X}$, the *statistical distance* between them is defined as $1/2 \sum_{x \in \mathcal{X}} \left| \Pr[x \leftarrow_\$ \mathcal{D}_1] - \Pr[x \leftarrow_\$ \mathcal{D}_2] \right|$. It essentially measures how far two probability distributions are from one another.

Miscellaneous. Unless stated otherwise, all logarithms are in base 2.

2.2 Algorithms

Turing Machines. Turing machines constitute an abstract model of computation introduced by Alan Turing in 1936 [Tur36] which formally captures the idea of an algorithm. Any computer algorithm can be represented by a Turing machine. The model uses an infinite tape as unlimited memory. It has a tape head that can read symbols on the tape head and perform operations (write, move left, move right) depending on the symbol read and on a state.

The Turing machines considered in this work are *probabilistic*, meaning that they are additionally supplied with another infinite tape of random bits. For a given algorithm A, $y \leftarrow A(x)$ denotes the execution of A on input x that results in y. To make the random bits used by A explicit, denoting the string of random bits by r, the notation $y \leftarrow A(x; r)$ is used.

Interactive Turing machines are Turing machines supplemented with special tapes to communicate with other Turing machines: a communication input tape to receive messages from other machines and a communication output tape to send messages to other machines. Interactive Turing machines are used to model interactive protocols.

Running Time. A Probabilistic algorithm is said to be *Polynomial Time* (PPT) if it runs in (strict, not simply expected) time polynomial in the length of its inputs.

Sampling Algorithms. Given a probability distribution $\mathcal{D}$, a PPT algorithm that samples a random element according to $\mathcal{D}$ is denoted by $x \leftarrow_{\$} \mathcal{D}$. For a finite set $\mathcal{X}$, $x \leftarrow_{\$} \mathcal{X}$ also denotes a PPT algorithm that samples an element uniformly at random from $\mathcal{X}$.

2.3 Provable Security

"Provable" security refers to the paradigm which consists in reducing, as in computability theory, the ability to jeopardize the security of a cryptosystem to solving an *a priori* hard computational problem. The underlying rationale is that if a certain computational problem is hard and that one can "efficiently" reduce the security of the scheme to solving the problem, then the system must be secure as otherwise an attacker can be converted into a solver. Giving a precise meaning to this intuition requires to formally define the security of a cryptosystem, which is often done via experiments as defined below, and to parametrize the hardness of the problem and characterize what it means for an algorithm to be efficient.

Security Parameter. Given enough time, any computational problem which has a computable solution can be solved by a computer. When relying on computational assumptions to build cryptosystems, the goal is simply to make sure that an attacker cannot break the system in reasonable time. Nowadays, if 2^{128} elementary operations are necessary for the currently available computers to break a cryptosystem with a probability not too small (e.g., 2^{-7}), then the system is considered secure.

To formalize this idea, a *security parameter* $\lambda \in \mathbb{N}$ is introduced. All algorithms considered in this thesis take a *unary* representation 1^{λ} of this parameter (although it may not always be made explicit). The unary representation is simply to make sure that all efficient algorithms run in time polynomial in λ.

Asymptotic Statements. Now that security parameters have been introduced, an algorithm is considered *efficient* if its runtime is polynomial time in the security parameter. A function (of the security parameter) is said to be *negligible* if it is asymptotically dominated by all polynomial functions (of the security parameter). The probability of an event will be said to be *overwhelming* if the probability of the complement event is negligible.

Adversaries, Experiments and Oracles. The security of cryptosystems is often formally defined via so-called "security experiments" or "games". A security experiment is essentially an interaction between two algorithms: a *challenger* and an *adversary*. The experiment specifies a series of actions

between the two algorithms that depends on the considered cryptosystem, and how much leeway the adversary has during the interaction.

During the security experiments, the adversaries are often also given access to *oracles* in addition to the inputs and the random bits. An oracle is an idealized black-box which returns a value on an input specified by the adversary, and it models the fact that an algorithm might in practice have access to the answers to these queries without any restriction on the way they are computed or obtained. Given an Oracle $\mathcal{O}$, the notation $(A, B, \cdot)$ means that (A, B) is its inner (secret) state, while $\cdot$ denotes the (adversarial) query.

Generally, at the end of the experiment, the challenger returns a decision bit indicating whether the adversary "won the game", i.e., broke the system. The goal of the experiment is to capture a clear range of attacks, and usually, the wider this range is, the more realistic the attack model is and the more difficult it is to build efficient schemes.

For a given experiment, the *success probability* or *advantage* of an adversary refers to the probability that it wins the game against the challenger. A cryptosystem is generally deemed secure in the sense specified by the experiment if no PPT algorithm has a non-negligible advantage in winning the game. Note that such statements are asymptotic, and although they give an intuition as to why a system should be secure in practice, they fail to give to quantify the actual advantage that an attacker can have in practice. That is why for certain problems, it might be more meaningful to rather say that a scheme is (T, q, ε)-secure if no adversary that runs in time at most T and makes at most q oracle queries has a success probability of at most ε.

Alternatively, the security of a scheme can rather be defined by requiring an attacker to distinguish two experiments. The adversary interacts with the challenger of either of them and returns a decision bit at the end of the interaction. The advantage of the adversary is then defined as the difference (in absolute value) of the probabilities that it returns the same decision bit.

Hybrid Arguments. Hybrid arguments are a useful tool to prove that cryptographic schemes are secure in the sense defined above. They generally consist in defining a sequence of experiments that are indistinguishable starting from the original experiment which defines the security of the scheme. The indistinguishability between two consecutive experiments can be either information theoretic or computational, i.e., in the second case, distinguishing two consecutive experiments can be reduced to solving a computational problem. The last experiment is usually one in which the advantage of the adversary is nil.

This technique is especially helpful when several schemes are involved to build a larger one, as it allows to gradually prove the security of the overall scheme by focusing on one building block at a time.

Random-Oracle Model. The *random-oracle model* was introduced in cryptography by Bellare and Rogaway [BR93]. It is an idealized model for hash functions. In short, a random oracle returns consistent answers to previous inputs, and a uniformly random value in its range on each new input. The model provides considerable leverage in security reductions as it allows the reduction algorithm to observe and program the random-oracle queries of the adversary. However, the random oracle cannot be realized in practice with a concrete hash function. In fact, there exist contrived schemes [CGH98, CGH04, MRH04] which can be proven secure in the random-oracle model but are insecure when instantiated with a concrete hash function – on the other hand, no such result for a "natural" protocol has been proved so far. On this ground, proofs in the random oracle model should rather be considered as heuristic indications of security.

2.4 Computational Assumptions.

This section covers classical computational assumptions that are used in this thesis. Contrarily to complexity theory, most assumptions in cryptography are about average-case hardness of problems rather than worst-case hardness. The assumptions presented hereafter are based on the discrete-logarithm problem or are closely related to the factorization problem. These types of assumptions underpin most of the now classical cryptosystems, and in particular the celebrated RSA [RSA78] and Elgamal [ElG85] cryptosystems.

Despite a result by Shor [Sho94] which shows that these problems can be solved in polynomial time with quantum computers, no classical polynomial-time algorithm to solve them is known today. They thus still are at the heart of many cryptographic schemes built nowadays, especially due to the algebraic properties they provide which allow for practical instantiations.

2.4.1 Group-Family Generators.

A generator of a group family is an algorithm G which takes as an input a security parameter 1^λ, and returns the description of a group $\mathbb{G}$ (groups are denoted multiplicatively) and potentially auxiliary information such as a group element. It is assumed that given the description of $\mathbb{G}$, the group law and the inversion of group elements can be efficiently computed and that group elements can be sampled uniformly at random. When exact-security statements are made, recall that the bit complexity of an elementary operation in a group $\mathbb{G}$ is denoted $T_{\mathbb{G}}$.

The group description may or may not allow to directly infer the group order. Actually, it is precisely this distinction which separates the assumptions based on the discrete-logarithm problem from those related to the factorization problem.

2.4.2 Assumptions on Public-Order-Group Generators

This section presents classical assumptions on generators of groups with public prime orders (note that these groups are necessarily cyclic). The assumptions are stated in asymptotic terms, even though they could of course also be stated in exact-security terms.

Discrete-Logarithm Assumption.

The discrete-logarithm assumption is one of the most central assumptions in cryptography. It has many stronger variants on which numerous practical schemes rely.

Definition 2.4.3 (Discrete-Logarithm Assumption)**.** *Let* G *be a group-family generator. The* discrete logarithm assumption *on* G *states that for any PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[x \leftarrow \mathcal{A}(\mathbb{G}, p, g, h) \colon \begin{array}{c} (\mathbb{G}, p, g) \leftarrow \mathsf{G}(1^\lambda) \\ x \leftarrow_\$ \mathbb{Z}_p^*; h \leftarrow g^x \end{array}\right]$$

is negligible.

Cryptanalysis. Shanks [Sha71] designed a generic (i.e., which only relies on exponentiations and group operations) deterministic algorithm to compute discrete logarithm in time $O(\sqrt{p})$. Pollard [Pol78] later improved this algorithm to use only constant memory and still run in time $O(\sqrt{p})$, but his algorithm is probabilistic. This latter algorithm is now known as Pollard's "rho" algorithm, and several improvements [Tes01,Mon87,BLS11] have been proposed since its introduction. Although more efficient algorithms can exist for specific groups, Shoup [Sho97] showed that any generic algorithm that solves the discrete-logarithm problem must run in time $\Omega(\sqrt{p})$. Pollard's rho algorithm is thus optimal in this sense. As a consequence, research on the discrete-logarithm problem has focused on designing more efficient algorithms for the specific groups commonly used in practical cryptography, and on elaborating groups for which no better algorithm is known.

Instantiation with Finite-Field Multiplicative Subgroups. A common construction of groups with public prime order is as follows. Let p be a large strong prime, i.e., $p' := (p-1)/2$ is also prime. The subgroup of squares[1] in $\mathbb{Z}_p^*$ is a (cyclic) group of prime order p' and is used in practice. To compute a generator of this subgroup, it suffices to generate $h \leftarrow_\$ \mathbb{G}$ and return $g \leftarrow h^2$. The best known algorithms for the discrete-logarithm problem in such groups are based on index calculus (with

[1] The reason why the subgroup of squares is considered is simply for related assumptions such as the decisional Diffie–Hellman assumption to hold.

seminal ideas harking back to Kraitchik's work [Kra22]) and run in time $O\left(\exp\left(a\log^b(t)(\log\log(t))^{1-b}\right)\right)$ for some constants a and b at most $1/3$. The most recent breakthroughs for this class of algorithms are due to Joux [Jou14], and Kleinjung and Wesolowski [KW19]. The existence of sub-exponential algorithms then entails that the parameters must be large; some recommendations advocate primes of 2048 bits for 128 bits of security.

Instantiation with Elliptic Curves. The discrete-logarithm assumptions is also believed to hold over elliptic curves, which are algebraic groups defined by equations of the form $y^2 - x^3 - ax - b = 0$. Despite strenuous cryptanalytic research on elliptic curves, Pollard rho's algorithm is still the best known algorithm to solve the discrete-logarithm problem in chosen particular curves for which the group operation can still be efficiently computed. For such curves, 256-bit primes are enough to achieve 128 bits of security.

Relation Assumptions.

The computational assumptions given below are related and can be reduced in polynomial time to the hardness of computing discrete logarithms, and there is no evidence that these problems are equivalent to the discrete-logarithm problem. Yet, to break these assumptions, no better attack than computing discrete logarithms is known.

Decisional Diffie–Hellman Assumption. The following assumption was introduced by Diffie and Hellman in their work [DH76] which initiated public-key cryptography. A plethora of cryptosystems are based on it, e.g., the Elgamal encryption scheme.

Definition 2.4.4 (Decisional Diffie-Hellman Assumption)**.** *Let* G *be a group family generator. The Decisional Diffie-Hellman assumption (DDH) over* G *states that the advantage (function of* λ*)*

$$\left|\Pr\left[b = \mathcal{A}(\mathbb{G}, \ell, g, g^x, g^y, g^{\alpha_b}) \colon \begin{array}{r}(\mathbb{G}, p, g) \leftarrow \mathsf{G}(1^\lambda)\\ (x, y) \leftarrow_\$ \mathbb{Z}_p^{*2}, b \leftarrow_\$ \{0,1\}\\ \alpha_0 \leftarrow xy \bmod p, \alpha_1 \leftarrow_\$ \mathbb{Z}_p^*\end{array}\right] - \frac{1}{2}\right|$$

of any PPT adversary $\mathcal{A}$ *is negligible.*

Decisional Diffie-Hellman-Inversion Assumption. The following assumption is somewhat related to the DDH assumption but for the problem of inversion.

Definition 2.4.5 (Decisional Diffie-Hellman-Inversion Assumption [BB04])**.** *Let* G *be a group family generator. The* q-Decisional Diffie-Hellman-Inversion

(DDHI) assumption *over* G *states that the advantage (function of* λ*)*

$$\left|\Pr\left[b \stackrel{?}{=} \mathcal{A}(\mathbb{G},p,g,\mathbf{y},z)\colon \begin{array}{r}(\mathbb{G},p,g)\leftarrow \mathsf{G}(1^\lambda)\\ x\leftarrow_\$ \mathbb{Z}_p^*, b\leftarrow_\$ \{0,1\}\\ y_i\leftarrow g^{x^i} \text{ for } i\in\{1,\ldots,q(\lambda)\}\\ \alpha_0\leftarrow 1/x \bmod \ell, \alpha_1\leftarrow_\$ \mathbb{Z}_p^*,\\ z\leftarrow g^{\alpha_b}\end{array}\right] - \frac{1}{2}\right|$$

of any PPT adversary $\mathcal{A}$ *is at most negligible.*

2.4.6 Pairing-Group Structures

An *(asymmetric) bilinear structure* or *pairing-group structure* is a tuple $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e)$ such that p is a prime, $\mathbb{G}_1=\langle g_1\rangle$, $\mathbb{G}_2=\langle g_2\rangle$ and $\mathbb{G}_T$ are p-order groups, and such that $e\colon \mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$ is a pairing, i.e., an efficiently computable non-degenerate ($e\neq 1_{\mathbb{G}_T}$) bilinear map. Bilinearity here means that $e\left(g_1^a,g_2^b\right)=e(g_1,g_2)^{ab}$ for all $a,b\in\mathbb{Z}_p$. In what follows, g_T denote $e(g_1,g_2)$, which is a generator of $\mathbb{G}_T$. Type-3 bilinear structures are bilinear structures for which there is no efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. They are advocated for their efficiency and security guarantees.

A *bilinear structure generator* is an algorithm G which, on the input of a security parameter 1^λ, returns the description of a bilinear structure. For any integers $n,m\geq 1$, given vectors $\mathbf{x}\in\mathbb{G}_1^n$ and $\mathbf{y}\in\mathbb{G}_2^m$, $f(\mathbf{x},\mathbf{y})$ denotes the matrix $\left[e\left(x_i,y_j\right)\right]_{i,j}\in\mathbb{G}_T^{n\times m}$.

Instantiation. Pairing-group structures are instantiated via elliptic curves. Consider an ordinary elliptic curve E over a prime finite field $\mathbb{F}_p$, and let r be the largest prime such that $r\mid \#E(\mathbb{F}_p)$. The embedding degree k of E is the minimal degree k for which all the r-th roots of unity are contained in $\mathbb{F}_{p^k}$. Usually, $\mathbb{G}_1$ and $\mathbb{G}_2$ are r-order subgroups of $E(\mathbb{F}_{p^k})$ and $\mathbb{G}_T$ is the subgroup of $\mathbb{F}_{p^k}^*$ of the r-th roots of unity. The most widely used pairings on ordinary elliptic curves are efficiently computable using variations of Miller's algorithm [Mil04].

Cryptanalysis. For the curves considered in cryptography, the best algorithm to compute discrete logarithms in $E(\mathbb{F}_{p^k})$ is still Pollard's rho algorithm, so it can be done in time $O(\sqrt{r})$. Computing discrete logarithms in $\mathbb{F}_{p^k}^*$ depends on k and p. Kim and Barbulescu [KB16] improved the Tower Number-Field Sieve (TNFS) to solve the discrete-logarithm problem when the embedding degree k is composite. Recently, Fotiadis and Martindale [FM19] then proposed TNFS-secure curves with composite embedding degree.

Assumptions.

The assumptions on generators of pairing-group structures are to a certain extend similar to these on group generators, after eliminating the trivial attacks due to the existence of a pairing. Boyen [Boy08] gave an extensive survey and analysis (with a generic approach) of assumptions on generators of pairing-group structures. The assumptions in this thesis are recalled in the specific chapters in which they are used.

2.4.7 Assumptions on Hidden-Order-Group Generators

Thi section presents assumptions on generator of hidden-order groups. These assumptions are given in exact-security term as all statements in the only chapter (i.e., Chapter 9) in which they appear are in exact-security terms.

A hidden-order-group generator G is an algorithm which takes as an input a security parameter 1^λ and returns the description of a finite Abelian group $(\mathbb{G}, \cdot)$ and an integer $P \geq 2$. Integer P is assumed to be smaller than the order of $\mathbb{G}$, but to still be a super-polynomial function of the security parameter. The role of P is mainly to adjust the soundness of the protocols herein, as their challenge spaces will typically be $[\![0; P^{\Omega(1)} - 1]\!]$.

It is also assumed that given the description of $\mathbb{G}$, the group law and the inversion of group elements can be efficiently computed, that group elements can be sampled uniformly at random and that an upper bound $2^{b_\mathbb{G}}$ on $\mathrm{ord}(\mathbb{G})$ can be efficiently computed, with $b_\mathbb{G} \coloneqq b_\mathbb{G}(\lambda)$ polynomial in λ (it is further assumed that $b_\mathbb{G} = \Omega(\lambda)$). Recall that the bit complexity of an elementary operation in a group $\mathbb{G}$ is denoted $T_\mathbb{G}$.

The following assumptions are classical for hidden-order-group generators and were introduced by Damgård and Fujisaki [DF02]. They are best illustrated for P such that natural integers less than P are factorizable in polynomial time in λ (e.g., $\lambda^{\log^{\Omega(1)}(\lambda)}$ given current knowledge in computational number theory), and for $\mathbb{G}$ as the group $\mathbb{Z}_N^*$ for an RSA modulus N with prime factors p and q such that $p = q = 3 \mod 4$, $\gcd(p-1, q-1) = 2$ and the number of divisors of $p-1$ and $q-1$ with prime factors less than P is of magnitude $O(\lambda)$. However, these assumptions are believed to also hold over generators of ideal-class groups.

Definition 2.4.8 (Strong-Root Assumption)**.** *A group generator* G *satisfies the* (T, ε)*-strong-root assumption if for all* $\lambda \in \mathbb{N}$*, for every adversary* $\mathcal{A}$ *that runs in time at most* $T(\lambda)$*,*

$$\Pr\left[g^n = h \wedge n > 1 \colon \begin{array}{c} (\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right) \\ h \leftarrow_\$ \mathbb{G} \\ (g, n) \leftarrow \mathcal{A}(\mathbb{G}, P, h) \end{array}\right] \leq \varepsilon(\lambda).$$

This assumption is simply a generalization of the strong RSA assumption [BP97, FO97] to hidden-order groups.

Definition 2.4.9 (Small-Order Assumption)**.** *A group generator* G *satisfies the* (T, ε)*-small-order assumption if for all* $\lambda \in \mathbb{N}$*, for every adversary* $\mathcal{A}$ *that runs in time at most* $T(\lambda)$*,*

$$\Pr\left[\begin{array}{c} g^n = 1_{\mathbb{G}} \wedge g^2 \neq 1_{\mathbb{G}} \\ 0 < n < P \end{array} : \begin{array}{c} (\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right) \\ (g, n) \leftarrow \mathcal{A}(\mathbb{G}, P) \end{array}\right] \leq \varepsilon(\lambda).$$

The small-order assumption simply states that it should be hard to find low-order elements in the group (different from $1_{\mathbb{G}}$), except for square roots of unity which may be easy to compute (e.g., -1 in RSA groups). In the group $\mathbb{Z}_N^*$ for $N = pq$ with p and q prime such that $\gcd(p-1, q-1) = 2$, Damgård and Fujisaki [DF02] showed that factoring N can be reduced to this problem in polynomial time if integers less than P are factorizable in polynomial time in λ.

Definition 2.4.10 (Orders with Low Dyadic Valuation)**.** *A group generator* G *satisfies the low-dyadic-valuation assumption on orders if for all* $\lambda \in \mathbb{N}$*, for every* $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$*, for every* $g \in \mathbb{G}$*,* $\mathrm{ord}(g)$ *is divisible by* 2 *at most once.*

Notice that in the group $\mathbb{Z}_N^*$ for $N = pq$ with p and q prime such that $p = q = 3 \mod 4$, the order of any element is divisible by 2 at most once since 2 divides $p-1$ and $q-1$ exactly once.

Definition 2.4.11 (Many Rough-Order Elements or μ-Assumption)**.** *An integer is said to be* P*-rough if all its prime factors are greater than or equal to* P*. A group generator* G *satisfies the* μ*-assumption that there are many rough-order elements in the groups generated by* G *(or simply the* μ*-assumption) if for all* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathrm{ord}(h) \text{ is } P\text{-rough} : \begin{array}{c} (\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right) \\ h \leftarrow_{\$} \mathbb{G} \end{array}\right] \geq \mu(\lambda).$$

2.5 Cryptographic Primivites

This section introduces primitives that are fundamental in cryptography.

2.5.1 Public-Key Encryption

Public-key encryption is a fundamental cryptographic primitives which allows two parties who have never met to confidentially exchange information. More precisely, public-key encryption enables a party, A, to privately send a message to another party, B (the user hereafter), given the public key of the latter, assuming it to be authentic. Using a corresponding secret key, B can decipher A's message.

Formally, a public-key encryption scheme consists of a setup algorithm $\mathsf{Setup}\left(1^{\lambda}\right) \to pp$, a key-generation algorithm $\mathsf{KG}(pp) \to (pk, sk)$ which returns a public encryption key and a secret decryption key, a probabilistic encryption algorithm $\mathsf{Enc}(pk, m; r) \to C$ (the randomness r may at times be omitted from the syntax) and a deterministic decryption algorithm $\mathsf{Dec}(sk, C) \to m/\perp$.

Golwasser and Micali [GM84] proposed the first security definitions of public-key encryption, and standard definitions nowadays include the so-called INDistinguishability under Chosen-Plaintext and Chosen-Ciphertext Attacks (IND-CPA and IND-CCA). A prominent example of IND-CPA scheme is the aforementioned Elgamal scheme [ElG85] under the DDH assumption, and under the same assumption, an example of IND-CCA scheme is the Cramer–Shoup encryption scheme [CS98].

2.5.2 Digital Signatures

(Digital) signature schemes constitute another fundamental cryptographic primitive which addresses the issue of authenticity in digital communication. They are to digital communication what seals were to physical communication in ancient times. A signature scheme allows a party to compute, with a secret key, signatures on messages that can be verified by anyone who has an authentic copy of the corresponding public verification key. A prime example of signature scheme is the cryptosystem due to Rivest, Shamir and Adleman [RSA78].

Formally, a signature scheme consists of a setup algorithm $\mathsf{Setup}\left(1^{\lambda}\right) \to pp$, a key-generation algorithm $\mathsf{KG}(pp) \to (vk, sk)$, a signing algorithm $\mathsf{Sign}(sk, m) \to \sigma$ and a verification algorithm $\mathsf{Vf}(vk, m, \sigma) \to b \in \{0, 1\}$ which returns a bit indicating whether the signature is valid on a message – bit 1 conventionally indicates validity.

A classical security notion for digital signature schemes is that of existential unforgeability against chosen-message attacks. It states that an adversary should not be able to compute a valid signature on a new message without the knowledge of the secret key.

2.5.3 Non-Interactive Commitments

(Non-Interactive) commitment schemes play a central role in cryptography as they allow a party to commit herself to a value in a *commitment* which does not disclose any information about the underlying value, and to later reveal that value to another party without any possibility to change it.

Formally, a (non-interactive) commitment scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^{\lambda}\right) \to pp$: generates public parameters (or common-reference string)

on the input of a security parameter 1^λ. These parameters are implicit inputs to the other algorithms.

$\mathsf{Com}(x) \to (C, d)$: computes a commitment C to a value and an opening or decommitment information d.

$\mathsf{ComVf}(C, x, d) \to b \in \{0, 1\}$: a deterministic returns a bit indicating whether the decommitment d is valid (bit 1) for C and x, or not (bit 0). It is assumed that if $x = \perp$, then it returns 0.

The following security definitions are given in asymptotic term, but could also be stated in exact-security terms.

A commitment scheme is *correct* if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ and for all x $\Pr\left[\mathsf{ComVf}(C, x, d) = 1 \colon (C, d) \leftarrow \mathsf{Com}(x)\right] = 1$.

A commitment scheme is *hiding* (statistically hiding) if for all $\lambda \in \mathbb{N}$, for every PPT (computationally unbounded) adversary $\mathcal{A}$,

$$\left| \Pr\left[b = b' \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ (x_0, x_1, st) \leftarrow \mathcal{A}(pp) \\ b \leftarrow_\$ \{0,1\} \\ (C, d) \leftarrow \mathsf{Com}(x_b) \\ b' \leftarrow \mathcal{A}(st, C) \\ \text{return } (b, b') \end{array} \right] - 1/2 \right|$$

is negligible. If the above difference is nil, then the scheme is said to be *perfectly hiding*.

A commitment scheme is *binding* if for all $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} \mathsf{ComVf}(C, x_0, d_0) = \mathsf{ComVf}(C, x_1, d_1) = 1 \\ \wedge x_0 \neq x_1 \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ (C, x_0, d_0, x_1, d_1) \leftarrow \mathcal{A}(pp) \end{array} \right]$$

is negligible.

Pedersen Commitments. The most commonly used commitment scheme in cryptography ought to be the Pedersen commitment scheme [Ped92]. It is additively homomorphic and allows to algebraic prove properties of a committed values. Given G a group-family generator, let $\mathbb{G}$ be a cyclic group of prime order p generated by G. Let g_1, g_2 denote two generators of $\mathbb{G}$. The common reference string is $(\mathbb{G}, p, g_1, g_2)$. For a value $a \in \mathbb{Z}_p$, the pedersen-commitment algorithm computes generates $r \leftarrow_\$ [\![0, p-1]\!]$ and computes $C \leftarrow g_1^a g_2^r$. Note that the commitments are perfectly hiding. To verify a commitment C to a with decommitment information r, the opening algorithm simply verifies that $C = g_1^a g_2^r$. Under the discret-logarithm assumption over G, the scheme is binding.

Extractable Commitments.

A commitment scheme $(\mathsf{Setup}, \mathsf{Com}, \mathsf{ComVf})$ is extractable (as defined by Abdalla et al. [ABB+13]) if there exists an algorithm $\mathsf{TSetup}\left(1^\lambda\right) \to (pp, \tau)$ which generates public parameters and a trapdoor, and another algorithm $\mathsf{ExtCom}\,(\tau, C) \to x/\perp$ which, on the input of a trapdoor and of a commitment, returns a message or $\perp$ if the commitment is invalid.

The scheme then satisfies *trapdoor correctness* if for all 1^λ, for all $(pp, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right)$, for all x, $\Pr\left[\mathsf{ExtCom}(\tau, C) = x \colon (C, \delta) \leftarrow \mathsf{Com}(x)\right] = 1$.

The commitment scheme satisfies *setup indistinguishability* if no PPT algorithm can distinguish the output of Setup from the first component of the output of TSetup with a non-negligible advantage.

Lastly, an extractable commitment scheme satisfies *binding extractability* if ExtCom runs in polynomial time and if for all 1^λ, for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|l} \mathsf{ComVf}(C, x, d) = 1 & (pp, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right) \\ \wedge x \neq x' & (C, x, d) \leftarrow \mathcal{A}^{\mathsf{ExtCom}(\tau, \cdot)}(pp) \\ & x' \leftarrow \mathsf{ExtCom}\,(\tau, C) \end{array}\right]$$

is negligible.

Public-key encryption schemes secure against chosen-message attacks are perfectly binding commitment schemes which are additionally extractable since the secret key can be used as a trapdoor.

2.6 Zero-Knowledge Arguments

This section introduce zero-knowledge arguments. They allow a prover to convince a computationally-bounded verifier that a statement is true without revealing any information beyond that. More precisely, a prover wants to convince a verifier that a certain word x belongs to an NP language $\mathcal{L}$. Arguments are distinguished from proofs in that provers in arguments are computationally bounded.

Interactive proof systems were first studied by Babai [Bab85], and Goldwasser, Micali and Rackoff [GMR85]. Arguments systems were later introduced by Brassard, Chaum and Crepeau [BCC88]. Their underlying idea is that a cheating prover should not be able to convince a verifier so long as it cannot solve a presumably hard computational task. This relaxation is very powerful as it for instance allows to build systems with sub-linear communication complexity, as for instance shown in Chapter 9.

This section first defines interactive proof systems, then non-interactive proof systems in both the random-oracle model and common-reference string model.

2.6.1 Interactive Systems

This section defines argument systems for families of languages. The languages are parametrized by public parameters and Common-Reference String (CRS). As a simple example, given an Abelian group $\mathbb{G}$ (which could be non-cyclic) and an element $h \in \mathbb{G}$ (the parameters) and another element $g \in \langle h \rangle$ (the CRS), consider the language of group elements $C \in \mathbb{G}$ such that there exists $x, y \in \mathbb{Z}$ for which $C = g^x h^y$. This language is clearly parametrized by the parameters and the CRS, and one can give an argument system for this parametrized language in the same vein as what is subsequently done in the paper. However, to lighten the notation, arguments will be (abusively) referred to as arguments for languages rather than arguments for families of languages.

Formally, an argument system (or protocol) for a language $\mathcal{L} = \mathcal{L}_{pp,crs}$ (or equivalently, for the corresponding relation $\mathcal{R} = \mathcal{R}_{pp,crs}$) consists of a quadruple $\Pi = (\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf})$ such that $\mathsf{Setup}\left(1^\lambda\right) \to pp$ returns public parameters on the input of a security parameter, $\mathsf{CRSGen}(pp) \to crs$ returns a CRS, and $\langle \mathsf{Prove}(crs, x, w) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle \to (\tau, b) \in \{0,1\}^* \times \{0,1\}$ are interactive algorithms (τ denotes the transcript of the interaction and b the decision bit of Vf). The public parameters are assumed to be tacit inputs to algorithms Prove and Vf, even though they may at times be made explicit for instantiated protocols, especially when the CRS is the empty string (in which case the CRS is omitted from the syntax).

Completeness. Π is complete if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$, $crs \leftarrow \mathsf{CRSGen}\,(pp)$, for all $(x, w) \in \mathcal{R}$,

$$\Pr\left[(*, 1) \leftarrow \langle \mathsf{Prove}(crs, x, w) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle\right] = 1.$$

Soundness. Π satisfies (T, ε)-soundness if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$,

$$\Pr\left[\begin{array}{ll} & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); crs \leftarrow \mathsf{CRSGen}\,(pp) \\ b = 1 \wedge x \notin \mathcal{L}\colon & (st, x) \leftarrow \mathcal{A}(crs) \\ & (\tau, b) \leftarrow \langle \mathcal{A}(st, x) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle \end{array}\right] \leq \varepsilon(\lambda).$$

This definition of soundness formalizes the idea that an adversary should not be able to convince the verifier that a word x is in $\mathcal{L}$ although it is actually in $\bar{\mathcal{L}} \coloneqq \{0,1\}^* \setminus \mathcal{L}$. Groth, Ostrovsky and Sahai [Gro06, GOS06] relaxed the notion of soundness so that a protocol which satisfies the relaxed notion only guarantees that an adversary cannot convince the verifier that a word is in $\mathcal{L}$ when it is actually in the complement $\bar{\Lambda}$ of language $\Lambda \supseteq \mathcal{L}$ (to be completely formal, Λ should also parametrized by pp and crs). They called this new notion co-soundness or *culpable soundness*. It means that a

malicious could still convince the verifier that a word is $\mathcal{L}$ when it is actually in $\Lambda \setminus \mathcal{L}$. However, for many applications, this notion is sufficient.

Formally, a protocol Π for a Language $\mathcal{L}$ satisfies $(T, \varepsilon, \Lambda \supseteq \mathcal{L})$-soundness if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$,

$$\Pr\left[b = 1 \wedge x \notin \Lambda \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x) \leftarrow \mathcal{A}(crs) \\ (\tau, b) \leftarrow \langle \mathcal{A}(st, x) \rightleftharpoons \mathsf{Vf}(crs, x)\rangle \end{array}\right] \leq \varepsilon(\lambda).$$

Extractability. Π is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon)$-extractable if for every deterministic algorithm Prove^* running in time at most $T_{\mathsf{Prove}^*}(\lambda)$, there exists an algorithm (called extractor) $\mathcal{E}(pp, x) \to w$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T_{\mathcal{A}}(\lambda)$,

$$* \Pr\left[(x, w) \notin \mathcal{R} \ : \ \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x, s) \leftarrow \mathcal{A}(crs) \\ w \leftarrow \mathcal{E}^{\langle \mathsf{Prove}^*(crs, x, s) \rightleftharpoons \mathsf{Vf}(crs, x)\rangle}(crs, x) \end{array}\right] \leq \varepsilon(\lambda),$$

- the running time of $\mathcal{E}$ is at most $T_{\mathcal{E}}$ in expectation, with $T_{\mathcal{E}}$ depending on T_{Prove^*} and

$$\varepsilon_{\mathcal{A}, \mathsf{Prove}^*}(\lambda) \coloneqq \Pr\left[\beta = 1 \ : \ \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x, s) \leftarrow \mathcal{A}(crs) \\ (\tau, \beta) \leftarrow \langle \mathsf{Prove}^*(crs, x, s) \rightleftharpoons \mathsf{Vf}(crs, x)\rangle \end{array}\right].$$

The above definition means that if Π is extractable, $\mathcal{E}$ can extract a valid witness from any prover Prove^* with the running time of $\mathcal{E}$ depending on the running time of Prove^* and on the success probability of $(\mathcal{A}, \mathsf{Prove}^*)$, except with probability at most $\varepsilon(\lambda)$. Π is thus an argument of knowledge, which implies that it is sound. The string s given to Prove^* can considered as an internal state which includes its random string.

Similarly to the notion of soundness, the notion of extractability can be extended to a notion of *culpable extractability* w.r.t. a relation $\Sigma \supseteq \mathcal{R}$.

Honest-Verifier Zero-Knowledge. Π is $(T, T_{\mathsf{Sim}}, \varepsilon)$-honest-verifier zero-knowledge ($(T_{\mathsf{Sim}}, \varepsilon)$-statistically honest-verifier zero-knowledge) if there exists an algorithm Sim running in time at most $T_{\mathsf{Sim}}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T(\lambda)$ (for every com-

putationally unbounded adversary $\mathcal{A}$),

$$\left|\Pr\left[\ (x,w)\in\mathcal{R}\wedge b=1 \quad : \begin{array}{l} pp\leftarrow \mathsf{Setup}\left(1^\lambda\right)\\ (crs,st,x,w)\leftarrow\mathcal{A}(pp)\\ (\tau,\beta)\leftarrow\langle\mathsf{Prove}(crs,x,w)\rightleftharpoons\mathsf{Vf}(crs,x)\rangle\\ b\leftarrow\mathcal{A}(st,(\tau,\beta))\end{array}\right]\right.$$
$$\left.-\ \Pr\left[\ (x,w)\in\mathcal{R}\wedge b=1\ :\ \begin{array}{l} pp\leftarrow \mathsf{Setup}\left(1^\lambda\right)\\ (crs,st,x,w)\leftarrow\mathcal{A}(pp)\\ (\tau,\beta)\leftarrow\mathsf{Sim}(crs,x)\\ b\leftarrow\mathcal{A}(st,(\tau,\beta))\end{array}\right]\right|$$
$$\leq\varepsilon(\lambda).$$

Note that the common-reference string is generated by the adversary in this definition. The reasons are the same as for the commitment key in the definition of the hiding property of commitment schemes. A weaker definition in which the common-reference string is honestly generated could also be considered.

Π is said to be *public coin* if all messages sent by Vf are chosen uniformly at random and independently of the messages sent by algorithm Prove.

Schnorr Proofs.

A simple example of a zero-knowledge proof is Schnorr's proof [Sch90] of knowledge of discrete logarithms in groups of public prime orders which perfectly illustrates the "cut-and-choose" paradigm.

In a group $\mathbb{G}$ of prime order p, given g and y in $\mathbb{G}$, the goal of the prover is to prove knowledge of $x\in\mathbb{Z}_p$ such that $y=g^x$. To do so, the prover generates $k\leftarrow_\$\mathbb{Z}_p$, and computes and sends $t\leftarrow g^k$ to the verifier. The verifier chooses a challenge $c\leftarrow_\$\mathbb{Z}_p$ and sends it to the prover. The prover then replies with $r\leftarrow k-cx$ and the verifier accepts if and only if $g^r y^c=t$.

Interactive Arguments in the Random–Oracle Model.

Interactive arguments could also be defined in a model in which the protocol algorithms are given access to a random oracle (in this paper, it will primarily be in case CRSGen needs to compute a non-interactive proof that the CRS is well-formed, as in Chapter 9 for instance). The soundness and zero-knowledge properties of the protocol may then be affected by the number of random-oracle queries made by the parties, e.g., if the protocol involves as sub-protocol the Fiat–Shamir (see the next section) non-interactive variant of another protocol, as it is the case in Section 9.3. The definitions of these properties should then be adapted to include an upper-bound $q_{\mathcal{H}}$ on the number of queries that the adversary can make.

For instance, given a random-oracle $\mathcal{H}$, one can define $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, q_{\mathcal{H}}, \varepsilon)$-*extractability* in the same way as $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon)$-extractability in the standard model was defined, but with the difference that $\mathcal{A}$ and Prove^* are given oracle access to $\mathcal{H}$ and can together make at most $q_{\mathcal{H}}(\lambda)$ queries. The other definitions are adapted in a similar fashion.

Fiat–Shamir Heuristic.

The Fiat–Shamir heuristic [FS87] can be used to turn a public-coin, interactive argument system into a non-interactive one in the random-oracle model [BR93]. Given a random oracle $\mathcal{H}$, the messages of the verifier are computed by evaluating $\mathcal{H}$ at the word and the transcript of the interactive protocol until that point of the computation of the prover. With oracle access to $\mathcal{H}$, the prover can then compute a full transcript (or argument), further denoted π instead of τ, without interacting with the verifier, and this latter can also verify the transcript without any interaction.

The non-interactive argument system derived from an interactive one $\Pi = (\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf})$ via the Fiat–Shamir heuristic with a random oracle $\mathcal{H}$ is denoted $FS.\Pi^{\mathcal{H}} \coloneqq \left(\mathsf{Setup}, \mathsf{CRSGen}, FS.\mathsf{Prove}^{\mathcal{H}}, FS.\mathsf{Vf}^{\mathcal{H}}\right)$. Note that the original interactive protocol Π may already be in the random-oracle model as in the previous section, but it is further assumed to be with a different oracle (that is not made explicit in the notation for simplicity).

Completeness. A non-interactive protocol $FS.\Pi^{\mathcal{H}}$ is said to be complete if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right)$, $crs \leftarrow \mathsf{CRSGen}(pp)$, for all $(x, w) \in \mathcal{R}$, $\Pr\left[FS.\mathsf{Vf}^{\mathcal{H}}\left(crs, x, FS.\mathsf{Prove}^{\mathcal{H}}(crs, x, w)\right) = 1\right] = 1$.

As in the interactive case, the following notions of soundness and extractability can be extended to those of culpable soundness and culpable extractability.

Soundness. $FS.\Pi^{\mathcal{H}}$ satisfies $(T, q_{\mathcal{H}}, \varepsilon)$-soundness if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$ and makes at most $q_{\mathcal{H}}(\lambda)$ queries to $\mathcal{H}$,

$$\Pr\left[FS.\mathsf{Vf}^{\mathcal{H}}(crs, x, \pi) = 1 \wedge x \notin \mathcal{L} \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}(pp) \\ (x, \pi) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(crs) \end{array}\right] \leq \varepsilon(\lambda).$$

Extractability. $FS.\Pi^{\mathcal{H}}$ is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathsf{Ext}}, q_{\mathcal{H}}, \varepsilon)$-extractable if for every deterministic algorithm Prove^* running in time at most $T_{\mathsf{Prove}^*}(\lambda)$ there exists an algorithm $(\mathsf{Ext}_0(Q, q) \to Q',\ \mathsf{Ext}_1(pp, x) \to w)$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T_{\mathcal{A}}(\lambda)$, if Prove^* and $\mathcal{A}$ together make at most $q_{\mathcal{H}}(\lambda)$ queries to $\mathcal{H}$,

$$* \ \Pr\left[\begin{array}{ll} & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); crs \leftarrow \mathsf{CRSGen}(pp); Q \leftarrow \emptyset \\ (x,w) \notin \mathcal{R} \ : & (x,s) \leftarrow \mathcal{A}^{\mathsf{Ext}_0(Q,\cdot)}(crs) \\ & w \leftarrow \mathsf{Ext}_1^{\mathsf{Prove}^{*\mathsf{Ext}_0(Q,\cdot)}(crs,x,s)}(crs,x) \end{array}\right] \leq \varepsilon(\lambda),$$

* the running time of Ext is at most T_{Ext} in expectation, with T_{Ext} depending on T_{Prove^*} and

$$\varepsilon_{\mathcal{A},\mathsf{Prove}^*} \coloneqq \Pr\left[\begin{array}{ll} & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); crs \leftarrow \mathsf{CRSGen}(pp) \\ b = 1 \ : & (x,s) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(crs) \\ & \pi \leftarrow \mathsf{Prove}^{*\mathcal{H}(\cdot)}(crs,x,s) \\ & b \leftarrow FS.\mathsf{Vf}^{\mathcal{H}(\cdot)}(crs,x,\pi) \end{array}\right].$$

Algorithm $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ is given access to a Prove^* oracle which can be rewound to any step of its computation and run anew with fresh Ext_0 randomness.

Zero-Knowledge. $\Pi^{\mathcal{H}}$ is $(T_{\mathcal{A}}, T_{\mathsf{Sim}}, q_{\mathcal{H}}, \varepsilon)$-zero-knowledge if there exists an algorithm Sim running in time at most $T_{\mathsf{Sim}}(\lambda)$ such that $\mathsf{Sim}(0,Q,q) \to (h,Q)$ and $\mathsf{Sim}(1,Q,crs,x) \to (\pi,Q)$, and such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$ and is given oracle access to $\mathcal{H}$ and Prove,

$$\left|\Pr\left[b=1 : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); (crs,st) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp) \\ b \leftarrow \mathcal{A}^{\mathcal{H}(\cdot),FS.\mathsf{Prove}^{\mathcal{H}}(crs,\cdot)}(st) \end{array}\right]\right.$$
$$\left. - \Pr\left[b=1 : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); Q \leftarrow \emptyset \\ (crs,st) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp) \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}_0}(Q,\cdot),\mathcal{O}_{\mathsf{Sim}_1}(Q,crs,\cdot)}(st) \end{array}\right]\right| \leq \varepsilon(\lambda),$$

with $\mathcal{O}_{\mathsf{Sim}_0}$ an oracle that computes $(h,Q) \leftarrow \mathsf{Sim}(0,Q,q)$ on input (Q,q) and returns h, and $\mathcal{O}_{\mathsf{Sim}_1}$ an oracle that computes $(\pi,Q) \leftarrow \mathsf{Sim}(1,Q,crs,x)$ if $(x,w) \in \mathcal{R}$ and returns π, and returns $\perp$ if $(x,w) \notin \mathcal{R}$. Set Q can be considered as a state which stores all pairs (q,h) of queries and responses. The total number of random-oracle calls incured by direct $\mathcal{H}$ queries and by Prove queries from $\mathcal{A}$ can be at most $q_{\mathcal{H}}$.

Notation. Whenever the relation is not made explicit because it is clear from the context and that the Fiat–Shamir transform is used, then the relation is embedded in the notation of algorithm Prove. For instance, for the Fiat–Shamir version of Schnorr's proof, the notation $FS.\mathsf{Prove}\{x\colon y = g^x\}$ is be used. The random oracle involved in the proof is not made explicit as it is here also assumed to be clear from the context.

2.6.2 Non-Interactive Systems

This section defines non-interactive systems in a model without random oracles. Note that a non-interactive proof system cannot be sound and zero-knowledge unless there exists at least a common-reference string used by both parties that could potentially be trapdoored. Indeed, would such a scheme exist, any malicious prover could simply run the algorithm of the simulator and contradict soundness. On this account, this section presents proof systems of which the properties rely on CRS and trapdoors.

A non-interactive proof system Π for a language $\mathcal{L} = \mathcal{L}_{crs}$ (with corresponding relation $\mathcal{R}$) consists of an algorithm $\mathsf{Setup}\left(1^\lambda\right) \to pp$ which returns public parameters, an algorithm $\mathsf{CRSGen}(pp) \to crs$ which returns a common reference string, an algorithm $\mathsf{Prove}(crs, x, w) \to \pi$ which computes a proof on the input of a word x and of a witness w, and an algorithm $\mathsf{Vf}(crs, x, \pi) \to b \in \{0, 1\}$ which returns a bit indicated whether the proof is considered valid.

Π is *complete* if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$, $crs \leftarrow \mathsf{CRSGen}(pp)$, for all $(x, w) \in \mathcal{R}$, $\Pr\left[\mathsf{Vf}(crs, x, \mathsf{Prove}(crs, x, w)) = 1\right] = 1$.

Π satisfies (T, ε)-*soundness* if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T(\lambda)$,

$$\Pr\left[\mathsf{Vf}(crs, x, \pi) = 1 \wedge x \notin \mathcal{L} \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ crs \leftarrow \mathsf{CRSGen}(pp) \\ (x, \pi) \leftarrow \mathcal{A}(crs) \end{array}\right] \leq \varepsilon(\lambda).$$

Π is $(T_{\mathsf{Ext}}, T_{\mathcal{A}}, \varepsilon)$-*extractable* if there exists an algorithm $\mathsf{TSetup}\left(1^\lambda\right) \to (crs, \tau)$ and an algorithm $\mathsf{Ext}(crs, \tau, x, \pi) \to w$ running in time at most $T_{\mathsf{Ext}}(\lambda)$ such that the distribution of the first component of TSetup is the same as that of CRSGen, and such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T_{\mathcal{A}}(\lambda)$,

$$\Pr\left[\mathsf{Vf}(crs, x, \pi) = 1 \wedge (x, \mathsf{Ext}(crs, \tau, x, \pi)) \notin \mathcal{R} \colon \begin{array}{l} (crs, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right) \\ (x, \pi) \leftarrow \mathcal{A}(crs) \end{array}\right] \leq \varepsilon(\lambda).$$

Π is ε-*witness-indistinguishable* if for all $\lambda \in \mathbb{N}$, for every PPT adver-

sary $\mathcal{A}$,

$$\left|\Pr\left[b = b' \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ crs \leftarrow \mathsf{CRSGen}(pp) \\ (st, x, w_0, w_1) \leftarrow \mathcal{A}(crs) \\ b \leftarrow_\$ \{0,1\} \\ \text{if } (x, w_0) \notin \mathcal{R} \text{ or } (x, w_1) \notin \mathcal{R} \\ \quad b' \leftarrow_\$ \{0,1\} \\ \quad \text{return } (b, b') \\ \pi \leftarrow \mathsf{Prove}(crs, x, w_b) \\ b' \leftarrow \mathcal{A}(st, \pi) \\ \text{return } (b, b') \end{array}\right] - 1/2\right| \leq \varepsilon(\lambda).$$

Π is (T, ε)-*composable zero-knowledge* if there exist two other algorithms $\mathsf{TSetup}\left(1^\lambda\right) \to (crs, \tau)$ and $\mathsf{Sim}(crs, \tau, x) \to \pi$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T(\lambda)$,

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(crs) \colon pp \leftarrow \mathsf{Setup}\left(1^\lambda\right), crs \leftarrow \mathsf{CRSGen}(pp)\right] - \Pr\left[1 \leftarrow \mathcal{A}(crs) \colon (crs, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right)\right]\right| \leq \varepsilon(\lambda)$$

and

$$\Pr\left[1 \leftarrow \mathcal{A}(st, crs, \pi) \colon \begin{array}{l} (crs, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right) \\ (st, x, w) \leftarrow \mathcal{A}(crs, \tau) \\ \pi \leftarrow \mathsf{Prove}(crs, x, w) \end{array}\right] = \Pr\left[1 \leftarrow \mathcal{A}(st, crs, \pi) \colon \begin{array}{l} (crs, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right) \\ (st, x, w) \leftarrow \mathcal{A}(crs, \tau) \\ \pi \leftarrow \mathsf{Sim}(crs, \tau, x) \end{array}\right].$$

Π is (q, ε)-*zero-knowledge* if there exist two algorithms $\mathsf{TSetup}\left(1^\lambda\right) \to (crs, \tau)$ and $\mathsf{Sim}(crs, \tau, x) \to \pi$ such that for all $\lambda \in \mathbb{N}$, the distribution of crs is the same as that of Setup, and such that for every PPT adversary $\mathcal{A}$ that makes at most q oracle queries,

$$\left|\Pr\left[b = 1 \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ crs \leftarrow \mathsf{CRSGen}(pp) \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Prove}}(crs, \cdot)}(crs) \end{array}\right] - \Pr\left[b = 1 \colon \begin{array}{l} (crs, \tau) \leftarrow \mathsf{TSetup}\left(1^\lambda\right) \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}}(crs, \tau, \cdot)}(crs) \end{array}\right]\right| \leq \varepsilon(\lambda),$$

with $\mathcal{O}_{\mathsf{Prove}}(crs, \cdot)$ an oracle that computes and returns $\mathsf{Prove}(crs, x, w)$ on input $(x, w) \in \mathcal{R}$, and returns $\perp$ on input $(x, w) \notin \mathcal{R}$; and $\mathcal{O}_{\mathsf{Sim}}$ an oracle that computes and returns $\mathsf{Sim}(crs, x, \tau)$ on input $(x, w) \in \mathcal{R}$, and returns $\perp$ on input $(x, w) \notin \mathcal{R}$.

Part I

Group Signatures and Protocols for Message Confidentiality

Chapter 3

Introduction

The first part of this thesis explores, through the prism of privacy, various real-world problematics which require confidential communication. More precisely, in a first subpart, it investigates the case of vehicle-to-vehicle communication in Chapter 5 and proposes a novel mechanism to securely exchange sensitive data to ensure traffic safety, and do so while protecting users from tracking. This does not only require vehicles to be able to communicate confidentially but also authentically, i.e., vehicles must be assured that the messages they receive originate from other accredited vehicles. Hence the study of group signatures in Chapter 4 as authentication mechanism, even though they constitute a contribution of independent interest. For stronger security guarantees, these group signatures are designed in a context with several authorities to prevent single points of failure.

In a second subpart, i.e., in Chapter 6, the problem tackled is that of encryption in a context in which users do not have access to secure storage for their secret keys and thus cannot be guaranteed that only they can read the messages intended to them. The goal therein is then to achieve security guarantees as strong as to those provided by secure storage without assuming it to be available, while requiring users solely to remember a password.

These cryptographic protocols are formally defined in the corresponding chapters, their security is approached from a property-based perspective, and efficient, practical solutions are proposed and proved secure via computational reductions following the principles of "provable security" laid by Goldwasser and Micali [GM82].

3.1 Group Signatures

Group signatures, introduced by Chaum and van Heyst [Cv91], are a fundamental cryptographic primitive which allows members of a user group to anonymously sign messages on behalf of the group, after having been added to the group by a party called *issuer*. That is, anyone can verify that a

signature was computed by a group member, but no one can identify the group member that did, nor even tell whether two signatures were computed by the same member. Only a dedicated authority called *opener* can reveal the identity of a signer, e.g., in case of dispute. In some definitions of group signatures [BMW03, BCN+10], the issuance and opening roles are assumed by the same authority who is then entirely responsible for administrating the group; in which case the authority is called *manager*.

In *static* group signatures, first formalized by Bellare, Micciancio and Warinschi [BMW03], the group members are fixed at the setup phase, whereas in *dynamic* group signatures, formalized by Bellare, Shi and Zhang [BSZ05], users can join the group at any time. This latter variant of group signatures is more relevant in practice due to its flexibility.

Extensions of dynamic group signatures have many a practical application. For instance, Camenisch, Neven and Rückert [CNR12] proposed a generalization that is intended to be used as a privacy-preserving authentication mechanism. In their generalization, the issuer generates for each user a credential associated to list of *attributes*. Such an attribute can for example be a date of birth or a social security number. The user then employs the credential to generate tokens to authenticate messages and only disclose the attributes required by the authentication policy. They called this generalization anonymous-attribute-token (with revocation) systems, but it is essentially a dynamic group-signature scheme that supports attributes.

Real-World Applications. A motivation for such schemes is that electronic services nowadays require superfluous information from users to authenticate them. A user who may want to register to an online betting website or access age-restricted content should only prove that she is old enough to do so. In practice, however, she also has to provide personal information that is irrelevant for the requested service such as her full date of birth or her phone number. Requiring unnecessary information from users thus harms their privacy and exposes them to identity theft and abuse by cyber-criminals. It also puts the onus on service providers to protect users' data and deploy extra effort to be compliant with data-protection regulations. Dynamic group signatures with attributes instead provide an attribute-based authentication mechanism with which users can authentication themselves while disclosing only minimal, necessary information. Such variants of group signatures also find applications in secure hardware computation with trusted platform modules [GPC+,BCC04,Gro14,fS15,CCD+17].

Another real-world application is that of vehicle-to-vehicle communication systems [WWKH13, PSFK15, NBCN17a], which is the main focus of Chapter 5. In such systems, vehicles must be able to communicate with each other and be assured that the messages they receive originate from other accredited vehicles. Yet, the communication should not expose users to mass

surveillance or tracking by criminals, i.e., the vehicles they use should remain anonymous. Still, in case of suspicious activity, the anonymity of a vehicle that sent a certain message involved in it should be revocable by a designated authority that can then ban it from the system. Rephrased in technical terms, vehicles should be able to anonymously send messages to each other so that a vehicle receiving a message should be certain that it originates from a member of the group of vehicles authorized *at that time*. Furthermore, if need be, the anonymity of the vehicle that sent a message should be revocable to allow for the exclusion of malicious vehicles and their associated credentials.

Threshold Schemes. Unfortunately, despite the extensive research on group signatures, the existing security models are so that no anonymity is guaranteed as soon as the opener is corrupt. It means that if a *single* authority is corrupt, no privacy is ensured and the whole benefit of group signatures is lost. Moreover, in these models, a malicious issuer can produce signatures that cannot be trace back to any group members, and therefore no one can be held accountable in case of dispute. The problem is here that the issuer and the opener constitute single points of failure and that trust is placed on a single authority for each role. A solution to mitigate the impact of corrupt authorities is usually use threshold cryptography [Sha79, DF90], i.e., in the present case, consider not just one but several authorities and distribute the capabilities of the issuer and opener over multiple entities, of whom a threshold number must collaborate to add a user or open a signature. As long as strictly less than a certain *threshold* number of them is corrupt, the security guarantees should be retained.

Short Group Signatures. Early schemes were based on the strong RSA assumption [ACJT00, CL02], but the focus later shifted to bilinear maps [BBS04, BS04, CL04, BCN$^+$10, LY10, PS16, DS18] for their better efficiency. Most of them follow a modular sign-encrypt-and-prove (or simply sign-and-encrypt) approach, whereby a user's signing key is a certificate on her identity, and a group signature contains an encryption of her identity and a zero-knowledge proof (bound to the signed message) that the user knows a valid certificate for the encrypted identity. The modular use of encryption to enable opening readily allows for threshold opening: it suffices to replace the underlying encryption scheme with another that supports threshold decryption [DF90]. Nevertheless, lifting a group-signature scheme to a threshold-issuance setting is not as straightforward. Besides, this modular approach typically results in signatures that can be prohibitively large for several applications.

The most efficient ("GetShorty") group signatures [BCN$^+$10, PS16] depart from the sign-and-encrypt paradigm though, yielding the shortest sig-

nature sizes to date. Instead of adding an encryption of the user's identity to every signature, they rely on the issuer for opening. To reveal the identity of the user that computed a signature, the issuer maintains a list of the membership credentials he has generated and tests the signature against each entry. It makes opening expensive for the benefit of having short signatures, which perfectly fits for all applications where signatures must be short and opening an uncommon practice, such as in vehicle-to-vehicle communication. A disadvantage of this approach is that it merges the roles of issuer and opener into a single party that has to be trusted for anonymity and unforgeability.

Unfortunately, these schemes are difficult to map to a threshold setting. A first obstacle is that to trace the signatures of a user, her identifier generated during the issuance protocol is necessary. A second hurdle is that their underlying base signature schemes, namely Camenisch–Lysyanskaya [CL04] and Pointcheval–Sanders [PS16, PS18] signatures, are not a priori suitable to a multi-signer setting as needed to distribute issuance. Indeed, with those signature schemes, all signers would have to agree on a common randomness.

Related Work. Ghadafi [Gha15] and Blömer et al. [BJL16] considered group signatures with threshold opening, but did not address the more challenging task of threshold issuance. Manulis [Man05] introduced democratic group signatures in which there is no group manager. All members must participate to add a user to the group, and any member can open all group signatures, i.e., there is no anonymity within the group. Zheng et al. [ZLM+08] extended democratic group signatures to enforce that at least a threshold number of members must collaborate to open signatures. In a sense, the extension of democratic group signatures due to Zheng et al. can be viewed as group signature schemes with distributed issuance and threshold opening. However, in addition to the poor anonymity guarantees that they provide, democratic group signature schemes are not applicable to a dynamic setting in which members join the group at a high frequency since public keys must then be refreshed. Furthermore, the public keys and signatures of the constructions of Manulis and of Zheng et al. are linear in the group size, making them impractical.

In their "Coconut" paper [SABD18], Sonnino et al. proposed an anonymous credential system with threshold issuance and selective disclosure of user attributes. Though their techniques to achieve threshold issuance are similar to ours, their solution does not consider the issue of threshold opening, and therefore leaves aside the difficulty of realizing both threshold issuance and threshold opening while having short signatures. Besides, the authors do not provide a security model to analyze the security of their scheme. They only informally state properties that a scheme with threshold issuance and selective disclosure should satisfy, and then argue that their

scheme does.

Gennaro, Goldfeder and Ithurburn recently proposed [GGI19] extensions of the Boneh–Boyen–Shacham [BBS04] and Camenisch–Lysyanskaya [CL04] group-signature schemes that support threshold issuance. To achieve threshold opening, since those schemes follow the sign-and-encrypt paradigm, the authors point out that it suffices to replace the underlying encryption scheme with a threshold one as did Ghadafi [Gha15] and Blömer et al. [BJL16]. Nonetheless, this paradigm results in large signatures as explained above. Furthermore, Gennaro et al. do not provide a security model for threshold group-signature schemes. For the BBS scheme, they give a simulation argument for their threshold issuance protocol. For the CL scheme, they give a game-based proof that an adversary controlling less than a threshold number of parties cannot issue new credentials. Without a model that takes into account all the other aspects of group signatures scheme, especially threshold opening, it is difficult to grasp the exact security guarantees of their schemes.

Problem Addressed. It appears that group signatures with threshold issuance and opening are yet to be formally defined, let alone efficiently constructed.

3.2 Vehicle-to-Vehicle Communication

The automotive industry and several governments around the world have made substantial progress towards deploying Cooperative Intelligent Transport Systems (C-ITSs), with the first deployment planned to start in 2019 [SABB16]. In C-ITSs, vehicles communicate with other vehicles (V2V) and with road-side infrastructure (V2I) to improve traffic safety and efficiency.

V2V and V2I communication, together often referred to as V2X, mainly consists of two types of messages: occasional event-triggered safety messages (e.g., emergency braking maneuver) and regular position beacon messages that each vehicle typically broadcasts 1–10 times per second. The latter category, known in the European C-ITS as Cooperative Awareness Messages (CAMs) [ETS14] and in the US as Basic Safety Messages (BSMs) [oTNHTSA17], carry dynamic information about the vehicle such as its position, speed, and heading, as well as (semi-)static information about the vehicle such as its length, width, and sensor accuracy.

Unencrypted Broadcast Messages. The CAMs are primarily broadcast in plaintext over an unprotected short-range radio channel (ETSI ITS-G5), and these messages are therefore easy to intercept and potentially leak sensitive information about people's whereabouts, travel itineraries, and driving habits. The current C-ITS proposals [ETS14, oTNHTSA17] have

therefore raised serious privacy concerns among civil right unions, scientists [RLS+17], and data protection authorities [Art17]. Concrete threats include burglars tracking which houses are left unoccupied in a neighborhood, stalkers following their victims from an out-of-sight location, and mass surveillance of entire cities (e.g., through connected road infrastructure) at an estimated amortized cost of dollar-cents per vehicle per year [RLS+17]. Privacy regulations prohibiting misuse of CAM data are hard to enforce because rogue eavesdropping devices are easy to build and nearly impossible to detect or localize.

Due to the open nature of C-ITSs and the problems of managing encryption keys among constantly changing groups of vehicles, encryption in V2X has mostly been considered impractical and of little use.

Privacy-Preserving Authentication. Most research in V2X security and privacy has focused on the authentication aspect, ensuring that messages originate from genuine vehicles without making individual vehicles traceable throughout the system. The work of Petit et al. [PSFK15] provides an excellent survey of this field.

The practical C-ITS systems which are currently considered for deployment in Europe [ETS18] and the US [oTNHTSA17] take a similar approach to authentication by letting vehicles sign outgoing V2X messages with short-lived pseudonym certificates. Some degree of privacy is obtained by letting vehicles frequently change or rotate their certificates from a small pool of pseudonyms. In the European approach, vehicles periodically reload unrevocable pseudonyms from an online authorization authority. In the American approach, vehicles come preloaded with three years' worth of revocable pseudonym certificates [WWKH13].

However, both approaches are forced into an uncomfortable trade-off between security, privacy, and efficiency. A larger pseudonym pool size gives more privacy, but is expensive to store or download, and provides less protection against Sybil attacks in which the keys of a single compromised vehicle are used to simultaneously impersonate several vehicles. Indeed, the compromises of 100 pseudonyms per vehicle per week (EU) or 20 (US) essentially combine poor privacy guarantees (especially for frequent drivers) with high bandwidth-and-storage costs, and no meaningful Sybil resistance.

Solutions that realize anonymous authentication via group signatures (as explained in the previous section), or privacy-preserving credentials [SF17, dFGSV17, NBCN17b] provide stronger security and privacy guarantees. However, none of them fits the stringent bandwidth constraint of 300 Bytes per CAM, and they are therefore not suitable for practical deployment. Besides, as group signatures have so far only been defined with single authorities, no authenticity is guaranteed for the solutions based on group signatures as soon as a single authority is corrupt.

No Privacy without Encryption. Finally, even though so much effort has been spent on privacy-preserving authentication in V2X, the main privacy problem in CAMs is actually the transmitted data itself. Indeed, when vehicles broadcast their position, speed, heading, and acceleration up to 10 times per second, then linking messages sent by the same vehicle is trivial simply by physical limitations, regardless of how often vehicles switch pseudonyms. The largely static information included in the CAMs such as the dimensions of a vehicle and its sensor accuracy further facilitates fingerprinting vehicles and tracking them over longer periods of time.

Related Work. As previously mentioned, the bulk of the literature in security and privacy of V2X communication focuses on anonymous authentication. Verheul, Hicks and Garcia [VHG19] proposed a variant of the solution with pseudonym certificates where vehicles come pre-loaded with encrypted batches of short-lived certificates for a long period of time (i.e., years). Each batch corresponds to an epoch and can only be decrypted with a key that the vehicle receives from an authority before the beginning of the epoch; malicious vehicles can be banned simply by not providing the key. Their approach effectively reduces bandwidth to a bare minimum at the cost of local storage, which makes sense because vehicles are often poorly connected while storage is cheap, but does not try to hide the content of CAMs. However, as argued above, no real privacy can be expected without encrypting CAMs.

Other solutions have been proposed based on different concepts including message-authentication codes [CJW05], digital signatures [GM02, HCL04, BMP+14, WWKH13], identity-based cryptography [KBT06], group signatures [BBS04,Zen06], and privacy-preserving attribute-based credentials [SF17, dFGSV17,NBCN17b]. The latter two come closest to the concept of dynamic group signatures with attributes, but all have signatures too large to fit the bandwidth constraints for up-to-date security parameters.

Some work has also been concerned with encryption for V2X communication. Encrypting CAMs seems an obvious choice, but doing so in a practical and useful way is not straightforward. The necessarily open nature of C-ITSs requires that all nearby vehicles can decrypt. Embedding the same symmetric key in all units is not feasible as no revocation is possible when the key gets compromised. Possibly better solutions such as multi-sender broadcast encryption [FKK17] or public-key traitor tracing [CFN94, BN08] do not scale to a setting with hundreds of millions of vehicles. Another drawback of symmetric encryption, broadcast encryption, or traitor tracing used alone is that it is almost impossible to detect, let alone localize, a rogue wiretapping device which eavesdrops on the communication. Public-key encryption is better in this respect since the receiving device has to make itself known to the senders so that these latter know which public key to encrypt

to. However, bandwidth restrictions prohibit one-to-one connections, and the CAM length of around 300 Bytes is too short to include a separate ciphertext for each receiving vehicle.

A number of previous proposals let vehicles organize dynamically into groups according to their speed and driving direction, and establish a common key to encrypt communication [RAF11, TBNM16]. These schemes are not practical, however, since key management in highly dynamic groups of vehicles is intricate. For instance, it is not clear whether the protocol to join a new group is fast enough to give a timely warning in case of a head-on collision with a group member.

Freudiger et al. [FRF+07] proposed to use cryptographic "mix zones" where V2X-enabled vehicles briefly encrypt all communication under a key provided by a traffic-infrastructure beacon to switch pseudonyms in an unlinkable way. Zone encryption, the main contribution of Chapter 5, could be seen as an extreme extension of that concept where the entire surface is covered in mix zones, but without relying on infrastructure support.

Problem Addressed. In summary, an authentication mechanism for V2X communication that satisfies the stringent bandwidth constraints of 300 Bytes per CAM, provides strong (revocable) anonymity guarantees and does not place all trust on a single authority, has so far remained elusive. In addition to that, the eventuality of encrypting CAMs in a practical manner has been disregarded altogether on efficiency grounds, although the CAM data actually constitutes the most significant threat to users' privacy.

3.3 Hardware Security without Secure Hardware

Mobile devices are ubiquitous nowadays: smart phones and tablets have not only become prevalent in daily communication but also in numerous security-critical tasks. These devices collect and compile a large amount of confidential information to which access must be controlled. If such a device is infected by a malware, an attacker may gain full access to the compromised device and be able to control it and steal any information stored on it. In addition to that, smart phones and tablets are easily lost or stolen even though they usually only use (human-memorizable) passwords to prevent unauthorized access.

Despite their practicality, the use of mobile devices to store sensitive data incurs various security issues (since they fail to protect sensitive information and passwords in particular): vulnerability to dictionary attacks (since passwords are weak-entropy secrets), re-use of passwords across multiple service, frequent leakage of password databases, and many more. A possible solution is to use a physical device that provides extra security. A hardware security module is a tamper-resistant device that strengthens en-

cryption practices and is used in addition to or in place of passwords. These modules often come in the form of plug-in cards or external devices from which secret information cannot be easily leaked to anyone who gets hold of it. However, such modules are often costly, inconvenient to use and may even be subverted, i.e., corrupt from their very production. It may thus be of interest to investigate how such tamper-resistant devices can be disposed of, while only relying on a smart phone (combined with a public computer) on which no hardware-security assumption is made. A device (such as a smart phone) is thus used as a token, meaning in particular that these two terms are used interchangeably in what follows.

Public-Key Encryption with Password-protected Assisted Decryption. Following the seminal work of Diffie and Hellman [DH76] and of Rivest, Shamir and Adleman [RSA78], Golwasser and Micali [GM84] proposed the first security definitions of public-key encryption, and an abundance of definitions exist nowadays. These definitions crucially rely on the fact that the attacker does not have access to B's secret key. Therefore, in practice, it amounts to assuming that B has some form of secure storage, e.g., a hardware-security module.

Chapter 6 considers the scenario in which B is not required to securely store her secret key and tackles the problem of achieving security guarantees equivalent to those of hardware security modules without making any assumption on the device. Therefore, no assumption is made on a user token, i.e., it is not presumed to be tamper-proof or malware-free. The token just acts as a virtual smart-card.

To mitigate this lack of security from the token, a server which assists the user with the decryption is introduced. In a real-world scenario, the user could log-in from a public computer, use her phone as a token, and communicate with a remote server to decrypt ciphertexts she receives. Concretely, the secret key of the user is shared between the token and the server, and a password is shared between the user and the server. Nevertheless, the introduction of a server should not weaken the security of the original scheme. In other words, an attacker should not be able to leverage the server alone to infer any information about the plaintexts of the user. Besides, introducing a token and a server should not jeopardize the privacy of a user: They should assist her in a *blind* manner, i.e., without being able to infer any information about the ciphertexts they help decrypt. This property is later referred to as *blindness*.

Verifiability for the user is also required. That is, although the user is not required to remember the decryption key, she should still be able to verify that the token and server both correctly performed their computation with respect to the public key. Furthermore, the user is not even required to remember the public encryption key, so a public key is assumed to be

attached to the ciphertext. Since a *different* key than that attached may have been used to compute the ciphertext, the protocol should again guarantee verifiability with respect to the attached key, so as to protect user and server-held passwords.

The advantage of having the user enter her password on the computer rather than on her token is twofold. First, if the token is infected with malware, still no one can decrypt her ciphertexts without knowledge of her password (assuming an appropriate throttling mechanism to prevent online dictionary attacks). Secondly, if the computer from which the user starts a decryption query is corrupt (e.g., has a keylogger on it), her password is leaked, but as long as her token is not corrupt, no one can decrypt her ciphertexts. In other words, separating the user algorithm and the token guarantees security if either the user password is not leaked or if the token is not corrupt.

Three Levels of Authentication. In the envisioned scenario, the user logs in from a public (untrusted) computer, and interacts with an (insecure) token (such as her smart phone or tablet) and a remote server. She shares a password with the server, and her decryption key is shared between the token and the server. Recall that no assumption is made on the hardware security of the token, but that the password has to be typed from another machine, such as a computer. The authentication model between these three parties is now discussed.

User–Token Authentication. Since the token would in practice be a smart phone or a tablet, a PIN is usually required to access them (in very few attempts), which is similar to having the user authenticates to the token. The user can then initiate the decryption protocol between the three entities, by logging-in to the server using her password. No secure channel between the user's machine and the token is assumed.

Similarly, no higher-level mechanism is assumed for the token to authenticate itself to the user. The user is supposed to recognize her token and have it at proximity. Nonetheless, if the scheme is verifiable and secure, the user is assured that even if she is led to interact with a malicious token, the result of the decryption protocol must be correct if it terminates, and that the protocol leaks no information about her password.

Token–Server Authentication. The token authenticates itself to the server using its share of the user decryption key. The reason is that if the token does not prove to the server that it indeed belongs to the user requesting the decryption of a ciphertext, an attacker could take advantage of the throttling mechanism of the server to block an honest-user account. Indeed, the attacker could make decryption requests on the user's behalf with an arbitrary token and make several password attempts until the server blocks her account. Even more damaging, the malicious token could also make

queries to the server to infer information about the share of the server. Likewise, the server must itself authenticate to the token using its share of the user decryption key, as otherwise a malicious server without the user password could exploit the token to get information about the token share of the secret key.
Server–User Authentication. In the present model, the user must only remember her password. She thus needs to be able to recover the address of a server which shares a decryption key with her token. This issue can be dealt with in various ways, but an obvious solution would be to retrieve the address of the server from the token. However, a corrupt token may lure her into executing the decryption protocol with a malicious server via a phishing attack. The user may also simply mistype the address of the server. The server on which the user lands may even be certified within a public-key infrastructure, but not be one with which she shares a password, or one which shares the secret key with the token. If this server is malicious, it could try to infer some information about the user's password or the secret-key share held by the token.

Therefore, since the secret values (passwords or shares of the secret key) must remain protected, the user and the server need to authenticate themselves to each other, and they do so via the help of their common password. Note that this authentication does not leak any information on the password. In particular, it protects the user's password from an adversarial server which does not know her password, and it also preserves the confidentiality of her messages against an attacker which does not know her password and tries to exploit her server. These requirements are captured by the upcoming security definitions.

A scheme satisfying all the aforementioned properties is called an Encryption scheme with Password-protected Decryption or EPAD scheme.

Related Work. For cryptographic authentication, Camenisch, Lehmann, Neven and Samelin introduced [CLNS16] *password-authenticated server-aided signatures* (Pass2Sign) to obtain the best of both worlds. Their approach aims to offer comparable security guarantees to hardware security modules even when using a potentially corrupt device. To do so, they introduce a server which shares the secret key with the device. To compute a signature, the user starts a protocol with the server from her device, and she uses a password to authenticate herself to the server. The secret key is never reconstructed during the protocol, so that if the device is later corrupt (assuming passwords previously entered have been erased), only a share of the secret key is lost and the attacker is still unable to compute valid signatures. The device thus simply acts as a virtual smart card.

However, a crucial assumption in the scenario considered by Camenisch, Lehmann, Neven and Samelin is that 1) the device is not corrupt at the mo-

ment the user enters her password, as an attacker would otherwise be able to impersonate the user and sign on her behalf and 2) the device securely erases the former passwords, since a corruption would directly leak the password that were previously typed. In this sense, they do not completely achieve their ambitious goal of making no assumptions on the security of the device.

Given the above description, the considered scenario (disregarding the blindness property) is a hybrid of that of Pass2Sign (in which the user enters her password from the device) and of *password-protected secret sharing* [BJSL11, CLLN14, ACNP16, JKKX17]. An important difference w.r.t. the latter is that the decryption key is never reconstructed, thereby protecting the user in case of corruption of her machine (i.e., the public computer from which she initiates the protocol). This property also allows the user to prevent further use of her device should it be stolen. She can do so by asking the server to block her account and thereby also hinder online dictionary attacks.

Moreover, interaction gives the server the opportunity to enforce a throttling mechanism and refuse to decrypt if it detects suspicious behavior (e.g., several failed password attempts). From a commercial perspective, interaction also allows the server to run a paid service and charge the user for decryption requests.

Contrarily to the model of Pass2Sign, the communication between the token and the server is here *not* assumed to be a priori authenticated (via TLS for instance). The security model ensures that interacting with a malicious party leaks no information about either the key shares or the passwords held by the user and the server. In the construction which follows, the token and the server leverage the fact that they share the user's secret key to authenticate themselves to each other.

Finally, separating the user's machine (on which she types the password) and the token provides a *strictly stronger security* than that of password-authenticated server-aided signatures (for which a malware-infected token leaks both the user's password and secret-key share); while being no less convenient than technologies leveraging two-factor authentication mechanisms.

3.4 Results

The contributions of the first part of this thesis are manifold.

3.4.1 Group Signatures

Chapter 4 presents the first provably-secure group signatures that no longer require trust in single authorities for issuance and opening, but instead distribute their roles over several parties. These group signatures, when augmented with attributes as done in Chapter 5, constitute one of the main building blocks of the construction therein.

Security Model for Threshold Group Signatures. Chapter 4 starts by formalizing threshold dynamic group signatures and define their security in the presence of multiple issuers and openers. The model features a number n_I of issuers and a number n_O of openers separate from the issuers. Any quorum of $t_I + 1$ issuers can add users to the group, whereas no collusion of t_I issuers can generate a valid credential. Besides, any $t_O + 1$ openers can recover the identity of a signer, but anonymity is guaranteed against up to t_O corrupt openers.

Short Threshold Group Signatures. Chapter 4 then continues with an efficient, provably secure instantiation based on Pointcheval–Sanders (PS) signatures [PS18]. It shares ideas with the "GetShorty" approach, the one leading to the shortest single-authority group signatures, and extends them to a threshold setting. It shows that the roles of issuer and opener can be separate even with this approach, as long as the openers can still access the opening information generated during issuance. Nevertheless, the openers do not partake in the issuance protocol and are the only parties who should be able to retrieve it. The challenge thus consists in making sure, during issuance, that the opening information is correct, and that the openers (and only them) can later retrieve it.

The resulting group signatures are short, fast to compute and verify, thereby making them suitable for practical privacy-preserving applications. The construction is proved secure in the random-oracle model under a so-called non-interactive q-type of assumption.

Simpler Distributed Group Signatures and Multi-Signatures. Chapter 4 also presents a variant of its main group-signature scheme that requires the participation of all n_I issuers to add users to the group, but still caters for threshold opening. It has the benefit of permitting the corruption of all issuers but one. It is based on a multi-signature variant of the PS scheme and proved secure in the plain public-key model (i.e., the signers do not have to prove knowledge of their secret keys) under the same q-assumption. This PS multi-signature scheme constitutes a contribution of independent interest. Multi-signatures compress the signatures of multiple signers on the same message into a single compact signature and are for instance used to optimize consensus protocols in distributed ledgers and blockchains. Unlike existing multi-signature schemes [MOR01, Bol03, BDN18, MPSW19], PS multi-signatures allow for efficient zero-knowledge proofs of signatures, making them an interesting tool for the design of privacy-enhancing cryptographic protocols.

3.4.2 Vehicle-to-Vehicle Communication

Chapter 5 tackles the problem of privacy in V2X communication by addressing, for the first time, the problem of authenticity and confidentiality in combination. As a result, it presents a protocol to encrypt and authenticate CAMs that is suitable for the stringent 300 Bytes bandwidth requirement of C-ITSs, and arguably offers better privacy than the existing proposal. Namely, it introduces an authentication scheme based on group signatures alike those presented in Chapter 4 that combines unlimited privacy with negligible bandwidth-and-storage costs; and based on this authentication mechanism, gives a practical way to encrypt CAMs to hide their content from eavesdroppers. Interestingly, this combination not only improves the security and privacy of C-ITSs, but the careful composition of symmetric and asymmetric building blocks even leads to better efficiency.

Zone Encryption. The novel concept of *zone encryption* is introduced as a practical means to transmit V2X data authentically and confidentially. The core idea of zone encryption is to rely on *symmetric authenticated* encryption for protecting V2X communication, using temporary keys that are exchanged among vehicles in the same vicinity. Only the key-exchange messages are signed with short-lived certificates, which results in an important efficiency gain compared to the existing solutions which sign *every* outgoing CAM. For short-lived certificates, the group signatures of Chapter 4, augmented with attributes tailored to the need of V2X communication, are used. Relying on group signatures instead of a pre-fetched batch of pseudonym certificates overcomes the trade-off between privacy and security of the existing approaches: vehicles need only store a single credential, but have full privacy that is equivalent to an unlimited pseudonym pool size. The certified attribute is a short time epoch during which the credential is valid. These credentials allow to create any number of unlinkable signatures to authenticate zone-key exchanges. The desired security and privacy properties are formally defined, and the chapter proposes an efficient, provably secure protocol that achieves them. Each CAM is 240 Bytes long in the instantiation, which is compliant with the stringent bandwidth requirements of C-ITSs.

Zone encryption certainly does not solve all privacy issues concerning V2X communication, but it does raise the bar of eavesdropping on CAM data to a level that is unaffordable for occasional criminals and notably more expensive for mass surveillance. Private criminals will rely on a black market to obtain eavesdropping devices; and to offset the costs of compromising hardware-protected vehicle keys, the market will most likely share the same long-term credentials across many rogue devices. Once the police confiscates rogue devices, it can run in a controlled setting to trace and revoke their underlying long-term credentials, thereby disabling all devices in the field that use the same credentials. This will in turn increase the

costs of producing such devices until they become too expensive for private criminals.

In the current plaintext-broadcasting C-ITS proposals, mass surveillance through sensitive antennas or traffic infrastructure is fairly cheap to deploy and hard to detect. Being inherently a semi-open system that enables all vehicles to communicate, mass surveillance of C-ITS data through a network of hidden or moving transmitters will always remain possible. Zone encryption cannot prevent this, but increases the cost of operating such a network.

Namely, central surveillance antennas have to send strong signals to engage in key exchanges in the observed zones, making it harder for them to covertly operate. A distributed network of less powerful relay stations, e.g., in driving vehicles or road infrastructure, is considerably more expensive to set up. Besides, most road infrastructure (e.g., traffic signs) has no need for privacy, nor to receive CAM data. Such infrastructure can thus be given a different type of credentials that enables it to broadcast unencrypted authenticated information, but not obtain zone keys. Any piece of infrastructure that is nevertheless caught engaging in zone-key exchanges would be considered suspect, and requires explanation from road operators.

3.4.3 Encryption with Password-Protected Assisted Decryption

Chapter 6 formalizes encryption with password-protected assisted decryption.

Security Model. First, the security properties expected from EPAD schemes are formalized. As explained above, the security guarantees of the scheme should tolerate a form of *malleability* so as to also preserve user privacy. Indeed, an EPAD scheme enables a user to query decryptions without revealing information about the ciphertext being decrypted. This property was defined and formalized by Green [Gre11] (for classical public-key encryption schemes) as blind decryption. To ensure the privacy of the user (particularly towards a malicious server), a similar *blindness* property formalizes the idea that neither the token nor the server should be able to infer any information about the ciphertexts the user wants to decrypt. It is a strong requirement as the token and the server are adversarial in the formal definition, and can therefore together reconstruct the entire secret key and the password.

Due to this mild form of malleability, which allows users to re-randomize their ciphertexts, the notion of confidentiality considered for EPAD schemes is similar to *Replayable Chosen-Ciphertext Attacks security* (RCCA) defined by Canetti, Krawczyk and Nielsen for classical public-key encryption [CKN03]. The notion is termed *Password-protected Indistinguishabil-*

ity under Replayable Chosen-Ciphertext Attacks (P-IND-RCCA) and takes into account the fact that decryption requests should be *protected by user passwords*. More precisely, it ensures that unless an adversary both knows the user's password (by corrupting the user's machine or the server) *and* has corrupted her token, it cannot infer any information about the user's plaintexts in reasonable time. It captures both indistinguishability under replayable chosen-ciphertext attacks and password authentication. The formal model for P-IND-RCCA security is inspired by the Bellare–Rogaway–Pointcheval (BPR) model for password-based authenticated-key-exchange protocols [BPR00]. It covers the cases of concurrent protocol executions, with potentially many users, tokens and servers. The third security notion in our model is *verifiability* which guarantees that the user accepts the result of the decryption protocol only if the token and the server performed their computations correctly.

Technical Challenges. The fact user passwords are not entered into the token may at first seem a simple solution to the problem in Pass2Sign, and thereby to achieve the level of security provided by secure hardware. However, it raises several challenges. Indeed, (1) the token – without knowing the password – must be able to ensure that the user and server have correctly authenticated themselves to each other, and that it shares its secret key with the server, *before* performing computations. Otherwise, an attacker which does not know the password or the other key share, could exploit the token and gain information about its share. (2) The parties must also make sure throughout the entire protocol that they are communicating with the right parties and are not victims of man-in-the-middle attacks (recall that the communication is not assumed to be a priori authenticated). (3) Moreover, although the token terminates decryption, the server must be able to hide the plaintext from it, even though it only shares a low-entropy secret with the user. (4) However, the token must be convinced that the server correctly performed its computation, even without knowing the only piece of information (i.e., the password) shared between the user and the server. (5) Lastly, the protocol should guarantee user privacy even though the token and the server together know all secrets.

Efficient and Secure Construction. Chapter 6 overcomes these challenges and proposes a concrete pairing-based EPAD scheme. It uses as building block the recent publicly verifiable RCCA-secure encryption scheme of Faonio, Fiore, Herranz and Ràfols [FFHR19], though similar techniques can be applied to other such schemes (e.g., Libert, Peters and Qian's [LPQ17]). The construction is proven secure in the standard model and heavily relies on malleable zero-knowledge proofs.

Chapter 4

Short Threshold Dynamic Group Signatures

Group signatures allow members of a user group to anonymously sign on behalf of the group after being added by an issuer. Only a party, an opener, can reveal the identity of a signer. This chapter defines group signatures with several authorities for issuance and opening, of whom a threshold number must collaborate to perform these critical tasks.

Section 4.2 gives a formal security model of threshold group signatures. Section 4.3 proposes a provably secure construction of threshold group-signature scheme which assumes a publicly available write-only ledger. Section 4.4 disposes of this assumption, but at the cost of combining the roles of issuers and openers. Finally, Section 4.6 proposes another variant of group signatures in which all issuers must participate to add a user, and not just a threshold number of them, even though opening still only requires a threshold number of authorities. The benefit of this variant is that it guarantees unforgeability so long as at most all but one issuers are corrupt, and not simply half of them as does the first variant. It builds on a novel multi-signature scheme presented in Section 4.5. This scheme is of independent interested as it allows to efficiently prove knowledge of multi-signatures, an important feature in many a privacy-preserving protocol.

Contents

4.1 Preliminaries

This section introduces preliminary material to this chapter. It comprises hardness assumptions and building blocks on which the constructions in this chapter rely. It also features a general forking lemma due to Bagherzandi et al. [BCJ08], which is a corner of the security proofs in this chapter.

4.1.1 Hardness Assumptions

Strong Diffie–Hellman Assumption. Pointcheval and Sanders introduced [PS18] a new non-interactive q-type of assumption that they called the Modified q-Strong Diffie–Hellman (q-MSDH-1) assumption. They proved that it holds in the generic bilinear group model.

Definition 4.1.2 (q-MSDH-1 Assumption)**.** *Let* G *be a type-3 pairing-group generator. The q-MSDH-1 assumption over* G *is that for all PPT adversary* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, for all* $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}\left(1^\lambda\right)$*, given* Γ*,* $g \in_R \mathbb{G}^*$*,* $\tilde{g} \in_R \tilde{\mathbb{G}}^*$*, and two tuples* $\left(g^{x^\ell}, \tilde{g}^{x^\ell}\right)_{\ell=0}^{q} \in (\mathbb{G} \times \tilde{\mathbb{G}})^{q+1}$ *and* $(g^a, \tilde{g}^a, \tilde{g}^{ax}) \in \mathbb{G} \times \tilde{\mathbb{G}}^2$ *for* $x, a \in_R \mathbb{Z}_p^*$*, the probability that* $\mathcal{A}$ *computes a tuple* $\Big(w, P, h^{1/x+w}, h^{a/P(x)}\Big)$*, with* $h \in \mathbb{G}^*$*,* P *a polynomial in* $\mathbb{Z}_p[X]$ *of degree at most* q *and* $w \in \mathbb{Z}_p$ *such that the polynomials* $X + w$ *and* P *are coprime, is negligible.*

Symmetric Discrete-Logarithm Assumption. The following assumption a generalization of the standard discrete-logarithm assumption to bilinear-group structures

Definition 4.1.3 (Symmetric Discrete-Logarithm Assumption)**.** *Let* G *be a type-3 pairing-group group generator. The Symmetric Discrete-Logarithm (SDL) assumption [BCN+10] over* G *is that for all PPT adversary* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*,* $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}\left(1^\lambda\right)$*,* $g \in_R G^*$*,* $\tilde{g} \in_R \tilde{\mathbb{G}}^*$*,* $x \in_R \mathbb{Z}_p^*$*, given* $(\Gamma, g, \tilde{g}, g^x, \tilde{g}^x)$ *as an input, the probability that* $\mathcal{A}$ *returns* x *is negligible.*

Note that given $(\Gamma, g, \tilde{g}, h, \tilde{h})$, one can always verify that it is a valid SDL tuple by testing the equality $e(g, \tilde{h}) = e(\tilde{g}, h)$. Notice also that the SDL assumption is implied by the q-MSDH-1 assumption (Definition 4.1.2).

Knowledge-of-Exponent Assumption. Fuchsbauer and Orru introduced an analog of the Diffie–Hellman Knowledge-of-Exponent assumption [BFS16] in an asymmetric setting called the Asymmetric Diffie–Hellman Knowledge-of-Exponent assumption [FO18]. It is primarily used in the context of subversion-resistant non-interactive witness-indistinguishable proofs.

Definition 4.1.4 (Asymmetric Diffie–Hellman Knowledge-of-Exponent Assumption)**.** *The (first-group) Asymmetric Diffie–Hellman Knowledge-of-Exponent (ADH-KE) game, parametrized by $\lambda \in \mathbb{N}$, for a type-3 pairing-group generator G, an adversary $\mathcal{A}$ and an extractor Ext is defined as follows:*

- $\Gamma := (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e) \leftarrow \mathsf{G}(1^\lambda)$; $g \in_R \mathbb{G}^*$
- $(X, Y, Z) \leftarrow \mathcal{A}(\Gamma, g)$
- $s \leftarrow \mathsf{Ext}\,(\Gamma, g, X, Y, Z)$
- *return* $b \leftarrow \left(g^s \neq X \wedge g^s \neq Y \wedge Z = Y^{\mathrm{dlog}_g(X)}\right)$.

In other words, $\mathcal{A}$ wins the game if (g, X, Y, Z) is a Diffie–Hellman tuple, but algorithm Ext can extract neither $\mathrm{dlog}_g(X)$ nor $\mathrm{dlog}_g(Y)$.

The ADH-KE assumption over a type-3 pairing-group generator G is that there exists an efficient algorithm Ext such that for all efficient adversary $\mathcal{A}$ for the ADH-KE game, $\Pr[b = 1]$ is negligible in λ.

4.1.5 Pointcheval–Sanders Signature Scheme

Pointcheval and Sanders [PS18] proposed an efficient signature scheme that allows to sign message blocks $(m_1, \ldots, m_k)$ at once, and also to efficiently prove knowledge of signatures in zero-knowledge. They proved this scheme to be existential unforgeable under the q-MDSH-1 assumption [PS18] stated in Definition 4.1.2.

Given a type-3 pairing-group generator G and security parameter $\lambda \in \mathbb{N}$, the PS-signature scheme in a pairing-group $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}(1^\lambda)$ consists of the following algorithms.

$\mathsf{PS.Setup}(1^\lambda, \Gamma, k) \to pp$: generate $\tilde{g} \in \tilde{\mathbb{G}}^*$. Return $pp \leftarrow (\Gamma, \tilde{g}, k)$.

$\mathsf{PS.KG}(pp) \to (vk, sk)$: generate $x, y_1, \ldots, y_{k+1} \in_R \mathbb{Z}_p$, compute $\tilde{X} \leftarrow \tilde{g}^x$, $\tilde{Y}_j \leftarrow \tilde{g}^{y_j}$ for $j \in [k+1]$, and set and return $vk \leftarrow (\tilde{X}, \tilde{Y}_1, \ldots, \tilde{Y}_{k+1})$ and $sk \leftarrow (x, y_1, \ldots, y_{k+1})$.

$\mathsf{PS.Sign}\,(sk, (m_1, \ldots, m_k)) \to \sigma$: choose $h \in_R \mathbb{G}^*$, $m' \in_R \mathbb{Z}_p$ and return $\left(m', h, h^{x+\sum_{j=1}^{k} y_j m_j + y_{k+1} m'}\right)$.

$\mathsf{PS.Vf}\,(vk, (m_1, \ldots, m_k), \sigma) \to b$: parse σ as (m', σ_1, σ_2), verify that $e\,(\sigma_1, \tilde{X} \prod_{j=1}^{k} \tilde{Y}_j^{m_j} \tilde{Y}_{k+1}^{m'}) = e(\sigma_2, \tilde{g})$ and that $\sigma_1 \neq 1_{\mathbb{G}}$. If so, return 1, otherwise return 0.

Pointcheval and Sanders proved this scheme to be existential unforgeable under the q-MDSH-1 assumption [PS18] stated in Definition 4.1.2.

In the random oracle model, the scheme remains secure under the same assumption if m' is computed as $\mathcal{H}(m_1, \ldots, m_k)$ [PS18, Corollary 12]. Noticing that the verification algorithm does *not* verify any property on m', and in particular that $m' = \mathcal{H}(m_1, \ldots, m_k)$, the scheme still allows for efficient zero-knowledge proofs of knowledge if m' is computed as such.

4.1.6 Multi-Signatures with Key Aggregation

A multi-signature scheme [MOR01] allows a number $n \geq 1$ of signers to jointly compute a short signature on the same message. Given the public verification keys of the n signers, one can verify that all n signers signed the message.

Syntax.

Formally, for an integer $n > 0$, an n-signature scheme MS (with key aggregation) consists of the following algorithms:

$\mathsf{MS.Setup}(1^\lambda, n, aux) \to pp$: returns public parameters on the input of a security parameter, a number of signers and an auxiliary input.

$\mathsf{MS.KG}(pp) \to (vk, sk)$: returns a pair of verification–signing keys on the input of public parameters.

$\mathsf{MS.KAggreg}(vk_1, \ldots, vk_n) \to avk$: aggregates the verification keys of n signers and returns a short aggregated key avk that can be used to verify aggregated signatures.

$\mathsf{MS.Sign}(sk, m) \to \sigma$: a signing algorithm which takes as an input a signing key sk and a message m. It returns a signature σ.

$\mathsf{MS.SAggreg}((vk_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n) \to \sigma$: an algorithm that aggregates the signatures on a single message m computed by n signers and returns a short aggregated signature σ.

$\mathsf{MS.Vf}(avk, m, \sigma) \to b \in \{0, 1\}$: on the input of an aggregated verification key, a message and an aggregated signature, returns a bit indicating whether the signature is valid w.r.t. the aggregated verification key.

An alternative definition in which the verification keys are aggregated during signature verification could be considered but would be less efficient in a setting in which the set of signers is fixed (or at least rarely changes). Indeed, if the set of signers is fixed, their keys can be aggregated once for all and the resulting aggregated key can be used every time a signature is to be verified.

Security Model.

The security of an n-signature scheme [BN06] MS is defined via a security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. At the beginning of the game, $\mathcal{C}$ generates parameters $pp \leftarrow \mathsf{MS.Setup}(1^\lambda, n, aux)$ and sends them to $\mathcal{A}$. Adversary $\mathcal{A}$ then sends a target honest-signer identity i^* to $\mathcal{C}$. Challenger $\mathcal{C}$ proceeds by generating keys $(vk_{i^*}, sk_{i^*}) \leftarrow \mathsf{MS.KG}(pp)$ and sending vk_{i^*} to $\mathcal{A}$. Adversary $\mathcal{A}$ is now allowed to issue signing queries on arbitrary messages m. To answer such a query, $\mathcal{C}$ computes and sends $\sigma_{i^*} \leftarrow \mathsf{MS.Sign}\,(sk_{i^*}, m)$ to $\mathcal{A}$. After the query phase, $\mathcal{A}$ outputs a set of verification keys K such that $vk_{i^*} \in K$, a message m for which no signing query has been made and a signature σ. Adversary $\mathcal{A}$ wins the game if and only if $\mathsf{MS.Vf}(\mathsf{MS.KAggreg}(K), m, \sigma) = 1$. A multi-signature scheme is existentially unforgeable under chosen-message attacks (or EUF-CMA secure) if no efficient adversary can win this security game with a non-negligible probability.

A weak unforgeability can also be defined via a variant of the previous game in which $\mathcal{A}$ outputs, along with index i^* (so before getting key vk_{i^*}), a list of messages that $\mathcal{C}$ signs with key sk_{i^*} and sends back the results with vk_{i^*}. Adversary $\mathcal{A}$ cannot make signing queries afterwards. Scheme MS is weakly existentially unforgeable (or EUF-wCMA secure) if no efficient adversary $\mathcal{A}$ has a non-negligible probability of winning this variant of the security game.

4.1.7 Generalized Forking Lemma

The original forking lemma was formulated by Pointcheval and Stern [PS00] to analyze the security of Schnorr signatures [Sch91]. The lemma rewinds a forger $\mathcal{A}$ against the Schnorr signature scheme in the random-oracle model (ROM) to a "crucial" random-oracle query (typically, the query involved in a forgery) and runs $\mathcal{A}$ again from the crucial query with fresh random-oracle responses. The lemma essentially translates that if $\mathcal{A}$ has non-negligible success probability in a single run, then the forking algorithm will generate two successful executions with non-negligible probability.

Bellare and Neven [BN06] generalized the forking lemma to apply to any algorithm $\mathcal{A}$ in the random-oracle model using a single rewinding, while Bagherzandi, Cheon, and Jarecki [BCJ08] generalized the lemma even fur-

ther to multiple subsequent rewindings on multiple crucial queries. This section recalls a slight modification of the latter version.

Let $\mathcal{A}$ be an algorithm that is given an input in as well as randomness $f = (\rho, h_1, \ldots, h_{q_\mathrm{H}})$, where ρ is $\mathcal{A}$'s random tape and $h_1, \ldots, h_{q_\mathrm{H}}$ are random values from $\mathbb{Z}_q$. Let Ω be the space of all such vectors f and let $f|_i = (\rho, h_1, \ldots, h_{i-1})$. Consider an execution of $\mathcal{A}$ on input in and randomness f with access to oracle $\mathcal{O}$, denoted $\mathcal{A}^{\mathcal{O}}(in, f)$, as *successful* if it outputs a tuple $(J, \{out_j\}_{j\in J}, aux)$, where J is a multi-set that is a non-empty subset of $\{1, \ldots, q_\mathrm{H}\}$, $\{out_j\}_{j\in J}$ is a multi-set of side outputs, and aux is an additional set of auxiliary outputs. $\mathcal{A}$ is said to have failed if it outputs $J = \emptyset$. Let ε be the probability that $\mathcal{A}(in, f)$ is successful for fresh randomness $f \leftarrow_\$ \Omega$ and for an input $in \leftarrow_\$ \mathsf{IG}$ generated by an input generator IG.

For a given input in, the generalized forking algorithm $\mathcal{GF}_\mathcal{A}$ is defined as follows:

$\mathcal{GF}_\mathcal{A}(in)$:
  $f = (\rho, h_1, \ldots, h_{q_\mathrm{H}}) \leftarrow_\$ \Omega$
  $(J, \{out_j\}_{j\in J}, aux) \leftarrow \mathcal{A}^{\mathcal{O}}(in, f)$
  If $J = \emptyset$ then return fail
  $Aux \leftarrow aux$
  Let $J = \{j_1, \ldots, j_n\}$ such that $j_1 \leq \ldots \leq j_n$
  For $i = 1, \ldots, n$ do
    $succ_i \leftarrow 0$; $k_i \leftarrow 0$; $k_{\max} \leftarrow 8nq_\mathrm{H}/\varepsilon \cdot \ln(8n/\varepsilon)$
    Repeat until $succ_i = 1$ or $k_i > k_{\max}$
      $f'' \leftarrow_\$ \Omega$ such that $f''|_{j_i} = f|_{j_i}$
      Let $f'' = (\rho, h_1, \ldots, h_{j_i-1}, h''_{j_i}, \ldots, h''_{q_\mathrm{H}})$
      $(J'', \{out''_j\}_{j\in J''}, aux) \leftarrow \mathcal{A}^{\mathcal{O}}(in, f'')$
      $Aux \leftarrow Aux \cup aux$
      If $h''_{j_i} \neq h_{j_i}$ and $J'' \neq \emptyset$ and $j_i \in J''$ then
        $out'_{j_i} \leftarrow out''_{j_i}$; $succ_i \leftarrow 1$
  If $succ_i = 1$ for all $i = 1, \ldots, n$
  Then return $(J, \{out_j\}_{j\in J}, \{out'_j\}_{j\in J}, Aux)$
  Else return fail

$\mathcal{GF}_\mathcal{A}$ is considered successful if it does not return fail. The main difference to Bagherzandi et al.'s forking lemma [BCJ08] is $\mathcal{A}$'s access to the oracle $\mathcal{O}$ and the additional auxiliary output aux that gets accumulated in Aux over all runs of $\mathcal{A}$, including failed runs. If the oracle $\mathcal{O}$ is deterministic, meaning that it always answers the same query with the same response, it is easy to see that these extensions do not impact the bounds of their forking lemma, so the following statement still holds.

Lemma 4.1.8 (Generalized Forking Lemma [BCJ08])**.** *Let* IG *be a randomized algorithm and* $\mathcal{A}$ *be a randomized algorithm running in time* τ *with access to a deterministic oracle* $\mathcal{O}$ *that succeeds with probability* ε*. If*

$q > 8nq_{\mathrm{H}}/\varepsilon$, then $\mathcal{GF}_{\mathcal{A}}(in)$ runs in time at most $\tau \cdot 8n^2 q_{\mathrm{H}}/\varepsilon \cdot \ln(8n/\varepsilon)$ and succeeds with probability at least $\varepsilon/8$, where the probability is over the choice of $in \leftarrow_{\$} \mathsf{IG}$ and over the coins of $\mathcal{GF}_{\mathcal{A}}$.

4.2 Threshold Dynamic Group Signatures

This section formally defines threshold dynamic group signatures. Classical dynamic group signatures [BSZ05] allow users to join a group of signers at any time by interacting with an issuer, and then sign anonymously on behalf of the group. A verifier is then assured that a valid signature stems from a group member but cannot infer any information about her identity. Only a dedicated authority, the opener, can recover the identity of a member who computed a valid signature.

In terms of security, group signatures should guarantee anonymity even in the presence of a corrupt issuer and unforgeability (also known as traceability) even when the opener is corrupt. Still, trust in each entity is necessary for the respective properties. It is even worse for schemes in which the roles of issuer and opener are assumed by the same party [BS04, CG05] who then has to be trusted for both anonymity and unforgeability. This holds in particular for the GetShorty-type of signatures [BCN+10, PS16] which yield the most efficient instantiations to date.

The capabilities of the issuer and the opener are here distributed over several entities to prevent them from becoming single points of failure. To reflect the difference between issuer and opener, two thresholds are introduced: schemes are defined with $n_I > 1$ issuers of whom $t_I + 1 \leq n_I$ are required to add users to the group. Similarly, there are $n_O > 1$ openers and at least $t_O + 1$ openers must collaborate to open a signature and reveal the signer's identity.

First comes the syntax of dynamic group signatures with threshold issuance and threshold opening. The security requirements that can be expected from such schemes are then formalized.

4.2.1 Syntax

Formally, a (t_I, t_O)-out-of-(n_I, n_O) DGS scheme, or $\binom{n_I, n_O}{t_I, t_O}$-DGS scheme, with identity space ID (assumed not to contain $\perp$) consists of the following algorithms:

$\mathsf{GSetup}(1^\lambda, n_I, n_O, t_I, t_O) \to pp$: on the input of a security parameter, a number n_I of issuers, a number n_O of openers and two integer threshold values, generates public parameters which are assumed to be an implicit input to all the other algorithms. Those parameters are also assumed to contain n_I, n_O, t_I and t_O. Moreover, each issuer is assigned

a public, fixed index $i \in [\![n_I]\!]$. Similarly, each opener is assigned a public index $i \in [\![n_O]\!]$.

$\langle \{\mathsf{IKG}(pp, i)\}_{i=1}^{n_I} \rangle \to \langle \{(ipk, isk_i, st_i)\}_{i=1}^{n_I} \rangle$: a key-generation protocol between all n_I issuers. At the end of the protocol, the issuers agree on a public key, and each of them holds a secret key and a state(initially empty). The state later contains the identities of the users that are added to the group. The issuer public key is used to add users to the group, and to compute and verify group signatures.

$\langle \{\mathsf{OKG}(pp, i)\}_{i=1}^{n_O} \rangle \to \langle \{(opk, osk_i, \mathbf{reg}_i)\}_{i=1}^{n_O} \rangle$: a key-generation protocol run by the openers. At the end of the protocol, the issuers agree on a public opening key opk, an each of them holds a secret key osk_i (assumed to contain i and opk) and a register $\mathbf{reg}_i$ initially empty. The public opening key is needed to add users to the group, and to compute and verify signatures. The secret keys and the registers are needed to open signatures.

The group public key gpk consists of ipk and opk, i.e., $gpk \leftarrow (ipk, opk)$.

$\langle \mathsf{GJoin.U}(id, I, gpk) \rightleftharpoons \{\mathsf{GJoin.I}(st_i, isk_i, id, I, gpk)\}_{i \in I} \rangle \to \langle \mathbf{gsk}[id]/\perp, st_i' \rangle$: is a protocol between a user with identity id and $t_I + 1 =: |I|$ issuers. If $id \in st_i$ for any $i \in [\![n_I]\!]$, then the ith issuer aborts the protocol. At the end of the protocol, the user algorithm returns a user group secret key $\mathbf{gsk}[id]$ (or $\perp$ if the protocol fails) and the state st_i of each issuer is updated.

$\mathsf{GSign}(gpk, \mathbf{gsk}[id], m) \to \sigma$: a probabilistic algorithm that computes a signature σ on a message m on behalf of the group.

$\mathsf{GVerf}(gpk, m, \sigma) \to b \in \{0, 1\}$: a deterministic algorithm that verifies a group signature σ on a message m w.r.t. to a group public key gpk.

$\langle \{\mathsf{GOpen}(\mathbf{reg}_i, osk_i, O, gpk, m, \sigma)\}_{i \in I} \rangle \to \langle \{id_i/\perp\}_{i \in O} \rangle$: is a protocol between $t_O + 1 =: |O|$ openers, at the end of which each algorithm returns the identity of the user who computed σ on m, or $\perp$ in case of failure. It is here assumed that the openers are given access to public and authentic information from all the successful issuance protocol executions, and that they use it to update their registers. Although this information is public, it is clear that for anonymity to hold, no information about a signer can be inferred from it without the opener secret keys.

Note that contrarily to the model of Bellare et al. [BSZ05], in the model above, each opener maintains a register separate from the state of the issuers. These registers are necessary to open signatures, in addition to the opener secret keys. Those registers should rather be thought as the registers in the model of Bichsel et al. [BCN+10].

Correctness.

Correctness captures the property that all honest issuers must agree on the same group public key. A signature σ computed on a message m with the secret key of a group-member *id* should also be accepted by the verification algorithm. Lastly, by executing the opening protocol, any set of t_O+1 openers should all return *id*. These properties should hold with overwhelming probability regardless of the order in which users are added to the group.

4.2.2 Security Model

The security requirements for threshold DGS schemes are similar to the conventional ones for dynamic group signatures with a single issuer and a single opener [BSZ05], but adapted to a threshold setting. Those requirements are *anonymity*, which guarantees that a group signature reveals no information about the member who computed it, and *traceability*, which expresses the unforgeability property of group signatures.

Essentially, no collusion of t_I issuers should be able to add users and no collusion of t_O openers should be able to open signatures. The definitions are flexible in the sense that they can require an additional fraction of openers to be honest for traceability, and of issuers to be honest for anonymity. This allows for more efficient schemes that may need slightly stronger assumptions to prove their security.

Global Variables.

In the security experiments for $\binom{n_I,n_O}{t_I,t_O}$-DGS schemes, the challenger maintains global variables which are accessible to the experiment oracles (defined hereunder). These variables are a group public *gpk*, a table of honest-user group secret keys **gsk** of size $|ID|$, and

- Q_{GJoin} a set of user identities *id* that have joined the group, whether honestly via a $\mathsf{GJoin.U}$ query or dishonestly via a $\mathsf{GJoin.I}_i$ query
- Q_{Corrupt} a set of user identities *id* either corrupt from the beginning via a $\mathsf{GJoin.I}_i$ query or of which the group secret key has been revealed
- Q_{GSign} a set of signing queries (id, m, σ) made by the adversary and the responses to those
- Q_{GOpen} a set of message–signature pairs (m, σ) for which the adversary has made an opening query.

The sets Q_{GJoin}, Q_{Corrupt}, Q_{GSign} and Q_{GOpen} are initially empty, and the entries of *gpk* and **gsk** are initially set to $\perp$.

Oracles.

This sections describes the oracles in the security experiments for $\binom{n_I,n_O}{t_I,t_O}$-DGS schemes. The oracles have access to global variables priorly defined and maintained by the challenger in each security experiment. Whenever the adversary queries a protocol-algorithm (for joining or opening) oracle, a protocol execution is triggered with all the other honest parties on the same inputs, and the adversary plays the role of the dishonest parties (dishonest users, or dishonest issuers or openers). During these executions, the adversary controls the network, i.e., it can forward, delay, drop or modify the messages sent by the various parties. However, as in prior models [BCN+10], protocols can only be executed in sequential order, i.e., the adversary cannot start a protocol execution if all the prior ones have not terminated. In particular, the adversary cannot interleave messages between protocol executions or execute multiple sessions of the same protocol in parallel.

In the following description of the oracles, if a verification fails, the oracle returns $\perp$. It is implicitly assumed that id is always in ID. Given a set Q, the statement "adds x to Q" means that $Q \leftarrow Q \cup \{x\}$. The oracles in the security experiments are then

$\mathcal{O}.\mathsf{GJoin.U}(id, I)$: checks that $I \in \binom{[\![n_I]\!]}{t_I+1}$. It adds id to Q_{GJoin}. It runs the user joining algorithm on (id, I, gpk). An execution of protocol GJoin is triggered and during it, the challenger plays the role of the (honest) user and of the honest issuers, and the adversary plays the role of the corrupt issuers. At the end of the protocol, if algorithm $\mathsf{GJoin.U}$ returns a key $\mathbf{gsk}[id]$, the challenger updates $\mathbf{gsk}$ accordingly.

$\mathcal{O}.\mathsf{GJoin.I}_i(id, I)$: (for each honest issuer i) checks that $i \in I \in \binom{[\![n_I]\!]}{t_I+1}$. It adds id to Q_{GJoin} and Q_{Corrupt}. It runs the issuer joining algorithm on $(st_i, isk_i, id, I, gpk)$. An execution of protocol GJoin is triggered and during it, the adversary plays the role of the (corrupt) user and of the corrupt issuers. The challenger plays the role of the honest issuers.

$\mathcal{O}.\mathsf{GSign}(id, m)$: checks that $id \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$. It computes $\sigma \leftarrow \mathsf{GSign}(gpk, \mathbf{gsk}[id], m)$. It adds (id, m, σ) to Q_{GSign} and returns σ.

$\mathcal{O}.\mathsf{GOpen}_i(O, m, \sigma)$: (for each honest opener i) checks that $i \in O \in \binom{[\![n_O]\!]}{t_O+1}$. It adds (m, σ) to Q_{GOpen}. It runs the opening algorithm on $(\mathbf{reg}_i, osk_i, O, gpk, m, \sigma)$. A GOpen protocol execution is triggered and the adversary plays the role of the corrupt openers, while the challenger plays that of the honest ones.

$\mathcal{O}.\mathsf{RevealU}(id)$: adds id to Q_{Corrupt} and returns $\mathbf{gsk}[id]$.

$\mathcal{O}.\mathsf{ReadReg}(i, id)$: returns $\mathbf{reg}_i[id]$.

Experiment $\mathbf{Exp}^{\text{ano}-b}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A})$:
$pp \leftarrow \mathsf{GSetup}(1^\lambda, n_I, n_O, t_I, t_O)$
$\langle st_{\mathcal{A}}, \{(ipk, isk_i, st_i)\}_{i>t_I^*}\rangle \leftarrow \langle \mathcal{A}(\mathsf{keygen}, pp), \{\mathsf{IKG}(pp,i)\}_{i>t_I^*}\rangle$
$\langle st'_{\mathcal{A}}, \{(opk, osk_i, st_i)\}_{i>t_O}\rangle \leftarrow \langle \mathcal{A}(st_{\mathcal{A}}), \{\mathsf{OKG}(pp,i)\}_{i>t_O}\rangle$
$gpk \leftarrow (ipk, opk)$
$\mathcal{O} \leftarrow \left\{\mathsf{GJoin.U}, (\mathsf{GJoin.I}_i)_{i>t_I^*}, \mathsf{GSign}, (\mathsf{GOpen}_i)_{i>t_O}, \mathsf{RevealIU}, \mathsf{WriteReg}\right\}$
$(st''_{\mathcal{A}}, id_0^*, id_1^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}(gpk,(\mathbf{reg}_i)_i,\mathbf{gsk},\cdot)}(\mathsf{choose}, st'_{\mathcal{A}})$
$\sigma^* \leftarrow \mathsf{GSign}(gpk, \mathbf{gsk}[id_b^*], m^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}(gpk,\mathbf{reg},\mathbf{gsk},\cdot)}(st''_{\mathcal{A}}, \sigma^*)$
if $id_0^*, id_1^* \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$ and $\mathbf{gsk}[id_0^*], \mathbf{gsk}[id_1^*] \neq \perp$ and $(m^*, \sigma^*) \notin Q_{\mathsf{GOpen}}$
    return b'
else
    return 0

Experiment $\mathbf{Exp}^{\text{trace}}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A})$:
$pp \leftarrow \mathsf{GSetup}(1^\lambda, n_I, n_O, t_I, t_O)$
$\langle st_{\mathcal{A}}, \{(ipk, isk_i, st_i)\}_{i>t_I}\rangle \leftarrow \langle \mathcal{A}(\mathsf{keygen}, pp), \{\mathsf{IKG}(pp,i)\}_{i>t_I}\rangle$
$\langle st'_{\mathcal{A}}, \{(opk, osk_i, st_i)\}_{i>t_O^*}\rangle \leftarrow \langle \mathcal{A}(st_{\mathcal{A}}), \{\mathsf{OKG}(pp,i)\}_{i>t_O^*}\rangle$
$gpk \leftarrow (ipk, opk)$
$\mathcal{O} \leftarrow \left\{\mathsf{GJoin.U}, (\mathsf{GJoin.I}_i)_{i>t_I}, \mathsf{GSign}, (\mathsf{GOpen}_i)_{i>t_O^*}, \mathsf{RevealIU}, \mathsf{ReadReg}\right\}$
$(O^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(gpk,(\mathbf{reg}_i)_i,\mathbf{gsk},\cdot)}(\mathsf{forge}, st'_{\mathcal{A}})$
if $O^* \notin \binom{\{t_O^*+1,\ldots,n_O\}}{t_O+1}$ then return 0
$\langle \{id_i^*\}_{i\in O^*}\rangle \leftarrow \langle \{\mathsf{GOpen}(\mathbf{reg}_i, osk_i, O^*, gpk, m^*, \sigma^*)\}_{i\in O^*}\rangle$
if $\mathsf{GVerf}(gpk, m^*, \sigma^*) = 1$ and (case 1 or case 2)
with
    case 1) opening failed i.e.,
        $\exists i \in O^*$: $id_i^* = \perp$ or $\exists i, j \in O^*$: $id_i^* \neq id_j^*$
    case 2) opening was "incorrect", i.e., setting $id^* \leftarrow id^*_{\max O^*}$,
        $id^* \notin Q_{\mathsf{GJoin}}$ or $(id^* \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$ and $(id^*, m^*, \sigma^*) \notin Q_{\mathsf{GSign}})$
    return 1
else
    return 0

Figure 4.1: Security Experiments for $\binom{n_I,n_O}{t_I,t_O}$-DGS Schemes.

$\mathcal{O}.\mathsf{WriteReg}(i, id, v)$: (for each honest opener i) sets $\mathbf{reg}_i[id] \leftarrow v$, i.e., it write value v on the register of the ith opener for user id.

Anonymity.

Anonymity ensures that a group signature reveals no information about the identity of the member who computed it as long as at most t_O openers are corrupt and the signature has not been opened. User identities are not hidden during the joining protocol, and it is in fact necessary to open signatures. In other words, anonymity is only guaranteed w.r.t. to group signatures, but it is not a restriction per se as in most practical scenarios, group signatures

are computed at a much higher frequency than members are added. Signatures are therefore much more critical from a privacy perspective.

The definition is indistinguishability-based as the adversary chooses two honest users and a message. It receives a group signature computed with the key of either of them, and it must determine the signer's identity better than by guessing. The adversary is given access to an opening oracle which it can query on all but the challenge signature, capturing a CCA-2 type of anonymity [BBS04]. Dynamic corruption of group members is allowed, i.e., a signer may initially be honest but later corrupt. However, anonymity is guaranteed only for fully honest users, i.e., there is no forward anonymity. See Figure 4.1 for the detailed experiment.

The classical notion of anonymity relies on the honesty of the opener, which is adjusted to a threshold setting and allows the adversary to corrupt up to t_O out of n_O openers. Without loss of generality, corrupt entities are always assumed to be the first ones, i.e., openers $1, \ldots, t_O$ are controlled by the adversary and $t_O + 1, \ldots, n_O$ are run by the challenger.

Concerning the issuers, the definition is flexible. Ideally, in schemes where the issuer and the opener are distinct entities, the issuer can be fully malicious in the anonymity game. In a distributed setting, it would translate in corrupting all n_I issuers. However, enforcing the corruption of all issuers may exclude some efficient schemes. The anonymity definition is thus parametrized to additionally limit the number of issuers that may be corrupt. In the experiment, the adversary corrupts t_I^* issuers, with t_I^* being a function of t_I. The strongest anonymity guarantees are achieved when $t_I^* = n_I$, in which case the adversary would output the issuer public key itself. The scheme presented in Section 4.3 realizes anonymity for $t_I^* = t_I < n_I/2$, i.e., the largest possible value for an interactive key-generation process to guarantee termination (robustness).

Lastly, it is worth noting that w.r.t. key generation, this model is stronger than that of Bellare et al. [BSZ05] in the sense that the keys of corrupt authorities are not assumed to be honestly generated.

Definition 4.2.3 (Anonymity)**.** *A* $\binom{n_I, n_O}{t_I, t_O}$*-DGS scheme* DGS *is anonymous if for every efficient adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{ano}}_{\mathsf{DGS},n_I,n_O,t_I,t_O,\mathcal{A}}(\lambda)$ *of* $\mathcal{A}$ *defined as*

$$\left|\Pr[\mathbf{Exp}^{\mathrm{ano}-0}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A}) = 1] - \Pr[\mathbf{Exp}^{\mathrm{ano}-1}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A}) = 1]\right|$$

is negligible in λ*.*

Traceability.

This notion captures the unforgeability property expected from dynamic group signatures and guarantees that only users who have joined the group can compute valid group signatures. With single authorities, the opener can

be corrupt but the issuer must be honest. Therefore, adapted to a threshold setting, up to t_I out of n_I issuers can be corrupt.

Traceability is then formalized through the opening capabilities of group signatures as described in Figure 4.1. It guarantees that for any valid signature σ on a message m, opening can neither fail (Case 1) nor reveal an "incorrect" identity (Case 2). The first case means that an opener cannot identify any signer or that the openers do not agree on the identity of the signer. The second case means that the revealed identity has either never joined the group of signers, or has joined and is honest, but never signed m. The latter is sometimes formalized through a dedicated *non-frameability* requirement, and the choice of combining both notions is discussed below.

Similarly to the case of anonymity, the number of openers that the adversary can additionally corrupt is parametrized via a bound t_O^*. The strongest traceability notion is achieved when $t_O^* = n_O$, i.e., when all openers can be corrupted. This is however not achievable when openers maintain state critical for opening, since the winning condition depends on a correct execution of protocol GOpen. In case of stateful opening, this requires the non-corrupt registers of at least $t_O + 1$ openers. Therefore, in such settings, at most $t_O^* = n_O - t_O - 1$ openers can be corrupt.

In comparison, in the traceability definition of Bellare et al. [BSZ05] for single-authority dynamic group signatures, the opener can be fully corrupt since the register needed to open signatures is rather maintained by the issuer (and the opener must have read access to it). However, this has the effect that in their anonymity definition, even if the honestly generated issuer key is given to the adversary, the challenger must maintain a register on its own to answer opening queries, and the adversary cannot read the register (though it can write on it) otherwise it would trivially win the anonymity game. It means that their model only captures a situation in which the issuer's key is compromised, but its state is not entirely. In this sense, even if their traceability definition captures a full corruption of the opener, the consequence is that their anonymity definition does not capture full corruption of the issuer.

Definition 4.2.4 (Traceability)**.** *A* $\binom{n_I,n_O}{t_I,t_O}$*-DGS scheme* DGS *is traceable if for all efficient adversary* $\mathcal{A}$*,* $\Pr\left[\mathbf{Exp}^{\text{trace}}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A}) = 1\right]$ *is negligible in* λ*.*

On Non-Frameability.

Classical definitions of group signatures with single authorities also include the notion of non-frameability. It reflects the idea that even if the issuer and opener are corrupt, they cannot falsely claim that an honest user computed a given valid signature. Since the opening algorithm in those definitions returns a long-term user public key, *in practice*, a public-key infrastruc-

ture would have to bind those keys to real-world identities; and such an infrastructure would be built with one or several certification authorities. However, if these certification authorities collude with the issuer and the opener, they would be able to frame an honest user. In other words, in real-world applications, users still need to trust some certification authority to protect them from malicious group-signature authorities even though the goal of non-frameability is precisely to avoid trust assumptions.

On the other hand, the rationale of threshold cryptography is that if there are many parties, some of them might in practice be corrupt but not all. Since group-signature schemes with several issuers and openers are now considered, the requirement is that if less than respective threshold numbers of them are corrupt, no efficient adversary can forge a signature and falsely claim that an honest user computed it. It is precisely this requirement that is captured by the last winning condition of the above definition of traceability. The scheme in Section 4.3 satisfies this property, but would not satisfy a definition in which all group-signature authorities are corrupt.

4.3 Main Construction

This section presents a threshold DGS scheme from (a variant of) PS signatures. It follows the GetShorty approach of Bichsel et al. [BCN+10] instead of the traditional sign-and-encrypt paradigm. This approach avoids the extra encryption of user identities and enables schemes with highly compact signatures despite supporting signature opening. The resulting signatures are short, and computing and verifying them only require a few exponentiations in the first group and some pairing computations (see Table 4.1).

The efficiency of the GetShorty scheme of Bichsel et al. [BCN+10] comes at the price of fully trusted authority responsible for both issuance and opening. A threshold setting allows to preserve the efficiency of the GetShorty approach, yet avoid the need for a single trusted entity. The scheme below shows that even with the GetShorty approach, the role of issuer and opener can be separated and distributed, and it enables t_I-out-of-n_I issuance and t_O-out-of-n_O opening. There is still a small price to pay for the efficiency, as not all issuers can be corrupt for anonymity and, likewise, not all openers can be malicious for traceability (contrarily to what might be expected). Still, moving to a threshold setting already avoids the most critical assumption, namely a fully trusted party, and instead tolerates corruption of some of them.

One challenge in designing the scheme is to separate the role of issuers and openers. It is necessary in the scheme of Bichsel et al. as the information needed for opening is created during issuance. The following scheme avoids that by assuming a public ledger to which users can upload their opening information during issuance. This information is encrypted under the opener

keys during issuance, and a user must prove to the issuers that she uploaded a valid ciphertext. Section 4.4 also presents a scheme that does not assume a ledger but instead combines the roles of issuer and opener anew, and supports threshold issuance and opening.

First comes a variant of the PS signature scheme on which the scheme is based, and then the description of the threshold group signatures.

4.3.1 Variant of the PS Signature Scheme

Consider the PS signature scheme (Section 4.1.5) in which the extra scalar m' is computed as $\mathcal{H}(m)$. In the same vein, the group element h could also be returned by the hash function, i.e., $(m', h) \leftarrow \mathcal{H}(m)$. This would allow several signers of the same message to agree on a common base h. The scheme remains unforgeable and the main difference from the unforgeability proof of Pointcheval and Sanders [PS18, Section 4.3] is that when $\mathcal{H}$ is queried on a message m different from the challenge message, the reduction algorithm already prepares (σ_1, σ_2) to be later returned in case the adversary makes a signing query on m. For more detail, see the unforgeability proof of the PS multi-signature scheme (introduced in Section 4.5) which relies on the same idea. This technique is similar to that of Sonnino et al. [SABD18] for their credential system. They hash a commitment to the signed message to obtain a base h, even though they apply it to the CT-RSA'16 version of the PS scheme (so without m') and do not formally prove that the scheme remains secure.

Besides, assume that the messages to be signed are publicly indexed, i.e., that for every message m there exists a unique value idx_m known to any signer. To sign a message m, instead of hashing to determine a scalar m' and a base h, a signer could instead compute m' as $\mathcal{H}(idx_m)$. If the number of messages to be signed is known in advance to be polynomial, the scheme remains unforgeable under the same assumption since indexing messages to determine (m', h) is equivalent to specifying in the public parameters a pair (m', h) for each message m. It is this variant of the scheme that is considered in the construction of threshold group signatures in Section 4.3.2. Therein, the messages signed are user secret keys sk_{id} indexed by the user identities id.

4.3.2 Construction with Separate Issuers and Openers

This section starts with a highlight of the mains ideas of the construction, and then proceeds with a detailed definition of the protocols and algorithms.

Key Generation. During set-up, the issuers run the distributed key-generation protocol of Gennaro et al. [GJKR99] with t_I as a threshold to generate a public key for the PS signature scheme so that each of them holds a share of the secret key. The protocol guarantees that if $t_I < n_I/2$, then

the protocol terminates (which cannot be enforced with a dishonest majority) and no colluding t_I issuers can infer any information about the secret key, whereas any $t_I + 1$ issuers can reconstruct it.

As for the openers, each of them simply generates a pair of ElGamal keys.

Join. For $t_I + 1$ issuers to add a user to the group, they all blindly sign with their PS secret-key share a random secret key sk_{id} chosen by the user. To do so, they need to agree on a common PS base h, so we used the variant from Section 4.3.1 of the PS signature scheme. The user group secret key consists of sk_{id} and the PS signature on it.

For each opener, the user encrypts a t_O-out-of-n_O Shamir share of sk_{id} and proves that the ciphertexts are correctly computed. With the ElGamal public keys of the openers, the issuers can verify the proofs and thus be convinced that the any $t_O + 1$ openers will later be able to retrieve correct shares of the user secret key if they can access the ciphertexts.

To make sure that these shares can later be retrieved by the openers, we assume the existence of an append-only ledger L accessible to all users, issuers and openers. Once the user has encrypted the shares of her secret key and has proved that she did so, she writes the encrypted shares and the proofs on the ledger. The issuers then send their PS signature-shares only after verifying the proofs. Therefore, each opener can later retrieve his shares of all group-member secret keys from the ledger.

Note that since honest issuers add a given user identity id to the group only once and when the proofs are correct, there is only one entry with valid proofs per user that can open her signatures. This entry is the one denoted $L[id]$ in the descripiton of the opening algorithm.

Sign & Verify. To compute a group signature on a message m, the user computes a signature of knowledge on m of a valid PS signature on a user secret key sk_{id}. Verifying a signature on a message simply consists in verifying the signature of knowledge on it.

Open. To open a signature, any $t_O + 1$ openers first retrieve from the ledger their shares of user secret keys and store them in their registers. Once the openers' registers are updated, they test the signature to be opened against each entry in their registers until they find a user for which the shares match. It makes opening expensive for the benefit of having short signatures, which perfectly fits most practical scenarios, in which signatures should be short and opening an uncommon practice performed by resourceful authorities.

Scheme Description.

To formally define the construction, let $\mathcal{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p \times \mathbb{G}^*$, and $\mathcal{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ be two random oracles (the latter is to compute non-interactive proofs of knownledge via the Fiat–Shamir heuristic). Let also $ID \subseteq \mathbb{Z}_p$ denote a user identity space. The construction requires as building blocks

- the Section-4.3.1 variant of the PS signature scheme, further denoted PS, to sign user secret keys.
- an append-only ledger L with user identities as keys that is available to all users, issuers and openers
- secure (i.e., authenticated and confidential) channels
- a broadcast channel, i.e., a protocol between several parties that allows a sender to distribute a value to all the other parties so that the following three properties are satisfied:
  1. (termination) the protocol terminates
  2. (consistency) all honest parties receive the same value and
  3. (validity) if the sender is honest, then the value received by all honest parties is that of the sender.

Given a type-3 pairing group generator G and a security parameter $\lambda \in \mathbb{N}$, the $\binom{n_I,n_O}{t_I,t_O}$-DGS scheme $\mathsf{PS\text{-}DGS}$ in a pairing group $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}\left(1^\lambda\right)$ consists of the following procedures.

$\mathsf{GSetup}(1^\lambda, n_I, n_O, t_I, t_O) \to pp$: generate $(g, \tilde{g}) \in_R \mathbb{G}^* \times \tilde{\mathbb{G}}^*$. Set $pp_{\mathsf{PS}} \leftarrow (\Gamma, \tilde{g}, 1)$ and return $pp \leftarrow (pp_{\mathsf{PS}}, g, n_I, n_O, t_I, t_O)$.

$\langle \{\mathsf{IKG}(pp, i)\}_{i=1}^{n_I} \rangle \to \langle \{(ipk, isk_i, st_i)\}_{i=1}^{n_I} \rangle$: is a protocol between all the issuers who proceed as follows:

- the issuers run three times the distributed key-generation protocol of Gennaro et al. [GJKR99] with t_I as a threshold in group $\tilde{\mathbb{G}}$ to obtain three uniformly distributed public values $\tilde{X}$, $\tilde{Y}_0$ and $\tilde{Y}_1$. At the end of the protocol, each issuer $i \in [\![n_I]\!]$ holds shares x_i, $y_{0,i}$ and $y_{1,i}$ such that for any $I \in \binom{[\![n_I]\!]}{t_I+1}$, if w_i denotes the Lagrange coefficient of issuer i, then $x \coloneqq \operatorname{dlog}_{\tilde{g}} \tilde{X} = \sum_{i \in I} x_i w_i$, and similarly for $y_0 \coloneqq \operatorname{dlog}_{\tilde{g}} \tilde{Y}_0$ and $y_1 \coloneqq \operatorname{dlog}_{\tilde{g}} \tilde{Y}_1$.
- issuer $i \in [\![n_I]\!]$ returns $\left(ipk \leftarrow \left(\tilde{X}, \tilde{Y}_0, \tilde{Y}_1\right), isk_i \leftarrow (i, x_i, y_{0,i}, y_{1,i}), st_I \leftarrow \perp\right)$. The issuers send ipk to a certification authority which is assumed to make it publicly available so that anyone can get an authentic copy of it.

$\langle\{\mathsf{OKG}(pp,i)\}_{i=1}^{n_O}\rangle \to \langle\{(opk, osk_i, \mathbf{reg}_i)\}_{i=1}^{n_O}\rangle$: for each opener, generate an ElGamal pair of keys $(\tilde{f}_i \leftarrow \tilde{g}^{z_i}, z_i) \in \tilde{\mathbb{G}}^* \times \mathbb{Z}_p^*$ and initialize an empty register $\mathbf{reg}_i$. Set $opk_i \leftarrow \tilde{f}_i$ and $osk_i \leftarrow (i, \tilde{f}_i, z_i)$, and $opk \leftarrow ((i, opk_i))_{i=1}^{n_O}$. For each opener, return $(opk, osk_i, \mathbf{reg}_i)$. The opener sends opk to a certification authority.

The group public key gpk is set to (ipk, opk).

GJoin : Assume that there is a broadcast channel between a user $\mathcal{U}$ and the $t_I + 1$ issuers $\mathcal{I}_i$ in $I \in \binom{[\![n_I]\!]}{t_I+1}$. Assume also that there is a secure channel between $\mathcal{U}$ and every issuer $\mathcal{I}_i$. In particular, this implies that an adversary cannot modify the messages sent by the user to the issuers: it can only forward, delay or drop them. Throughout the following description of the protocol, whenever an algorithm receives an abort or an ill-formed message, or when a verification fails, it interrupts the protocol execution by broadcasting an abort message to all participants and returning $\perp$. The joining protocol between the user and the issuers is as follows:

1. GJoin.U, on input $\Big(id, I, gpk = \Big(ipk = (\tilde{X}, \tilde{Y}_0, \tilde{Y}_1), opk\Big)\Big)$,
   - choose $sk_{id} \in_R \mathbb{Z}_p^*$
   - $(a', h) \leftarrow \mathcal{H}_0(id)$
   - $h_{sk} \leftarrow h^{sk_{id}}$; $g_{sk} \leftarrow g^{sk_{id}}$. Therefore, (g, h, g_{sk}, h_{sk}) is a DDH tuple. It helps the reduction algorithm of the traceability proof to efficiently extract the secret keys of adversarial users (under the ADH-KE assumption).
   - $\pi \leftarrow FS.\mathsf{Prove}\{sk_{id} : h_{sk} = h^{sk_{id}} \wedge g_{sk} = g^{sk_{id}}\}$
   - generate $p_1, \ldots, p_{t_O} \in_R \mathbb{Z}_p$ and set $P \leftarrow sk_{id} + \sum_{\ell=1}^{t_O} p_\ell X^\ell \in \mathbb{Z}_p[X]$
   - for $i \in [\![n_O]\!]$, compute $s_i \leftarrow P(i)$, i.e., Shamir shares of sk_{id} for each opener
   - for $\ell \in [t_O]$, compute $h_\ell \leftarrow h^{p_\ell}$, i.e., verification values as in the Feldman verifiable secret sharing scheme [Fel87]
   - for all $i \in [\![n_O]\!]$:
     - $r_i \leftarrow_\$ \mathbb{Z}_p$
     - $\tilde{C}_i := (\tilde{C}_{i,0}, \tilde{C}_{i,1}) \leftarrow \left(\tilde{g}^{r_i}, \tilde{f}_i^{r_i} \tilde{Y}_0^{s_i}\right)$
     - $\pi_i \leftarrow FS.\mathsf{Prove}\Big\{r_i : \tilde{C}_{i,0} = \tilde{g}^{r_i}, e\left(h, \tilde{C}_{i,1}/\tilde{f}_i^{r_i}\right) = e\left(h_{sk} \prod_{\ell=1}^{t_O} h_\ell^{i^\ell}, \tilde{Y}_0\right)\Big\}$, i.e., compute a proof that $\tilde{C}_i$ encrypts the ith share of sk_{id}
   - set $L[id] \leftarrow \left(g_{sk}, h_{sk}, h_1, \ldots, h_{t_O}, \pi, \left(\tilde{C}_i, \pi_i\right)_{i \in [\![n_O]\!]}\right)$

- broadcast written to all $\mathcal{I}_i$

2. GJoin.I, for $i \in I$, on input $(st_i, isk_i = (i, x_i, y_{0,i}, y_{1,i}), id, I, gpk)$
   - abort if $id \in st_i$
   - upon receiving written from $\mathcal{U}$:
     * $(a', h) \leftarrow \mathcal{H}_0(id)$
     * parse $L[id]$ as $\left(g_{sk}, h_{sk}, h_1, \ldots, h_{t_O}, \pi, \left(\tilde{C}_i, \pi_i\right)_{i \in [\![n_O]\!]}\right)$
     * $FS.\mathsf{Vf}(g, h, g_{sk}, h_{sk}, \pi) \stackrel{?}{=} 1$, i.e., verify that it is a DDH tuple
     * for $j \in [\![n_O]\!]$, $FS.\mathsf{Vf}\left(h, (h_\ell)_{\ell=1}^{t_O}, \tilde{Y}_0, \tilde{f}_j, \tilde{C}_j, \pi_j\right) \stackrel{?}{=} 1$, i.e., verify that the ciphertexts encrypt correct shares for each opener
     * $\Sigma_{i,2} \leftarrow h^{x_i + y_{i,1} a'} h_{sk}^{y_{i,0}}$ (i.e., blindly sign sk_{id} via h_{sk})
     * $st_i \leftarrow st_i \cup \{id\}$
     * send $\Sigma_{i,2}$ to $\mathcal{U}$ over a secure channel
3. GJoin.U, upon receiving $\Sigma_{i,2}$ from all $\mathcal{I}_i$ for $i \in I$,
   - $\Sigma \leftarrow \left(a', h, \prod_{i \in I} \Sigma_{i,2}^{w_i} = h^{x + y_0 sk_{id} + y_1 a'}\right)$, i.e., reconstruct the PS signature w.r.t. to ipk
   - $\mathsf{PS.Vf}(ipk, sk_{id}, \Sigma) \stackrel{?}{=} 1$
   - return $\mathbf{gsk}[id] \leftarrow (sk_{id}, \Sigma)$

$\mathsf{GSign}(ipk, \mathbf{gsk}[id], m) \to \sigma$: parse $\mathbf{gsk}[id] = (sk_{id}, \Sigma = (a', \Sigma_1, \Sigma_2))$. Generate $r \in_R \mathbb{Z}_p^*$. Compute $(\Sigma_1', \Sigma_2') \leftarrow (\Sigma_1^r, \Sigma_2^r)$ and

$$\pi \leftarrow FS.\mathsf{Prove}\{(sk_{id}, a') \colon \mathsf{PS.Vf}(ipk, sk_{id}, (a', \Sigma_1', \Sigma_2')) = 1\}(m).$$

That is, compute, with $\mathcal{H}_1$ as random oracle, a Schnorr signature of knowledge $\pi = (c, v_{sk}, v_{a'}) \in \mathbb{Z}_p^3$ on m of a pair (sk_{id}, a') such that $e\left({\Sigma_1'}^{sk_{id}}, \tilde{Y}_0\right) e\left({\Sigma_1'}^{a'}, \tilde{Y}_1\right) = e(\Sigma_2', \tilde{g}) e(\Sigma_1', \tilde{X})^{-1}$. Return $\sigma \leftarrow (\Sigma_1', \Sigma_2', \pi)$.

$\mathsf{GVerf}(ipk, m, \sigma) \to b \in \{0, 1\}$: parse σ as $(\Sigma_1, \Sigma_2, \pi)$. Return $FS.\mathsf{Vf}(gpk, \Sigma_1, \Sigma_2, m, \pi)$. That is, return 1 if

$$c = \mathcal{H}_1\left(ipk, \Sigma_1, \Sigma_2, e\left({\Sigma_1}^{v_{sk}}, \tilde{Y}_0\right) e\left({\Sigma_1}^{v_{a'}}, \tilde{Y}_1\right) e\left({\Sigma_2}^{c}, \tilde{g}\right) e\left({\Sigma_1}^{-c}, \tilde{X}\right), m\right)$$

and 0 otherwise.

GOpen : is run by $t_O + 1$ openers $\mathcal{O}_i$ (for $i \in O$) to recover the identity of the user who computed a valid group signature $\sigma = (\Sigma_1, \Sigma_2, \pi)$ on a message m. To do so, the openers first update their registers by

checking the public ledger L. Then, the opening algorithms loop over the entries of their registers $\mathbf{reg}_i$ containing encryptions of shares $\tilde{Y}_0^{s_i}$ recorded during executions of protocol GJoin. For each identity id for which they have a share, the opening algorithms use their shares to determine whether (a', Σ_1, Σ_2) (with $(a', h) = \mathcal{H}_0(id)$) is a valid PS signature on the unique value determined by their $t_O + 1$ shares of the secret key of user id. If it is the case, the algorithms return id. If no such identity is found, the opening algorithm returns $\perp$. The protocol assumes a broadcast channel between the participating openers, and also that the protocol is aborted as soon as an algorithm receives an abort or an ill-formed message.

Formally, $\mathsf{GOpen}\left(\mathbf{reg}_i, osk_i = (i, z_i), I, gpk, m, \sigma = (\Sigma_1, \Sigma_2, \pi)\right)$ proceeds as follows:

1. if $\mathsf{GVerf}(ipk, m, \sigma) = 0$ then return $\perp$
2. for all id such that $L[id] \neq \perp$, if $\mathbf{reg}_i[id] = \perp$ then parse $L[id]$ as $\left(g_{sk}, h_{sk}, h_1, \ldots, h_{t_O}, \pi, (\tilde{C}_i, \pi_i)_{i \in [\![n_O]\!]}\right)$ and set $\mathbf{reg}_i[id] \leftarrow \tilde{C}_{i,1} / \tilde{C}_{i,0}^{z_i}$
3. for all id such that $\mathbf{reg}_i[id] \neq \perp$, compute $T_{id,i} \leftarrow e\left(\Sigma_1, \tilde{C}_{i,1} / \tilde{C}_{i,0}^{z_i}\right)$
4. broadcast $S_i \leftarrow \{(id, T_{id,i})\}_{id\colon \mathbf{reg}_i[id] \neq \perp}$ to all the openers in I
5. upon receiving S_j from all the other openers $\mathcal{O}_j$ (for $j \in I \backslash \{i\}$),
   - for $j \in I$, compute $w_j \leftarrow \prod_{\ell \in I \backslash \{j\}} \ell / (\ell - j)$, i.e., the jth Lagrange coefficient
   - for all $(id, T_{id,i}) \in S_i$ (in lexicographic order of user identities)
     - if $\exists j \in I\colon (id, *) \notin S_j$ then continue
     - $(a', h) \leftarrow \mathcal{H}_0(id)$
     - for all $j \in I \backslash \{i\}$, retrieve $(id, T_{id,j})$ from S_j
     - if $e\left(\Sigma_1, \tilde{X}\, \tilde{Y}_1^{a'}\right) \prod_{j \in I} T_{id,j}^{w_j} = e(\Sigma_2, \tilde{g})$ then return id
6. return $\perp$ (i.e., in case the previous equality holds for no tuple in $\mathbf{reg}_i$).

Remark 4.3.3. *The signing and verification algorithms only need the short issuer public key, not the entire group public key.*

Theorem 4.3.4 (Correctness)**.** PS-DGS *is correct.*

Proof. The correctness of the scheme follows from the correctness of the key-generation protocol of Gennaro et al., the correctness of the Shamir sharing scheme, the correctness of the Elgamal encryption scheme (relevant during issuance and opening) and the completeness of Schnorr proofs. □

The lemmas below give important arguments for the proofs of anonymity and traceability.

Lemma 4.3.5 (Unanimity)**.** *Protocol* GOpen *is unanimous between honest openers in the anonymity and traceability security experiments. That is, for all* (O, m, σ)*, if* id_i *denotes the output of the* i*th opening algorithm and* $\mathcal{HO}$ *the index set of honest openers, then* $id_i = id_j$ *for all* $i, j \in O \cap \mathcal{HO}$*.*

Proof. During GOpen protocol executions, the broadcast protocol and the fact that the ledger is publicly available ensure that all honest openers receive the same opening sets S_j. Moreover, the opening algorithms test the signature verification equality only on identities for which they all have an entry in their registers. As they proceed in the same order (i.e., lexicographic order), the claim follows. □

Lemma 4.3.6 (Forgery)**.** *If the SDL assumption over the pairing-group generator* G *holds, then in the anonymity and traceability security experiments for* PS-DGS*, no efficient adversary* $\mathcal{A}$ *can forge, with non-negligible probability, a signature that opens to an honest user for which it has not queried the secret keys. That is, no efficient adversary* $\mathcal{A}$ *can, with non-negligible probability, make an oracle query* $\mathsf{GOpen}(O, m, \sigma)$*, or, in the traceability experiment, output* (I^*, m^*, σ^*)*, such that the identity* id/id^* *output by the opening algorithms of the honest openers in* O/O^* *is so that* $id/id^* \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$ *(which implies* $id/id^* \neq \perp$*) and* $(id, m, \sigma)/(id^*, m^*, \sigma^*) \notin Q_{\mathsf{GSign}}$*.*

Proof. This lemma can be proved by contradiction as follows. Assume that there exists an adversary $\mathcal{A}$ that computes such a forgery in either of the games with probability at least ε and in time $\tau_{\mathcal{A}}$. Assume $p > 8q_{\mathcal{H}_1}/\varepsilon$. Consider a reduction algorithm $\mathcal{B}$ which interacts with the SDL challenger.

On the input of an SDL tuple $(g, \tilde{g}, g^{\mu}, \tilde{g}^{\mu})$, the idea of the proof is to compute a PS signature on μsk_{id^*} with $sk_{id^*} \in_R \mathbb{Z}_p^*$ chosen by $\mathcal{B}$ for a random user identity id^*. If this identity is the one for which $\mathcal{A}$ forges a signature, then $\mathcal{B}$ can extract μsk_{id^*}, so also μ and win the SDL game. To extract μsk_{id^*}, algorithm $\mathcal{B}$ runs the forking algorithm of the generalized forking lemma (Lemma 4.1.8) on an algorithm $\mathcal{A}'$ of which $\mathcal{A}$ is a sub-routine.

In more detail, $\mathcal{A}'$ runs on the input of public parameters pp as in the scheme, of a tuple $(g_1, \tilde{g}_1, g_2, \tilde{g}_2) \in \left(\mathbb{G}^* \times \tilde{\mathbb{G}}^*\right)^2$ and of a random tape ρ.

$\mathcal{A}'$ answers random-oracle queries as follows:

- $\mathcal{H}_0(id)$: forward the query to $\mathcal{B}$.
- $\mathcal{H}_1(b \in \{0,1\}^*)$: if $(b, *) \notin Q_{\mathcal{H}_1}$ then generate $c \in_R \mathbb{Z}_p$, do $ctr \leftarrow ctr + 1$, add (b, c, ctr) to $Q_{\mathcal{H}_1}$ and return c; else retrieve (b, c, ι) from $Q_{\mathcal{H}_1}$ and return c.

During the key-generation phase, $\mathcal{A}'$ proceeds as in the real scheme. At the end of the protocol, it reconstructs the secret keys x, y_0 and y_1 using the shares of the honest issuers. It can do so since there are at least $t_I + 1$ honest

issuers by assumption. If $y_0 = 0 \mod p$, which occurs with probability at most $1/p$, algorithm $\mathcal{B}$ aborts its interaction with $\mathcal{A}$ and sends a uniformly random $\mathbb{Z}_p^*$ element to the SDL challenger.

$\mathcal{A}'$ then answers game-oracle queries as follows (if a check fails or if it ever aborts, $\mathcal{A}'$ returns $(\emptyset, \emptyset, \emptyset)$):

- $\mathsf{GJoin.U}(id, I)$: check that $I \in \binom{[\![n_I]\!]}{t_I+1}$. Add id to Q_{GJoin}. A GJoin execution is triggered and $\mathcal{B}$ plays the role of the user and of the honest issuers.
  - if $id \neq id^*$, follow the protocol but also save sk_{id} in $\mathbf{reg}_j[id]$ for each honest opener $\mathcal{O}_j$.
  - if $id = id^*$, first check that $id^* \notin st_i$ for all honest issuer $\mathcal{I}_i$. Make an internal query $(a'^*, r^*) \leftarrow \mathcal{H}_0(id^*)$. Next, generate $sk_{id^*} \in_R \mathbb{Z}_p^*$ and shares $(s_i)_{i \in [\![n_O]\!]}$ of $0_{\mathbb{Z}_p}$ (, so $\sum_{j \in I} s_j w_j = 0 \mod p$ for all $O \in \binom{[\![n_O]\!]}{t_O+1}$). Compute $h_{sk} \leftarrow g_2^{r^* sk_{id^*}} = g_1^{r^* \mu sk_{id^*}}$ and $g_{sk} \leftarrow g_2^{sk_{id^*}}$. To compute verification values $(h_1, \ldots, h_{t_O})$ that convince all corrupt issuers and openers that s_i is a valid share of $\mathrm{dlog}_{g_1^{r^*}}(h_{sk}) = \mu sk_{id^*}$, solve via Gaussian elimination the linear system (with the notation $x \cdot g \coloneqq g^x$ for $x \in \mathbb{Z}_p$ and $g \in \mathbb{G}$)

$$\begin{bmatrix} i & \cdots i^{t_O} \end{bmatrix}_{i \in \mathcal{CO}} \begin{bmatrix} h_1 \\ \vdots \\ h_{t_O} \end{bmatrix} = \begin{bmatrix} (g_1^{r^*})^{s_i} (h_{sk}^*)^{-1} \end{bmatrix}_{i \in \mathcal{CO}}$$

    with unknowns $h_1, \ldots, h_{t_O}$ and $\mathcal{CO}$ as the set of corrupt openers. This linear system has a solution since $\begin{bmatrix} i & \cdots i^{t_O} \end{bmatrix}_{i \in \mathcal{CO}}$ is full-rank, for it is a sub-matrix of a Vandermonde matrix and the positions $i \in \mathcal{CO}$ are pairwise distinct. It should be noted that unlike the case of the real protocol, there is no relation between the value μsk_{id}^* signed by the issuers and the shares s_i.

    Now, program $\mathcal{H}_1$ to simulate a proof of knowledge π of $\mathrm{dlog}_{g_1}(g_{sk}) = \mathrm{dlog}_{g_1^{r^*}}(h_{sk})$. For $i \in \mathcal{CO}$, compute as in the scheme ciphertexts $\tilde{C}_i$ using the $h_1, \ldots, h_{t_O}$ values obtained by solving the linear system, as well as proofs π_i. For each honest opener i, choose uniformly random $\tilde{C}_i$ and program oracle $\mathcal{H}_1$ to simulate proofs of knowledge π_i.

    Set $L[id]$ as in the real protocol, and playing the role of the user, broadcast $\mathsf{written}$. Under the DDH assumption over $\tilde{\mathbb{G}}$, from the point of view of $\mathcal{A}$, the distribution of $L[id]$ is indistinguishable from its distribution in the real protocol since $\mathcal{A}$ can decrypt at most t_O shares. The distinguishing advantage of $\mathcal{A}$ is at most n_O times the advantage of $\mathcal{B}$ running it as a sub-routine and attempting to win DDH game in $\tilde{\mathbb{G}}$.

Compute signature shares on μsk_{id^*} (using h_{sk}) for each the honest issuers and send them to the user after adding id^* to their states. Upon receiving signature shares from $\mathcal{A}$, reconstruct the PS signature and verify the result as in the proof of each of the properties (anonymity of traceability). The resulting signature is

$$\left(a'^*, \Sigma_1^* \leftarrow g_1^{r^*}, \Sigma_2^* \leftarrow \left(g_1^{r^*}\right)^{x+y_0\mu sk_{id^*}+y_1 a'^*}\right),$$

i.e., it is a signature on μsk_{id^*}. Set $\mathbf{gsk}[id^*] \leftarrow (\perp, (a'^*, \Sigma_1^*, \Sigma_2^*))$ and for all honest $\mathcal{O}_j$, set $\mathbf{reg}_j[id^*]$ as in the scheme but also append $(s_i)_{i=1}^{n_O}$.

- $\mathsf{GJoin}.\mathsf{I}_i(id, I)$: check that $i \in I \in \binom{[\![n_I]\!]}{t_I+1}$. Add id to Q_{GJoin} and Q_{Corrupt} (note that $id \neq id^*$ in the event in which the adversary outputs a forgery, necessarily). A GJoin protocol execution is triggered. Play the role of the honest issuers and follow the protocol. If the protocol succeeds, reconstruct $\tilde{Y}_0^{sk_{id}}$ with the shares given by the adversary (recall that there are at least t_O+1 honest openers in either game) for issuers in I, and append $\tilde{Y}_0^{sk_{id}}$ to $\mathbf{reg}_j[id]$ for each honest opener $\mathcal{O}_j$. If the reconstructed value is not consistent with the share of an honest opener (which can be verified with the pairing), abort the protocol for it means that $\mathcal{A}$ broke the soundness of the zero-knowledge proof. It occurs with probability at most $(q_{\mathcal{H}_1}+1)/p$. The probability that it occurs for any of the identities is then at most $|ID|(q_{\mathcal{H}_1}+1)/p$.

- $\mathsf{GSign}(id, m)$: check that $id \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$. If $id \neq id^*$ then compute $\sigma \leftarrow \mathsf{GSign}(ipk, \mathbf{gsk}[id], m)$, add (id, m, σ) to Q_{GSign} and return σ. If $id = id^*$ then fetch $(\perp, (a'^*, \Sigma_1^*, \Sigma_2^*)) \leftarrow \mathbf{gsk}[id^*]$, generate $\alpha \in_R \mathbb{Z}_p^*$, compute $(\Sigma_1', \Sigma_2') \leftarrow (\Sigma_1^*, \Sigma_2^*)^\alpha$, simulate a proof π of knowledge of μsk_{id}^* and a'^* by programming $\mathcal{H}_1$, add $(id, m, \sigma = (\Sigma_1', \Sigma_2', \pi))$ to Q_{GSign}, and return σ.

- $\mathsf{GOpen}_i(O, m, \sigma = (\Sigma_1, \Sigma_2, \pi))$: check that $i \in O \in \binom{[\![n_O]\!]}{t_O+1}$. Add (m, σ) to Q_{GOpen}. A GOpen execution is triggered. Play the role of the honest openers in O. Check that $\mathsf{GVerf}(ipk, m, \sigma) = 1$. Update the registers as in the real scheme. Test the equality

$$e\left(\Sigma_1, \tilde{g}_1^{x+y_1 a'^*} \tilde{g}_2^{y_0 sk_{id^*}}\right) = e(\Sigma_2, \tilde{g}_1).$$

  - If the equality does not hold but $e\left(\Sigma_1, \tilde{g}_1^{x+y_1 a'} \tilde{Y}_0^{sk_{id}}\right) = e(\Sigma_2, \tilde{g}_1)$ for an identity $id \neq id^* \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$, where $(a', h) \leftarrow \mathcal{H}_0(id)$, then $\mathcal{A}$ forged a signature on an honest identity $id \neq id^*$ for which it does not have the secret key. Abort the interaction with $\mathcal{A}$ and send a uniformly random $\mathbb{Z}_p^*$ element to the SDL

challenger. This is to make sure that the first identity for which $\mathcal{A}$ forges a signature is id^* with probability $1/|ID|$.

* If it holds, if $id^* \in Q_{\mathsf{GJoin}}$ and if $(id^*, m, \sigma) \notin Q_{\mathsf{GSign}}$ then $\mathcal{A}$ has forged σ on m for id^*.

  Parsing σ as $(\Sigma_1, \Sigma_2, \pi = (c, v_{sk}, v_{a'}))$, let ι_σ be the computation step at which $\mathcal{A}$ queried $\mathcal{H}_1$ on $b_\sigma \leftarrow (ipk, \Sigma_1, \Sigma_2, u, m)$ with

$$u := e\left(\Sigma_1^{v_{sk}}, \tilde{Y}_0\right) e\left(\Sigma_1^{v_{a'}}, \tilde{Y}_1\right) e\left(\Sigma_1^{-c}, \tilde{X}\right) e\left(\Sigma_2^{c}, \tilde{g}_1\right),$$

  be it the step at which it outputs the forgery, i.e., ι_σ is such that $(b_\sigma, c, \iota_\sigma) \in Q_{\mathcal{H}_1}$. Algorithm $\mathcal{A}'$ sets $J \leftarrow \{\iota_\sigma\}$ and $aux \leftarrow \emptyset$, and returns $(J, \{\sigma\}, aux)$.

* If the equality holds and if $(id^*, m, \sigma) \in Q_{\mathsf{GSign}}$ (which implies $id^* \in Q_{\mathsf{GJoin}}$) then fetch $\mathbf{reg}_i[id^*] = (*, (s_i)_{i=1}^{n_O})$. For $j \in I \setminus \{i\}$, compute $T_{id^*,j} \leftarrow e\left(\Sigma_1, \tilde{Y}_0\right)^{s_j}$. Compute

$$T_{id^*,i} \leftarrow \left(e\left(\Sigma_1, \tilde{g}_2^{y_0 sk_{id^*}}\right) \prod_{j \in I \setminus \{i\}} T_{id^*,j}^{-w_j}\right)^{1/w_i}$$
$$= e\left(\Sigma_1, \tilde{Y}_0\right)^{\left(\mu sk_{id^*} - \sum_{j \neq i} s_j w_j\right)/w_i} = e\left(\Sigma_1, \tilde{Y}_0\right)^{\mu sk_{id^*}/w_i + s_i}.$$

  The values $(T_{id^*,j})_{j \in I}$ are indistinguishable from the values in the real protocol since $\mu sk_{id^*}/w_i + s_i$ and $(s_j)_{j \in I \setminus \{i\}}$ are valid shares of μsk_{id^*} (recall that $\sum_{j \in I} s_j w_j = 0$ for all $O \in \binom{[\![n_O]\!]}{t_O+1}$). Therefore, with $T_{id^*,i}$ thus computed, the corrupt participating openers, if they follow the protocol, will open to id^*. For the identities $id \neq id^*$ such that $\mathbf{reg}_i[id] \neq \perp$, compute the $T_{id,i}$ values as in the scheme. Follow the rest of the protocol.

* If the equality does not hold and $id^* \in Q_{\mathsf{GJoin}}$, proceed as in the previous case in which the equality held and $(id^*, m, *) \in Q_{\mathsf{GSign}}$. The same indistinguishability arguments apply.

* If the equality does not hold and $id^* \notin Q_{\mathsf{GJoin}}$ then follow the protocol.

- $\mathsf{RevealU}(id)$: if $id = id^*$ then abort the interaction with $\mathcal{A}$ and send a uniformly random $\mathbb{Z}_p^*$ element to the SDL challenger. Add id to Q_{RevealU}. Return $\mathbf{gsk}[id]$.

- $\mathsf{ReadReg}(i, id)$: (in the traceability experiment only) return $\mathbf{reg}_i[id]$.

- $\mathsf{WriteReg}(i, id, v)$: (in the anonymity experiment only) set $\mathbf{reg}_i[id] \leftarrow v$.

- In the anonymity experiment $\mathbf{Exp}^{\mathrm{ano}-b}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A})$ for $b \in \{0,1\}$, at the challenge phase, if $id^* \neq id_b$ then return $\mathsf{GSign}(ipk, \mathbf{gsk}[id_b], m^*)$. If $id_b = id^*$ then generate $\alpha \in_R \mathbb{Z}_p^*$, compute $\Sigma_1 \leftarrow g_1^{\alpha}$ and $\Sigma_2 \leftarrow \Sigma_1^{x+y_1 a'^*}(g_2)^{\alpha y_2 sk^*_{id}}$. Simulate a proof π of knowledge of μsk_{id^*} and a'^* by programming $\mathcal{H}_1$. Return $(\Sigma_1, \Sigma_2, \pi)$.

If $\mathcal{A}$ never forges a signature for an honest identity, $\mathcal{A}'$ returns $(\emptyset, \emptyset, \emptyset)$.

The reduction algorithm $\mathcal{B}$ then proceeds as follows. Upon receiving Γ and $(g_1, \tilde{g}_1, g_2, \tilde{g}_2)$ from the challenger, $\mathcal{B}$ sets $pp_{\mathsf{PS}} \leftarrow (\Gamma, \tilde{g}_1, 1)$ and $pp \leftarrow (pp_{\mathsf{PS}}, g, n_I, n_O, t_I, t_O)$. It then runs $\mathcal{A}'$ on the input of pp, $(g_1, \tilde{g}_1, g_2, \tilde{g}_2)$ and a uniformly random tape ρ.

Throughout its interaction with $\mathcal{A}'$, algorithm $\mathcal{B}$ answers random-oracle queries as follows:

- $\mathcal{H}_0(id)$: if $(id, *) \notin Q_{\mathcal{H}_0}$ then forward the query to $\mathcal{C}$, receive (a', h), add $(id, (a', h))$ to $Q_{\mathcal{H}_0}$ and return (a', h); else retrieve $(id, (a', h))$ from $Q_{\mathcal{H}_0}$ and return (a', h).

Recall that $p > 8tq_{\mathcal{H}_1}/\varepsilon$ by assumption. With probability at least $\varepsilon/8$ and with running time at most $8q_{\mathcal{H}_1}/\varepsilon \cdot \ln(8/\varepsilon) \cdot \tau_{\mathcal{A}}$, the forking algorithm of the generalized forking lemma (Lemma 4.1.8) applied to $\mathcal{A}'$ returns $(\{\iota_\sigma\}, \{\sigma\}, \{\sigma'\}, Aux)$. That is, it returns two forgeries for honest identities.

Parsing σ as $(\Sigma_1, \Sigma_2, \pi = (c, v_{sk}, v_{a'}))$, the computation step at which $\mathcal{A}$ queried $\mathcal{H}_1$ on $(ipk, \Sigma_1, \Sigma_2, u, m)$ with

$$u \coloneqq e\left(\Sigma_1^{v_{sk}}, \tilde{Y}_0\right) e\left(\Sigma_1^{v_{a'}}, \tilde{Y}_1\right) e\left(\Sigma_1^{-c}, \tilde{X}\right) e\left(\Sigma_2^{c}, \tilde{g}_1\right)$$

is the same step at which it queried $\mathcal{H}_1$ on $(gpk, \Sigma'_1, \Sigma'_2, u', m')$ for u' similarly defined. The answers c and c' to these queries are also distinct modulo p.

As the inputs to $\mathcal{A}$ and its randomness are identical until that $\mathcal{H}_1$ query, $\Sigma_1 = \Sigma'_1$, $\Sigma_2 = \Sigma'_2$, $u = u'$ and $m = m'$ necessarily. It also implies $id^* = id'$ by the initial equality test. It follows that

$$\begin{aligned} e\left(\Sigma_1, \tilde{Y}_0^{(v_{sk}-v'_{sk})/(c'-c)} \tilde{Y}_1^{(v_{a'}-v'_{a'})/(c'-c)}\right) &= e(\Sigma_2, \tilde{g}_1) e\left(\Sigma_1, \tilde{X}\right)^{-1} \\ &= e\left(\Sigma_1, \tilde{g}_1^{y_1 a'^*} \tilde{g}_2^{y_0 sk_{id^*}}\right) \end{aligned}$$

with the second equality due to the initial test equality. Therefore,

$$\tilde{g}_1^{\left(y_0(v_{sk}-v'_{sk})+y_1(v_{a'}-v'_{a'})\right)/(c'-c)} = \tilde{g}_1^{y_1 a'^*} \tilde{g}_2^{y_0 sk_{id^*}}.$$

Compute and send to the SDL challenger the value

$$\left(y_0(v_{sk}-v'_{sk}) + y_1\left((v_{a'}-v'_{a'}) - a'^*(c'-c)\right)\right)/(c'-c)y_0 sk_{id^*},$$

thereby winning the SDL game.

In the event in which $\mathcal{A}$ forges with non-negligible probability a signature that opens to an honest user for which it has not queried the secret keys, the first identity for which it does is id^* with probability $1/|ID|$. Algorithm $\mathcal{B}$ then wins the SDL game with probability at least

$$1/|ID| \cdot \varepsilon/8 \cdot (1 - 1/p) - |ID|(q_{\mathcal{H}_1} + 1)/p - n_O|ID|\mathbf{Adv}^{\mathrm{DDH}-2}_{\mathsf{G},\lambda}(\mathcal{B}(\mathcal{A}))$$

and with running time at most $8q_{\mathcal{H}_1}/\varepsilon \cdot \ln(8/\varepsilon) \cdot \tau_{\mathcal{A}} + O(|ID|)$. If ε were negligible, $\mathcal{B}$ would contradict the SDL assumption. Consequently, ε is necessarily negligible. □

Theorem 4.3.7 (Anonymity)**.** *Scheme* PS-DGS *is anonymous under the first-group DDH and the SDL assumptions over the pairing-group generator* G *for any* $t_O < n_O/2$ *and* $t_I = t_I^* < n_I/2$.

Proof. The anonymity of DGS can be proved via the following hybrid argument. Denote by $\mathcal{C}_b$ the challenger of experiment $\mathbf{Exp}^{\mathrm{ano}-b}_{\mathsf{DGS},\lambda,n_I,n_O,t_I,t_O}(\mathcal{A})$. Let $\mathcal{H}$ be an algorithm that proceeds exactly like $\mathcal{C}_0$ except for the challenge query. To answer the challenge query, $\mathcal{H}$ sends two random $\mathbb{G}$ elements as re-randomized group elements of the PS multi-signature in $\mathbf{gsk}[id_b]$, and programs oracle $\mathcal{H}_1$ to compute a proof of knowledge of sk_{id_b} and a'_b, and complete the challenge group signature.

To show that DGS satisfies anonymity, it suffices to show that $\mathcal{C}_0$ and $\mathcal{C}_1$ are both computationally indistinguishable from $\mathcal{H}$ under the first-group DDH and the SDL assumptions.

Assume that the SDL assumption holds over G and that there exists an efficient adversary $\mathcal{A}$ that can distinguish $\mathcal{C}_0$ from $\mathcal{H}$ with probability at least ε and in time $\tau_{\mathcal{A}}$. Assume that $p > 8q_{\mathcal{H}_1}/\varepsilon$. Consider an algorithm $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and interacts with a DDH challenger $\mathcal{C}^{\mathrm{DDH}}_\beta$ that outputs a Diffie–Hellman tuple if $\beta = 1$ or a uniformly random tuple if $\beta = 0$.

Upon receiving the description of a pairing group $\Gamma = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$ and of a tuple $(g_1, g_2, g_3, g_4) = \left(g_1, g_1^{\mu}, g_1^{\nu}, g_1^{\xi}\right)$ from the DDH challenger, $\mathcal{B}$ sets g_1 as the generator of $\mathbb{G}$ in the public parameters and generates the other DGS parameters.

$\mathcal{B}$ starts by executing the key-generation protocol with $\mathcal{A}$ and reconstructing the shares of the dishonest managers (which is possible since there are at least $t_I + 1$ honest issuers). Algorithm $\mathcal{B}$ then chooses two identities id_0^* and id_1^* uniformly at random. To answer game-oracle queries, $\mathcal{B}$ proceeds as algorithm $\mathcal{A}'$ in Lemma 4.3.6, except that everything done for id^* is duplicated for id_0^* and id_1^*, and except for GSign queries on $(id_d^*, *, *)$ for $d \in \{0, 1\}$, for all GOpen queries and for the challenge query. For those queries, $\mathcal{B}$ proceeds as follows:

- $\mathsf{GSign}(id_d^*, m)$: check that $id^* \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$. Fetch $(\perp, (a_d'^*, \Sigma_{d,1}^*, \Sigma_{d,2}^*)) \leftarrow \mathbf{gsk}[id_d^*]$, generate $\alpha \in_R \mathbb{Z}_p^*$, compute $(\Sigma_1', \Sigma_2') \leftarrow (\Sigma_{d,1}^*, \Sigma_{d,2}^*)^\alpha$, simulate a proof π of knowledge of $\mu sk_{id_d^*}$ and $a_d'^*$ by programming $\mathcal{H}_1$. Add $(id, m, \sigma = (\Sigma_1', \Sigma_2', \pi), \alpha)$ to Q_{GSign} and return σ. The random α used to compute the signature is necessary to later correctly simulate the opening of group signatures as $\mathcal{B}$ does not have access to $\tilde{g}_2^\mu$ as in the previous lemma.

- $\mathsf{GOpen}_i(O, m, \sigma = (\Sigma_1, \Sigma_2, \pi))$: check that $i \in O \in \binom{[\![n_O]\!]}{t_O+1}$. Add (m, σ) to Q_{GOpen}. A GOpen execution is triggered. Play the role of the honest openers in I. Check that $\mathsf{GVerf}(ipk, m, \sigma) = 1$. By Lemma 4.3.6, this query cannot be a forgery for id_0^* or id_1^* (in the event in which $\mathcal{A}$ queries Ch on $(id_0^*, id_1^*, *)$ and receives an answer different from $\perp$). For all (id, m, σ, α) in Q_{GSign}, test the equality

$$e\left(\Sigma_1, \tilde{g}_1^{x+y_1 a'^*}\right) e\left(g_2^{r_d^* \alpha}, \tilde{Y}_0^{sk_{id_d^*}}\right) = e(\Sigma_2, \tilde{g}_1).$$

  for all $d \in \{0,1\}$. If it holds for some $d \in \{0,1\}$ and if $(id_d^*, m, *) \in Q_{\mathsf{GSign}}$ (which implies $id_d^* \in Q_{\mathsf{GJoin}}$) then fetch $\mathbf{reg}_i[id_d^*] = (*, (s_{d,i})_{i=1}^{n_O})$. For $j \in I \setminus \{i\}$, compute $T_{id_d^*,j} \leftarrow e(\Sigma_1, \tilde{Y}_0)^{s_{d,j}}$. Compute

$$T_{id_d^*,i} \leftarrow \left(e\left(g_2^{r_d^* \alpha sk_{id_d^*}}, \tilde{Y}_0\right) \prod_{j \in I \setminus \{i\}} T_{id_d^*,j}^{-w_j} \right)^{1/w_i}$$
$$= e(\Sigma_1, \tilde{Y}_0)^{\left(\mu sk_{id_d^*} - \sum_{j \neq i} s_j w_j\right)/w_i}.$$

  The values $(T_{id_d^*,j})_{j \in I}$ are indistinguishable from the values in the real protocol (in which case it would be $sk_{id_d^*}$ instead of $\mu sk_{id_d^*}$) as $\mathcal{A}$ never receives g_2. For the identities $id \neq id_d^*$ such that $\mathbf{reg}_i[id] \neq \perp$, compute the $T_{id,i}$ values as in the scheme. Continue as in Lemma 4.3.6.

- For the challenge query, if $(id_0, id_1) \neq (id_0^*, id_1^*)$ then halt the interaction with $\mathcal{A}$ and send 0 to the DDH challenger. Check that id_0^* and id_1^* are in $Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$, that $\mathbf{gsk}[id_d^*] \neq \perp$ for both $d \in \{0,1\}$ and that $(*, m^*, \sigma^*) \notin Q_{\mathsf{GSign}}$. Set $\Sigma_1 \leftarrow g_3$ and $\Sigma_2 \leftarrow g_3^{x+y_1 a_0'^*} g_4^{sk_{id_0^*}}$. Compute a simulated proof π of knowledge of $\mu sk_{id_0^*}$ and $a_0'^*$ by programming $\mathcal{H}_1$. Return $\sigma \leftarrow (\Sigma_1, \Sigma_2, \pi)$.

  Note that if $\beta = 1$, then the σ has the same distribution as the challenge signature computed by $\mathcal{C}_0$, whereas in case $\beta = 0$, Σ_2 is uniformly random in $\mathbb{G}^*$, and therefore σ has the same distribution as the signature computed by $\mathcal{H}$.

If $\beta = 1$ then algorithm $\mathcal{B}$ is perfectly indistinguishable from $\mathcal{C}_0$, and if $\beta = 0$ then it is perfectly indistinguishable from $\mathcal{H}$. The advantage of $\mathcal{B}$ in the DDH game is then at least ε.

It follows that

$$\begin{aligned}\varepsilon \leq & |ID|^2 \mathbf{Adv}^{\mathrm{DDH}-1}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda) + \frac{8p}{p-1}|ID|\mathbf{Adv}^{\mathrm{SDL}}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda) \\ & + \frac{|ID|(q_{\mathcal{H}_1}+1)}{p} + n_O|ID|\mathbf{Adv}^{\mathrm{DDH}-2}_{\mathsf{G},\lambda}(\mathcal{B}(\mathcal{A}))\,.\end{aligned}$$

Similarly, if $\mathcal{A}$ can distinguish $\mathcal{H}$ from $\mathcal{C}_1$, then $\mathcal{B}$ (by using $\mathbf{gsk}[id_1^*]$ instead of $\mathbf{gsk}[id_0^*]$ for the challenge query) wins the DDH game with probability at least ε.

Therefore,

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ano}}_{\mathsf{G},\mathsf{PS\text{-}DGS},n_I,n_O,t_I,t_O,\mathcal{A}}(\lambda)/2|ID| \leq & |ID|\mathbf{Adv}^{\mathrm{DDH}-1}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda) + \frac{8p}{p-1}\mathbf{Adv}^{\mathrm{SDL}}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda) \\ & + \frac{(q_{\mathcal{H}_1}+1)}{p} + n_O\mathbf{Adv}^{\mathrm{DDH}-2}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda)\end{aligned}$$

and the theorem follows. □

Theorem 4.3.8 (Traceability)**.** *Denoting by $q_{\mathcal{H}_0}$ the number of $\mathcal{H}_0$ queries, scheme* PS-DGS *satisfies traceability under the $q_{\mathcal{H}_0}$-MSDH-1, the ADH-KE and the SDL assumptions over the pairing-group generator* G *for any $t_I < n_I/2$, $t_O < n_O$ and $t_O^* = \min(t_O, n_O - t_O - 1)$.*

Proof. The proof consists in reducing the traceability of DGS to the existential unforgeability of the PS signature scheme. As its unforgeability relies on the $q_{\mathcal{H}_0}$-MSDH-1 assumption, the theorem follows. (Notice that GJoin.U, GJoin.I$_i$ and GOpen queries induce $\mathcal{H}_0$ queries.)

The proof idea is to apply the forking algorithm of the generalized forking lemma (Lemma 4.1.8) to an algorithm $\mathcal{A}'$ that runs $\mathcal{A}$ as a subroutine in order to obtain two distinct group signatures from which a PS signature forgery can be computed.

Suppose that there exists an adversary $\mathcal{A}$ that wins the $\binom{n_I,n_O}{t_I,t_I}$ traceability game for DGS with probability at least ε and in time $\tau_{\mathcal{A}}$. Further assume that $p > 8q_{\mathcal{H}_1}/\varepsilon$.

We first define $\mathcal{A}'$ and then a reduction algorithm $\mathcal{B}$ that interacts with the forgery-game challenger $\mathcal{C}$ for PS and applies the general forking lemma to $\mathcal{A}'$ to win the forgery game.

$\mathcal{A}'$ is an algorithm that runs on the input of public parameters pp as in the scheme, of a public key $(\tilde{X}, \tilde{Y}_0, \tilde{Y}_1)$ for the PS signature scheme and of a random tape ρ.

Throughout its interaction with $\mathcal{A}$, algorithm $\mathcal{A}'$ maintains the same global variables as the challenger of the traceability game. It initializes a counter $ctr \leftarrow 0$ and sets $Q_{\mathcal{H}_1} \leftarrow \emptyset$.

Throughout the experiment, $\mathcal{A}'$ answers random-oracle queries as follows:

- $\mathcal{H}_0(id)$: forward the query to $\mathcal{B}$.
- $\mathcal{H}_1(b \in \{0,1\}^*)$: if $(b,*) \notin Q_{\mathcal{H}_1}$ then generate $c \in_R \mathbb{Z}_p$, do $ctr \leftarrow ctr + 1$, add (b, c, ctr) to $Q_{\mathcal{H}_1}$ and return c; else retrieve (b, c, ι) from $Q_{\mathcal{H}_1}$ and return c.

At the beginning of the experiment, $\mathcal{A}'$ runs three times the simulator of key-generation protocol of Gennaro et al. with $\mathcal{A}$ on each $\tilde{X}$, $\tilde{Y}_0$ and $\tilde{Y}_1$ respectively, and plays the role of the honest issuers. At the end of the protocol executions, all honest issuers return an issuer public key ipk. Moreover, as there are at least $t_I + 1$ honest issuers, so $\mathcal{A}'$ can reconstruct the share x_i, $y_{i,0}$ and $y_{i,1}$ of the dishonest issuers. Let $\tilde{X}'$, $\tilde{Y}'_0$ and $\tilde{Y}'_1$ be the group elements defined by the shares generated during the simulation, i.e., $\tilde{X}' = \tilde{g}^{\sum_{i \in I} x_i w_i}$ (and $x' \coloneqq \sum_{i \in I} x_i w_i$) for any set $I \in \binom{[\![n_I]\!]}{t_I+1}$, where w_i is the Lagrange interpolation coefficient at position i; and similarly for $\tilde{Y}'_0$ and $\tilde{Y}'_1$.

For each of the honest openers, $\mathcal{A}'$ generates a pair of keys as in the scheme.

Algorithm $\mathcal{A}'$ then answers oracles queries as follows (if a check fails or if it ever aborts, $\mathcal{A}'$ returns $(\emptyset, \emptyset, \emptyset)$):

- $\mathsf{GJoin.U}(id, I)$: check that $I \in \binom{[\![n_I]\!]}{t_I+1}$. Add id to Q_{GJoin}. A GJoin execution is triggered. Play the role of the user and of the honest issuers with the shares generated by the simulator of the distributed key-generation protocol.

  Upon receiving signature shares $\Sigma_{2,i}$ from each of the corrupt issuers played by the adversary, test whether the equality

  $$e\left(\Sigma_1, \tilde{X}' \tilde{Y}_0'^{sk_{id}} \tilde{Y}_1'^{a'}\right) = e\left(\prod_{i \in I} \Sigma_{i,2}^{w_i}, \tilde{g}\right)$$

  holds, i.e., test whether the signature is valid for the key $(\tilde{X}', \tilde{Y}'_0, \tilde{Y}'_1)$. If not, abort the protocol, otherwise make a signing query to $\mathcal{C}$ on sk and receive a signature $\Sigma = (a', \Sigma_1, \Sigma_2)$. Note that $\Sigma_2 \neq \prod_{i \in I} \Sigma_{2,i}^{w_i}$ with overwhelming probability, because of the simulator of the key-generation protocol. Follow the joining protocol and set $\mathbf{gsk}[id] \leftarrow (sk, \Sigma)$. Note also that even if $\mathcal{A}$ causes the protocol to abort, or delays or drops messages, $\mathcal{B}$ has already added to Q_{GJoin} at the beginning of the protocol.

- $\mathsf{GJoin.I}_i(id, I)$: check that $i \in I \in \binom{[\![n_I]\!]}{t_I+1}$. Adds id to Q_{GJoin} and Q_{Corrupt}. A GJoin protocol execution is triggered. Play the role of the honest issuers.

Upon receiving $\mathsf{written}$ from the user played by $\mathcal{A}$, parse $L[id]$ as in the protocol. Make an internal query $(a', h) \leftarrow \mathcal{H}_0(id)$. Perform the same verifications as in the real scheme for each of the honest issuers.

If the verification of the proof on the encrypted shares holds but that the shares are not actually valid i.e., the soundness of the NIZK proof was broken, abort the protocol and return $(\emptyset, \emptyset, \emptyset)$. Note since there are at least $t_O + 1$ honest openers, $\mathcal{A}'$ can test whether the shares are valid as it then has enough shares to recompute the user polynomial.

If the verification $FS.\mathsf{Vf}(g, h, g_{sk}, h_{sk}, \pi) \stackrel{?}{=} 1$ succeeds, then (g, g_{sk}, h, h_{sk}) is a Diffie–Hellman tuple. Run $sk_{id} \leftarrow \mathsf{Ext}(\Gamma, g, g_{sk}, h, h_{sk})$. Abort the interaction with $\mathcal{A}$ if the extraction fails. Unless $\mathcal{A}$ can solve the discrete-logarithm problem in $\mathbb{G}$, which occurs only with negligible probability under the SDL assumption over G (which is itself implied by the q-MSDH-1 assumption), $sk_{id} = \mathrm{dlog}_g(g_{sk}) = \mathrm{dlog}_h(h_{sk})$. If the equality does not hold, then abort the interaction with $\mathcal{A}$. Otherwise, make a signing query on sk_{id} to $\mathcal{C}$ and receive a signature $\Sigma = (a', \Sigma_1 = h, \Sigma_2)$.

For the honest issuers $j \neq i$, compute $\Sigma_{2,j} \leftarrow h^{x_j + y_{j,0} sk_{id} + y_{j,1} a'}$.

For issuer i, compute

$$\begin{aligned}\Sigma_{2,i} &\leftarrow \left(\Sigma_2 \prod_{j \in I \setminus \{i\}} \Sigma_{2,j}^{-w_j}\right)^{1/w_i} \\ &= h^{\left(x - x' + (y_0 - y'_0) sk_{id} + (y_1 - y'_1) a'\right)/w_i + x_i + y_{i,0} sk_{id} + y_{i,1} a'}.\end{aligned}$$

Follow the rest of the protocol.

Note that even if $\mathcal{A}$ causes the protocol to abort, or delays or drops messages, $\mathcal{A}'$ has already added to Q_{GJoin} and Q_{Corrupt} at the beginning of the protocol. It implies that if $\mathcal{B}$ has made a signing query on an identity id, then $id \in Q_{\mathsf{GJoin}}$.

- $\mathsf{GSign}(id, m)$: check that $id \in Q_{\mathsf{GJoin}} \setminus Q_{\mathsf{Corrupt}}$. Compute $\sigma \leftarrow \mathsf{GSign}(ipk, \mathbf{gsk}[id], m)$. Add (id, m, σ) to Q_{GSign}. Return σ.
- $\mathsf{GOpen}_i(O, m, \sigma)$: (for $i \in \mathcal{HM}$) check that $i \in O \in \binom{[\![n_O]\!]}{t_O+1}$. Add (m, σ) to Q_{GOpen}. Play the role of the honest openers in I and follow the protocol.
- $\mathsf{RevealU}(id)$: add id to Q_{RevealU}. Returns $\mathbf{gsk}[id]$.
- $\mathsf{ReadReg}(i, id)$: return $\mathbf{reg}_i[id]$.

In the event in which $\mathcal{A}$ wins the game, it ultimately outputs a tuple (O^*, m^*, σ^*). Algorithm $\mathcal{A}'$ verifies that $O^* \notin \binom{\{t_O^*+1,\ldots,n_O\}}{t_O+1}$ and that

$\mathsf{GVerf}(ipk, m^*, \sigma^*) = 1$. If $\mathcal{A}$ does not output such a tuple or if the verifications fail, then $\mathcal{A}'$ returns $(\emptyset, \emptyset, \emptyset)$, otherwise it runs

$$\langle \{id_i^*\}_{i \in O^*} \rangle \leftarrow \langle \{\mathsf{GOpen}(\mathbf{reg}_i, osk_i, O^*, gpk, m^*, \sigma^*)\}_{i \in O^*} \rangle.$$

By Lemma 4.3.5, for all $i, j \in O^*$, $id_i^* = id_j^*$.

Moreover, since for each honest opener, any id for which $\mathbf{reg}_i[id] \neq \perp$ is such that $id \in Q_{\mathsf{GJoin}}$ (for $\mathcal{A}'$ starts by adding to Q_{GJoin} the identity of any joining query), it is impossible that the opening algorithms return an identity that is not in Q_{GJoin}. That is, setting $id^* \leftarrow \max O^*$, if $id^* \neq \perp$, then $id^* \in Q_{\mathsf{GJoin}}$ necessarily.

The winning conditions thus imply that in the event in which $\mathcal{A}$ wins the traceability game, either $id^* = \perp$, or $id^* \in Q_{\mathsf{GJoin}} \backslash Q_{\mathsf{Corrupt}}$ and $(id^*, m^*, \sigma^*) \notin Q_{\mathsf{GSign}}$.

By Lemma 4.3.6, $(id^*, m^*, \sigma^*) \notin Q_{\mathsf{GSign}}$, i.e., the second case occurs with negligible probability under the SDL assumption.

If $\mathcal{A}$ did not win the traceability game, $\mathcal{A}'$ returns $(\emptyset, \emptyset, \emptyset)$. Otherwise, parsing σ^* as $(\Sigma_1^*, \Sigma_2^*, \pi^* = (c^*, v_{sk}^*, v_{a'}^*))$, let ι_{σ^*} be the computation step at which $\mathcal{A}$ queried $\mathcal{H}_1$ on $b_{\sigma^*} \leftarrow (ipk, \Sigma_1^*, \Sigma_2^*, u^*, m^*)$ with

$$u^* \coloneqq e\left(\Sigma_1^{*v_{sk}^*}, \tilde{Y}_0\right) e\left(\Sigma_1^{*v_{a'}^*}, \tilde{Y}_1\right) e\left(\Sigma_1^{*-c}, \tilde{X}\right) e\left(\Sigma_2^{*c}, \tilde{g}_1\right),$$

be it the step at which it outputs the forgery, i.e., $(b_{\sigma^*}, c^*, \iota_{\sigma^*}) \in Q_{\mathcal{H}_1}$. Algorithm $\mathcal{A}'$ returns $(\{\iota_{\sigma^*}\}, \sigma^*, \emptyset)$.

Set

$$\tilde{\varepsilon} \leftarrow \varepsilon - |ID| \left(\mathbf{Adv}_{\mathsf{G},\lambda}^{\mathrm{ADH-KE}}(\mathcal{B}(\mathcal{A})) + \mathbf{Adv}_{\mathsf{G},\lambda}^{\mathrm{DLOG-1}}(\mathcal{B}(\mathcal{A}))\right) - \frac{8p}{p-1}|ID|\mathbf{Adv}_{\mathsf{G},\mathcal{B}(\mathcal{A})}^{\mathrm{SDL}}(\lambda) - \frac{|ID|(q_{\mathcal{H}_1}+1)}{p} - n_O|ID|\mathbf{Adv}_{\mathsf{G},\mathcal{B}(\mathcal{A})}^{\mathrm{DDH-2}}(\lambda).$$

It is a lower bound on the probability that $\mathcal{A}$ wins the traceability game for DGS, that $\mathcal{A}'$ never aborts during GJoin execution (prompted by a $\mathsf{GJoin.I}_i$ query) due to extraction failure of a user secret key and that the identity output by the honest openers at the forgery phase is $\perp$.

Let then $\mathcal{B}$ be an algorithm that runs $\mathcal{A}'$ as subroutine and interacts with the forgery-game challenger $\mathcal{C}$ for PS.

Upon receiving pp_{PS} and vk from the challenger, $\mathcal{B}$ generates $g \in_R \mathbb{G}^*$ and sets $pp \leftarrow (pp_{\mathsf{PS}}, g, n_I, n_O, t_I, t_O)$. It then runs $\mathcal{A}'$ on the input of pp, $(\tilde{X}, \tilde{Y}_0, \tilde{Y}_1)$ and a uniformly random tape ρ.

Throughout its interaction with $\mathcal{A}'$, algorithm $\mathcal{B}$ answers random-oracle queries as follows:

- $\mathcal{H}_0(id)$: if $(id, *) \notin Q_{\mathcal{H}_0}$ then forward the query to $\mathcal{C}$, receive (a', h), add $(id, (a', h))$ to $Q_{\mathcal{H}_0}$ and return (a', h); else retrieve $(id, (a', h))$ from $Q_{\mathcal{H}_0}$ and return (a', h).

Recall that $p > 8tq_{\mathcal{H}_1}/\varepsilon$ by assumption. With probability at least $\tilde{\varepsilon}/8$ and with running time at most $8q_{\mathcal{H}_1}/\tilde{\varepsilon} \cdot \ln(8/\tilde{\varepsilon}) \cdot \tau_{\mathcal{A}}$, the forking algorithm of the generalized forking lemma (Lemma 4.1.8) applied to $\mathcal{A}'$ returns $(\{\iota_{\sigma^*}\}, \{\sigma^*\}, \{\sigma^{*\prime}\})$ with σ^* and $\sigma^{*\prime}$ as group-signature forgeries that open to $\perp$. The lemma ensures that for $\sigma^* = (\Sigma_1^*, \Sigma_2^*, \pi^* = (c^*, v_{sk}^*, v_{a'}^*))$ and $\sigma^{*\prime}$ similarly parsed, the computation step at which the challenges c^* and $c^{*\prime}$ were computed are the same, and those challenge are distinct modulo p.

As the inputs to $\mathcal{A}$ and its randomness are identical until that $\mathcal{H}_1$ query, $\Sigma_1^* = \Sigma_1^{*\prime}$, $\Sigma_2^* = \Sigma_2^{*\prime}$, $u^* = u^{*\prime}$ and $m^* = m^{*\prime}$ necessarily. It follows that

$$e\left(\Sigma_1^*,\ \tilde{Y}_0^{(v_{sk}^* - v_{sk}^{*\prime})/(c^{*\prime}-c^*)}\ \tilde{Y}_1^{(v_{a'}^* - v_{a'}^{*\prime})/(c^{*\prime}-c^*)}\right) = e(\Sigma_2^*, \tilde{g}_1)e(\Sigma_1^*, \tilde{X})^{-1}.$$

Set $sk^* \leftarrow (v_{sk}^* - v_{sk}^{*\prime})/(c^{*\prime} - c^*)$, $a'^* \leftarrow (v_{a'}^* - v_{a'}^{*\prime})/(c^{*\prime} - c^*)$ and $\Sigma^* \leftarrow (a'^*, \Sigma_1^*, \Sigma_2^*)$. The latter is then a valid multi-signature on sk^*.

However, no honest issuer ever signed sk^* as there would otherwise exist an identity $id^* \neq \perp$ such that $\mathbf{reg}_i[id^*] = s_i^*$ for all opener $i \in O^*$ and $sk^* = \sum_{i \in I^*} s_i^* w_i$. It is the case since in each execution of GOpen, the openers always start by updating their registers. That identity id^* would have then be returned by the honest openers.

Algorithm $\mathcal{B}$ then sends sk^* and Σ^* to $\mathcal{C}$ as forgery and wins the existential forgery game for PS with probability at least $\tilde{\varepsilon}/8$. Therefore,

$$\begin{aligned}\mathbf{Adv}^{\text{trace}}_{\mathsf{G},\mathsf{DGS},N,t,\mathcal{A}}(\lambda) \leq\ & 8\mathbf{Adv}^{\text{EUF-CMA}}_{\mathsf{G},\mathsf{PS},1,\mathcal{A}}(\lambda) \\ & + |ID|\left(\mathbf{Adv}^{\text{ADH-KE}}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda) + \mathbf{Adv}^{\text{DLOG}-1}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda)\right) \\ & + \frac{8p}{p-1}|ID|\mathbf{Adv}^{\text{SDL}}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda) + \frac{|ID|(q_{\mathcal{H}_1}+1)}{p} \\ & + n_O|ID|\mathbf{Adv}^{\text{DDH}-2}_{\mathsf{G},\mathcal{B}(\mathcal{A})}(\lambda)\end{aligned}$$

As the first-group discrete logarithm and the SDL assumptions are both implied by the $q_{\mathcal{H}_0}$-MSDH-1 assumption, the theorem follows. □

Discussion.

The anonymity of the scheme is only guaranteed if less than half of the issuers are corrupt, and not all as one would hope. One reason is that the generation of the issuer keys is interactive, so for the protocol to terminate, there cannot be a dishonest majority. Another reason is that the reduction to the DDH requires to know all issuer secret keys which are shares of the PS secret key obtained during the key-generation protocol. To be able to reconstruct the shares of the corrupt issuers, the number $n - t_I^*$ of honest issuers must be greater than t_I^*.

Concerning traceability, it requires the number of corrupt openers not only to be smaller than $n_O - t_O - 1$ as explained in Section 4.2, but also

Scheme	Sig. Size	Sign	Verify
PS-DGS [Section 4.3]	$2\mathbb{G} + 3\mathbb{Z}_p$	$4\mathbb{G} + 1P^2$	$4\mathbb{G} + 1P^4$
PS [PS16]	$2\mathbb{G} + 2\mathbb{Z}_p$	$2\mathbb{G} + 1\mathbb{G}_T$	$3\mathbb{G} + 1P^3$
Bichsel et al. [BCN+10]	$3\mathbb{G} + 2\mathbb{Z}_p$	$3\mathbb{G} + 1\mathbb{G}_T$	$1\mathbb{G} + 1\mathbb{G}^2 + 2P^2$
CL [CL04, GGI19]	$7\mathbb{G} + 4\mathbb{Z}_p$	$11\mathbb{G} + 1\mathbb{G}^2 + 1\mathbb{G}_T^2$	$1\mathbb{G} + 2\mathbb{G}^2 + 2\mathbb{G}^3$ $+1\tilde{\mathbb{G}}^2 + 2P^2$
BBS* [BCN+10, GGI19]	$4\mathbb{G} + 5\mathbb{Z}_p$	$5\mathbb{G} + 3\mathbb{G}^2 + 1\mathbb{G}_T^5$	$4\mathbb{G}^2 + 1\mathbb{G}^3 + 1\mathbb{G}^4$ $+1P^2$
Coconut [SABD18]	$3\mathbb{G} + 1\tilde{\mathbb{G}} + 3\mathbb{Z}_p$	$4\mathbb{G} + 1\tilde{\mathbb{G}}^3 + 1\tilde{\mathbb{G}}^2$	$1\mathbb{G} + 1\mathbb{G}^2 + 1\tilde{\mathbb{G}}^4$ $+P^2$

Table 4.1: Comparison of the PS-DGS scheme with other schemes in terms of signature sizes, and the cost to compute and verify signatures. For the signing and verification costs, $\mathbb{G}^\ell$ indicates an ℓ-exponentiation in $\mathbb{G}$, and similarly for $\tilde{\mathbb{G}}$ and $\mathbb{G}_T$. P denotes the number of pairing computations required, and P^ℓ stands for the product of ℓ pairing values, which is more efficient than computing ℓ pairings separately.

smaller than t_O. It is due to the fact that even though the openers are separate from the issuers, they must still obtain user secret key shares from the joining protocol to be able to open signatures. In this sense, opening is not completely independent of issuance, and it is precisely what allows the signatures of the scheme to be so short. This constraint on the number of corrupt openers appears in the proof of Lemma 4.3.6 in which the forked algorithm must simulate shares of user secret keys, and it can only do so if at most t_O openers are corrupt since the opening threshold is $t_O + 1$.

Efficiency.

On a Cocks–Pinch pairing curve [GMT19] defined over a field of order 2^{544} and with embedding degree 8, group elements in $\mathbb{G}$ take 68 Bytes for a group of 256-bit order. Note that this curve provides 131 bits of security [GMT19].

A group signature from the above scheme consists of two $\mathbb{G}$ elements and three $\mathbb{Z}_p$ elements, totalling 232 Bytes. The hash value in the proof of knowledge of a multi-signature can actually be shortened to second-preimage resistant length, further shortening a group signature to 216 Bytes.

Considering only group operations, computing a signature costs 4 exponentiations in $\mathbb{G}$ and the product of 2 pairing values. Verifying a signature costs 4 exponentiations in $\mathbb{G}$ and the multiplication of 4 pairing values.

Comparison with other Schemes.

Table 4.1 gives a comparison of our threshold DGS scheme of with other CCA-anonymous dynamic group signatures schemes based on pairings. Lattice-based schemes are absent from the table since have considerably larger signa-

tures than pairing-based ones, and are therefore less preferred for practical applications.

Note that unlike the scheme of Bichsel et al. [BCN+10] and the scheme of Pointcheval and Sanders [PS16, Appendix A.1], our scheme supports threshold issuance and threshold opening, and does not rely on an interactive assumption, but rather a q-type of assumption.

The table features the CL and BBS* scheme. As explained in the introduction, Gennaro et al. extended [GGI19] those schemes to make them support threshold issuance. Since those schemes follow the sign-and-encrypt paradigm, the CL and BBS* scheme also easily allow for threshold opening. Note that generating the decryption key in a distributed and robust way (e.g., with the protocol of Gennaro et al.) would require an honest majority of openers. However, the security guarantees of those schemes in a threshold setting are unclear since Gennaro et al. did not provide a formal model for threshold group signatures schemes.

The table also includes the "Coconut" credential system which is based on the CT-RSA16 version of the PS signature scheme. The first reason is that their system and our scheme are related as presenting a credential consists in proving that a user knows a signature from an authority on her attributes. It is also how our group signatures are computed. It should be noted that the verification of a coconut token involves a multiplication in $\mathbb{G}$, which is approximately as expensive as a squaring, so it was counted as an exponentiation. Their system is further related to ours in that it supports threshold issuance. However, since it is a credential system, there is no need for opening, and it therefore avoids the challenge of also achieving (threshold) opening while maintaining signatures short. Besides, the authors did not provide a security model to analyze their scheme.

One can see that, except for the PS group signatures (of which the non-threshold traceability relies on an interactive assumption) which are not defined in a threshold setting, the signatures of our scheme are the shortest. Indeed, even when compared to the signatures of Bichsel et al. [BCN+10], our signatures are shorter since $\mathbb{Z}_p$ elements are typically shorter than $\mathbb{G}$ elements.

4.4 Threshold Group Signatures without Ledger

The construction in Section 4.3 requires an append-only ledger for users to communicate their secret-key shares to the openers. Such ledgers can be implemented in practice, but it is yet an additional assumption. This section thus proposes a scheme that does not require a ledger. However, this comes for the price of combining the roles of issuer and opener, as it is the case in some of the prior models for group signatures [BS04, CG05, BCN+10, PS16]. The authorities are now referred to as managers, and suppose that there are

n of them.

For the sake of simplicity, assume now that the issuance and opening thresholds are the same and denote it t. However, instead of having the same threshold for corruption, one could define (t, t_c)-out-of-n threshold group signatures as group signatures with n managers, of which $t+1$ must collaborate to add a user or to open a signature, and of which at most t_c can be corrupt. Defining a separate corruption threshold gives the flexibility to corrupt some managers, but not too many so that any two sets of t have at least one honest manager in common. This property ensure that any manager who did not add a user can recover her secret-key shares from an honest manager who added her when a signature is to be opened.

In this model, the main changes from the previous construction are as follows.

Key Generation. The managers now simply generate ElGamal keys separately and run the distributed key-generation protocol of Gennaro et al. [GJKR99] to generate PS keys. They use t as reconstruction threshold and t_c as corruption threshold.

Note that the original protocol of Gennaro et al. does not distinguish the reconstruction from the corruption threshold. However, their protocol can be adapted to a setting with two thresholds. It remains secure and robust (i.e., it terminates) as long as the number of honest parties (at least $n - t_c$) is greater than the reconstruction threshold (t in the present case), so that the honest parties can reconstruct the shares of qualified participants who received a valid complaint during the extraction phase [GJKR99, Fig. 2]. The two thresholds t and t_c must only satisfy $n - t_c > t$.

Join. As before, the user encrypts a t-out-of-n Shamir share of sk_{id} for each manager, even for the non-participating ones, and proves the validity of the ciphertexts. Each participating manager then verifies the correctness of the proofs. The difficulty is to ensure that the non-participating managers indeed get those ciphertexts now that there is no ledger.

Assuming $t \geq (n + t_c - 1)/2$ ($\geq t_c$), any two manager sets I and J of size $t+1$ have an intersection $I \cap J$ of size $|I| + |J| - |I \cup J| \geq 2(t+1) - n \geq t_c + 1$. It means that for every group member, any set of $t+1$ managers will always contain at least one honest manager that added her.

To ensure that the non-participating managers can later recover the shares, it suffices to have each manager in the joining protocol sign the encrypted shares of the user secret key with a multi-signature scheme (BLS [BDN18, Section 3.1] for instance) after verifying the

shares. Each manager then saves the shares and their multi-signatures in registers.

Update Registers. Afterwards, any set of $t+1$ managers can synchronize their registers and retrieve shares of every group-member secret key by first broadcasting the list of users they added. Next, turn by turn in lexicographic order, they broadcast the encrypted shares and multi-signatures for each user such that there is a manager who did not add her, and for which shares with a valid multi-signature have not not been broadcast yet.

Since $t \geq t_c$, a valid multi-signature by $t+1$ managers on a set of shares implies that at least one honest manager has signed them (after verifying them), so they are authentic. The managers can then update their registers by decrypting the shares they receive. Moreover, for every group member, the set of $t+1$ managers that added her has at least t_c+1 managers in common with the $t+1$ managers synchronizing their registers. Therefore, at least one honest manager will broadcast valid shares of her secret key, and each manager is guaranteed that his register now contains a valid secret-key share for every user that was ever added.

Sign&Verify. Computing and verifying signatures are done as in the Section 4.3 construction.

Open. As for opening, after synchronizing their registers, the managers participating in the opening protocol proceed as before.

Overall, the scheme is secure if $n - t_c > t \geq (n + t_c - 1)/2$ (which implies $t_c < \min(t, n/3)$). An interpretation of these bounds is that the joining threshold should be large enough so that for every group member, any set of openers contains at least one honest manager who added her, but not too large for the honest managers to be able to securely generate issuance keys.

4.5 Pointcheval–Sanders Multi-Signatures

This section introduces a novel multi-signature scheme based on the CT-RSA'18 version of the Pointcheval–Sanders signature scheme. It is the main building block of the scheme presented in Section 4.6. The CT-RSA'18 version of the PS signature scheme is referred to as the *modified* PS scheme, and the unforgeability proof of the multi-signature scheme makes a gradual reduction starting from the original PS signatures, i.e., the CT-RSA'16 version [PS16]. The core difference between the original and modified versions of PS signatures is that the original one can only be proved to be *weakly* unforgeable from the q-MDSH-1 assumption, whereas the modified scheme leverages an extra scalar to lift the security to standard unforgeability.

For each version of their signatures, Pointcheval and Sanders also distinguish the *single-message* variant that allows to sign a single message in $\mathbb{Z}_p$ from the *multi-message* variant that allows to sign blocks of messages. This distinction is also made for PS multi-signatures and is further relevant in the unforgeability proof.

The modified PS signature scheme can be turned into a multi-signature scheme by having each signer generate her pair of keys separately. However, to produce a signature for a given message, the signers need to agree on a common base h and a common extra-message m'. They can do so by hashing the message to be signed with a random oracle $\mathcal{H}_0 \colon \mathbb{Z}_p^k \to \mathbb{Z}_p \times \mathbb{G}^*$ to obtain these[1]. Moreover, to securely aggregate keys, as for BLS multi-signatures [BDN18], another random oracle $\mathcal{H}_1 \colon \tilde{\mathbb{G}}^{n(k+2)} \to \Theta^n \subseteq \mathbb{Z}_p^n$ (with $1/|\Theta|$ negligible in λ) is introduced.

Given a type-3 pairing-group generator G and a security parameter $\lambda \in \mathbb{N}$, the (modified) PS (multi-message) multi-signature scheme in a pairing group $\Gamma = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e) \leftarrow \mathsf{G}(1^\lambda)$ consists of the following algorithms:

$\mathsf{PSM.Setup}(1^\lambda, n, k, \Gamma) \to pp$: generate $g \in_R \mathbb{G}^*$, $\tilde{g} \in_R \tilde{\mathbb{G}}^*$ and return $pp \leftarrow (\Gamma, g, \tilde{g}, n, k)$. Integer n is the number of signers, and k is the number of messages to be signed.

$\mathsf{PSM.KG}(pp) \to (vk, sk)$: generate $x, y_1, \ldots, y_{k+1} \in_R \mathbb{Z}_p^*$, compute $\tilde{X} \leftarrow \tilde{g}^x$, $\tilde{Y}_j \leftarrow \tilde{g}^{y_j}$ for $j \in [k+1]$, and set $vk \leftarrow \left(\tilde{X}, \tilde{Y}_1, \ldots, \tilde{Y}_{k+1}\right)$, $sk \leftarrow (x, y_1, \ldots, y_{k+1})$. Return (vk, sk).

$\mathsf{PSM.KAggreg}(vk_1, \ldots, vk_n) \to avk$: compute $(t_1, \ldots, t_n) \leftarrow \mathcal{H}_1(vk_1, \ldots, vk_n)$ and return $avk \leftarrow \prod_{i=1}^n vk_i^{t_i}$.

$\mathsf{PSM.Sign}(sk, m = (m_1, \ldots, m_k)) \to \sigma$: compute $(m', h) \leftarrow \mathcal{H}_0(m) \in \mathbb{Z}_p \times \mathbb{G}^*$ and return $\sigma \leftarrow \left(m', h, h^{x + \sum_{j=1}^k y_j m_j + y_{k+1} m'}\right)$.

$\mathsf{PSM.SAggreg}((vk_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n) \to \sigma$: parse σ_i as $(m'_i, \sigma_{i,1}, \sigma_{i,2})$ for $i \in [\![n]\!]$. If $m'_1 = \cdots = m'_n$ and $\sigma_{1,1} = \cdots = \sigma_{n,1}$, compute $(t_1, \ldots, t_n) \leftarrow \mathcal{H}_1(vk_1, \ldots, vk_n)$ and $\sigma_2 \leftarrow \prod_{i=1}^n \sigma_{i,2}^{t_i} = \sigma_{1,1}^{\xi + \sum_{j=1}^k u_j m_j + u_{k+1} m'}$, with $\xi = \sum_{i=1}^n x_i t_i$, $u_j = \sum_{i=1}^n y_{i,j} t_i$ for $j \in [k]$ and $u_{k+1} = \sum_{i=1}^n y_{i,k+1} t_i$. Return $\sigma \leftarrow (m'_1, \sigma_{1,1}, \sigma_2)$. Otherwise, return $\perp$.

$\mathsf{PSM.Vf}(avk, m = (m_1, \ldots, m_k), \sigma) \to b$: parse σ as (m', σ_1, σ_2). If $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e\left(\sigma_1, \tilde{X} \prod_{i=1}^k \tilde{Y}_i^{m_i} \ \tilde{Y}_{k+1}^{m'}\right) = e(\sigma_2, \tilde{g})$ then return 1, else return 0.

[1] Note that Pointcheval and Sanders already suggested [PS18, Section 4.3] to use a random oracle to generate m' deterministically in their single-signer signature scheme. Using a hash function to generate h is a variation of this idea which has also been considered by Sonnino et al. [SABD18], but with the original PS signature scheme and without proving that the scheme remains secure.

Remark 4.5.1. *Note that the original PS signature scheme can also be turned into a multi-signature scheme in the random oracle model in the same way, except for m' which is omitted.*

Theorem 4.5.2. *In the random oracle model, denoting by $q_{\mathcal{H}_0}$ the maximum number of $\mathcal{H}_0$ oracle queries, the PS multi-signature scheme in a pairing group generated by G is EUF-CMA under the $q_{\mathcal{H}_0}$-MSDH-1 assumption over G.*

Proof. Similarly the proof [PS18, Section 5] of the existential unforgeability under CMA of the modified PS multi-message signature scheme, this proof is in two steps. First, the original single-message multi-signature scheme is proved to be EUF-wCMA secure under the $q_{\mathcal{H}_0}$-MSDH-1 assumption (this proof requires the following rewinding lemma, then the existential unforgeability of the modified multi-message multi-signature scheme under adaptive CMAs is reduced to the weak existential unforgeability of the original single-message multi-signature scheme (notice that each signing query implies an $\mathcal{H}_0$ query). The fact that the $q_{\mathcal{H}_0}$-MSDH-1 assumption implies the SDL assumption then implies the theorem.

Lemma 4.5.3 (Rewinding Lemma [BS17, Variant of Lemma 19.2])**.** *Let S, Θ and T be non-empty finite sets. Consider a function $f\colon S\times\Theta\times T\to\{0,1\}$. Let X, Y, Y', Z and Z' be mutually independent random variables such that X takes values in S, Y and Y' take values in Θ, and Z and Z' take values in T. Setting $\varepsilon=\Pr[f(X,Y,Z)=1]$,*

$$\Pr[f(X,Y,Z)=1, f(X,Y',Z')=1, Y\neq Y']\geq\varepsilon^2-\varepsilon/|\Theta|.$$

Lemma 4.5.4. *In the random oracle model, denoting by $q_{\mathcal{H}_0}$ the amount of non-adaptive signing queries, the original PS single-message multi-signature scheme is EUF-wCMA secure under the $q_{\mathcal{H}_0}$-MSDH-1 assumption.*

Proof. For $\lambda\in\mathbb{N}$, consider a pairing group $\Gamma\leftarrow\mathsf{G}(1^\lambda)$. Suppose that there exists an efficient adversary $\mathcal{A}$ that wins the weak existential unforgeability game with at most one query to oracle $\mathcal{H}_1$ (the general case will be considered further) with probability at least ε. Consider a reduction algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as a subrountine and receives a q-MSDH-1 instance from a challenger, i.e., receives two tuples $\left(g^{x^\ell},\tilde{g}^{x^\ell}\right)_{\ell=0}^{q}\in(\mathbb{G}^*)^{q+1}$ and $(g^a,\tilde{g}^a,\tilde{g}^{ax})\in\mathbb{G}^*\times(\tilde{\mathbb{G}}^*)^2$. Algorithm $\mathcal{B}$ has to output a tuple $\left(w,P,h^{1/x+w},h^{a/P(x)}\right)$ with $h\in_R\mathbb{G}^*$, P a polynomial in $\mathbb{Z}_p[X]$ of degree at most q and $w\in\mathbb{Z}_p$ such that the polynomials $X+w$ and P are coprime. At the beginning of the weak unforgeability game, adversary $\mathcal{A}$ sends to $\mathcal{B}$ a tuple of messages $(w_1,\ldots,w_q)\in\mathbb{Z}_p^q$ for which it expects signatures. $\mathcal{B}$ computes group elements $G\leftarrow g^{\prod_{\ell=1}^{q}(x+w_\ell)}$ and $\tilde{G}\leftarrow\tilde{g}^{\prod_{\ell=1}^{q}(x+w_\ell)}$, and sends $(G,\tilde{G},n,1)$ to adversary $\mathcal{A}$ as public parameters. $\mathcal{A}$ sends the index $i^*\in[\![n]\!]$ of a target

(honest) signer. $\mathcal{B}$ sets $\tilde{X}_{i^*} \leftarrow \tilde{g}^{ax}$ and $\tilde{Y}_{i^*} \leftarrow \tilde{g}^{a}$, and outputs $\left(\tilde{X}_{i^*}, \tilde{Y}_{i^*}\right)$ as the verification key of signer i^*. Notice that this implicitly sets

$$x_{i^*} = \frac{ax}{\prod_{\ell=1}^{q}(x+w_\ell)}, \qquad y_{i^*} = \frac{a}{\prod_{\ell=1}^{q}(x+w_\ell)}.$$

For $\ell = 1, \ldots, q$, algorithm $\mathcal{B}$ computes a signature on w_ℓ as follows: it generates $t_\ell \in_R \mathbb{Z}_p^*$, and computes and stores

$$\sigma_\ell = (\sigma_{\ell,1}, \sigma_{\ell,2}) = \left(\left(g^{\prod_{r\neq l}(x+w_r)}\right)^{t_\ell}, (g^a)^{t_\ell}\right).$$

Algorithm $\mathcal{B}$ can compute σ_ℓ from the q-MSDH-1 instance as the discrete logarithm of $\sigma_{\ell,1}$ to base g is a polynomial of degree $q-1$ in x and $\sigma_{\ell,2}$ is a single exponentiation of g^a. Note that

$$\begin{aligned}\sigma_{\ell,1}^{x_{i^*}+y_{i^*}w_\ell} &= \left(G^{\frac{t_\ell}{x+w_\ell}}\right)^{\frac{ax+aw_\ell}{\prod_{\ell=1}^{q}(x+w_\ell)}} = \left(G^{\frac{t_\ell}{x+w_\ell}}\right)^{\frac{a}{\prod_{r\neq l}(x+w_\ell)}} \\ &= g^{at_\ell} = \sigma_{\ell,2},\end{aligned}$$

and thus σ_ℓ is a valid signature on w_ℓ for the verification key $\left(\tilde{X}_{i^*}, \tilde{Y}_{i^*}\right)$. Algorithm $\mathcal{B}$ then returns $\sigma_1, \ldots, \sigma_q$ to adversary $\mathcal{A}$.

Whenever adversary $\mathcal{A}$ queries random oracle $\mathcal{H}_0$ on a message m, algorithm $\mathcal{B}$ checks whether $m \in \{w_\ell\}_{\ell=1}^{q}$. If $m = w_\ell$ for some $\ell \in [\![q]\!]$, algorithm $\mathcal{B}$ returns $\sigma_{\ell,1}$ as an answer to the random oracle query, otherwise if m has not been queried before, it generates, stores and returns $h \in_R \mathbb{G}^*$, otherwise (m is none of the w_ℓ and has been queried before), algorithm answers the query as it priorly did.

Whenever $\mathcal{A}$ makes its (unique) query to oracle $\mathcal{H}_1$ on $(vk_1, \ldots, vk_n) \in \tilde{\mathbb{G}}^{2n}$, algorithm $\mathcal{B}$ generates and returns $(t_1, \ldots, t_n) \in_R \Theta^n$. Adversary $\mathcal{A}$ eventually outputs a list of verification keys $\left(\tilde{X}_i, \tilde{Y}_i\right)_{i\neq i^*}$ and a valid forgery σ on a message $w \notin \{w_\ell\}_{\ell=1}^{q}$. If the unique query of $\mathcal{A}$ to $\mathcal{H}_1$ is not such that $vk_i = \left(\tilde{X}_i, \tilde{Y}_i\right)$ for $i = 1, \ldots, n$, then the forgery can be valid with probability at most $1/|\Theta|$, in which case algorithm $\mathcal{B}$ returns $\perp$ to the q-MSDH-1 challenger. Otherwise (the query to $\mathcal{H}_1$ is as such), as σ is a valid forgery, $\sigma_1 \neq 1_{\mathbb{G}}$ and

$$e\left(\sigma_1, \prod_{i=1}^{n} \tilde{X}_i^{t_i} \cdot \tilde{Y}_i^{wt_i}\right) = e\left(\sigma_1, \prod_{i\neq i^*} \tilde{X}_i^{t_i}\, \tilde{Y}_i^{wt_i}\right) e\left(\sigma_1, \tilde{X}_{i^*}^{t_{i^*}}\, \tilde{Y}_{i^*}^{wt_{i^*}}\right) = e\left(\sigma_2, \tilde{G}\right).$$

Algorithm $\mathcal{B}$ then rewinds $\mathcal{A}$ to the computation step at which it made its query to $\mathcal{H}_1$. Algorithm $\mathcal{B}$ generates $t'_{i^*} \in_R \Theta$ and replies with $(t_1, \ldots, t_{i^*-1}, t'_{i^*},$

$t_{i^*+1}, \ldots, t_n$). Adversary $\mathcal{A}$ can make queries anew, and if it eventually outputs another forgery σ', then $\sigma'_1 \neq 1_{\mathbb{G}}$ and

$$e\left(\sigma'_1, \prod_{i \neq i^*} \tilde{X}_i^{t_i} \tilde{Y}_i^{w t_i}\right) e\left(\sigma'_1, \tilde{X}_{i^*}^{t'_{i^*}} \tilde{Y}_{i^*}^{w t'_{i^*}}\right) = e\left(\sigma'_2, \tilde{G}\right).$$

On that account,

$$e\left(\sigma_1'^{t'_{i^*}} / \sigma_1^{t_{i^*}}, \tilde{X}_{i^*} \tilde{Y}_{i^*}^{w}\right) = e\left(\sigma'_2/\sigma_2, \tilde{G}\right),$$

so

$$e\left(\sigma_1'^{t'_{i^*}} / \sigma_1^{t_{i^*}}, \tilde{g}^{a(x+w)}\right) = e\left(\sigma'_2/\sigma_2, \tilde{g}^{\prod_{\ell=1}^{q}(x+w_\ell)}\right).$$

It follows that $\left(\sigma_1'^{t'_{i^*}} / \sigma^{t_{i^*}}, \sigma'_2/\sigma_2\right)$ is then of the form $\left(h^{x+w}, h^{\frac{a}{\prod_{\ell=1}^{q}(x+w_\ell)}}\right)$.

Setting $P = \prod_{\ell=1}^{q}(x + w_\ell)$, note that P is coprime with $(X + w)$ as $w \notin \{w_\ell\}_{\ell=1}^{q}$. Algorithm $\mathcal{B}$ then returns the tuple $\left(w, P, \sigma_1'^{t'_{i^*}} / \sigma^{t_{i^*}}, \sigma'_2/\sigma_2\right)$ and wins the q-MSDH-1 challenge. The rewinding lemma (Lemma 4.5.3) implies that adversary $\mathcal{A}$ outputs another forgery with probability at least $(\varepsilon^2 - \varepsilon/|\Theta|)$ by considering (for the application of the lemma) X as the inputs of $\mathcal{A}$ (including its random tape) up to its to query to $\mathcal{H}_1$ in the first run and its response, excluding t_{i^*}; Y and Y' as t_{i^*} and t'_{i^*} respectively; Z and Z' as the inputs given to $\mathcal{A}$ strictly after the query to $\mathcal{H}_1$ in the first and second run respectively; f as the function which outputs 1 if $\mathcal{A}$ outputs another forgery and 0 otherwise. If adversary $\mathcal{A}$ does not output another forgery, $\mathcal{B}$ returns $\perp$. It follows that algorithm $\mathcal{B}$ wins the q-MSDH-1 challenge with probability at least $(1 - 1/|\Theta|)(\varepsilon^2 - \varepsilon/|\Theta|)$. As $1/|\Theta|$ is negligible, if ε were non-negligible, $\mathcal{B}$ would win the q-MSDH-1 game with non-negligible probability: a contradiction. Such an adversary $\mathcal{A}$ can therefore not exist.

In the general case, i.e., for an adversary which makes $q_{\mathcal{H}_1} > 1$ queries to $\mathcal{H}_1$ and wins the weak forgery game with probability ε, there exists an adversary $\mathcal{A}'$ which runs $\mathcal{A}$ as a subroutine, makes only one query to $\mathcal{H}_1$, and wins the q-MSDH-1 challenge with probability at least $\varepsilon/q_{\mathcal{H}_1}$. Indeed, consider $\mathcal{A}'$ an algorithm which chooses uniformly at random one of the $q_{\mathcal{H}_1}$ queries to $\mathcal{H}_1$ and forwards it to the challenger, and replies to the rest of the $\mathcal{H}_1$-queries with by choosing uniformly random values itself. It also forwards to the challenger all the other type of queries that $\mathcal{A}$ makes. When $\mathcal{A}$ outputs a forgery together with a list of public keys, if those latter are the ones in the query $\mathcal{A}'$ chose to forward, $\mathcal{A}'$ submits the forgery, otherwise it returns $\perp$. Algorithm $\mathcal{A}'$ then wins the weak forgery game with probability at least $\varepsilon/q_{\mathcal{H}_1}$. Therefore, if there exists an efficient adversary $\mathcal{A}$ that wins the weak forgery game with probability at least ε by making q non-adaptive signing queries and $q_{\mathcal{H}_1}$ queries to oracle $\mathcal{H}_1$, there exists an algorithm $\mathcal{B}$ with

running time essentially twice that of $\mathcal{A}$, which wins the q-MSDH-1 game with probability at least $(1-1/|\Theta|)(\varepsilon^2-\varepsilon/|\Theta|)/q_{\mathcal{H}_1}$. Under the q-MSDH-1 assumption, ε must be negligible, and the PS single-message multi-signature is thus EUF-wCMA secure. □

Lemma 4.5.5. *If the SDL assumption holds over the group-generator* G *and the PS single-message multi-signature scheme is EUF-wCMA secure, then the modified PS multi-message multi-signature scheme is EUF-CMA secure.*

Proof. The proof consists in showing that if there exists an efficient algorithm that can win the forgery game for the modified PS k-message multi-signature scheme, then, if the SDL assumption holds, there exists an efficient algorithm that can win the weak forgery game of the original single-message signature-scheme. Let $\mathcal{A}$ be an adversary that wins the forgery game for the modified PS k-message multi-signature scheme, for an integer $k \geq 1$, with a non-negligible probability ε. As a signing query implies a query to $\mathcal{H}_0$, adversary $\mathcal{A}$ makes at most $q_{\mathcal{H}_0}$ signing queries. For $\ell \in [\![q_{\mathcal{H}_0}]\!]$, denote by $m_\ell = (m_{\ell,1}, \ldots, m_{\ell,k})$ the messages for which $\mathcal{A}$ makes a signing query. Under the SDL assumption, if i^* is the index of the target (honest) signer, then the message $m = (m_1, \ldots, m_k)$ for which $\mathcal{A}$ outputs a forgery is such that $\sum_{j=1}^{k} y_{i^*,j} m_j + y_{i^*,k+1} m_{k+1} \neq \sum_{j=1}^{k} y_{i^*,j} m_{\ell,j} + y_{i^*,k+1} m_{\ell,k+1}$ for $\ell \in [\![q_{\mathcal{H}_0}]\!]$ (if ℓ if strictly greater than the number of signing queries, set $m_\ell = \perp$).

Indeed, would it not be the case, consider a algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as a subroutine and interacts with an SDL challenger which outputs an instance $\left(g, \tilde{g}, Y = g^y, \tilde{Y} = \tilde{g}^y\right)$. Given a target (honest) signer index $i^* \in [\![n]\!]$ from $\mathcal{A}$, algorithm $\mathcal{B}$ generates $x_{i^*} \in_R \mathbb{Z}_p$ and $a_j, b_j \in_R \mathbb{Z}_p$ for $j = 1, \ldots, k+1$, and sets and sends $\tilde{X}_{i^*} = \tilde{g}^{x_{i^*}}$, $\tilde{Y}_{i^*,j} = \tilde{g}^{a_j} \tilde{Y}^{b_j}$ for $j = 1, \ldots, k+1$ to $\mathcal{A}$. It implicitly sets $y_{i^*,j} = a_j + y b_j$.

To answer $\mathcal{H}_1$ queries, $\mathcal{B}$ chooses uniformly random values. To answer the ℓ-th $\mathcal{H}_0$ query on a new message m_ℓ, algorithm $\mathcal{B}$ prepares and stores a signature on m_ℓ, i.e., generates $m'_\ell, t_\ell \in_R \mathbb{Z}_p$, sets $\sigma_{\ell,1} \leftarrow g^{t_\ell}$ and

$$\begin{aligned}\sigma_{\ell,2} &\leftarrow \left(g^{x_{i^*} + \sum_{j=1}^{k} a_j m_{\ell,j} + a_{k+1} m'_\ell} \cdot Y^{\sum_{j=1}^{k} y_j m_{\ell,j} + y_{k+1} m'_\ell}\right)^{t_\ell} \\ &= \sigma_{\ell,1}^{x_{i^*} + \sum_j y_{i^*,j} m_{\ell,j} + y_{\ell,k+1} m'_\ell}.\end{aligned}$$

Algorithm $\mathcal{B}$ then replies with $\sigma_{\ell,1}$. Later, if $\mathcal{A}$ makes a signing query on m_ℓ, algorithm $\mathcal{B}$ replies with $(m'_\ell, \sigma_{\ell,1}, \sigma_{\ell,2})$. If $\mathcal{A}$ makes a signing query on a message m for which it has not made a $\mathcal{H}_0$-query yet, algorithm $\mathcal{B}$ proceeds as before but also outputs the signature instead of only storing it. If $\mathcal{A}$ makes a $\mathcal{H}_0$-query for a message for which it has already made a signing or $\mathcal{H}_0$ query, algorithm $\mathcal{B}$ answers as it priorly did.

When $\mathcal{A}$ eventually outputs a forgery (m', σ_1, σ_2) on a message m such that $\sum_{j=1}^{k} y_{i^*,j} m_j + y_{i^*,k+1} m_{k+1} = \sum_{j=1}^{k} y_{i^*,j} m_{\ell,j} + y_{i^*,k+1} m_{\ell,k+1} \mod p$ for

some $\ell \in [\![q_{\mathcal{H}_0}]\!]$ and a message m_ℓ for which it has made a signing query, then

$$\tilde{g}^{\sum_{j=1}^{k} a_j(m_j - m_{\ell,j}) + a_{k+1}\left(m' - m'_{\ell,j}\right)} = \tilde{Y}^{\sum_{j=1}^{k} b_j(m_j - m_{\ell,j}) + b_{k+1}\left(m' - m'_{\ell,j}\right)}.$$

Since the distribution of the b_j values conditioned on the input of adversary $\mathcal{A}$ is uniformly random (because of the a_j values), $\sum_{j=1}^{k} b_j\,(m_j - m_{\ell,j}) + b_{k+1}\,(m' - m'_{\ell,j})\Big) = 0 \mod p$ with probability $1/p$. Consequently, algorithm $\mathcal{B}$ wins the SDL challenge with probability at least $(1-1/p)\varepsilon$, which is non-negligible and the SDL assumption is thus contradicted. It follows that if the SDL assumption holds, then the message $m = (m_1, \ldots, m_k)$ for which $\mathcal{A}$ outputs a forgery is necessarily such that $\sum_{j=1}^{k} y_{i^*,j} m_j + y_{i^*,k+1} m_{k+1} \neq \sum_{j=1}^{k} y_{i^*,j} m_{\ell,j} + y_{i^*,k+1} m_{\ell,k+1}$ for $\ell \in [\![q_{\mathcal{H}_0}]\!]$.

Consider then an algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as a sub-routine and interacts with a challenger for the weak unforgeability game of the PS k-message multi-signature scheme. On the input of public parameters, $\mathcal{B}$ forwards them to $\mathcal{A}$, receives a target signer index i^*, and forwards it to the challenger together with $q_{\mathcal{H}_0}$ messages $w_1, \ldots, w_{q_{\mathcal{H}_0}}$ chosen uniformly at random. The challenger outputs a verification key $\left(\tilde{X}_{i^*}, \tilde{Y}_{i^*}\right)$ for signer i^* and signatures $\sigma_1, \ldots, \sigma_{q_{\mathcal{H}_0}}$.

Next, algorithm $\mathcal{B}$ generates $u_j \in_R \mathbb{Z}_p$ for $j = 1, \ldots, k+1$, and computes $\tilde{Y}_{i^*,1} \leftarrow \tilde{Y}$ and $\tilde{Y}_{i^*,j} \leftarrow \tilde{Y}^{u_j}$ for $j = 2, \ldots, k+1$. Algorithm $\mathcal{B}$ then outputs $vk_{i^*} \leftarrow \left(\tilde{X}_{i^*}, \tilde{Y}_{i^*,1}, \ldots, \tilde{Y}_{i^*,k+1}\right)$. This implicitly sets $sk_{i^*} = (x_{i^*}, y_{i^*,1} = y_{i^*}, y_{i^*,2} = u_2 y_{i^*}, \ldots, y_{i^*,k+1} = u_{k+1} y_{i^*})$. To answer $\mathcal{H}_1$ queries, $\mathcal{B}$ chooses uniformly random values. To answer the ℓ-th $\mathcal{H}_0$ query on a new message m_ℓ, algorithm $\mathcal{B}$ prepares and stores a signature on m_ℓ by setting

$$m'_\ell \leftarrow u_{k+1}^{-1} \left(w_\ell - \sum_{j=1}^{k} u_j m_{\ell,j} \right) \mod p$$

with $u_1 = 1$. Since $y_{i^*,1} w_\ell = \sum_{j=1}^{k} y_{i^*,1} u_j m_{\ell,j} + y_{i^*,1} u_{k+1} m'_\ell = \sum_{j=1}^{k} y_j m_{\ell,j} + y_{k+1} m'_\ell$, the tuple $(m'_\ell, \sigma_{\ell,1}, \sigma_{\ell,2})$ is a valid signature on m_ℓ. Algorithm $\mathcal{B}$ then replies the $\mathcal{H}_0$-query with $\sigma_{\ell,1}$. Later, if $\mathcal{A}$ makes a signing query on m_ℓ, algorithm $\mathcal{B}$ replies with $(m'_\ell, \sigma_{\ell,1}, \sigma_{\ell,2})$. If $\mathcal{A}$ makes a signing query on a message m for which it has not made a $\mathcal{H}_0$-query yet, algorithm $\mathcal{B}$ proceeds as before but also outputs the signature instead of only storing it. If $\mathcal{A}$ makes a $\mathcal{H}_0$-query for a message for which it has already made a signing or $\mathcal{H}_0$ query, algorithm $\mathcal{B}$ answers as it priorly did.

When $\mathcal{A}$ eventually outputs a list of verifications keys for $i \neq i^*$ and a forgery (m', σ_1, σ_2) on a new message m (i.e., for which no signing query was made) such that $\sum_{j=1}^{k} y_{i^*,j} m_j + y_{i^*,k+1} m_{k+1} \neq \sum_{j=1}^{k} y_{i^*,j} m_{\ell,j} + y_{i^*,k+1} m_{\ell,k+1} \mod p$ for all $\ell \in [\![q_{\mathcal{H}_0}]\!]$, setting $w = \sum_{j=1}^{k} u_j m_j + u_{k+1} m'$, note that $y_{i^*,1} w \neq y_{i^*,1} w_\ell$ for $\ell \in [\![q_{\mathcal{H}_0}]\!]$. Therefore, (σ_1, σ_2) is a valid forgery for the new

message m. Algorithm $\mathcal{B}$ then perfectly simulates to $\mathcal{A}$ the challenger of the forgery game for the modified PS k-message multi-signature scheme, and wins with the same probability with which $\mathcal{A}$ wins the weak forgery game for the PS single-message multi-signature scheme. Hence the statement of the lemma. □

This concludes the proof of theorem. □

4.6 Distributed Group Signatures from Multi-Signatures

In practice, the ability to open signatures with a threshold number of openers seems to be a natural requirement. For privacy concerns, it is desirable to distribute the role of the opener over many entities. If this number is high, not all potential openers can be expected to be permanently available, or to agree in case of legal dispute for instance. Therefore, only a threshold amount of them should be required to open signatures.

The threshold aspect is, however, less critical for issuance. Although the ability to add users to the group should be *distributed* to several entities for stronger security, the number of issuers generally does not need to be as high as the number of potential openers. Effectively, issuers would be a few service providers, whereas openers would be judges or jurors. Besides, the eventuality of a disagreement about adding a user is less likely than a dispute about opening a signature. Distributed issuance instead of threshold issuance might then be satisfactory for many real-world scenarios.

Distributed Group Signatures. On this account, this section presents a group signature scheme with distributed issuance and threshold opening. The benefit of this dedicated distributed scheme is that the traceability of the scheme now holds even if all issuers but one are corrupt. It means that even if *all but one* issuer collude, they can neither compute an untraceable valid group signature nor forge a group signature that opens to an honest user who never computed it. In contrast, the PS-DGS scheme in Section 4.3 does not allow to corrupt $n_I - 1$ issuers because of the $t_I < n_I/2$ bound stemming from the key-generation protocol of Gennaro et al. [GJKR99].

Restricting issuance to a distributed setting rather than a threshold one allows to dispense with a key-generation protocol. Instead, the distributed scheme is based on PS multi-signatures. Relying on multi-signatures not only overcomes the $t_I < n_I/2$ bound but also features a simpler, non-interactive key generation phase.

In terms of signature size and computational costs, this group signature scheme still has the same efficiency as the scheme in Section 4.3, and it also supports the same threshold opening capabilities as the previous schemes.

4.6.1 Construction

Building on PS multi-signatures, the dynamic group signatures with distributed issuance and threshold opening can now be described. The main differences compared to the scheme in Section 4.3 are in the key-generation phase and the issuance protocol.

Key Generation. Given that PS multi-signatures support public key aggregation, the issuers can now generate their PS keys separately instead of executing a protocol. The PS public keys generated by the issuers can later be aggregated with a random oracle $\mathcal{H}_1$ to obtain a global issuer public key that can be used for signing and verifying signatures.

Nevertheless, the security proofs still require to be able to extract the keys of dishonest issuers, so they additionally have to prove knowledge of their PS secret keys. It is worth stressing that proving knowledge of secret keys is not needed for the stand-alone PS multi-signature scheme, but rather needed to prove the group-signature scheme secure.

The opener keys are generated as in the scheme in Section 4.3.

The group public key is now the concatenation of all issuer-and-opener public keys.

Issuance. During issuance, each issuer blindly signs h_{sk} with his secret key. After receiving signatures from all issuers, the user aggregates them and verifies their validity with respect to the global issuer public key, i.e., the PS aggregated public key. Therefore, user certificates are now PS multi-signatures instead of PS signatures.

Update Registers, Sign, Verify & Open. Updating registers, and computing, verifying and opening signatures are done as in the scheme of Section 4.3, except that signing and verifying are now done with another hash function $\mathcal{H}_2$ modeled as a random oracle.

Formally, the key-generation and key-aggregation algorithms of the issuers and the issuance protocol are as follows.

$\mathsf{IKG}(pp, i \in [\![n]\!]) \to (ipk_i, isk_i, st_i))$: generate $(vk, sk) \leftarrow \mathsf{PSM.KG}(pp)$ and set $(pk_i, sk_i) \leftarrow (vk, sk)$. Denote by $\mathcal{R}_{\mathsf{PSM}}$ the relation of honestly generated PS multi-signature public and secret keys. Compute $\pi_i \leftarrow FS.\mathsf{Prove}\{sk_i : (pk_i, sk_i) \in \mathcal{R}_{\mathsf{PSM}}\}$. Set $ipk_i \leftarrow (i, pk_i, \pi_i)$ and $isk_i \leftarrow (ipk_i, sk_i)$. Initialize an empty state st_i. Return (ipk_i, isk_i, st_i). The vector of all issuer public keys is denoted ipk.

$\mathsf{IKAggreg}(ipk_1, \ldots, ipk_n) \to ipk$: for $i \in [\![n]\!]$, parse ipk_i as (i, pk_i, π_i). Verify that for all $i \in [\![n]\!]$, $FS.\mathsf{Vf}(\tilde{g}, pk_i, \pi_i) = 1$. Compute an aggregated key $avk \leftarrow \mathsf{PSM.KAggreg}(pk_1, \ldots, pk_n)$. Set an return $ipk \leftarrow avk$.

The group public key gpk is set to (ipk, opk).

GJoin The protocol assumes a secure channel between $\mathcal{U}$ and every issuer $\mathcal{I}_i$ as well as a broadcast channel. Formally,

1. GJoin.U, on input (id, gpk),
   - choose $sk_{id} \in_R \mathbb{Z}_p^*$
   - $(a', h) \leftarrow \mathcal{H}_0(id)$
   - $h_{sk} \leftarrow h^{sk_{id}}$; $g_{sk} \leftarrow g^{sk_{id}}$
   - $\pi \leftarrow FS.\mathsf{Prove}\{sk_{id} \colon h_{sk} = h^{sk_{id}} \wedge g_{sk} = g^{sk_{id}}\}$
   - generate $p_1, \ldots, p_{t_O} \in_R \mathbb{Z}_p$ and set $P \leftarrow sk_{id} + \sum_{\ell=1}^{t_O} p_\ell X^\ell \in \mathbb{Z}_p[X]$
   - for $i \in [\![n_O]\!]$, compute $s_i \leftarrow P(i)$
   - for $\ell \in [\![t_O]\!]$, compute $h_\ell \leftarrow h^{p_\ell}$
   - for all $i \in [\![n_O]\!]$:
     - $r_i \leftarrow_\$ \mathbb{Z}_p$
     - $\tilde{C}_i := (\tilde{C}_{i,0}, \tilde{C}_{i,1}) \leftarrow \left(\tilde{g}^{r_i}, \tilde{f}_i^{r_i} \tilde{Y}_0^{s_i}\right)$
     - $\pi_i \leftarrow FS.\mathsf{Prove}\left\{r_i \colon \tilde{C}_{i,0} = \tilde{g}^{r_i}, e\left(h, \tilde{C}_{i,1}/\tilde{f}_i^{r_i}\right) = e\left(h_{sk} \prod_{\ell=1}^{t_O} h_\ell^{i^\ell}, \tilde{Y}_0\right)\right\}$
   - set $L[id] \leftarrow \left(g_{sk}, h_{sk}, h_1, \ldots, h_{t_O}, \pi, \left(\tilde{C}_i, \pi_i\right)_{i \in [\![n_O]\!]}\right)$
   - broadcast written to all $\mathcal{I}_i$
2. GJoin.I, for $i \in [\![n_I]\!]$, on input $(st_i, isk_i = (i, x_i, y_{0,i}, y_{1,i}), id, I, gpk)$
   - abort if $id \in st_i$
   - upon receiving written from $\mathcal{U}$:
     - $(a', h) \leftarrow \mathcal{H}_0(id)$
     - parse $L[id]$ as $\left(g_{sk}, h_{sk}, h_1, \ldots, h_{t_O}, \pi, \left(\tilde{C}_i, \pi_i\right)_{i \in [\![n_O]\!]}\right)$
     - $FS.\mathsf{Vf}(g, h, g_{sk}, h_{sk}, \pi) \stackrel{?}{=} 1$
     - for $j \in [\![n_O]\!]$, $FS.\mathsf{Vf}\left(h, (h_\ell)_{\ell=1}^{t_O}, \tilde{Y}_0, \tilde{f}_j, \tilde{C}_j, \pi_j\right) \stackrel{?}{=} 1$
     - $\Sigma_{i,2} \leftarrow h^{x_i + y_{i,1} a'} h_{sk}^{y_{i,0}}$
     - $st_i \leftarrow st_i \cup \{id\}$
     - send $\Sigma_{i,2}$ to $\mathcal{U}$ over a secure channel
3. GJoin.U, upon receiving $\Sigma_{i,2}$ from all $\mathcal{I}_i$,
   - $ipk \leftarrow \mathsf{PSM.KAggreg}(pk)$
   - $\Sigma \leftarrow \mathsf{PSM.SAggreg}(pk, sk_{id}, (a', h, \Sigma_{i,2})_{i=1}^{n_I})$
   - $\mathsf{PSM.Vf}(ipk, sk_{id}, \Sigma) \stackrel{?}{=} 1$
   - return $\mathbf{gsk}[id] \leftarrow (sk_{id}, \Sigma)$.

Security.

The anonymity of the scheme holds under the DDH and the SDL assumptions over the group generator if $t_O < n_O/2$. It is not necessary to assume that less than half of the issuers are corrupt since the issuer secret keys can be extracted from their proofs of knowledge.

As for traceability, it holds under the $q_{\mathcal{H}_0}$-MSDH-1, the ADH-KE and the SDL assumptions if at most $n_I - 1$ issuers are corrupt. It is because no colluding $n_I - 1$ PS signers can, with non-negligible probability, compute a multi-signature that is valid w.r.t. the aggregated key of all the n_I signers. The assumption that at most $\min(t_O, n_O - t_O - 1)$ openers are corrupt is still necessary though. Indeed, no more than t_O openers can be corrupt in the simulation of shares of μsk_{id} in the proof that no honest user can be framed by $n_I - 1$ corrupt issuers; and the winning condition still depends on a correct execution of protocol GOpen which requires the non-corrupt registers of at least $t_O + 1$ openers.

Chapter 5

Zone Encryption with Anonymous Authentication

Vehicle-to-vehicle (V2V)communication systems are currently being prepared for real-world deployment, but they face strong opposition over privacy concerns. Position beacon messages are the main culprit, being broadcast in cleartext and pseudonymously signed up to 10 times per second. So far, no practical solutions have been proposed to encrypt or anonymously authenticate V2V messages.

This chapter introduces Zone Encryption to enhance the privacy of V2V communication. In a zone-encryption scheme, vehicles generate and authentically distribute encryption keys associated to static geographic zones close to their location. Zone encryption provides security against eavesdropping, and, combined with the group signatures from Chapter 4 as authentication scheme, ensures that messages can only be sent by genuine vehicles, while adding only 224 Bytes of cryptographic overhead to each message. The group signatures from Chapter 4 are actually augmented with attributes in Section 5.2. In the context of zone encryption, these attributes are only the validity periods of short-term credentials vehicles need to authentically distribute keys. Section 5.3.1 formally defines zone encryption and its security, and Section 5.3.7 proposes an efficient constructions that meet the requirements of C-ITSs. Lastly, Section 5.3.14 addresses the limitations of zone encryption.

Contents

5.1 Preliminaries

This section introduces preliminary material to this chapter.

5.1.1 Deterministic Authenticated Encryption

Deterministic Authenticated Encryption (DAE) [RS06] is a type of symmetric encryption. It is mainly used in the context of key wrapping, i.e., transmitting a secret key from one party to another. It is one of the main cryptographic building blocks of the construction given in Section 5.3.7.

Security Properties.

(Deterministic) Privacy [RS06, Appendix B] is modeled via a distinction experiment. In the real game, a key is chosen uniformly at random and the adversary can make (, without loss of generality, non-repeated) encryption queries. In the ideal game, encryption queries are answered with uniformly random bit strings. A DAE scheme satisfies privacy if no efficient adversary has a non-negligible advantage in distinguishing the real game from the ideal game. (Deterministic) Authenticity is formalized via a key at the beginning of which a key is chosen uniformly at random. The adversary can make (non-repeated) encryption queries, and can submit ciphertexts none of which is the output of a previous encryption query; that is to say, it can make forgery attempts. The adversary wins the game as soon as one of those forgery attempts does not fail. A DAE scheme satisfies authenticity if no efficient adversary has non-negligible advantage in winning this game.

SIV Construction.

Rogaway and Shrimpton constructed a DAE scheme from IV-based encryption schemes and Pseudo-Random Functions (PRFs). They called it the Synthetic-IV (SIV) construction. Their construction requires the ciphertexts of the encryption scheme to be unpredictable if the initialization vector is a uniformly random n-bit string, with n the IV length of the encryption scheme. (The latter property is referred to as privacy of IV-based encryption schemes.) They therefore use a PRF in the SIV construction to compute

the initialization vector from the message and the header, so as to make the IV unpredictable.

Note that the PRF must thereby support vectors of bit strings even though most PRFs in the literature are designed to be computed on a single bit string. Of course, in case a PRF must be computed on a vector of bit strings, the string could be concatenated, but it would incur an important efficiency loss [RS06, Section 5]. Rogaway and Shrimpton consequently proposed a String-to-Vector (S2V) transformation [RS06, Section 5] of string PRFs to PRFs that directly support string vectors without the efficiency loss of trivial solutions.

Formally, the SIV construction is the following. Let $\mathsf{PRF}\colon \mathcal{K}_1 \times \{0,1\}^{**} \to \{0,1\}^n$ be a pseudo-random function with key space $\mathcal{K}_1$ and the set of vectors of bit strings $\{0,1\}^{**}$ as message space. Consider also $(\mathsf{Enc}, \mathsf{Dec})$ (the setup algorithm is omitted) an IV-based encryption scheme with key space $\mathcal{K}_2$. To generate keys for the DAE scheme, generate independently and uniformly at random two keys K_1 and K_2 from $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively. To encrypt, on the input of K_1, K_2, a header H and a message M, compute $IV \leftarrow \mathsf{PRF}(K_1, (H, M))$ then $\mathcal{C} \leftarrow \mathsf{Enc}(K_2, IV, M)$, and output $IV \| \mathcal{C}$. To decrypt a ciphertext $\mathcal{C}$, if it is less than n-bit long, abort. Otherwise parse it as $IV \| \mathcal{C}'$ with IV the first n bits, and compute $\mathsf{Dec}(K_2, IV, \mathcal{C}')$ then $IV' \leftarrow \mathsf{PRF}(K_1, H, M)$. If $IV = IV'$, then output M, otherwise output $\perp$.

Rogaway and Shrimpton proved [RS06, Theorem 2] that if the IV-based scheme $(\mathsf{Enc}, \mathsf{Dec})$ satisfies privacy and if PRF is a pseudo-random function (i.e., such that its outputs are computationally indistinguishable from uniformly random n-bit strings), then the SIV construction satisfies privacy and authenticity.

To instantiate their construction, one can use the S2V transform of a block cipher such as AES in CMAC mode [Dwo16] as PRF and a block cipher in counter (CTR) mode [Dwo07] as IV-based encryption scheme.

5.2 Group Signatures with Attributes

This section introduces an important building block for the zone-encryption protocol, namely *dynamic group signatures with attributes* (DGS+A). These group signatures are simply an extension of those presented in Chapter 4 and constitute the main authentication mechanism later used in the zone-encryption construction of Section 5.3.7. For simplicity, they are defined with a single authority responsible for both issuance and opening. The underlying reason is that zone encryption will itself be formally defined with a single authority to stress its privacy and security aspect rather than the threshold aspect. Of course, DGS+A could also be defined in a threshold setting in the same vein as in Chapter 4.

5.2.1 Definition

In the extension of group signatures with attributes, users obtain membership credentials which are associated to a set of attributes by interacting with an issuer. Signatures, further referred to as *authentication tokens*, are verified w.r.t. those attributes, i.e., a message m can only be signed for a set of attributes A if the signer has a valid membership credential for A. What is called credential here is nothing but a group-secret key according to the terminology in Section 4.2. A similar generalization of group signatures with attributes was already introduced by Camenisch, Neven and Rückert [CNR12], but without interactive credential issuance.

Syntax.

Formally, a DGS+A scheme consists of the following algorithms.

$\mathsf{Setup}(\lambda, aux) \to pp$: Generates public parameters on the input of a security parameter and of auxiliary inputs. These public parameters are assumed to be an implicit input to all the other algorithms.

$\mathsf{KG}(pp) \to (pk, (sk, st))$: A key-generation algorithm for the issuer. It is assumed that pk can be recovered from sk. Variable st represents a state.

$\langle \mathsf{Issue.U}(id, A, pk) \rightleftharpoons \mathsf{Issue.I}(sk, st, id, A) \rangle \to \langle cred, st' \rangle$: A credential-issuance protocol for an attribute set A and user identity id. At the end of the protocol, the user outputs a membership credential $cred$ (or $\perp$ if the protocol fails) and the issuer updates its state to st'. Credential $cred$ is assumed to contain the attributes A.

$\mathsf{Auth}(pk, cred, m) \to tok$: A probabilistic authentication algorithm which signs a message m w.r.t. A and returns tok.

$\mathsf{Vf}(pk, m, A, tok) \to b \in \{0, 1\}$: Returns $b = 1$ if tok is a valid token for message m and attributes A w.r.t. pk.

$\mathsf{Open}(sk, st, m, A, tok) \to id/\perp$: An opening algorithm which allows the issuer to identify the user who generated a valid authentication token. The algorithm returns an identity id or $\perp$.

Correctness & Security Properties.

Similarly to the group signatures in Chapter 4, group signatures should satisfy correctness, anonymity and traceability. The main difference is here that there is only authority in the present case (again, for simplicity) which is assumed to be honest. Another slight distinction is that the adversary in the definition of anonymity and traceability must also specify a challenge

attribute set which also intervenes in the winning conditions. Except for those aspects, the properties of the extension with group signatures are akin to those of the group signatures in Chapter 4.

5.2.2 Construction of Group Signatures with Attributes

The high-level idea of the DGS+A construction is similar to that of the construction of group signatures in Section 4.3. More precisely, to compute a user membership credential as a PS signature on her identity and her (public) attributes. To compute an anonymous authentication token for a message, the user re-randomizes the group elements of her signature and computes a signature of knowledge, on the message, of her signed identity.

To allow for compact authentication tokens yet enable traceability, the issuer maintains a list of the credentials that it generated and traces a token by testing the re-randomized PS signature in the token against each entry. This approach makes tracing more expensive for the benefit of having short tokens, which perfectly fits our application to V2V communication in which bandwidth is limited and tracing an uncommon practice.

Scheme Description. Let G be a type-3 pairing-group generator, $\mathcal{H}$ a random oracle and PS the modified Pointcheval–Sanders signature scheme (Section 4.1.5). Denoting by k the number of attributes of each user, the DGS+A constructionDGSA consists of the following algorithms.

$\mathsf{Setup}(\lambda, k) \to pp$: Generate $\Gamma = (p, \mathbb{G}, \tilde{g}, \mathbb{G}_T, e) \leftarrow \mathsf{G}(\lambda)$. Return $pp \leftarrow (\Gamma, k+1)$.

$\mathsf{KG.I}(pp) \to (pk, (sk, st))$: Generate $\tilde{g} \in_R \tilde{g}^*$, $(x, y_{id}, y_1, \ldots, y_{k+1}) \in_R \mathbb{Z}_p^{k+3}$, compute $\tilde{X} \leftarrow \tilde{g}^x$, $\tilde{Y}_{id} \leftarrow \tilde{g}^{y_{id}}$, and $\tilde{Y}_j \leftarrow \tilde{g}^{y_j}$ for $j = 1, \ldots, k+1$, and return $pk \leftarrow \left(\tilde{g}, \tilde{X}, \tilde{Y}_{id}, \tilde{Y}_1, \ldots, \tilde{Y}_{k+1}\right)$, $sk \leftarrow (pk, x, y_{id}, \ldots, y_{k+1})$ and an intially empty state $st \leftarrow \emptyset$.

Issue : For issuance between a user $\mathcal{U}$ and an issuer $\mathcal{I}$, we assume a secure channel. If a party aborts the protocol, it returns $\perp$. We further assume that the identity space ID is a polynomial-size (in λ) subset of $\mathbb{Z}_p$.

1. $\mathsf{Issue.I}\left(sk, st, id, A = (a_i)_{i=1}^k\right)$,
   - abort if a record $(id, A, *)$ exists in st
   - compute $\sigma = (a', \sigma_1, \sigma_2) \leftarrow \mathsf{PS.Sign}(sk, (id, a_1, \ldots, a_k))$
   - send σ to $\mathcal{U}$ and return $st' \leftarrow st \cup (id, A, a')$
2. $\mathsf{Issue.U}(id, A, pk)$ upon receiving σ from $\mathcal{I}$:
   - verify that $\mathsf{PS.Vf}(pk, (id, A), \sigma) = 1$ and abort if not

- return $cred \leftarrow \left(id, A, \sigma, e\left(\sigma_1, \tilde{Y}_{id}\right), e\left(\sigma_1, \tilde{Y}_{k+1}\right)\right)$. It is actually not necessary to store $e(\sigma_1, \tilde{Y}_{id})$ and $e(\sigma_1, \tilde{Y}_{k+1})$, but it helps avoiding pairing computations when tokens are generated.

$\mathsf{Auth}(pk, cred, m) \rightarrow tok$: Parse $cred = (id,\ A,\ \sigma,\ e\left(\sigma_1, \tilde{Y}_{id}\right), e\left(\sigma_1, \tilde{Y}_{k+1}\right))$ with $A = (a_i)_{i=1}^k$, generate $r \in_R \mathbb{Z}_p^*$, compute $(\sigma_1', \sigma_2') \leftarrow (\sigma_1^r, \sigma_2^r)$ and a non-interactive proof of knowledge π of (id, a') such that

$$e\left(\sigma_1', \tilde{X}\, \tilde{Y}_{id}^{id} \prod_{i=1}^{k} \tilde{Y}_i^{a_i}\, \tilde{Y}_{k+1}^{a'}\right) = e(\sigma_2', \tilde{g}).$$

That is, compute $u \leftarrow e\left(\sigma_1^{rs_{id}}, \tilde{Y}_{id}\right) e\left(\sigma_1^{rs_{a'}}, \tilde{Y}_{k+1}\right)$ for $s_{id}, s_{a'} \in_R \mathbb{Z}_p$, compute a challenge $c \leftarrow h\,(u, A, m, \sigma_1', \sigma_2', pk) \in \mathbb{Z}_p$ and a response $\mathbf{v} \leftarrow (s_{id} - cid, s_{a'} - ca')$. Set $\pi \leftarrow (c, \mathbf{v})$, and return $tok \leftarrow (\sigma_1', \sigma_2', \pi)$.

$\mathsf{Vf}(pk, m, A, tok) \rightarrow b$: Parse $tok = (\sigma_1, \sigma_2, \pi)$ with $\pi = (c, v_{id}, v_{a'})$, $A = (a_i)_{i=1}^k$, and return 1 if $\sigma_1 \neq 1_{\mathbb{G}}$ and $c = h_0\,(u, A, m, \sigma_1, \sigma_2, pk)$ for $u \leftarrow e\left(\sigma_1^{v_{id}}, \tilde{Y}_{id}\right) e\left(\sigma_1^{v_{a'}}, \tilde{Y}_{k+1}\right) e\,(\sigma_2^c, \tilde{g})\, e\left(\sigma_1^c, \tilde{X}^{-1} \prod_{j=1}^{k} \tilde{Y}_j^{-a_j}\right)$.

$\mathsf{Open}(sk, st, m, A, tok) \rightarrow id/\perp$: Recovers the identity id of the user who generated an authentication token $tok = (\sigma_1, \sigma_2, \pi)$ for a message m and attribute set A. It first verifies that tok is valid for m and A. If so, it goes through (in lexicographic order of the identities) the tuples (id, A, a') in st until it finds one such that (a', σ_1, σ_2) is a valid PS signature on (id, A), and then returns id. If no such tuple is found, it returns $\perp$.

Algorithm 5.2.1 Open.

Require: (sk, st, m, A, tok)
Ensure: An identity id or $\perp$.

```
if Vf(pk, m, A, tok) = 0 then
    return ⊥
end if
for all id such that (id, A, a') ∈ st do
    if e(σ1, X̃ Ỹ_id^id ∏_{i=1}^k Ỹ_i^{a_i} Ỹ_{k+1}^{a'}) = e(σ2, g̃) then
        return id
    end if
end for
return ⊥
```

To open a signature on a given message, the opening algorithm loops over all id such that a credential was issued for a tuple (id, A). The complexity of the opening algorithm is then of order $O(|ID|)$. This approach allows to have much shorter group signatures than those obtained with the traditional sign-and-encrypt paradigm. An expensive opening procedure seems appropriate to the case of V2X communication as the issuer should revoke the anonymity of vehicles only on solid grounds.

Efficiency.

With a Cocks–Pinch pairing curve [GMT19] defined over a field of order 2^{544} and with embedding degree 8, group elements in $\mathbb{G}$ take 68 Bytes for a group of 256-bit order. Note that this curve provides 131 bits of security [GMT19].

An authentication token consists of two $\mathbb{G}$ elements and three $\mathbb{Z}_p$ elements, totalling 232 Bytes. The hash value in the proof of knowledge of a multi-signature can actually be shortened to second-preimage resistant length, further shortening a group signature to 216 Bytes.

Application to Zone Encryption. With a token size of 216 Bytes, our pairing-based instantiation is sufficiently compact to be used in combination with our zone-encryption scheme. Therein, tokens are only computed and sent during key requests and responses. Compared to the 160-Byte overhead of ECDSA signatures with certificates, our scheme could even be considered to sign each individual CAM.

Threshold Group Signatures with Attributes.

In the above definition and scheme, the issuer can alone open all tokens, which makes him a single point of failure in terms of privacy. Yet, notice that the group signatures in Chapter 4 can be augmented with attributes as done above, and this would result in threshold scheme in which the issuers and openers are separated.

5.3 Zone Encryption

This section introduces *zone encryption*, a novel mechanism to authentically and confidentially send CAMs between vehicles. It lets a vehicle securely communicate with the other vehicles in its vicinity, encrypting all CAMs. A vehicle can do so only after *anonymously authenticating* itself to the other vehicles.

To authenticate itself, a vehicle uses a short-term credential that it requests at regular intervals from an issuer to whom it authenticates with a

long-term credential. If necessary, the issuer can revoke the anonymity of a vehicle and potentially ban it from the system by revoking its long-term credential.

Overall, the goal is that the V2X communication is authenticated, confidential, i.e., only authorized vehicles can decrypt messages, and that the privacy of vehicles in this communication is preserved. This section starts by describing the high-level concept of zone encryption for V2X communication, then formally defines the desired properties and finally propose a provably secure instantiation.

Geo-Local Shared Keys. The core idea behind zone encryption is to leverage the fact that only vehicles in close proximity need to communicate. More precisely, zone encryption assumes that the surface of the earth is divided into disjoint zones, and lets the vehicles that are present in a particular zone agree on the shared encryption key for that zone. For example, the zone boundaries could be derived statically from the GPS coordinates and are chosen so that the longest straight-line distance within a zone is less than the transmission radius of a radio signal (typically 300–500m), enabling any two vehicles in the same zone to communicate.

Of course, it should be avoided that two vehicles that are physically close but at opposite sides of a zone boundary cannot communicate because they broadcast to different zones. On this account, vehicles broadcast to multiple zones simultaneously.

Short-Lived Zone Keys. The zone keys are also required to be periodically refreshed, e.g., every 15 minutes. This ensures that a rogue eavesdropping device cannot simply stay silent and listen to ongoing traffic, but has to send key requests or responses to other vehicles, exposing itself to detection and localization through triangulation.

Authenticated encryption. Zone encryption takes a significantly different approach for authentication than the existing C-ITS proposals. Instead of signing every CAM with an anonymous authentication scheme, we simply use authenticated symmetric encryption with the short-lived zone keys. Anonymous credential-based authentication is only necessary when a vehicle enters a zone and keys are exchanged in an authenticated manner. Given that each vehicle has to process up to 3000 incoming CAMs per second, relying (mostly) on symmetric primitives instead of asymmetric authentication leads to a significant computational speed-up.

Besides, smart traffic infrastructure that has no need to receive CAMs can be equipped with certificates only for broadcasting authenticated but unencrypted messages (as their content is not privacy sensitive), so that it cannot be abused for mass surveillance.

Identity Resolution & Revocation. In case of dispute or malicious activity in a certain zone at a given time, the messages that each vehicle had to send to receive the zone key can be *opened* by a dedicated entity. The opening algorithm run by this entity reveals the identity of the vehicle that computed a message, which in turn allows to revoke its long-term credential. Recovering the long-term identity of rogue vehicles is commonly known as *identity resolution*. It has been established as an essential requirement to balance the privacy and accountability needs in vehicular communication systems [SMK09, WWKH13, PSFK15]. In the current C-ITS proposal, identity resolution is realized by keeping mappings between pseudonyms and long-term identities [WWKH13, Section IV.D].

For revocation, the approach followed is that of *passive revocation* which is advocated by the European standard [ETS19, Section 6.1.4], meaning that vehicles must regularly request new short-term credentials. These requests are rejected once the corresponding long-term credential has been revoked. Revocation of the long-term credential does not only disable the decryption capabilities of the detected device, but also of any other rogue devices based on the same compromised credential, making mass production of rogue devices less lucrative.

Privacy & Efficiency vs. Sybil Resistance & Non-Repudiation. Zone encryption does pay a price in some other security aspects, though. By relying on symmetric authenticated encryption to authenticate CAMs, neither Sybil resistance nor non-repudiation can be achieved. The former is not a major change since with a pseudonym pool size of up to 100 simultaneously valid certificates, the current proposals essentially gave up on Sybil resistance as well. The loss of non-repudiation should only have minor effects: V2X logs will still be a useful tool to analyze accidents in court, and transmitters of false information can still be uncovered, albeit with slightly more effort, by tracing key requests and responses at the time of the accident. The loss of non-repudiation (which is not a requirement of the standards) is, on this account, a small price to pay for the privacy gains that zone encryption achieves.

5.3.1 Syntax of Zone Encryption Schemes

A *Zone-Encryption* (ZE) scheme allows vehicles in a geographic zone at a given time to securely and anonymously communicate which each other. A ZE scheme features an *enrollment authority* $\mathcal{E}$, an *issuer* $\mathcal{I}$, and *vehicles* with unique identities $\mathcal{V} \in \{0,1\}^*$. The enrollment authority provides vehicles with revocable *long-term credentials* and may in practice be a state authority. A vehicle that has obtained a long-term credential is considered enrolled, and a vehicle identity can be enrolled (only once) in the system at any time.

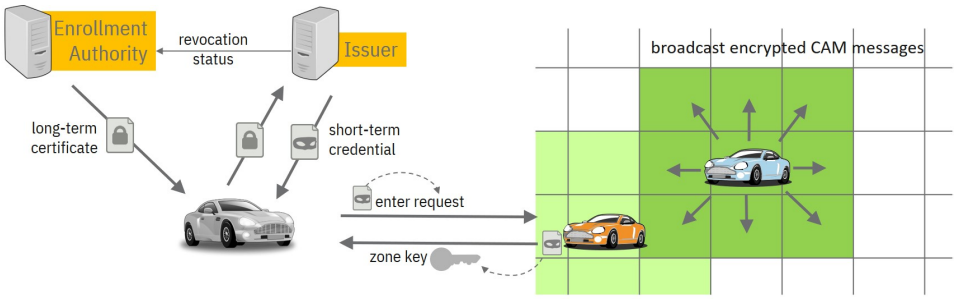

Figure 5.1: Illustration of Zone Encryption with its Anonymous-Authentication Approach.

The long-term credential is used to obtain *short-term credentials* from the issuer (which may in practice be another legal authority or a representative of a car-manufacturer consortium). That is, the time is assumed to be divided into (revocation) *epochs*, and all parties are assumed to be roughly synchronized, i.e., they share a common clock (e.g., a network clock). The duration of an epoch (e.g., a week) is the validity period of short-term credentials, and before the beginning of each epoch, a vehicle must interact with the issuer to obtain the short-term credential. These short-term credentials are irrevocable as they have limited validity anyway. However, the issuer learns the identity $\mathcal{V}$ of the vehicle and can check if its long-term credential has been revoked by the enrollment authority. For the sake of simplicity, we do not explicitly model revocation. As revocation would only be needed for standard, i.e., non-anonymous authentication, this can be added in a straightforward way.

A vehicle being equipped with a short-term credential can, during the epoch for which the credential was issued, communicate with other vehicles in an authenticated yet anonymous manner. More precisely, it uses the short-term credential to exchange so-called *zone keys* with other (anonymously) authenticated vehicles. These keys are valid for a particular zone and short time period, e.g., 15 minutes and enable vehicles to securely send and receive payloads that are encrypted under these keys.

A ZE scheme allows vehicles to communicate anonymously, but if need be, the issuer can recover the identity of the vehicle which computed a certain message. It can then revoke (i.e., blacklist) the vehicle identity and reject its authorization requests in the future.

To formally define a ZE scheme, let Z be a set of zones that cover the road network and let $\mathcal{P}$ be the payload space. Consider also a set of epochs *Epoch* and a set of time periods T, both non-empty finite integer sets such that for all $t \in T$, there exists a unique $e \in \mathit{Epoch}$ for which $e \leq t < e + 1$. Denote it by $e(t)$. These are parameters for the scheme. A ZE scheme then consists of the following algorithms:

Setup & Key Generation. A ZE scheme features an algorithm generating public parameters $\mathsf{Setup}\,(\lambda, Z, \mathit{Epoch}, T) \to \mathit{pp}$, as well as key-generation algorithms $\mathsf{KG.E}(\mathit{pp}) \to (\mathit{pk}_{\mathcal{E}}, (\mathit{sk}_{\mathcal{E}}, \mathit{st}_{\mathcal{E}}))$ and $\mathsf{KG.I}(\mathit{pp}) \to (\mathit{pk}_{\mathcal{I}}, (\mathit{sk}_{\mathcal{I}}, \mathit{st}_{\mathcal{I}}))$ respectively for the enrollment authority and the issuer. The private outputs also contain state $\mathit{st}_{\mathcal{E}}$ and $\mathit{st}_{\mathcal{I}}$ that are used to keep track of enrolled vehicles and open messages sent during key requests. Moreover, we assume that the public keys can be recovered from the secret keys.

Receiving Long-term and Short-term Credentials. A ZE scheme has two interactive protocols for $\mathcal{V}$ to obtain authentication credentials.

$\langle \mathsf{Enroll.V}(\mathit{pk}_{\mathcal{E}}, \mathcal{V}) \rightleftharpoons \mathsf{Enroll.E}(\mathit{sk}_{\mathcal{E}}, \mathit{st}_{\mathcal{E}}, \mathcal{V}) \rangle \to \langle \mathit{cert}_{\mathcal{V}}, \mathit{st}'_{\mathcal{E}} \rangle$: The Enroll protocol is run between a vehicle $\mathcal{V}$ and enrollment authority $\mathcal{E}$. If successful, $\mathcal{V}$ obtains a long-term certificate $\mathit{cert}_{\mathcal{V}}$.

$\langle \mathsf{Authorize.V}(\mathit{cert}_{\mathcal{V}}, e, \mathit{pk}_{\mathcal{I}}) \rightleftharpoons \mathsf{Authorize.I}(\mathit{sk}_{\mathcal{I}}, \mathit{st}_{\mathcal{I}}, \mathcal{V}, e, \mathit{pk}_{\mathcal{E}}) \rangle \to \langle \mathit{cred}_{\mathcal{V}}, \mathit{st}'_{\mathcal{I}} \rangle$: A vehicle $\mathcal{V}$ can use its long-term certificate to obtain from issuer $\mathcal{I}$ a short-term credential $\mathit{cred}_{\mathcal{V}}$ for an epoch e by running protocol Authorize.

Entering and Exiting Zones. Protocol Enter is run when a vehicle $\mathcal{V}$ enters a zone z at time t. It is run with other responding vehicles $\mathcal{W}_i$, all authenticated via their short-term credentials $\mathit{cred}_{\mathcal{W}_i}$. If successful, the protocol allows the entering vehicle $\mathcal{V}$ to obtain the zone key $K_{z,t}$ for the zone–time pair (z, t). Algorithm Exit is used to remove key material from the zone-key list L_K of a vehicle when it exits a zone or when the time period has expired. The latter is crucial for our security model in which a vehicle, *after* leaving a zone, should no longer be able to decrypt messages or compute valid ciphertexts for it.

$\langle \mathsf{Enter.V}(\mathit{cred}_{\mathcal{V}}, L_K, \mathit{pk}_{\mathcal{I}}, z, t, \mathit{requester})$
$\rightleftharpoons \mathsf{Enter.W}(\mathit{cred}_{\mathcal{W}_i}, L_{K_i}, \mathit{pk}_{\mathcal{I}}, z, t, \mathit{responder}_i)_{i \geq 0} \rangle \to \langle L_K, \perp \rangle$: Protocol Enter is run between a requesting vehicle $\mathcal{V}$ and other responding vehicles $\mathcal{W}_i$. List L_K consists of tuples $(z', t', K_{z',t'})$ to which, if the protocol is successful, a new key $K_{z,t}$ for the requested zone–time pair is added.

$\mathsf{Exit}(L_K, z, t) \to L'_K$: Removes $(z, t, K_{z,t})$ from L_K.

Sending and Receiving Payloads. Algorithms Send and Receive are used by a vehicle to exchange encrypted payloads. Note that these algorithms only need to access the zone keys stored in L_K, but not the short-and-long-term credentials, which is a security benefit compared with existing C-ITS solutions.

$\mathsf{Send}(L_K, P, Y \subseteq Z, t) \to \gamma/\perp$: Computes a ciphertext γ for a payload P for all zones Y in time period t (if L_K contains the corresponding keys). The ciphertext γ is assumed to carry public information about t and Y, i.e., it can be parsed as (t, Y, γ').

$\mathsf{Receive}(L_K, \gamma) \to P/\perp$: Recovers the payload P from ciphertext γ if L_K contains a zone key under which γ is encrypted.

Identity Escrow. When suspicious behaviour is detected or when an accident occurs, the issuer $\mathcal{I}$ of the short-term credentials can reveal the identity of a vehicle that sent a given message during an execution of protocol Enter.

$\mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m) \to \mathcal{V}/\perp$**:** returns the identity of a vehicle that it identifies as the sender of a message m during an execution of protocol Enter, or $\perp$.

Note that Open runs on a single anonymous message sent during an execution of protocol Enter, not on a full record of all messages ever sent by vehicles. It means that in practice, in case of a dispute or a suspicious event in a certain zone the issuer only needs to de-anonymize the messages sent during executions of protocol Enter for that zone at the time (period) of the event.

Correctness of Zone Encryption. A ZE scheme should satisfy *correctness*, i.e., if a vehicle is authorized during a given epoch and has entered a zone in a certain time period, then every message sent by that vehicle to this zone should be successfully received by any other vehicle in the zone in that time period. Moreover, the identity of a vehicle that sent a given message during an Enter protocol execution should be recoverable by the issuer. These properties should hold independently of the order in which certificates and credentials are issued for vehicles, and with overwhelming probability.

5.3.2 Security of Zone Encryption Schemes

This section describes the security and privacy properties a ZE scheme must satisfy.

The payloads sent by the vehicles should be confidential. This property is formalized as **Payload-Hiding security against Chosen-Ciphertext Attacks** (PH-CCA). Intuitively, PH-CCA security ensures that no efficient adversary can infer any information about the payload underlying a ciphertext, unless it has entered the zone in the time period of the ciphertext.

The privacy of vehicles should also be preserved, and this requirement is defined through an **anonymity** game. Essentially, anonymity guarantees

that ciphertexts and enter-protocol messages do not reveal any information about the identity of the sending vehicle.

Note that there is no anonymity requirement for the authorization process, i.e., for receiving short-term credentials, as it is performed once per epoch (e.g., a week) and leaks very little information about the whereabouts of the vehicles. It is not an issue assuming that users have control on when it occurs.

Furthermore, despite strong privacy properties, zone-encryption should ensure that only legitimate vehicles can send valid ciphertexts. This is captured via two related security definitions.

First, the **traceability** notion guarantees that if a vehicle knows a key $K_{z,t}$ for zone z at time t, then it must have explicitly entered the zone z at time t, meaning that it must have sent an enter message that can be traced back by the issuer to its long-term identity.

Secondly, the related notion of **ciphertext integrity** guarantees that an adversary cannot compute a valid ciphertext γ for a particular zone–time pair without knowing the zone key.

Common Oracles.

We first introduce the oracles we give the adversary in all our security games. In the formal definitions, $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$ denotes that the adversary is given access to oracles {Enroll.E, Enroll.V&E, Authorize.I, Authorize.V&I, Enter, Exit, Send, Receive, Open, Corrupt} as defined hereunder and initialized with secret keys $sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}}$. The public keys $pk_{\mathcal{E}}, pk_I$ are not made explicit, but are assumed to be recoverable from the corresponding secret keys.

Throughout the security experiments, the challenger maintains several lists which reflect the information $\mathcal{A}$ learns through his interaction with the oracles. These are summarized on Figure 5.2.

$\mathcal{L}_{\mathsf{honest}}$	list of all enrolled honest vehicles $\{(\mathcal{V})\}$
$\mathcal{L}_{\mathsf{corrupt}}$	enrolled vehicles $\{(\mathcal{V})\}$ that where corrupt either from the beginning or later on
$\mathcal{L}_{\mathsf{auth}}$	authorized vehicles $\{(\mathcal{V}, e)\}$ per epoch e
$\mathcal{L}_{\mathsf{enter}}$	contains all messages $\{(\{\mathcal{V}, \mathcal{A}\}, z, t, m)\}$ that honest vehicles or the adversary $\mathcal{A}$ exchanged during executions of protocol Enter
$\mathcal{L}_{\mathsf{sent}}$	ciphertexts $\{\gamma\}$ generated by honest vehicles
$\mathcal{L}_{\mathsf{received}}$	decrypted ciphertexts $\{\gamma = (t, Y, \gamma')\}$
$\mathcal{L}_{\mathsf{opened}}$	opened transcripts m
$\mathcal{L}_{\mathsf{keys}}$	zone-keys $\{(z, t, K_{t,z})\}$ that the adversary $\mathcal{A}$ learned by corrupting honest vehicles

Figure 5.2: Lists maintained by the Challenger in the ZE Security Experiments.

Notation. "$\mathcal{O}$.algorithm.P" denotes the oracle which lets the adversary interact with honest party $\mathcal{P}$ running algorithm.P. Similarly, an oracle "$\mathcal{O}$.algorithm.P&R" allows the adversary to trigger the interactive protocol $\langle$algorithm.P $\rightleftharpoons$ algorithm.R$\rangle$ between two honest parties $\mathcal{P}$ and $\mathcal{R}$. In the latter case, the adversary does not learn the outputs of honest parties, but their internal states are updated accordingly. Moreover, when an oracle is said to be running an algorithm on behalf of an *honest* vehicle $\mathcal{V}$, it is implicitly assumed that the oracle checks that $\mathcal{V} \in \mathcal{L}_{\mathsf{honest}}$. Finally, the state of an honest vehicle $\mathcal{V}$ is referred to as $\mathcal{V}[st_{\mathcal{V}}]$, e.g., $\mathcal{V}[L_K]$ denotes the zone keys L_K maintained by $\mathcal{V}$.

Oracles for Obtaining Credentials. There are a number of oracles to model enrollment and issuance of short-term credentials, depending on whether the requesting vehicle is honest or corrupt.

$\mathcal{O}$.Enroll.V&E$(sk_{\mathcal{E}}, st_{\mathcal{E}}, \cdot)$ on input $\mathcal{V}$, lets the adversary trigger the enrollment protocol between an honest vehicle with identity $\mathcal{V}$ and the honest enrollment authority $\mathcal{E}$. If Enroll.V ends with a private output $\mathcal{V}[cert]$, it adds $\mathcal{V}$ to $\mathcal{L}_{\mathsf{honest}}$.

$\mathcal{O}$.Enroll.E$(sk_{\mathcal{E}}, st_{\mathcal{E}}, \cdot)$ on input $\mathcal{V}$, lets the adversary run an enrollment protocol (in the role of the corrupt vehicle $\mathcal{V}$) with the honest enrollment authority. If Enroll.E ends with a private output $st'_{\mathcal{E}}$, it adds $\mathcal{V}$ to $\mathcal{L}_{\mathsf{corrupt}}$.

$\mathcal{O}$.Authorize.V&I$(sk_{\mathcal{I}}, st_{\mathcal{I}}, \cdot)$ on input $(\mathcal{V}, e)$, triggers an Authorize protocol between the honest vehicle $\mathcal{V}$ and honest issuer $\mathcal{I}$. If Authorize.V ends with private output $\mathcal{V}[e, cred_{\mathcal{V}}]$, it adds $(\mathcal{V}, e)$ to $\mathcal{L}_{\mathsf{auth}}$.

$\mathcal{O}$.Authorize.I$(sk_{\mathcal{I}}, st_{\mathcal{I}}, \cdot)$ on input $(\mathcal{V}, e)$, allows a corrupt vehicle $\mathcal{V}$, played by the adversary, to run the Authorize protocol with the honest issuer $\mathcal{I}$. If Authorize.I ends with private output $st'_{\mathcal{I}}$, it adds $(\mathcal{V}, e)$ to $\mathcal{L}_{\mathsf{auth}}$.

Oracles for Entering and Exiting Zones. The adversary is further given access to an oracle which lets it actively participate in the Enter protocol as well as eavesdrop on enter-protocol executions between honest vehicles. Another oracle lets the adversary make an honest vehicle exit a zone.

$\mathcal{O}$.Enter$(\cdot)$ on input $(\mathcal{V}, z, t, role)$, triggers a zone-key request or response protocol (according to *role*) for a honest vehicle $\mathcal{V}$ in zone z and time period t.

- For $role = requester$, the oracle starts Enter.V for $\mathcal{V}$ in the *requester* role and also internally invokes all other honest vehicles $\mathcal{W}_i$ which have zone keys for (z, t) to run Enter.W with $role = responder$.

The adversary can intercept and inject messages sent by these honest vehicles, and also participate in the *responder* role with a corrupt vehicle. Eventually, the key state $\mathcal{V}[L_K]$ of the honest requester is updated to include $K_{z,t}$.

- For $role = responder$ the oracle lets an honest vehicle $\mathcal{V}$ respond to a zone-key request that the adversary runs for a corrupt vehicle.

All messages $(\mathcal{V}, z, t, m)$ sent by honest vehicles $\mathcal{V}$ are tracked with list $\mathcal{L}_{\mathsf{enter}}$. Similarly, when an honest vehicle receives a message m that no other honest vehicle has sent, the message is recorded as adversarial by adding $(\mathcal{A}, z, t, m)$ to $\mathcal{L}_{\mathsf{enter}}$.

Note that this oracle captures both *active* and *passive* attacks. The latter can be done if the adversary queries $\mathcal{O}.\mathsf{Enter}$ for $role = requester$ and does not participate as corrupt responder or manipulates messages, but merely observes the traffic between the honest vehicles in a certain zone and time period (z, t). There is then no message $(\mathcal{A}, z, t, m) \in \mathcal{L}_{\mathsf{enter}}$; and $\mathcal{A}$ is considered successful if it infers information for (z, t) that is supposed to only be known to vehicles which entered the zone, e.g., if it manages to distinguish ciphertexts encrypted for (z, t) (PH-CCA) or if it can produce a valid ciphertext (ciphertext integrity).

$\mathcal{O}.\mathsf{Exit}(\cdot)$ on input $(\mathcal{V}, z, t)$, deletes key $K_{z,t}$ from the key state of the honest vehicle $\mathcal{V}$.

Oracles for Sending and Receiving Payloads, Opening and Corruption. Finally, the adversary is given access to oracles that can trigger honest vehicles to encrypt or decrypt messages, recover the identity of sending vehicles, and adaptively corrupt vehicles.

$\mathcal{O}.\mathsf{Send}(\cdot)$ on input $(\mathcal{V}, P, Y, t)$ for an honest vehicle $\mathcal{V}$, returns $\gamma/\perp \leftarrow \mathsf{Send}(\mathcal{V}[L_K], P, Y, t)$ and adds γ to $\mathcal{L}_{\mathsf{sent}}$.

$\mathcal{O}.\mathsf{Receive}(\cdot)$ on input $(\mathcal{V}, \gamma)$ for an honest vehicle $\mathcal{V}$, returns $m/\perp \leftarrow \mathsf{Receive}(\mathcal{V}[L_K], \gamma)$ and adds γ to $\mathcal{L}_{\mathsf{received}}$.

$\mathcal{O}.\mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, \cdot)$ on input m, returns $\mathcal{V}/\perp \leftarrow \mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m)$ and adds m to $\mathcal{L}_{\mathsf{opened}}$.

$\mathcal{O}.\mathsf{Corrupt}(\cdot)$ on input $\mathcal{V}$, returns the current state of the honest vehicle $\mathcal{V}$, i.e., it returns $\mathcal{V}[cert_{\mathcal{V}}]$, all $\mathcal{V}[\{(e_j, cred_{\mathcal{V},j})\}]$, and $\mathcal{V}[L_K]$ to the adversary. It also adds $\mathcal{V}$ to $\mathcal{L}_{\mathsf{corrupt}}$ and all keys $(z, t, K_{z,t})$ in $\mathcal{V}[L_K]$ to $\mathcal{L}_{\mathsf{keys}}$.

Payload Hiding.

Payload-hiding security against chosen-ciphertext attacks (see Figure 5.3) guarantees that an adversary cannot infer any information about messages encrypted for a zone it is not supposed to be in. The definition follows the classical CCA definition and requires the adversary to output two payloads P_0, P_1 together with a time t^* and zones Y^*, upon which it receives the zone encryption of P_b for a random $b \in \{0,1\}$. The adversary must then determine b better than by guessing. The adversary is given access to honest participants in the system, e.g., by allowing it to enroll and authorize vehicles, enter zones, encrypt and decrypt messages of its choice, and to corrupt vehicles.

The adversary wins as long as its interactions with these oracles do not lead to a trivial win. Clearly, the adversary is not allowed to query the decryption oracle $\mathcal{O}.\mathsf{Receive}$ on (parts of) the challenge ciphertext (condition 1), or corrupt an honest vehicle that has a zone-key for one of the challenge zones in Y^* at time t^* (condition 2). Furthermore, the adversary must not have entered any challenge zone at time t^* with a corrupt vehicle, or if it did, it must not have a valid authorization credential for epoch $e(t^*)$ (condition 3). The latter condition is crucial as the PH-CCA notion should guarantee message confidentiality for all zones the adversary was not *supposed* to be in.

Experiment $\mathbf{Exp}^{\text{ph-cca-}b}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(\mathcal{V}^*, P_0, P_1, Y^*, t^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{choose}, pp, pk_{\mathcal{E}}, pk_{\mathcal{I}})$
abort if $\mathcal{V}^* \in \mathcal{L}_{\mathsf{corrupt}}$

$\gamma^* \leftarrow \mathsf{Send}(\mathcal{V}^*[L_K], P_b, Y^*, t^*)$ with $\gamma^* = (t^*, Y^*, \gamma^{*'})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, \gamma^*, st_{\mathcal{A}})$
return b' if $\mathcal{A}$ did not trivially win, i.e.,
1) $\forall (t^*, Y, \gamma) \in \mathcal{L}_{\mathsf{received}} : \gamma \cap \gamma^{*'} = \emptyset$ and

2) $\forall y^* \in Y^* : (y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{keys}}$, i.e., *$\mathcal{A}$ has not corrupted a vehicle in a challenge zone*, and

3a) $\forall y^* \in Y^* : ((\mathcal{A}, y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{enter}})$ **or**

3b) $\exists (\mathcal{A}, y^*, t^*, \cdot) \in \mathcal{L}_{\mathsf{enter}}$ and $\forall \mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}, \nexists (\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}$
i.e., *$\mathcal{A}$ has not entered a challenge zone in time t^* or entered but was not authorized*

Figure 5.3: PH-CCA Experiment for ZE Schemes.

Definition 5.3.3 (PH-CCA Security)**.** *A ZE scheme $\mathcal{Z}$ is PH-CCA secure if there exists a negligible function* negl *such that for all efficient adversary*

$\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,

$$\left|\Pr\left[\mathbf{Exp}^{\text{ph-cca-0}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] - \text{pr}\left[\mathbf{Exp}^{\text{ph-cca-1}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Anonymity.

Anonymity (see Figure 5.4) captures the idea that ZE ciphertexts and the messages sent during executions of protocol Enter do not reveal any information about the identity of the sending vehicle. This includes *unlinkability*, i.e., the adversary cannot tell whether two ciphertexts or two enter-protocol messages stem from the same vehicle.

The definition follows the indistinguishability style, and it grants the adversary oracle access to honest participants. In particular, the adversary can enter and exit zones with honest vehicles, as well as send payloads and receive ciphertexts with them. The adversary must eventually output two challenge vehicle identities $\mathcal{V}_0$ and $\mathcal{V}_1$, after which it gets access to vehicles $\mathcal{V}_b$ and $\mathcal{V}_{1-b}$ and has to determine b. In the experiment, it is captured by turning all oracles that should not leak information about the vehicles identity into challenge oracles. That is, the oracles to enter and exit zones, or to send payloads and receive ciphertexts are restricted to no longer respond to queries for identities $\mathcal{V}_0$ or $\mathcal{V}_1$. If the adversary wants to make such a query for either of them, it has to provide a bit d and the query is answered with vehicle $\mathcal{V}_{d\oplus b}$, i.e., either the chosen vehicle $\mathcal{V}_b$ (for $d = 0$) or its counterpart $\mathcal{V}_{1-b}$ (for $d = 1$).

To avoid trivial wins, the oracles to enter and exit zones cannot be queried at a time t for a challenge vehicle if $\mathcal{V}_0$ and $\mathcal{V}_1$ have not *both* been authorized in epoch $e(t)$. Besides, the adversary can never open a message sent by one of the challenge vehicles during an execution of protocol Enter.

Note that this definition does not require the authorization protocol to be anonymous, but only ZE ciphertexts and messages exchanged during executions of protocol Enter. As authorization is performed only once per epoch, it is not critical for the privacy guarantees we aim for in V2X communication.

Definition 5.3.4 (Anonymity)**.** *A ZE scheme $\mathcal{Z}$ satisfies anonymity if there exists a negligible function* negl *such that for all efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\left|\Pr\left[\mathbf{Exp}^{\text{ano-0}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ano-1}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Traceability.

The notion of traceability ensures that if a vehicle knows a secret key $K_{z,t}$ for a zone–time pair, then it must have entered the zone–time pair by sending a message that can be traced back to the sending vehicle. This is captured via

Experiment $\mathbf{Exp}^{\text{ano-b}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(\mathcal{V}_0, \mathcal{V}_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{choose}, pp, pk_{\mathcal{E}}, pk_{\mathcal{I}})$
abort if the vehicles have different cred/key states, i.e., check that:
for $d \in \{0,1\}$: $\nexists e_i$ s.t. $(\mathcal{V}_d, e_i) \in \mathcal{L}_{\mathsf{auth}}$ and $(\mathcal{V}_{1-d}, e_i) \notin \mathcal{L}_{\mathsf{auth}}$ and $\mathcal{V}_0[L_K] = \mathcal{V}_1[L_K]$

use challenge oracles $\mathcal{O}^*(b)$ for $\{\mathsf{Enter}^*, \mathsf{Exit}^*, \mathsf{Send}^*, \mathsf{Receive}^*\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^*}(\mathsf{guess}, st_{\mathcal{A}})$
return b' if $\mathcal{A}$ did not trivially win, i.e.,
for $d \in \{0,1\} : \mathcal{V}_d \in \mathcal{L}_{\mathsf{honest}}$ and $\forall m \in \mathcal{L}^*_{\mathsf{enter}} : m \notin \mathcal{L}_{\mathsf{opened}}$

Figure 5.4: Anonymity Experiment for ZE Schemes.

a game (on Figure 5.5) where the adversary must output a key K_{z^*,t^*} for a zone z^* and time t^* of its choice. The adversary wins if at least one honest vehicle $\mathcal{V}$ has accepted the key, but none of the messages in executions of protocol Enter for (z^*, t^*) can be traced with algorithm Open to a corrupt vehicle (that was authorized to enter). To avoid trivial wins we further request that the adversary has not corrupted an honest vehicle that held the key K_{z^*,t^*} output by the adversary (condition 1). Moreover, no corrupt vehicle can be authorized in epoch $e(t^*)$ (conditions 2) as otherwise the adversary would be able to impose a zone key.

In particular, for a ZE scheme that satisfies traceability, if an efficient adversary knows the zone key of an honest vehicle, then it has either corrupted another honest vehicle in the zone, or it must have sent at least one message that traces back to a corrupt vehicle that was authorized in epoch $e(t^*)$.

Experiment $\mathbf{Exp}^{\text{trace}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(z^*, t^*, K_{z^*,t^*}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{forge}, pp, pk_{\mathcal{E}}, pk_{\mathcal{I}})$
look up $\mathcal{V} \in \mathcal{L}_{\mathsf{honest}}$ with $K_{z^*,t^*} \in \mathcal{V}[L_K]$, abort if no such $\mathcal{V}$ exists

return 1 if knowledge of K_{z^*,t^*} cannot be traced to a corrupt vehicle:
1) $K_{z^*,t^*} \notin \mathcal{L}_{\mathsf{keys}}$ and 2) $\forall(\cdot, z^*, t^*, m_j) \in \mathcal{L}_{\mathsf{enter}}$ with $\mathcal{V}_j \leftarrow \mathsf{Open}(sk_I, st_I, m_j)$:
$\mathcal{V}_j \notin \mathcal{L}_{\mathsf{corrupt}}$ or $((\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}})$ and $(\nexists(\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}))$

Figure 5.5: Traceability Experiment for ZE Schemes.

Definition 5.3.5 (Traceability)**.** *A ZE scheme $\mathcal{Z}$ satisfies traceability if there exists a negligible function* negl *such that for all efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\Pr\left[\mathbf{Exp}^{\text{trace}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] \leq \mathsf{negl}(\lambda).$$

Ciphertext Integrity.

The notion of ciphertext integrity (see Figure 5.6) complements the traceability property as it guarantees that without knowing a secret zone key $K_{z,t}$ an adversary should not be able to compute a valid ciphertext for that zone and time. This is modeled by asking the adversary to produce a fresh and valid ciphertext γ^* for zones for which it is not supposed to know the keys. The adversary also outputs an honest vehicle $\mathcal{V}$ that must decrypt γ^* to $P \neq \perp$.

Freshness means that γ^* should not contain any honestly generated ciphertexts (or parts thereof) the adversary has received via oracle $\mathcal{O}.\mathsf{Send}$ (condition 1). Moreover, the same conditions as in the PH-CCA game are used to check that the adversary is *not supposed* to know the key. That is, the adversary must not have corrupted an honest vehicle that knows a valid key for the forged ciphertext (condition 2). Besides, it must not have entered any challenge zone at time t^* with a corrupt vehicle, or if it entered, it must not have a valid authorization credential for epoch $e(t^*)$ (condition 3). The last condition means that the adversary *does* win the game if it knows the key of a zone–time pair it was *not allowed* to enter.

Experiment $\mathbf{Exp}^{\text{integrity}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(\mathcal{V}, \gamma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{forge}, pp, pk_{\mathcal{E}}, pk_I)$
parse $\gamma^* = (t^*, Y^*, \gamma^{*'})$, abort if $\mathcal{V} \in \mathcal{L}_{\mathsf{corrupt}}$

return 1 if $\mathsf{Receive}(\mathcal{V}[L_K], \gamma^*) \neq \perp$ and $\mathcal{A}$ did not trivially win, i.e.,
1) $\forall (t^*, Y, \gamma) \in \mathcal{L}_{\mathsf{sent}} : \gamma \cap \gamma^{*'} = \emptyset$ and

2) $\forall y^* \in Y^* : (y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{keys}}$ and
i.e., *$\mathcal{A}$ has not corrupted a vehicle in a challenge zone*

3a) $\forall y^* \in Y^* : ((\mathcal{A}, y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{enter}})$ **or**

3b) $\exists (\mathcal{A}, y^*, t^*, \cdot) \in \mathcal{L}_{\mathsf{enter}}$ and $\forall \mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}} : \nexists (\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}$
i.e., *$\mathcal{A}$ has not entered a challenge zone in time t^* or entered but was not authorized*

Figure 5.6: Ciphertext-Integrity Experiment for ZE Schemes.

Definition 5.3.6 (Ciphertext Integrity)**.** *A ZE scheme $\mathcal{Z}$ satisfies ciphertext-integrity if there exists a negligible function* negl *such that for every efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\Pr\left[\mathbf{Exp}^{\text{integrity}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] \leq \mathsf{negl}(\lambda).$$

Zone Encryption with Multiple Authorities.

The previous definitions feature a single authorization authority that is responsible for both issuing vehicle credentials and revoking their anonymity. The definitions could of course be readily extended to support multiple, separate authorities for each role and require a threshold number of them for each task, following the same ideas as in Chapter 4. For simplicity, the definitions were kept with a single authority to highlight the aspects other than the threshold one and ease their readability.

5.3.7 Construction of a Zone-Encryption Scheme

This section gives a generic zone-encryption scheme constructed from authenticated encryption, public-key encryption, digital signatures and a dynamic group signature scheme with attributes. For the latter, see the practical, pairing-based instantiation in Section 5.2.2. The ZE scheme involves the following building blocks:

- SIG a signature scheme to generate long-term credentials and thereby certify vehicle identities
- DGSA a group-signature scheme (Section 5.2.1) used to compute short-term authentication credentials. Group-membership credentials are issued w.r.t. the current time epoch $e(t)$
- PKE a public-key encryption scheme to encrypt zone keys during key requests and responses
- SE a symmetric-key encryption scheme to encrypt payloads
- DAE a deterministic authenticated encryption scheme to wrap payload keys with each zone key, and thereby bind payload ciphertexts to their zones.

The reason symmetric encryption is used to encrypt payloads and only authenticate key wraps is that payloads may a priori be long. Authenticating the payload part of the ciphertext would increase its length. Only authenticating the key wraps and bind the payload part to them results in shorter ciphertext.

Formal Description.

Recall that each short-term credential is only valid in a certain epoch (e.g., a week), and that zone keys must be refreshed at the beginning of every time period (e.g., every 15 minutes). It is assumed that, during protocol executions, whenever an algorithm receives an abort or invalid message, or a verification step fails, it aborts by returning $\perp$. Likewise, when an algorithm must retrieve a key from its internal state, it aborts if no such key can be found.

The ZE scheme $\mathcal{Z}$, parametrized by a zone set Z, an epoch set *Epoch* and a time-period set T, is defined as follows.

Setup & Key Generation. The setup and key generation algorithms simply run the respective algorithms of the building blocks and generate the keys and parameters accordingly.

$\mathsf{Setup}\,(\lambda, Z, \mathit{Epoch}, T)$: Generate public parameters for SIG, DGSA (with one attribute), PKE, SE and DAE and return $pp \leftarrow (pp_{\mathsf{SIG}}, pp_{\mathsf{DGSA}}, pp_{\mathsf{PKE}}, pp_{\mathsf{SE}}, pp_{\mathsf{DAE}}, Z, \mathit{Epoch}, T)$.

$\mathsf{KG.E}(pp)$: Run $(vk, sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$, set keys as $(pk_{\mathcal{E}}, sk_{\mathcal{E}}) \leftarrow (vk, sk)$, $st_{\mathcal{E}} \leftarrow \emptyset$, and return the tuple $(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}}))$.

$\mathsf{KG.I}(pp)$: Run and return $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{DGSA.KG}(pp_{\mathsf{DGSA}})$.

Issuance of Long-Term and of Short-Term Credentials. To enroll in the communication system, a vehicle with identity $\mathcal{V}$ must request a long-term certificate, which is simply a signature on $\mathcal{V}$ by the enrollment authority. From then on, it can request short-term credentials at the beginning of each epoch from issuer $\mathcal{I}$. Credentials are DGS+A membership credentials for an epoch e as attribute, and are used to authenticate vehicles during protocol Enter.

$\mathsf{Enroll.V} \rightleftharpoons \mathsf{Enroll.E}$: The vehicle and the EA proceed as follows.

1. $\mathsf{Enroll.V}(pk_{\mathcal{E}}, \mathcal{V})$:
   - $(vk_{\mathcal{V}}, sk_{\mathcal{V}}) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$, send $(\mathcal{V}, vk_{\mathcal{V}})$ to $\mathcal{E}$
2. $\mathsf{Enroll.E}(sk_{\mathcal{E}}, st_{\mathcal{E}}, \mathcal{V})$ upon receiving $(\mathcal{V}, vk_{\mathcal{V}})$:
   - check that $\mathcal{V} \notin st_{\mathcal{E}}$ (to ensure that a vehicle identity can be enrolled only once), send a signature $\sigma_{\mathcal{E}} \leftarrow \mathsf{SIG.Sign}(sk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}))$ to $\mathcal{V}$ and return $st'_{\mathcal{E}} \leftarrow st_{\mathcal{E}} \cup \mathcal{V}$
3. $\mathsf{Enroll.V}$ upon receiving $\sigma_{\mathcal{E}}$ from $\mathcal{E}$:
   - if $\mathsf{SIG.Vf}(pk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}), \sigma_{\mathcal{E}}) = 1$, return $\mathit{cert}_{\mathcal{V}} \leftarrow (sk_{\mathcal{V}}, vk_{\mathcal{V}}, \sigma_{\mathcal{E}})$.

$\mathsf{Authorize.V} \rightleftharpoons \mathsf{Authorize.I}$:

1. $\mathsf{Authorize.V}(cert_{\mathcal{V}}, e, pk_{\mathcal{I}})$ with $cert_{\mathcal{V}}$ parsed as $(sk_{\mathcal{V}}, vk_{\mathcal{V}}, \sigma_{\mathcal{E}})$:
   - compute $\sigma_{\mathcal{V}} \leftarrow \mathsf{SIG.Sign}(sk_{\mathcal{V}}, e)$
   - send $(vk_{\mathcal{V}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{V}})$ to $\mathcal{I}$
2. $\mathsf{Authorize.I}(sk_I, st_{\mathcal{I}}, \mathcal{V}, e, pk_{\mathcal{E}})$ upon receiving $(vk_{\mathcal{V}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{V}})$:
   - abort if $\mathcal{V}$ is revoked (this is handled outside the scheme)
   - test whether $\mathsf{SIG.Vf}(pk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}), \sigma_{\mathcal{E}}) = 1$ and $\mathsf{SIG.Vf}(vk_{\mathcal{V}}, e, \sigma_{\mathcal{V}}) = 1$
3. $\mathcal{I}$ and $\mathcal{V}$ run the issuance protocol of DGSA with $id = \mathcal{V}$ and attribute $A = e$, i.e., run

$$\langle \mathsf{DGSA.Issue.U}(\mathcal{V}, e, pk_{\mathcal{I}}) \rightleftharpoons \mathsf{DGSA.Issue.I}(sk_{\mathcal{I}}, st_{\mathcal{I}}, \mathcal{V}, e) \rangle$$

   and return their respective outputs *cred* and $st'_{\mathcal{I}}$.

Entering and Exiting Zones. A vehicle which approaches a zone z in time period t obtains the key for (z, t) by sending an anonymously authenticated key request wich includes a fresh public-key encryption key ek. Any vehicle which receives the request and knows the zone key $K_{z,t}$ can send an anonymously authenticated response which contains an encryption of $K_{z,t}$ under ek. The tokens authenticate messages consisting of z, t, and the fresh key ek for requests or encryptions C under ek of the zone key $K_{z,t}$.

If the requesting vehicle receives no response, it generates a random key $K_{z,t}$ and waits for requests from new vehicles that join the zone.

A vehicle determines whether it should reply according to a predetermined strategy, e.g., the vehicle closest to the requesting vehicle should reply to the key request. The optimal strategy depends on the zone structure, the traffic and other practical factors, and it is an engineering problem on its own. The assumption is here that such a pre-established key-response strategy exists among all vehicles. Finally, once the time period t has elapsed, $\mathcal{V}$ simply deletes $K_{z,t}$ from its internal key state.

$\mathsf{Enter.V} \rightleftharpoons [\mathsf{Enter.W}_i]$: The inputs are assumed to be well-formed, i.e., credentials $cred_{\mathcal{V}}$ and $cred_{\mathcal{W}_i}$ are valid for epoch $e(t)$.

1. $\mathcal{V}$ running $\mathsf{Enter.V}(cred_{\mathcal{V}}, L_K, pk_{\mathcal{I}}, z, t, requester)$:
   - return L_K if $\exists\ (z, t, K_{z,t}) \in L_K$
   - $(ek, dk) \leftarrow \mathsf{PKE.KG}(pp_{\mathsf{PKE}})$
   - $tok_{\mathcal{V}} \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{V}}, (z, t, ek))$
   - broadcast $(z, t, ek, tok_{\mathcal{V}})$
2. $\mathcal{W}_i$ running $\mathsf{Enter.W}(cred_{\mathcal{W}_i}, L_{K_i}, pk_{\mathcal{I}}, z, t, responder_i)$ upon receiving $(z, t, ek, tok_{\mathcal{V}})$ from a vehicle $\mathcal{V}$:

- verify that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, ek), e(t), tok_{\mathcal{V}}) = 1$
- retrieve $(z, t, K_{z,t}) \in L_{K_i}$
- $C \leftarrow \mathsf{PKE.Enc}(ek, K_{z,t})$
- $tok_{\mathcal{W}} \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{W}_i}, (z, t, C))$
- send $(z, t, C, tok_{\mathcal{W}})$ to vehicle $\mathcal{V}$

(3a) Vehicle $\mathcal{V}$ upon receiving $(z, t, C, tok_{\mathcal{W}})$ from a vehicle $\mathcal{W}_i$:

- verify that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, C), e(t), tok_{\mathcal{W}}) = 1$
- decrypt $K_{z,t} \leftarrow \mathsf{PKE.Dec}(dk, C)$, return $L_K \leftarrow L_K \cup (z, t, K_{z,t})$

(3b) If $\mathcal{V}$ does not receive a response after a predetermined waiting time:

- $K_{z,t} \leftarrow \mathsf{DAE.KG}(\lambda)$, return $L_K \leftarrow L_K \cup (z, t, K_{z,t})$

$\mathsf{Exit}(L_K, z, t)$: If $(z, t, K_{z,t}) \in L_K$ return $L'_K \leftarrow L_K \backslash (z, t, K_{z,t})$.

Sending and Receiving Payloads. To encrypt CAMs, referred to as payloads in the construction, a vehicle generates a fresh symmetric key K_P and encrypts the payload with it. It then wraps the payload key with the key $K_{z,t}$ of each of the zones to which it intends to send the CAM. By using a deterministic[1]authenticated encryption scheme [RS06] to wrap fresh payload keys, it is guaranteed that the message originates from a genuine vehicle, as it had to authenticate itself to obtain the zone key. The need for a separate signature on each CAM message is thus eliminated, yielding considerable savings in terms of computation. In addition to that, the authentication provided by scheme DAE is extended to the payload ciphertext C by including C as the header when K_P is encrypted under the zone keys.

$\mathsf{Send}(L_K, P, Y \subseteq Z, t)$: to send a payload P,

1. retrieve keys $\{K_{y,t}\}$ for all zones $y \in Y$ and time t from L_K
2. $C \leftarrow \mathsf{SE.Enc}(K_P, P)$ with $K_P \leftarrow \mathsf{SE.KG}(pp_{\mathsf{SE}})$
3. for all $y \in Y$: $\gamma_{y,t} \leftarrow \mathsf{DAE.Enc}(K_{y,t}, C, K_P)$
4. return $\gamma \leftarrow (t, Y, ((y, \gamma_{y,t})_{y \in Y}, C))$

1. retrieve keys $\{K_{y,t}\}$ for all zones $y \in Y$ and time t from L_K
2. $C \leftarrow \mathsf{SE.Enc}(K_P, P)$ with $K_P \leftarrow \mathsf{SE.KG}(pp_{\mathsf{SE}})$
3. for all $y \in Y$: $\gamma_{y,t} \leftarrow \mathsf{DAE.Enc}(K_{y,t}, C, K_P)$
4. return $\gamma \leftarrow (t, Y, ((y, \gamma_{y,t})_{y \in Y}, C))$

[1] A deterministic authenticated encryption scheme is here used as a key-wrapping algorithm should in practice not rely on nonces [Dwo04].

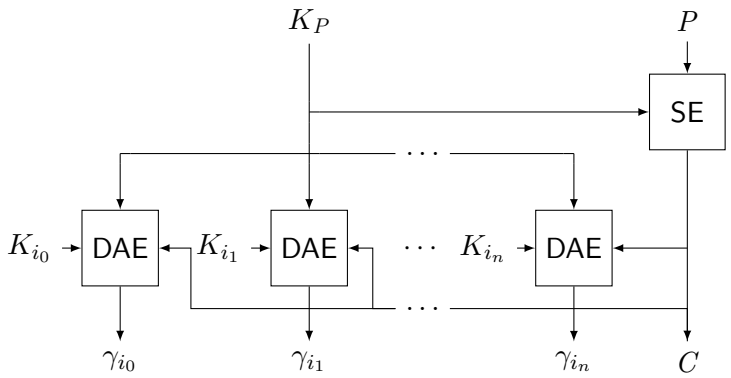

Figure 5.7: Encryption Procedure with $Y \coloneqq \{y_1, \ldots, y_n\}$ and $i_j \coloneqq (y_j, t)$.

$\mathsf{Receive}(L_K, \gamma)$: To recover the payload of a ciphertext γ,

1. parse $\gamma = (t, Y, ((y, \gamma_{y,t})_{y \in Y}, C))$
2. retrieve from L_K a key $K_{y,t}$ for a zone $y \in Y$
3. $K_P \leftarrow \mathsf{DAE.Dec}(K_{y,t}, C, \gamma_{y,t})$
4. return $P \leftarrow \mathsf{SE.Dec}(K_P, C)$.

Identity Escrow. If needed, the issuer can recover the identity of a vehicle that sent an authenticated key request or response during an execution of protocol Enter. He does so by executing the opening protocol of DGSA.

$\mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m)$: parse message m as (z, t, m', tok) with $m' \in \{ek, C\}$ and return identity $\mathcal{V}/\perp \leftarrow \mathsf{DGSA.Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, (z, t, m'), e(t), tok)$.

Threshold Issuance and Identity Resolution. In the above scheme, the issuer can alone issue vehicle credentails and de-anonymize messages sent during executions of protocol Enter, making it a single point of failure. To distribute the issuance and opening capabilities over several authorities, one can instead use the threshold DGS+A scheme mentioned in Section 5.2.2, so that at least a threshold number of authorities must collaborate to issue a credential or to link a message to a vehicle long-term credential. The ZE scheme thereby inherits the security properties of the threshold DGS+A scheme.

Correctness & Security.

This section shows that $\mathcal{Z}$ is correct and satisfies the security requirements introduced in Section 5.3.2.

Theorem 5.3.8 (Correctness)**.** *$\mathcal{Z}$ is correct if the signature scheme* SIG*, the DGS+A scheme* DGSA*, the public-key encryption scheme* PKE*, the symmetric encryption scheme* SE *and the authenticated encryption scheme* DAE *are correct.*

Proof. If the signature scheme SIG is correct, vehicles that honestly execute the vehicle enrollment algorithm obtain a certificate that is accepted in the authorization protocol with probability 1. If scheme DGSA is also correct, the credentials obtained during the authorization protocol in an epoch allows the vehicles to generate authentication tokens that are later accepted when they enter new zones in the same epoch with overwhelming probability. If the encryption scheme PKE is correct, during the Enter protocol for a zone in a given time period, the same vehicles can successfully unwrap the authenticated-encryption key for the zone in that time period. The correctness of the symmetric encryption scheme SE and that of the authenticated encryption scheme DAE then suffice to conclude that scheme $\mathcal{Z}$ is correct. □

Theorem 5.3.9 (PH-CCA Security)**.** *$\mathcal{Z}$ is PH-CCA secure if* SIG *is EUF-CMA secure, if* DGSA *satisfies traceability, if* SE *is IND-CPA secure, if* PKE *is IND-CPA secure, and if* DAE *satisfies privacy and authenticity.*

Proof. Under the assumptions of the theorem, the PH-CCA security of $\mathcal{Z}$ can be proved via the following hybrid argument. Let $\mathcal{A}$ be an adversary for the PH-CCA security distinction experiment (i.e., $\mathcal{A}$ tries to tell apart the PH-CCA challenger $\mathcal{C}_0^{\text{ph-cca}}$ that encrypts P_0 and the challenger $\mathcal{C}_1^{\text{ph-cca}}$ that encrypts P_1). Suppose that $\mathcal{A}$ wins the game with $(\mathcal{V}^*, P_0, P_1, Y^*, t^*)$ as a challenge tuple. Number the zones in Y^* from 1 to $n_{Y^*} \coloneqq |Y^*|$, i.e., $Y^* = \{y_1, \ldots, y_{n_{Y^*}}\}$. For $i = 0, \ldots, n_{Y^*}$, consider the hybrid algorithm $\mathcal{H}_i$ that proceeds exactly like the PH-CCA challenger $\mathcal{C}_0^{\text{ph-cca}}$, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K' and the remaining $z - i$ zones with another key K, and encrypts P_0 with K. Namely, the challenge ciphertext of $\mathcal{H}_i$ is of the form

$$\begin{aligned}&\mathsf{DAE.Enc}(K_{y_1,t}, K'), \ldots, \mathsf{DAE.Enc}(K_{y_i,t}, K'),\\ &\mathsf{DAE.Enc}(K_{y_{i+1},t}, K), \ldots, \mathsf{DAE.Enc}(K_{n_{Y^*},t}, K),\\ &\mathsf{SE.Enc}(K, P_0).\end{aligned}$$

Consider also, for $i = 0, \ldots, n_{Y^*}$, the hybrid algorithm $\mathcal{H}'_i$ that proceeds exactly like the PH-CCA challenger $\mathcal{C}_1^{\text{ph-cca}}$, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K and the remaining $z - i$ zones with another key K', and encrypts P_1 with K.

Namely, the challenge ciphertext of $\mathcal{H}'_i$ is of the form

$$\begin{aligned}&\mathsf{DAE.Enc}(K_{y_1,t}, K), \ldots, \mathsf{DAE.Enc}(K_{y_i,t}, K),\\ &\mathsf{DAE.Enc}(K_{y_{i+1},t}, K'), \ldots, \mathsf{DAE.Enc}(K_{n_{Y^*},t}, K'),\\ &\mathsf{SE.Enc}(K, P_1).\end{aligned}$$

By definition, $\mathcal{H}_0 = \mathcal{C}_0^{\text{ph}-\text{cca}}$, the challenger that encrypts P_0 and $\mathcal{H}'_{n_{Y^*}} = \mathcal{C}_1^{\text{ph}-\text{cca}}$, the challenger that encrypts P_1. To show that $\mathcal{A}$ has a negligible advantage in the PH-CCA distinction experiment, it suffices to show that the advantage of $\mathcal{A}$ in distinguishing two consecutive hybrids is negligible.

If adversary $\mathcal{A}$ can distinguish $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$, then it can be used to win the DAE privacy game for DAE as follows. Assume SIG to be existentially unforgeable, DGSA to satisfy traceability, SE to be IND-CPA secure, PKE to be IND-CPA secure, and DAE to satisfy authenticity. Let $\mathcal{B}$ be an algorithm that features $\mathcal{A}$ and interacts with a DAE privacy game $\mathcal{C}_b^{\text{priv}}$ for $b \in \{0,1\}$, which generates a secret key K_{priv}. At the beginning of the game, $\mathcal{B}$ receives parameters pp_{DAE} for DAE from $\mathcal{C}_b^{\text{priv}}$ and generates the other parameters itself. It generates $(pk_{\mathcal{E}} = vk, sk_{\mathcal{E}} = sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$ and $(pk_{\mathcal{I}} = pk, sk_{\mathcal{I}} = sk) \leftarrow \mathsf{DGSA.KG}(pp_{\mathsf{DGSA}})$. It then sends all the parameters and $pk_{\mathcal{E}}$ and pk_I to $\mathcal{A}$.

$\mathcal{B}$ chooses a time period $\tilde{t}$ uniformly at random and implicitly sets $K_{y_{i+1},\tilde{t}} := K_{\text{priv}}$ (i.e., it will query $\mathcal{C}_b^{\text{priv}}$ to answer queries Send queries involving zone y_{i+1} and time $\tilde{t}$).

Throughout the game, $\mathcal{B}$ locally maintains the same lists as the PH-CCA challenger does.

For Enroll.V&Enroll.E queries, $\mathcal{B}$ runs the protocol and stores the generated certificates.

For Enroll.E queries, $\mathcal{B}$ runs the corresponding algorithms with $sk_{\mathcal{E}}$.

For Authorize.V&I queries, $\mathcal{B}$ runs the protocol and stores the generated credentials.

For Authorize.I queries, $\mathcal{B}$ runs the corresponding algorithm with pk_I.

For an Enter query on input $(\mathcal{V}, z, t, role)$, ($\mathcal{V} \notin \mathcal{L}_{\mathsf{corrupt}}$ by definition of this oracle) for any *role*, if $(z,t) = (y_{i+1}, \tilde{t})$ then $\mathcal{B}$ starts an execution of protocol Enter with $\mathcal{A}$.

If $role = requester$, then algorithm $\mathcal{B}$ generates $(ek, dk) \leftarrow \mathsf{PKE.KG}(\lambda)$, computes a token $tok \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{V},e(t)}, (z,t,ek))$ and sends $(z,t,ek,tok_{\mathcal{V}})$ to $\mathcal{A}$.

If $role = responder_i$, and that $\mathcal{V}$ should respond according to the strategy of $responder_i$, then $\mathcal{B}$, upon receiving $(y_{i+1}, \tilde{t}, ek, tok)$ from $\mathcal{A}$, determines whether it comes from a non-corrupt vehicle identity in the same protocol execution, i.e., whether $\mathcal{A}$ is simply performing a passive attack by relaying a message from a non-corrupt vehicle identity.

If so, then $\mathcal{B}$ encrypts a random message instead of K with PKE. The IND-CPA security of PKE is important here to argue for indistinguishability

between the two consecutive hybrids. If $\mathcal{V} \in \mathcal{L}_{\mathsf{corrupt}}$, algorithm $\mathcal{B}$ returns $\perp$.

If $(y_{i+1}, \tilde{t}, ek, tok)$ does not come from a non-corrupt vehicle in the same protocol execution, it is an active attack. $\mathcal{B}$ then aborts the protocol execution. In the event in which $\mathcal{A}$ wins the PH-CCA game, if $t^* = \tilde{t}$ ($\mathcal{B}$ will abort if it is not the case), the winning condition 3b) implies that no vehicle $\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}$ can be authorized in $e(\tilde{t})$, so that

- either there exists a vehicle identity $\mathcal{W}$ such that $(\mathcal{W}, e(\tilde{t})) \in \mathcal{L}_{\mathsf{auth}}$ but $\mathcal{W} \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), i.e., it has obtain a credential for $e(\tilde{t})$ but has never been enrolled neither as an honest vehicle nor a corrupt one; and it happens with negligible probability if SIG is EUF-CMA secure

- or no such vehicle exists and the token sent the adversary can be valid w.r.t. $pk_{\mathcal{I}}$ and $e(\tilde{t})$ with only negligible probability if DGSA satisfies traceability.

Therefore, by aborting the protocol once a token is received from $\mathcal{A}$ during the Enter protocol execution, $\mathcal{B}$ is computationally indistinguishable from both $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$.

If $(z, t) \neq (y_{i+1}, \tilde{t})$, then $\mathcal{B}$ simply runs the $\mathsf{Enter.V}$ algorithm on input $(\mathcal{V}, z, t, role)$, and never has to send K_{priv}.

For an Exit on input $(\mathcal{V}, z, t)$, algorithm $\mathcal{B}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.

For a Send query on $(\mathcal{V}, P, Y \ni y_{i+1}, \tilde{t})$, algorithm $\mathcal{B}$ checks whether all the zones in Y are active for $\mathcal{V}$, and in particular if $(y_{i+1}, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If not, it returns $\perp$, otherwise $\mathcal{B}$ generates a payload key $K \leftarrow \mathsf{DAE.KG}(pp_{\mathsf{DAE}})$, computes $C \leftarrow \mathsf{DAE.Enc}(K, P)$, and computes $\gamma_{y_{i+1},\tilde{t}}$ for the zone–time pair $(y_{i+1}, \tilde{t})$ by sending K to $\mathcal{C}_b^{\mathsf{priv}}$. Simulator $\mathcal{B}$ then encrypts K with the keys for the other zone–time pairs in the query, and sets the ciphertext as the Enter algorithm does. For Send queries such that $y_{i+1} \notin Y$ or $t \neq \tilde{t}$, algorithm $\mathcal{B}$ runs algorithm Send on the inputs.

For a $\mathsf{Receive}$ query on a vehicle identity $\mathcal{V}$ and a ciphertext $\gamma = (\tilde{t}, Y, ((y, \gamma_{y,\tilde{t}})_{y \in Y}, C))$ such that $y_{i+1} \in Y$, algorithm $\mathcal{B}$ first determines whether y_{i+1} is the only zone in Y that is active for $\mathcal{V}$ in time $\tilde{t}$. If not, then $\mathcal{B}$ can answer the query using any other zone z that is active for $\mathcal{V}$, i.e., by computing $K_P \leftarrow \mathsf{DAE.Dec}(K_{z,\tilde{t}}, C, \gamma_{z,\tilde{t}})$, and returning $P \leftarrow \mathsf{SE.Dec}(K_P, C)$. If y_{i+1} is the only zone in γ that is active for $\mathcal{V}$ in time t, then $\mathcal{B}$ checks whether $\gamma_{y_{i+1},\tilde{t}}$ and C are the output of a same previous Send query that it answered. If not, then $\mathcal{B}$ replies with $\perp$. This is where the authenticity of DAE comes into play. Indeed, if DAE satisfies authenticity, $\mathcal{A}$ can submit a ciphertext that does not decrypt to $\perp$ with only negligible probability. If $\gamma_{y_{i+1},\tilde{t}}$ and C are part of the answer to a previous Send query, then $\mathcal{B}$ replies with the payload

on which it was queried. For Receive oracle involving other zone–time pairs, $\mathcal{B}$ simply runs algorithm Receive on the queried input.

For an Open query on input m, algorithm $\mathcal{B}$ parses m as (m', t, tok) and runs $\mathsf{DGSA.Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, (m, e(t), tok))$. It returns the output to $\mathcal{A}$.

For a Corrupt query on an identity $\mathcal{V}$, algorithm $\mathcal{B}$ replies by sending to $\mathcal{A}$ the certificate $\mathcal{V}[cert_{\mathcal{V}}]$, all the credentials $\mathcal{V}[e_j, cred_{V,j}]$ and the key list $\mathcal{V}[L_K]$. Note that in the event in which $\mathcal{A}$ wins the game, if $t = \tilde{t}$, no challenge zone $y^* \in Y^*$ can be active for $\mathcal{V}$ in time t^* (condition 2). In particular, $\mathcal{B}$ never has to send $K_{y_{i+1},\tilde{t}}$ to $\mathcal{A}$.

At the challenge phase, adversary $\mathcal{A}$ outputs the challenge tuple $(\mathcal{V}^*, P_0, P_1, Y^*, t^*)$. If $\mathcal{V}^* \in \mathcal{L}_{\mathsf{corrupt}}$, then $\mathcal{B}$ returns 0 as the PH-CCA challenger does. If $\tilde{t} \neq t^*$, then $\mathcal{B}$ aborts and outputs $\perp$; otherwise, $\mathcal{B}$ generates two payload keys K and K', sends them to $\mathcal{C}$, and receives a ciphertext $\gamma^*_{y_{i+1},\tilde{t}}$ from $\mathcal{C}_b^{\text{priv}}$. For $k = 1, \ldots, i$, Simulator $\mathcal{B}$ computes $\gamma_{y_k,\tilde{t}} \leftarrow \mathsf{DAE.Enc}(K_{y_k,\tilde{t}}, K')$, and for $i < k \leq n_{Y^*}$, it computes $\gamma_{y_k,\tilde{t}} \leftarrow \mathsf{DAE.Enc}(K_{y_k,\tilde{t}}, K)$. It also compute $C \leftarrow \mathsf{DAE}(K, P_0)$, and then sends the ciphertext

$$\gamma^* \leftarrow (\tilde{t}, Y^*, ((y_k, \gamma_{y_k,\tilde{t}})_{k \leq i}, (y_{i+1}, \gamma^*_{y_{i+1},\tilde{t}}), (y_k, \gamma_{y_k,\tilde{t}})_{k > i+1}, C))$$

to adversary $\mathcal{A}$.

After the challenge phase, for Receive on $(\mathcal{V}, \gamma)$ queries from $\mathcal{A}$, algorithm $\mathcal{B}$ first parses γ as (t, Y, γ'). Conditioned on the event in which $\mathcal{A}$ wins the PH-CCA game, no part of γ' is replayed from the challenge ciphertext γ^*, i.e., $\gamma' \cap \gamma^{*'} \neq \emptyset$. $\mathcal{B}$ can then proceeds as before the challenge phase.

For the other queries, $\mathcal{B}$ replies as before the challenge phase.

At the end of the experiment, $\mathcal{B}$ forwards the decision bit of $\mathcal{A}$. Up to the existential unforgeability of SIG and the traceability of DGSA, $\mathcal{B}$ perfectly simulates $\mathcal{C}_b^{\text{ph-cca}}$ to adversary $\mathcal{A}$ except for Receive queries involving $(y_{i+1}, \tilde{t})$ as the only zone in the ciphertext that is active for the queried vehicle and for passive Enter queries. Consequently, as $\tilde{t} = t^*$ with probability $1/|T|$,

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{H}_i,\mathcal{H}_{i+1}}(\mathcal{A}) &- q_{\mathsf{Enter.act}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{euf-cma}}_{\mathsf{SIG}}(\mathcal{B}(\mathcal{A})) \\
&- q_{\mathsf{Enter.act}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{trace}}_{\mathsf{DGSA}}(\mathcal{B}(\mathcal{A})) \\
&- q_{\mathsf{Receive}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{auth}}_{\mathsf{DAE}}(\mathcal{B}(\mathcal{A})) \\
&- q_{\mathsf{Enter.pass}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathcal{B}(\mathcal{A})) \\
&\leq |T|\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE}}(\mathcal{B}(\mathcal{A}))\,,
\end{aligned}$$

with

- $q_{\mathsf{Enter.act}}(y_{i+1}, \tilde{t})$ the number of active enter queries in $(y_{i+1}, \tilde{t})$
- $q_{\mathsf{Receive}}(y_{i+1}, \tilde{t})$ the number of Receive queries involving $(y_{i+1}, \tilde{t})$ as the only zone in the ciphertext that is active for the queried vehicle

- $q_{\mathsf{Enter.pass}}(y_{i+1}, \tilde{t})$ the numer of passive Enter queries made by for zone y_{i+1} in time $\tilde{t}$.

Therefore, if SIG is EUF-CMA secure, if DGSA satisfies traceability, if PKE is IND-CPA secure and if DAE satisfies authenticity, then $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ are computationally indistinguishable.

The IND-CPA security of SE can be reduced to the computational indistinguishability of $\mathcal{H}'_i$ and $\mathcal{H}'_{i+1}$ in the very same manner.

Note also that the IND-CPA security of SE can be reduced to the computational indistinguishability of $\mathcal{H}_{n_{Y^*}}$ and $\mathcal{H}'_0$. The reduction algorithm can implicitly set K as the challenger key, and forward the challenge tuple (P_0, P_1) at the challenge phase. It can answer of all the other Send queries (i.e., other that the challenge one) by generating fresh payload keys. Moreover, as all the key for active zones are known to $\mathcal{B}$, it can answer all the other queries.

Overall,

$$\mathbf{Adv}^{\mathrm{ph-cca}}_{\mathcal{Z}}(\mathcal{A}) - \mathsf{negl}(\lambda) \leq 2n_{Y^*}|T|\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{DAE}}(\mathcal{B}(\mathcal{A})) + \mathbf{Adv}^{\mathrm{ind-cpa}}_{\mathsf{SE}}(\mathcal{B}(\mathcal{A})),$$

hence the statement of the theorem. □

Theorem 5.3.10 (Anonymity)**.** *The ZE scheme $\mathcal{Z}$ satisfies anonymity if the DGS+A scheme* DGSA *satisfies anonymity and if* SIG *is EUF-CMA secure.*

Proof. Assuming that DGSA satisfies anonymity, and that SIG is EUF-CMA secure, the anonymity of $\mathcal{Z}$ can be proved via the following hybrid argument. Let $\mathcal{A}$ be an adversary for the ZE anonymity game that makes q Enter^* queries. One can assume that $q > 0$. Indeed, an adversary that wins the game with $q = 0$ can always be run as a sub-routine by an adversary that makes one arbitrary Enter^* query. For $i = 0, \ldots, q$, let $\mathcal{H}_i$ be an algorithm that proceeds exactly that the ZE anonymity game challenger, except that to answer the (*) queries with a bit d up to the ith Enter^*, it uses $\mathcal{V}_{1-d}$. For the remaining (*) queries (including the remaining $q - i$ Enter^*), it uses $\mathcal{V}_d$. By definition, if $\mathcal{C}_b$ denotes the ZE anonymity challenger that uses $\mathcal{V}_b$, then $\mathcal{H}_0 = \mathcal{C}_0$ and $\mathcal{H}_q = \mathcal{C}_1$. The advantage of $\mathcal{A}$ in the ZE anonymity game is therefore at most q times its advantage in distinguishing $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$ for some $0 \leq i \leq q - 1$. However, if $\mathcal{A}$ can distinguish $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$, then it can be used to win the DGS+A anonymity game as follows.

Consider a simulator that runs $\mathcal{A}$ as a subroutine and interacts with the DGS+A anonymity challenger $\mathcal{C}_{\mathsf{DGSA},b}$ for $b \in \{0, 1\}$. Throughout the game, $\mathcal{B}$ locally maintains the same lists as the PH-CCA challenger does.

Upon receiving parameters pp_{DGSA} for DGSA and a public pk_I, simulator $\mathcal{B}$ generates the parameters for the other schemes itself, generates $(pk_{\mathcal{E}} = vk, sk_{\mathcal{E}} = sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$ and sends all the parameters to $\mathcal{A}$ as well as $pk_{\mathcal{E}}$ and $pk_{\mathcal{I}}$.

For Enroll.V&Enroll.E queries, $\mathcal{B}$ runs the protocol and stores the generated certificates.

For Enroll.E queries, simulator runs the corresponding algorithms with $sk_{\mathcal{E}}$.

For an Authorize.V&I on $(\mathcal{V}, e)$, simulator $\mathcal{B}$ queries the Auth oracle of $\mathcal{C}_{\mathsf{DGSA},b}$ for $(\mathcal{V}, e)$ and stores the output credential.

For an Authorize.I query on $(\mathcal{V}, e)$, simulator $\mathcal{B}$ starts an execution of protocol Authorize with $\mathcal{A}$. Conditioned on the event in which $\mathcal{A}$ wins the game with a pair of identities $\mathcal{V}_0$ and V_1, if $\mathcal{V} = \mathcal{V}_d$ for $d \in \{0, 1\}$, upon receiving $(vk_{\mathcal{V}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{V}})$ from $\mathcal{A}$, simulator $\mathcal{B}$ checks whether $\mathcal{V}_d \in \mathcal{L}_{\mathsf{honest}}$ (i.e., it $\mathcal{V}_d$ is enrolled). If not, then $\mathcal{B}$ aborts and is indistinguishable from the ZE anonymity challenger under the assumption that SIG is EUF-CMA secure. If $\mathcal{V} \neq \mathcal{V}_0, \mathcal{V}_1$, then simulator $\mathcal{B}$ first checks that $\mathsf{SIG.Vf}(pk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}), \sigma_{\mathcal{E}}) = 1$ and that . If not, $\mathcal{B}$ aborts; otherwise it starts an execution of protocol DGSA.Issue with $\mathcal{A}$ on $(\mathcal{V}, e)$ and simply forwarding every message from $\mathcal{A}$ to the Issue.I oracle provided by $\mathcal{C}_{\mathsf{DGSA},b}$ and vice versa.

$\mathcal{B}$ answers Enter queries by calling on oracle Auth to generate and verify authentication tokens.

For an Exit on input $(\mathcal{V}, z, t)$, simulator $\mathcal{B}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.

For Send and Receive queries, $\mathcal{B}$ runs the corresponding algorithms on the inputs.

For an Open query on input m, simulator $\mathcal{B}$ queries the Open oracle provided by the DGS+A-anonymity challenger on m.

To answer Corrupt queries on a vehicle identity $\mathcal{V}$, simulator $\mathcal{B}$ sends to the Corrupt oracle provided by $\mathcal{C}$ all the pairs $(\mathcal{V}, e_j)$ such that $(\mathcal{V}, e_j) \in \mathcal{L}_{\mathsf{auth}}$, gets a credential $cred_j$ for each e_j, and sets $cred_{\mathcal{V},j} \leftarrow cred_j$. It can then answer the query by returning the certificate $\mathcal{V}[cert_{\mathcal{V}}]$ (that it maintains locally), $\mathcal{V}[(e_j, cred_{\mathcal{V},j}]$ and the list $\mathcal{V}[L_K]$ of keys that it maintains for $\mathcal{V}$.

At the challenge phase, after the adversary outputs two challenge vehicle identities $\mathcal{V}_0$ and $\mathcal{V}_1$. Simulator $\mathcal{B}$ checks that they are authorized in exactly the same epochs and that $\mathcal{V}_0[L_K] = \mathcal{V}_1[L_K]$. If it is not the case, $\mathcal{B}$ aborts.

$\mathcal{B}$ simulator randomly chooses a zone–time pair $(\tilde{z}, \tilde{t})$ such that both vehicles are authorized in $e(\tilde{t})$.

After the challenge phase, the oracles Enter, Exit, Send and Receive are respectively replaced with the Enter*, Exit*, Send* and Receive* oracles.

For all oracles queries except the queries to these oracles with a bit d, simulator $\mathcal{B}$ replies as before the challenge phase (and recall that conditioned on the event in which $\mathcal{A}$ wins the game, neither $\mathcal{V}_0$ nor V_1 can be corrupt).

For the (*) queries up to the ith Enter* query,

1. if Enter* is queried on $(d, z, t, requester)$, $\mathcal{B}$
   - checks that $(e(t), \mathcal{V}_{1-d}) \in \mathcal{L}_{\mathsf{auth}}$ (aborts if not)

- checks if $\exists(z, t, K_{z,t}) \in \mathcal{V}_{1-d}[L_K]$ (does nothing if it is the case)
- makes an authenticated key request: it generates a $(ek, dk) \leftarrow \mathsf{PKE.KG}(\lambda)$, queries the Auth oracle of $\mathcal{C}_{\mathsf{DGSA},b}$ on the tuple $(\mathcal{V}_{1-d}, e(t), (z, t, ek))$, receives an authentication token tok and sends (z, t, ek, tok) to $\mathcal{A}$
- upon receiving (z, t, C, tok') from $\mathcal{A}$, checks that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, C), e(t), tok') = 1$. If not, $\mathcal{B}$ aborts, otherwise it decrypts $K_{z,t} \leftarrow \mathsf{PKE.Enc}(ek, C)$ and adds $(z, t, K_{z,t})$ to $\mathcal{V}_{1-d}[L_K]$

2. if Enter^* is queried on $(d, z, t, responder_i)$, $\mathcal{B}$ upon receiving (z, t, ek, tok) from $\mathcal{A}$, if $\exists(z, t, K_{z,t}) \in \mathcal{V}_{1-d}[L_K]$ and if it should reply according to the strategy of $responder_i$,
   - checks that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, ek), e(t), tok) = 1$ (aborts if not)
   - computes $C \leftarrow \mathsf{PKE.Enc}(ek, K_{z,t})$
   - queries the Auth oracle of $\mathcal{C}_{\mathsf{DGSA},b}$ on $(\mathcal{V}_{1-d}, e(t), (z, t, C))$ and receives a token tok'
   - sends (z, t, C, tok') to $\mathcal{A}$
3. if Exit^* is queried on (d, z, t), simulator $\mathcal{B}$ deletes $(z, t, K_{z,t})$ from both $\mathcal{V}_{1-d}[L_K]$
4. if Send^* is queried on (d, P, Y, t), simulator $\mathcal{B}$
   - computes $\gamma \leftarrow \mathcal{Z}.\mathsf{Enc}(V_{1-d}[L_K], P, Y, t)$ and sends it to $\mathcal{A}$

For the i+1th Enter^* query on input $(d, z, t, role)$, if $(z, t) \neq (\tilde{z}, \tilde{t})$, simulator $\mathcal{B}$ aborts, otherwise if it is an Enter^* with $role = requester$, to compute an authenticated key request, it sends $(\mathcal{V}_0, \mathcal{V}_1, e(\tilde{t}), (z, t, ek))$ as a challenge tuple to $\mathcal{C}_{\mathsf{DGSA},b}$. If it is an Enter^* query and that $role = responder_i$ and that it should reply according to the strategy to the strategy of $responder_i$, to compute its authenticated key response, simulator $\mathcal{B}$ sends $(\mathcal{V}_0, \mathcal{V}_1, e(\tilde{t}), (z, t, C))$ as a challenge tuple to $\mathcal{C}_{\mathsf{DGSA},b}$.

For the remaining (*) queries $\mathcal{B}$ uses the state of $\mathcal{V}_d$ instead of $\mathcal{V}_{1-d}$.

Note that the winning condition enforces that neither $\mathcal{V}_0$ nor $\mathcal{V}_1$ can be corrupt throughout the game so $\mathcal{B}$ never has to return their states.

Moreover, the winning condition also implies that $\mathcal{A}$ never made an Open query on any message exchanged during the executions protocol Enter^*. As a consequence, the distribution of the answers of $\mathcal{B}$ to oracle queries except for the $i + 1$th Enter^* query are identically to those of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$.

At the end of the game, $\mathcal{B}$ forwards the decision bit b' of $\mathcal{A}$ to $\mathcal{C}_{\mathsf{DGSA},b}$. If $\mathcal{A}$ has made no Enter^* query on z and t, then $\mathcal{B}$ returns $\perp$ to $\mathcal{C}$, otherwise $\mathcal{B}$ forwards b' to $\mathcal{C}_{\mathcal{Z},b}$, and

$$\mathbf{Adv}_{\mathcal{H}_i,\mathcal{H}_{i+1}}(\mathcal{A}) - \mathsf{negl}(\lambda) \leq |Z||T|\mathbf{Adv}^{\text{ano}}_{\mathsf{DGSA}}(\mathcal{B}(\mathcal{A}))$$

with the negligible factor coming from the unforgeability of SIG.

Therefore,

$$\mathbf{Adv}^{\text{ano}}_{\mathcal{Z}}(\mathcal{A}) - \mathsf{negl}(\lambda) \leq q|Z||T|\mathbf{Adv}^{\text{ano}}_{\mathsf{DGSA}}(\mathcal{B}(\mathcal{A})).$$

As $1/q|Z||T|$ is non-negligible, if $\mathcal{A}$ has a non-negligible advantage in the ZE anonymity game, then so does $\mathcal{B}$ in the DGS+A anonymity game; hence the theorem. □

Theorem 5.3.11 (Traceability)**.** *The ZE scheme $\mathcal{Z}$ satisfies traceability if the DGS+A scheme* DGSA *satisfies traceability, if* PKE *is IND-CPA secure and if* SIG *is EUF-CMA secure.*

Proof. Consider an adversary $\mathcal{A}$ that wins the ZE traceabilty game with a non-negligible probability. To win the game, at the challenge phase, $\mathcal{A}$ outputs a tuple (z^*, t^*, K_{z^*,t^*}) such that there exists an honest vehicle identity $\mathcal{V}$ such that $(z^*, t^*, K_{z^*,t^*}) \in \mathcal{V}[L_K]$.

The winning condition implies that $K_{z^*,t^*} \notin \mathcal{L}_{\mathsf{keys}}$, so for K_{z^*,t^*} to be active for $\mathcal{V}$, either there exists at least one message m_j (with $m_j = (z^*, t^*, C, tok)$ or (z^*, t^*, ek, tok)) such that $(\mathcal{A}, z^*, t^*, m_j) \in \mathcal{L}_{\mathsf{enter}}$ or $(\mathcal{W}, z^*, t^*, m_j) \in \mathcal{L}_{\mathsf{enter}}$ for $\mathcal{W} \in \mathcal{L}_{\mathsf{corrupt}}$, or there does not exist such a message.

If there is no such message (case 0), then the traceability of $\mathcal{Z}$ can be reduced to the IND-CPA security of PKE since $\mathcal{A}$ only sees transcripts of Enter protocol executions.

If there exists at least one such message m_j, then the winning condition ensures that for $\mathcal{V}_j \leftarrow \mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m_j)$,

1. $\mathcal{V}_j \notin \mathcal{L}_{\mathsf{corrupt}}$ or
2. $(\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}) \wedge (\mathcal{V}_j, e(t^*)) \notin \mathcal{L}_{\mathsf{auth}}$.

In case 1), the traceability of $\mathcal{Z}$ can be reduced to the unforgeability of SIG or the IND-CPA security of PKE. Indeed, the fact that $K_{z^*,t^*} \in \mathcal{V}[L_K]$ means that $\mathcal{A}$ has sent an authentication token tok for a message m_j that was accepted by an honest vehicle in (z^*, t^*), be it $\mathcal{V}$, (since $K_{z^*,t^*} \in \mathcal{V}[L_K]$) during an Enter protocol execution, i.e., $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, m_j, e(t^*), tok) = 1$. However, as $\mathcal{V}_j \notin \mathcal{L}_{\mathsf{corrupt}}$, if

1.1) $\mathcal{V}_j \in \mathcal{L}_{\mathsf{honest}}$, then $\mathcal{A}$ either 1.1.1) simply relayed a message between $\mathcal{V}_j$ and that honest vehicle, and the traceability of $\mathcal{Z}$ can be reduced to the IND-CPA security of PKE, or 1.1.2) the adversary forged a token that opens to an honest vehicle that never computed it, in which case the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA

1.2) $\mathcal{V}_j \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), then the traceability of $\mathcal{Z}$ can be reduced to the unforgeability of SIG if 1.2.1) $(\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}$ ($\mathcal{V}_j$ was never enrolled, i.e., $\mathcal{A}$ forged a certificate for it) or the traceability of DGSA if 1.2.2) $(\mathcal{V}_j, e(t^*)) \notin \mathcal{L}_{\mathsf{auth}}$.

In case 2), the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA as $\mathcal{V}_j$ is corrupt, but was not authorized in $e(t^*)$.

The reduction to the IND-CPA security of PKE in case 0) and 1.1.1), is done by encrypting random a random key instead of the challenger in passive Enter queries.

To reduce to the traceability of DGSA in cases 1.1.2), 1.2.2) and 2), a reduction algorithm $\mathcal{B}$ running $\mathcal{A}$ as a subroutine uses message m_j and token tok sent by $\mathcal{A}$ as a forgery for the vehicle identity $\mathcal{V}_j$ in epoch $e(t^*)$.

To reduce to the unforgeability of SIG in case 1.2.1), $\mathcal{B}$ uses the signature of the certificate of $\mathcal{V}_j$ as a forgery. □

Theorem 5.3.12 (Ciphertext Integrity)**.** *The ZE scheme $\mathcal{Z}$ satisfies ciphertext integrity if* DAE *satisfies authenticity, if* SIG *is EUF-CMA secure, if* DGSA *satisfies traceability, and if* PKE *is IND-CPA secure.*

Proof. Assume SIG to be EUF-CMA secure, DGSA to satisfy traceability and PKE to be IND-CPA secure. The ciphertext integrity of $\mathcal{Z}$ can be reduced to the authenticity of DAE as follows. Let $\mathcal{A}$ be an adversary that wins the authenticity game of $\mathcal{Z}$ with probability at least ε. Let $\mathcal{B}$ be a reduction algorithm which runs $\mathcal{A}$ as a subroutine and interacts with the challenger $\mathcal{C}$ of the authenticity game of DAE (which generates a secret key K). At the beginning of the game, $\mathcal{B}$ receives parameters pp_{DAE} for DAE from $\mathcal{C}_b^{\mathrm{priv}}$ and generates the other parameters itself. It generates $(pk_{\mathcal{E}} = vk, sk_{\mathcal{E}} = sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$ and $(pk_{\mathcal{I}} = pk, sk_{\mathcal{I}} = sk) \leftarrow \mathsf{DGSA.KG}(pp_{\mathsf{DGSA}})$. It then sends all the parameters and $pk_{\mathcal{E}}$ and pk_I to $\mathcal{A}$.

Algorithm $\mathcal{B}$ chooses a zone–time pair $(\tilde{z}, \tilde{t})$ uniformly at random and implicitly sets $K_{\tilde{z},\tilde{t}} \coloneqq K$ (i.e., it will query $\mathcal{C}$ to answer queries involving $\tilde{z}$ and $\tilde{t}$).

Throughout the game, $\mathcal{B}$ locally maintains the same lists as the PH-CCA challenger does.

For Enroll.V&Enroll.E queries, $\mathcal{B}$ runs the protocol and stores the generated certificates.

For Enroll.E queries, $\mathcal{B}$ runs the corresponding algorithms with $sk_{\mathcal{E}}$.

For Authorize.V&I queries, $\mathcal{B}$ runs the protocol and stores the generated credentials.

For Authorize.I queries, $\mathcal{B}$ runs the corresponding algorithm with pk_I.

For an Enter query on input $(\mathcal{V}, z, t, role)$, ($\mathcal{V} \notin \mathcal{L}_{\mathsf{corrupt}}$ by definition of this oracle) for any $role$, if $(z, t) = (\tilde{z}, \tilde{t})$ then $\mathcal{B}$ starts an execution of protocol Enter with $\mathcal{A}$.

If $role = requester$, then $\mathcal{B}$ generates $(ek, dk) \leftarrow \mathsf{PKE.KG}(\lambda)$, computes $tok \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{V},e(t)}, (z, t, ek))$ and sends $(z, t, ek, tok_{\mathcal{V}})$ to $\mathcal{A}$.

If $role = responder_i$, and that $\mathcal{V}$ should respond according to the strategy of $responder_i$, then $\mathcal{B}$, upon receiving $(\tilde{z}, \tilde{t}, ek, tok)$ from $\mathcal{A}$, determines whether it comes from a non-corrupt vehicle identity in the same protocol execution, i.e., whether $\mathcal{A}$ is simply performing a passive attack by relaying a message from a non-corrupt vehicle identity.

Algorithm $\mathcal{B}$ then encrypts a random message instead of K with PKE. The IND-CPA security of PKE is important here to argue for indistinguishability between the two consecutive hybrids. If $\mathcal{V} \in \mathcal{L}_{\mathsf{corrupt}}$, algorithm $\mathcal{B}$ returns $\perp$.

If $(\tilde{z}, \tilde{t}, ek, tok)$ does not come from a non-corrupt vehicle in the same protocol execution, it is an active attack. $\mathcal{B}$ then aborts the protocol execution. Conditioned on the event in which $\mathcal{A}$ wins the PH-CCA game, if $t^* = \tilde{t}$ ($\mathcal{B}$ will abort if it is not the case), the winning condition 3b) implies that no vehicle $\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}$ can be authorized in $e(\tilde{t})$, so that

- either there exists a vehicle identity $\mathcal{W}$ such that $(\mathcal{W}, e(\tilde{t})) \in \mathcal{L}_{\mathsf{auth}}$ but $\mathcal{W} \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), i.e., it has obtain a credential for $e(\tilde{t})$ but has never been enrolled whether as an honest vehicle or not; and it happens with negligible probability if SIG is EUF-CMA secure

- or no such vehicle exists and the token sent the adversary can be valid w.r.t. $pk_{\mathcal{I}}$ and $e(\tilde{t})$ with only negligible probability if DGSA satisfies traceability.

Therefore, by aborting the protocol once a token is received from $\mathcal{A}$ during the Enter protocol execution, $\mathcal{B}$ is computationally indistinguishable from both $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$.

If $(z, t) \neq (\tilde{z}, \tilde{t})$, then $\mathcal{B}$ simply runs the $\mathsf{Enter.V}$ algorithm on input $(\mathcal{V}, z, t, role)$, and never has to send K_{priv}.

For an Exit on input $(\mathcal{V}, z, t)$, algorithm $\mathcal{B}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.

For a Send query on $(\mathcal{V}, P, Y \ni \tilde{z}, \tilde{t})$, algorithm $\mathcal{B}$ checks whether all the zones in Y are active for $\mathcal{V}$, and in particular if $(\tilde{z}, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If not, it returns $\perp$, otherwise $\mathcal{B}$ generates a payload key $K \leftarrow \mathsf{SE.KG}(pp_{\mathsf{SE}})$, computes $C \leftarrow \mathsf{SE.Enc}(K, P)$, and computes $\gamma_{\tilde{z},\tilde{t}}$ for the zone–time pair $(\tilde{z}, \tilde{t})$ by sending K to $\mathcal{C}_b^{\mathsf{priv}}$. Algorithm $\mathcal{B}$ then encrypts K with the keys for the other zone–time pairs in the query, and sets the ciphertext as the Enter algorithm does. For Send queries such that $\tilde{z} \notin Y$ or $t \neq \tilde{t}$, algorithm $\mathcal{B}$ runs algorithm Send on the inputs.

For every $\mathsf{Receive}$ query on vehicle identity $\mathcal{V}$ and a ciphertext $\gamma = (t, Y \ni \tilde{z}, ((y, \gamma_{y,t})_{y \in Y}, C))$ such that $t = \tilde{t}$, algorithm $\mathcal{B}$ first checks whether there exists a zone $y \neq \tilde{z}$ in Y such that $(y, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If so, it decrypts

using $K_{y,\tilde{t}}$ and replies as algorithm Receive does, otherwise it checks whether $(\tilde{z}, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If not, it returns $\perp$; otherwise it sends $\gamma_{\tilde{z},\tilde{t}}$ together with C as a header to $\mathcal{C}$ and forwards its answer to $\mathcal{A}$. For Receive queries such that $\tilde{z} \notin Y$ or $t \neq \tilde{t}$, algorithm $\mathcal{B}$ runs the corresponding algorithm on the inputs.

For an Open query on input m, algorithm $\mathcal{B}$ parses m as (m', t, tok) and runs $\mathsf{DGSA.Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, (m, e(t), tok))$. It forwards the output to $\mathcal{A}$.

For a Corrupt query on an identity $\mathcal{V}$, algorithm $\mathcal{B}$ replies by sending to $\mathcal{A}$ the certificate $\mathcal{V}[cert_{\mathcal{V}}]$, all the credentials $\mathcal{V}[e_j, cred_{V,j}]$ and the key list $\mathcal{V}[L_K]$. Note that in the event in which $\mathcal{A}$ wins the game, no challenge zone $y^* \in Y^*$ can be active for $\mathcal{V}$ in time t^* (condition 2). In particular, if $t = \tilde{t}$ (simulator $\mathcal{B}$ will abort otherwise), $\mathcal{B}$ never has to send $K_{\tilde{z},t}$ to $\mathcal{A}$.

$\mathcal{A}$ ultimately outputs a challenge tuple $(\mathcal{V}, \gamma^*)$. Algorithm $\mathcal{B}$ parses γ^* as $(t^*, Y^*, \gamma^{*'})$. If $\tilde{t} \neq t^*$ or $\tilde{z} \notin Y^*$, algorithm $\mathcal{B}$ aborts; otherwise, in the event in which $\mathcal{A}$ wins, $\mathsf{Receive}(\mathcal{V}[L_k], \gamma^*) \neq \perp$, meaning that there exists $y^* \in Y^*$ such that $\mathsf{DAE.Dec}(K_{y^*,\tilde{t}}, \gamma_{y^*,\tilde{t}}) \neq \perp$. Such a y^* is equal to $\tilde{z}$ with probability at least $1/|Z|$ and $t = \tilde{t}$ with probability $1/|T|$. Moreover, since $\forall (t^*, Y, \gamma) \notin \mathcal{L}_{\mathsf{sent}}, \gamma \cap \gamma^* = \emptyset$, it follows that $\gamma_{\tilde{z},\tilde{t}}$ was never output by $\mathcal{C}_b^{\text{priv}}$. Algorithm $\mathcal{B}$ then sends $\gamma_{\tilde{z},\tilde{t}}$ to $\mathcal{A}$.

As SIG is assumed to be EUF-CMA secure, DGSA to satisfy traceabilty and PKE to be IND-CPA secure, $\mathcal{B}$ is computationally indistinguishable from the ZE-scheme integrity challenger. Adversary $\mathcal{A}$ then wins the authenticity game with probability at least $(\varepsilon - \mathsf{negl}(\lambda))/|Z||T|$. As DAE satisfies authenticity and as $1/|Z||T|$ is non-negligible, ε must be negligible. □

5.3.13 Efficiency & Comparison

This section specifies how the building blocks can be instantiated such that the bandwidth constraint of 300 Bytes per message can be satisfied. Some design choices for the C-ITS deployment are then discussed and compared to the current C-ITS proposal.

Efficiency.

To instantiate the above ZE construction at a 128-bit security level, one can use

- SIG as the BLS signature scheme [BLS01] since no zero-knowledge proof must be computed during enrollment and authorization. On a Cocks–Pinch pairing curve [GMT19] defined over a field of order 2^{544} and with embedding degree 8, group elements in $\mathbb{G}$ and $\tilde{\mathbb{G}}$ respectively take 68 Bytes and 136 Bytes (using their quadratic twists which have degree 4 [GMT19]) for a group of 256-bit order. Therefore, vehicle certificates, each of which consist of a pair of keys and a signature, are 236 Bytes long.

- DGSA as the DGS+A scheme of Section 5.2.2. Authentication tokens sent during protocol Enter are then 246 Bytes.
- PKE as the Hash-ElGamal encryption scheme on the 256-bit first group of the previous Cocks–Pinch curve. A public key is a group element in $\mathbb{G}$, and a ciphertext consist of a group element and a bit string of same length as the plaintext (a 128-bit DAE zone key).
- SE as AES-CTR (Counter Mode) with 128-bit keys.
- DAE as AES-128-GCM-SIV [RS06, Section 5].

The complexity of the opening algorithm is the same as for DGSA, i.e., it grows linearly in the number of enrolled vehicles. This makes tracing expensive but allows for short authentication tokens, which is the appropriate trade-off for V2V communication in which CAMs should be short and tracing only be done in case of exceptional events, e.g., an accident or to revoke the key of a rogue device.

C-ITS Deployment and Comparison.

Suppose that the road network is divided into hexagonal zones, and that a vehicle broadcasts messages to the zone it currently is and its 6 neighboring zones, i.e., to 7 zones in total. With the parameters of Section 5.3.13, the ciphertexts of the ZE scheme (ignoring the time-period and zone indicators) consist of 7 AES-128-GCM-SIV ciphertexts (256 bits each) and an AES ciphertext (128 bits), amounting to 240 Bytes; well within the 300 Bytes bandwidth requirements for C-ITSs. With payloads of 128 bits, it corresponds to a cryptographic overhead of 224 Bytes.

For messages during protocol Enter, the cryptographic overhead of the ZE scheme is a PKE public key (83 Bytes) and an authentication token in a key request, and a PKE encryption of the DAE key (16 Bytes) and an authentication token in a key response. With tokens of size 216 Bytes, this yields a total of 284 Bytes for request and 300 Bytes for response messages.

Table 5.1 gives, for various curve choices, the security level, the size of certificates and the cryptographic overhead for key requests and key responses. Note that Cocks–Pinch curves are not vulnerable to TNFS attacks [BGK15, KB16, MSS16, KJ17, BD19, GS19, Gui19] which affects the security of some curves constructed from different methods, and these attacks may be improved in the future. The CP8-544 curve therefore seems to be the safest choice in terms of security at a 128-bit level or higher. On the other hand, although operations on CP8-544 are very efficient [Gui19], the BLS12-446 and FM12-446 curves are the most efficient pairing curves [Gui19] at that security level.

	p (bits)	Sec. Lvl. (bits)	Req. (B)	Resp. (B)
CP8-544 [GMT19]	256	131 [GMT19]	284	300
BLS12-446	299	132 [GS19]	263	279
FM12-446 [FM19]	296	136 [Gui19]	261	277

Table 5.1: Sizes of Key Requests and Responses with various Curves and their associated field sizes and security levels.

With the previous ZE construction used in combination with the DGS+A scheme of Section 5.2.2, a vehicle can create an unlimited number of unlinkable (even by other vehicles thanks to the anonymity of group signatures) signatures by downloading a single credential in every epoch. Compared to the current C-ITS proposals [ETS14, oTNHTSA17], DGS+A combines the equivalent of an infinite pseudonym pool size with the negligible costs of downloading and storing a single constant-size credential per epoch. The latter aspect is a significant improvement not only in terms of storage, but also communication: in the current C-ITS proposals, vehicles have to spread out requests for individual pseudonyms over time, rather than downloading them in batches, to avoid that issuers are able to link the pseudonyms belonging to the same vehicle.

Moreover, with the ZE scheme, each CAM carries only 64 Bytes more of cryptographic overhead than the current proposals with ECDSA signatures (160 Bytes), for all the additional security and privacy benefits. Besides, symmetric cryptography is typically significantly faster than elliptic-curve operations. Therefore, the verification of the authenticity of incoming CAMs is also faster thanks to the use of a deterministic authenticated (symmetric) encryption scheme to encrypt payload keys.

Table 5.2 provides a brief overview of the core differences between zone encryption and the current C-ITS proposal.

5.3.14 Threat Model and Design Choices

By nature, V2X communication is an open system that enables all participating vehicles to communicate with each other; the security that one can hope to achieve is therefore also inherently limited. This section discusses the threat model of zone encryption in more detail and provides some insights into the design choices.

Passive vs. Active Eavesdropping. Because all vehicles must be able to decrypt messages from other close vehicles, no system can protect against eavesdropping attacks by insiders that have access to legitimate vehicle credentials and roam around to actively listen into nearby zones. However, zone encryption does force such an attacker to actively participate in zone key exchange protocols, thereby exposing its credentials to being traced and

	Zone Encryption	C-ITS Proposal
Encrypted CAM	✓	✘
Anonymity	✓	✗
Pseudonyms per week	unlimited	100 (EU) / 20 (US)
CAM Authentication	DAE	ECDSA
Overhead per CAM	224 Bytes	160 Bytes
+ per entered Zones	284/300 Bytes	—

Table 5.2: Comparison of zone encryption to current C-ITS proposals at a 128-bit security level. "Pseudonyms" refers to the number of unlinkable authentication tokens a vehicle can generate per epoch.

revoked by authorities. Doing so may not be straightforward in practice, but it is a considerable step up from the passive and covert eavesdropping attacks that are trivial to deploy in the current C-ITS proposals where all vehicles broadcast *plaintext* messages.

As priorly discussed, zone encryption enables authorities to considerably increase the manufacturing cost of black-market decryption devices, hopefully beyond the point of economical feasibility for ordinary criminals. As also mentioned, the threat of abusing traffic infrastructure for mass surveillance can be limited by giving infrastructure that has no need for privacy, nor to decrypt CAM traffic, a different type of credentials that cannot be used to obtain zone keys. Nevertheless, mass surveillance by a powerful adversary remains possible, e.g., through a network of (parked or moving) vehicles, or by road infrastructure that does have a legitimate need to read CAM traffic. Therefore, even when using zone encryption, the information in CAMs must still be minimized as much as possible. Maintaining a pool of vehicles or infrastructure for performing such eavesdropping attacks becomes more expensive with zone encryption though, because by forcing the adversary to actively participate in zone key exchanges, suspicious behavior can be traced and the corresponding credentials revoked.

Cloning & Insider Attacks. An adversary that compromises and clones the keys of a vehicle, short-term credentials, or even its long-term certificate obviously allows the adversary to "impersonate" that vehicle. For zone keys and short-term (DGSA) credentials, the impact is limited by the timed aspect of zone encryption to, e.g., 15 minutes and a week, respectively. Corruption of long-term credentials is more damaging, but the certificate is also likely

to enjoy stronger protection, e.g., from trusted hardware. Furthermore, the issuer of short-term credentials could monitor and detect suspicious use of long-term credentials, such as too frequent requests or requests from very distant locations, and block or revoke the long-term credential accordingly.

Apart from decrypting CAM traffic from other vehicles, the adversary can also use the compromised credentials to broadcast fake information. Indeed, the ciphertext integrity property of zone encryption protects against malicious information being inserted by outsiders, but not by inside attackers. Moreover, since zone keys are shared among vehicles, it is nearly impossible to exactly identify the culprit vehicle. This is indeed a drawback with respect to the current proposal for C-ITSs where each CAM message is signed. However, if an abnormal event occurs in a zone, the issuer, can with our scheme, reduce the list of suspects to just the devices that have entered the zone at the time of the event. Besides, with the current proposal, the issuer can only fully trace back malicious information, not prevent it, and information from CAMs will always be double-checked by other sensors such as cameras and LiDARs. Overall, it seems that the privacy advantages of zone encryption outweigh not being able to directly identify malicious senders.

Alternative Authentication Mechanisms. The zone-encryption construction uses group signatures to authenticate messages of protocol Enter, which guarantees privacy for vehicles while enabling the issuer to trace and revoke compromised credentials.

One could consider resorting to different authentication mechanisms, such as anonymous credentials [CL03]. Generic anonymous credential schemes have been proposed for use in V2X applications [SF17,dFGSV17,NBCN17b] but typically have much larger signature sizes.

In theory, limited-spending techniques [CHK+06] for such schemes might seem suitable to avoid the credential cloning and allow to trace compromised keys. However, doing so would require some authority to collect and cross-check *all* pseudonyms across *all* zones to identify overused credentials. Apart from being detrimental to privacy, such data collection is infeasible in a continent-scale V2X communication system with tens of millions of zones and billions of messages exchanged.

In comparison, the opening algorithm of the group signatures in Section 5.2.2 runs on a *single* authenticated message sent during a key request or a key response. There is no need for a synchronization between all zones. The anonymous messages exchanged in a zone can easily be recoverable if e.g., road infrastructures are required to maintain a record of messages exchanged in the zone they are in for a certain duration (e.g., a day), or if vehicles maintain a record of the messages they exchange when requesting or communicating zone keys for a short amount of time (e.g., an hour).

Some anonymous attestation mechanisms such as EPID [BL09] support signature-based revocation, meaning that the key behind a signature can be revoked by adding the signature to a blacklist. However, the size and/or the verification time of such non-revocation proofs grow linearly with the number of revoked users, which is impractical in a V2X system with hundreds of millions of vehicles.

Finally, note that the authentication mechanism could be replaced with a quantum-safe one if large-scale quantum computers were to be built in the future. However, no quantum-safe scheme known today is even close to fitting the 300-Byte limit at a 128-bit security level.

5.3.15 Deployment Challenges

The cryptographic concept of zone encryption significantly improves the security and privacy of V2X communication. To be ready for real-world usage there are a number of interesting deployment challenges that need to be solved, and potential solutions are discussed below.

Key Agreement Strategy. The Enter protocol assumes the availability of an appropriate key agreement strategy. In order to avoid that all vehicles respond to a key request, a predetermined strategy should be used to decide which should reply, e.g., the vehicle closest to the requesting vehicle. The optimal strategy depends on the zone structure, the traffic and other practical factors, and is an engineering problem on its own.

Another important aspect are mechanisms to avoid and resolve different key clusters within a zone. These might occur when different groups of vehicles are unable to communicate with each other, e.g., due to physical constraints or jamming attacks, and hence establish independent keys within their clusters. Consequently, vehicles in different clusters would not be able to communicate with each other. A deployed system would need a mechanism to detect such clustering and to resolve the duplication issue by agreeing on a common zone key.

Besides, a clear strategy to refresh zone keys needs to be established. These keys are supposed to be valid for a short amount of time only. The expiration time thus needs to be communicated with the key and a mechanism must determine which vehicle will choose the new key, similarly to the strategy when entering a new zone.

Robustness. It is crucial that the increase of cryptographic security does not come for the price of reduced reliability of inter vehicle communication. Therefore, to ensure that vehicles can communicate in a timely manner, they do not only encrypt to the zone they are in, but also to the neighboring ones. Note that this approach also limits the impact of key clustering events described above. This robustness strategy is captured by the zone-encryption

protocol which encrypts the payload under *several* zone keys. The actual deployment solution still needs to decide which zone keys are used and requested in time by the vehicle, based e.g., on the direction of driving, as well as available traffic flow and route information.

Besides, to smoothly transition between time periods, periods should be required to slightly overlap. The concrete layout of such zones or the chosen time periods will depend on practical factors such as density of traffic and the range of the communication signal.

Finally, the communication medium of the deployed system should be robust to prevent package loss. In practice, both key-exchange messages and payloads would be repeated over a strong signal as done in other real-world protocols.

Chapter 6

Hardware Security without Secure Hardware: How to Decrypt with a Password and a Server

Hardware security tokens have now been used for several decades to store cryptographic keys. When deployed, the security of the corresponding schemes fundamentally relies on the tamper-resistance of the tokens, which is a very strong assumption in practice. Moreover, secure tokens are often expensive, cumbersome and can even be subverted.

This chapter introduces a new cryptographic primitive called *Encryption schemes with Password-protected Assisted Decryption* (EPAD schemes), in which the decryption key of a user is shared between a user device (or token) on which no assumption is made, and an online server. The user shares a human-memorizable password with the server. To decrypt a ciphertext, the user launches, from a public computer, a distributed protocol with the device and the server, authenticating herself to the server with her password (unknown to the device); and her secret key is never reconstructed during the interaction.

Section 6.3 gives a strong security model which guarantees that (1) for an efficient adversary to infer any information about a user's plaintexts, it must know her password *and* have corrupted her device (secrecy is guaranteed if only one of the two conditions is fulfilled), (2) the device and the server are unable to infer any information about the ciphertexts they help to decrypt (even though they could together reconstruct the secret key), and (3) the user is able to verify that they both correctly performed their computations. These EPAD schemes are in the password-only model, meaning that the user is not required to remember a trusted public key, and her password remains safe even if she is led to interact with a wrong server and a malicious device.

Section 6.4 then gives a practical pairing-based EPAD scheme. The construction is provably secure under standard computational assumptions, using malleable non-interactive proof systems which can be efficiently instantiated in the standard security model, i.e., without relying on the random oracle heuristic. Section 6.2 recalls the properties of malleable proof systems and provides a generic construction based on signatures and witness-indistinguishable proof systems.

Contents

6.1 Preliminaries

This section introduces preliminary material to this chapter.

6.1.1 Hardness Assumptions

We introdcue the Symmetric eXternal Diffie–Hellman assumption, the assumption on which some constructions in this chapter rely.

Definition 6.1.2 (SXDH Assumption)**.** *The Symmetric eXternal Diffie–Hellman (SXDH) assumption over a bilinear structure generator* G *is that given*

$\lambda \in \mathbb{N}$, for $\Gamma = (p,\ \mathbb{G}_1 = \langle g_1 \rangle,\ \mathbb{G}_2 = \langle g_2 \rangle,\ \mathbb{G}_T,\ e) \leftarrow \mathsf{G}\left(1^\lambda\right)$, the Decisional Diffie–Hellman assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$ with overwhelming probability. That is, no efficient adversary has a non-negligible advantage (in λ) in distinguishing $\left(g_i, g_i^a, g_i^b, g_i^{ab}\right)$ from $\left(g_i, g_i^a, g_i^b, g_i^c\right)$, for $i \in \{1,2\}$, $a, b, c \leftarrow_\$ \mathbb{Z}_p$ and $\Gamma \leftarrow \mathsf{G}\left(1^\lambda\right)$.

6.1.3 Signatures

This section first gives the definition of strong one-time security of signature schemes. It then presents Groth's one-time signature scheme as well as a structure-preserving existentially unforgeable signature scheme due to Jutla and Roy [JR17].

Strong One-Time Security. A signature scheme is one-time strongly unforgeable against chosen-message attacks if no efficient adversary can forge a signature on a message even after obtaining a single signature (potentially on the same message). That is to say, for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{Vf}(vk, m^*, \sigma^*) = 1 & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); Q_{\mathsf{Sign}} \leftarrow \emptyset \\ \wedge (m^*, \sigma^*) \notin Q_{\mathsf{Sign}} \ : & (vk, sk) \leftarrow \mathsf{KG}(pp) \\ & (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Sign}^{\mathrm{OT}}(sk,\cdot)}(vk) \end{array}\right]$$

is negligible. Oracle $\mathcal{O}.\mathsf{Sign}^{\mathrm{OT}}(sk, \cdot)$ can be queried only once, say on a message m. It computes and returns $\sigma \leftarrow \mathsf{Sign}(sk, m)$, and adds (m, σ) to Q_{Sign}.

6.1.4 Groth's Strong One-Time Signatures

Given a group generator G and family of hash functions $\mathcal{H}$, Groth's [Gro06] signature scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $(p, \mathbb{G} = \langle g \rangle) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Generate $H \leftarrow_\$ \mathcal{H}$. Set and return $pp \leftarrow (p, \mathbb{G}, H)$.

$\mathsf{KG}(pp) \to (vk, sk)$**:** generate $x, y \leftarrow_\$ \mathbb{Z}_p^*$. Compute $f \leftarrow g^x$ and $h \leftarrow g^y$. Generate $r, s \leftarrow_\$ \mathbb{Z}_p^*$ and compute $c \leftarrow f^r h^s$. Set $vk \leftarrow (f, h, c)$ and $sk \leftarrow (x, y, r, s)$. Return (vk, sk).

$\mathsf{Sign}\,(sk, m \in \{0,1\}^*) \to \sigma$: generate $t \leftarrow_\$ \mathbb{Z}_p$, and compute and return $\sigma \leftarrow (t, (x(r-t) + ys - H(m))y^{-1})$.

$\mathsf{Vf}(vk, m, \sigma) \to b \in \{0,1\}$: parse σ as (t, u). Return 1 if $c = g^{H(m)} f^t h^u$, else return 0.

Groth proved [Gro06, Theorem 18] that it is one-time strongly unforgeable against chosen-message attacks if the discrete-logarithm assumption over G holds and if $\mathcal{H}$ is a family of collision-resistant hash functions.

6.1.5 Jutla and Roy's Signature Scheme

The following signature scheme, parametrized by a bilinear structure generator G, is due to Jutla and Roy [JR17]. It allows to sign vectors of first-group elements, and it is existentially unforgeable under the SXDH assumption.

$\mathsf{Setup}\left(1^\lambda, n\right) \to pp$: run $\Gamma \leftarrow (p,\ \mathbb{G}_1 = \langle g_1 \rangle,\ \mathbb{G}_2 = \langle g_2 \rangle,\ \mathbb{G}_T,\ e) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Return $pp \leftarrow (\Gamma, n)$.

$\mathsf{KG}(pp) \to (vk, sk)$**:** Generate $b, d, f, k_0, A, z \leftarrow_\$ \mathbb{Z}_p$, $\mathbf{k} \leftarrow_\$ \mathbb{Z}_p^n$, $\mathbf{K} \leftarrow_\$ \mathbb{Z}_p^{n+4}$. Set vk[1] $\leftarrow \left(g_2^{AK_1}, \ldots, g_2^{AK_{n+4}}, g_2^{Az}, g_2^{A}\right)$ and $sk \leftarrow (b, d, f, k_0, \mathbf{k}, \mathbf{K}, z)$. Return (vk, sk).

$\mathsf{Sign}\,(sk, \mu \in \mathbb{G}_1^n) \to \sigma$: Generate $r, t \leftarrow_\$ \mathbb{Z}_p$. Compute

$$\rho \leftarrow g_1^r, \hat{\rho} \leftarrow g_1^{rb}, \psi \leftarrow g_1^{rt}, \gamma \leftarrow \prod_{i=1}^{n} \mu_i^{k_i} g_1^{k_0 + dr + frt}$$

$$\tau \leftarrow g_2^t, \pi \leftarrow \prod_{i=1}^{n} \mu_i^{K_i} \rho^{K_{n+1}} \hat{\rho}^{K_{n+2}} \psi^{K_{n+3}} \gamma^{K_{n+4}} g_1^z$$

Set and return $\sigma \leftarrow (\rho, \hat{\rho}, \psi, \gamma, \pi, \tau) \in \mathbb{G}_1^5 \times \mathbb{G}_2$.

$\mathsf{Vf}(vk, \mu, \sigma) \to b \in \{0,1\}$: If $e(\rho, \tau) = e(\psi, g_2)$ and

$$e\left(\pi, g_2^A\right) = \prod_{i=1}^{n} e\left(\mu_i, g_2^{AK_i}\right) e\left(\rho, g_2^{AK_{n+1}}\right) e\left(\hat{\rho}, g_2^{AK_{n+2}}\right) e\left(\psi, g_2^{AK_{n+3}}\right) e\left(\gamma, g_2^{AK_{n+4}}\right) e\left(g_1, g_2^{Az}\right)$$

then return 1, else return 0.

6.1.6 Public-Key Encryption

This section introduces labeled public-key encryption schemes, their security, as well as instantiations.

[1] The verification key is independent of $b, d, f, k_0, \mathbf{k}$ as signatures are split-CRS quasi-adaptive proofs for an affine language defined by the message, b, d, f, k_0 and $\mathbf{k}$, and these proofs are simulated with $(\mathbf{K}, z)$ as trapdoor.

Labeled Public-Key Encryption. A public-key encryption scheme (Setup, $\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}$) is *labeled* if the encryption and decryption algorithm additionally take as input a label or public data ℓ which is non-malleably attached to the ciphertext. In this case, the label is indicated on these algorithms by a superscript.

IND-PCA Security. INDistinguishability under Chosen *Plaintext-Checkable* Attacks [ABP15] (IND-PCA) guarantees that an encryption scheme reveals no information about plaintexts even if an adversary can check whether ciphertexts encrypt messages of its choice. Abdalla, Benhamouda and Pointcheval [ABP15] argued that this weakening of IND-CCA of security is actually enough for many password-related applications. Note, however, that it is equivalent to IND-CCA security if the message space is small since it is then possible to enumerate all messages. Formally, a labeled encryption scheme is IND-PCA secure if for every $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{ll} & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); Q \leftarrow \emptyset; (pk, sk) \leftarrow \mathsf{KG}(pp) \\ & (st, \ell^*, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}(Q,sk,\cdot)}\,(pk) \\ & b \leftarrow_\$ \{0,1\}; C^* \leftarrow \mathsf{Enc}^{\ell^*}(pk, m_b) \\ b = b' : & b' \leftarrow_\$ \mathcal{A}^{\mathcal{O}(Q,sk,\cdot)}\,(st, C^*) \\ & \text{if } (\ell^*, C^*) \in Q \\ & \quad b' \leftarrow_\$ \{0,1\} \\ & \quad \text{return } (b, b') \\ & \text{return } (b, b') \end{array}\right]$$

is negligibly close to $1/2$ (the *advantage* of $\mathcal{A}$ is then the distance of that probability to $1/2$). Oracle $\mathcal{O}$, with a state (Q and sk), replies to a (ℓ, C, m) query by returning the truth value of $\left(\mathsf{Dec}^\ell(sk, C) = m\right)$ and setting $Q \leftarrow Q \cup \{(\ell, C)\}$. If $\mathcal{A}$ ever queries $\mathcal{O}$ on (ℓ^*, C^*), its advantage is set to 0 since b' is overwritten by a uniformly random bit.

Short Cramer–Shoup Encryption. Given a *group generator* G (i.e., an algorithm which returns a prime p and the description of a p-order group on the input of a security parameter 1^λ) and a family $\mathcal{H}$ of hash functions, the (labeled) short Cramer–Shoup encryption scheme [ABP15] consists of the following algorithms.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: generate $(p, \mathbb{G} = \langle g \rangle) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and $H \leftarrow_\$ \mathcal{H}$. Set and return $pp \leftarrow (p, \mathbb{G}, g, H)$.

$\mathsf{KG}(pp) \to (pk, sk)$: $sk \leftarrow (\zeta, \alpha, \beta, \alpha', \beta') \leftarrow_\$ \mathbb{Z}_p^5$. Compute $pk \leftarrow (pp, h, \gamma, \delta)$ $\leftarrow \left(pp, g^\zeta,\ g^\alpha h^\beta,\ g^{\alpha'} h^{\beta'}\right)$. (Parameters pp may further be omitted in the syntax.) Return (pk, sk).

$\mathsf{Enc}^{\ell}(pk, m \in \mathbb{G}; r \in \mathbb{Z}_p) \to C$: compute $U \leftarrow g^r$, $E \leftarrow h^r m$, $\xi \leftarrow H(U, E, \ell)$ and $V \leftarrow \left(\gamma \delta^{\xi}\right)^r$. Set and return $C \leftarrow (U, E, V)$.

$\mathsf{Dec}^{\ell}(sk, C) \to m/\perp$: compute $m \leftarrow U/E^{\zeta}$ and $\xi \leftarrow H(U, E, \ell)$. If $V = U^{\alpha+\xi\alpha'} (E/m)^{\beta+\xi\beta'}$ then return m, else return $\perp$.

Abdalla, Benhamouda and Pointcheval proved that this scheme is IND-PCA secure if the DDH assumption over G holds and if $\mathcal{H}$ is second-preimage resistant.

RCCA Security. Indistinguishability under Replayable Chosen-Ciphertext Attacks [CKN03] (RCCA) is a relaxation of the classical CCA security which tolerates a mild form of malleability. It allows for the re-randomization of ciphertexts while still providing strong security guarantees. Formally, an encryption scheme is IND-RCCA secure if for every $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,

$$\Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); (pk, sk) \leftarrow \mathsf{KG}(pp) \\ (st, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}(sk, \perp, \perp, \cdot)}(pk) \\ b \leftarrow_{\$} \{0,1\}; C^* \leftarrow \mathsf{Enc}(pk, m_b) \\ b' \leftarrow_{\$} \mathcal{A}^{\mathcal{O}(sk, m_0, m_1, \cdot)}(st, C^*) \\ \text{return } (b, b') \end{array}\right]$$

is negligibly close to $1/2$. Oracle $\mathcal{O}$, with state (sk, m_0, m_1), replies to C queries by first computing $m \leftarrow \mathsf{Dec}(sk, C)$. If $m = m_0$ or $m = m_1$, it returns a special string replay indicating that C encrypts one of the challenge messages, otherwise it returns m.

Publicly Verifiable Encryption. A public-key encryption scheme is verifiable if there exists a deterministic algorithm $\mathsf{Vf}(pk, C) \to b \in \{0, 1\}$ such that no efficient adversary can, on the input of pk and with non-negligible probability, produce a ciphertext C such that $\mathsf{Vf}(pk, C) = 1$ and $\mathsf{Dec}(sk, C) = \perp$.

Re-randomizable Encryption. A public-key encryption scheme is re-randomizable if there exists an algorithm $\mathsf{Rand}(pk, C) \to \hat{C}$ which computes, from a public key pk and a ciphertext C, a new ciphertext $\hat{C}$. It is assumed to return $\perp$ if any of its inputs are ill-formed.

Unlinkability. A re-randomizable encryption scheme is perfectly unlinkable [PR07, CKLM12, LPQ17] if the re-randomized valid ciphertexts have the same distribution as fresh encryptions of their underlying plaintexts.

Threshold Decryption. Threshold encryption schemes [CDN01] are schemes in which decryption keys are shared between several parties. For a given ciphertext, each party can compute a decryption share with her key share, and a threshold number of those decryption shares is necessary to reconstruct the plaintext. If there are n parties, the security requirement of a t-out-of-n scheme is that no information about the plaintext can be inferred from less than $t+1$ shares. In the RCCA variant of this security notion, during the second query phase (which targets a specific honest party), the challenger first decrypts the ciphertext of the query with the secret key it has generated, and checks whether it results in one of the challenge messages before answering with a decryption share if it is not the case.

Encryption Scheme of Faonio, Fiore, Herranz and Ràfols. Faonio, Fiore, Herranz and Ràfols [FFHR19] constructed a publicly verifiable, re-randomizable, structure-preserving encryption scheme which is secure under the matrix decisional Diffie–Hellman assumption [EHK+13]. The explicit version of their scheme under the SXDH assumption is given below.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $\Gamma \leftarrow (p,\ \mathbb{G}_1 = \langle g_1 \rangle,\ \mathbb{G}_2 = \langle g_2 \rangle,\ \mathbb{G}_T,\ e) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Generate a Groth–Sahai common reference string, i.e., $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1^2$ and $\mathbf{v}, \mathbf{w} \in \mathbb{G}_2^2$. Set and return $pp \leftarrow (\Gamma, \mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{w})$.

$\mathsf{KG}(pp) \to (pk, sk)$: generate $D, E \leftarrow_\$ \mathbb{Z}_p$, $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 \leftarrow_\$ \mathbb{Z}_p$, $\mathbf{F} \leftarrow_\$ \mathbb{Z}_p^2$ and $\mathbf{G} \leftarrow_\$ \mathbb{Z}_p^{2\times 3}$. Compute

$$\mathbf{M}_1 \leftarrow \begin{bmatrix} g_1^D & & & & & & \\ g_1^{\beta_1 D+\beta_2} & a_1 & b_1 & & & & \\ 1_{\mathbb{G}_1} & a_2 & b_2 & & & & \\ g_1^{F_{1,1}D+F_{2,1}} & & & a_1 & b_1 & & \\ 1_{\mathbb{G}_1} & & & a_2 & b_2 & & \\ g_1^{F_{1,2}D+F_{2,2}} & & & & & a_1 & b_1 \\ 1_{\mathbb{G}_1} & & & & & a_2 & b_2 \end{bmatrix}$$

and

$$\mathbf{M}_2 \leftarrow \begin{bmatrix} g_2^E & & & & & & & & \\ g_2^{\gamma_1 D+\gamma_2} & v_1 & w_1 & & & & & & \\ 1_{\mathbb{G}_2} & v_2 & w_2 & & & & & & \\ g_2^{G_{1,1}E+G_{2,1}} & & & v_1 & w_1 & & & & \\ 1_{\mathbb{G}_2} & & & v_2 & w_2 & & & & \\ g_2^{G_{1,2}E+G_{2,2}} & & & & & v_1 & w_1 & & \\ 1_{\mathbb{G}_2} & & & & & v_2 & w_2 & & \\ g_2^{G_{1,3}E+G_{2,3}} & & & & & & & v_1 & w_1 \\ 1_{\mathbb{G}_2} & & & & & & & v_2 & w_2 \end{bmatrix}.$$

For $i = 1, 2$, generate $A_i \leftarrow_\$ \mathbb{Z}_p$. Generate $\mathbf{K}_i \leftarrow_\$ \mathbb{Z}_p^{(5+2i)\times 1}$. Compute matrices $\mathbf{P}_i \leftarrow \mathbf{M}_i^{\mathrm{T}}\mathbf{K}_i$ and $\mathbf{C}_i \leftarrow A_i\mathbf{K}_i$. Set

$$\begin{aligned} pk \leftarrow \Big(&\Gamma, g_1^D, g_1^{\alpha_1 D+\alpha_2}, g_1^{\beta_1 D+\beta_2}, g_2^E, g_2^{\gamma_1 E+\gamma_2}, \\ &\begin{bmatrix} g_1^{F_{1,1}D+F_{2,1}} \\ g_1^{F_{1,2}D+F_{2,2}} \end{bmatrix}, \begin{bmatrix} g_1^{G_{1,1}D+G_{1,2}+G_{1,3}(\alpha_1 D+\alpha_2)} \\ g_1^{G_{2,1}D+G_{2,2}+G_{2,3}(\alpha_1 D+\alpha_2)} \end{bmatrix}, \\ &\begin{bmatrix} g_2^{G_{1,1}E+G_{2,1}} \\ g_2^{G_{1,2}E+G_{2,2}} \\ g_2^{G_{1,3}E+G_{2,3}} \end{bmatrix}, \begin{bmatrix} g_2^{F_{1,1}E+F_{1,2}} \\ g_2^{F_{2,1}E+F_{2,2}} \end{bmatrix}, A_1, A_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{C}_1, \mathbf{C}_2 \Big) \\ sk \leftarrow (&\Gamma, \alpha_1, \alpha_2, A_1, A_2, \mathbf{C}_1, \mathbf{C}_2)\,. \end{aligned}$$

Return (pk, sk). If ciphetext blinding is necessary as in Section 6.4.1, store also $\mathbf{G}$ in sk.

Matrices $\mathbf{M}_i$, $\mathbf{P}_i$ and $\mathbf{C}_i$, and scalars A_i are parameters for the quasi-adaptive non-interactive zero-knowledge proof system of Kiltz and Wee for linear spaces [KW15] with witness-sampleable distribution.

$\mathsf{Enc}(pk, m \in \mathbb{G}_1) \to C$: ∘ compute a tag, i.e.,

- $r, s \leftarrow_\$ \mathbb{Z}_p$
- $\mathbf{x}^{\mathrm{T}} \leftarrow \begin{bmatrix} g_1^{rD} & g_1^r & g_1^{r(\alpha_1 D+\alpha_2)} m \end{bmatrix}$
- $\mathbf{y}^{\mathrm{T}} \leftarrow \begin{bmatrix} g_2^{sE} & g_2^s \end{bmatrix}$

∘ compute a smooth projective hash π of the tag, i.e.,

$$\begin{aligned} \pi \leftarrow & e\left(g_1^{r(\beta_1 D+\beta_2)}, g_2\right) \prod_{i=1,2} e\left(g_1^{r(F_{1,i}D+F_{2,i})}, y_i\right) \\ & e\left(g_1, g_2^{s(\gamma_1 E+\gamma_2)}\right) \prod_{i=1,2,3} e\left(x_i, g_2^{s(G_{1,i}E+G_{2,i})}\right) \end{aligned}$$

∘ prove that π is well-formed, i.e.,

- $r_{av}, r_{bv}, r_{aw}, r_{bw}, \rho_1, \rho_2, \sigma_{i,j}, r_i, s_{k,j} \leftarrow_\$ \mathbb{Z}_p$ for $i, j \in \{1, 2\}$ and $k = 1, 2, 3$
- $\mathbf{c}(\pi) \leftarrow \left[\pi_{i,j}\right]_{i,j=1,2} f\left(\mathbf{a}^{r_{av}}, \mathbf{v}\right) f\left(\mathbf{b}^{r_{bv}}, \mathbf{v}\right) f\left(\mathbf{a}^{r_{aw}}, \mathbf{w}\right) f\left(\mathbf{b}^{r_{bw}}, \mathbf{w}\right)$, with $\pi_{i,j} \coloneqq \pi$ if $i = j = 2$ and $1_{\mathbb{G}_T}$ otherwise
- $\mathbf{c}_0 \leftarrow \begin{bmatrix} 1 & g_1^{r(\beta_1 D+\beta_2)} \end{bmatrix}^{\mathrm{T}} \mathbf{a}^{\rho_1}\mathbf{b}^{\rho_2}$
- $\mathbf{c}_{1,i} \leftarrow \begin{bmatrix} 1 & g_1^{r(F_{1,i}D+F_{2,i})} \end{bmatrix}^{\mathrm{T}} \mathbf{a}^{\sigma_{i,1}}\mathbf{b}^{\sigma_{i,2}}$ for $i = 1, 2$
- $\mathbf{d}_0 \leftarrow \begin{bmatrix} 1 & g_2^{s(\gamma_1 E+\gamma_2)} \end{bmatrix}^{\mathrm{T}} \mathbf{v}^{r_1}\mathbf{w}^{r_2}$
- $\mathbf{d}_{1,i} \leftarrow \begin{bmatrix} 1 & g_2^{s(G_{1,i}E+G_{2,i})} \end{bmatrix}^{\mathrm{T}} \mathbf{v}^{s_{i,1}}\mathbf{w}^{s_{i,2}}$ for $i = 1, 2, 3$

– $\mathbf{T} \leftarrow_{\$} \mathbb{Z}_p^{2\times 2}$

– $\Theta \leftarrow \begin{bmatrix} a_1^{T_{1,1}} b_1^{T_{2,1}} & a_2^{T_{1,1}} b_2^{T_{1,2}} g_1^{r_1} x_1^{s_{1,1}} x_2^{s_{2,1}} x_3^{s_{3,1}} \\ a_1^{T_{1,2}} b_1^{T_{2,2}} & a_2^{T_{2,1}} b_2^{T_{2,2}} g_1^{r_2} x_1^{s_{1,2}} x_2^{s_{2,2}} x_3^{s_{3,2}} \end{bmatrix}^{\mathrm{T}} =: \begin{bmatrix} \Theta_1 & \Theta_2 \end{bmatrix}$

– $\tilde{\Pi} \leftarrow \begin{bmatrix} v_1^{-T_{1,1}} w_1^{-T_{2,1}} & v_2^{-T_{1,1}} w_2^{-T_{2,1}} g_2^{\rho_1} y_1^{\sigma_{1,1}} y_2^{\sigma_{2,1}} \\ v_1^{-T_{1,2}} w_1^{-T_{2,2}} & v_2^{-T_{1,2}} w_2^{-T_{2,2}} g_2^{\rho_2} y_1^{\sigma_{1,2}} y_2^{\sigma_{2,2}} \end{bmatrix}$

– $\Pi_1 \leftarrow \tilde{\Pi}_1^{\mathrm{T}} \mathbf{v}^{r_{av}} \mathbf{w}^{r_{aw}}; \Pi_2 \leftarrow \tilde{\Pi}_2^{\mathrm{T}} \mathbf{v}^{r_{bv}} \mathbf{w}^{r_{bw}}; \Pi \leftarrow \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix}$

∘ prove that the commitments are well-formed, i.e., that $\begin{bmatrix} g_1^{Dr} & \mathbf{c}_0^{\mathrm{T}} & \mathbf{c}_{1,1}^{\mathrm{T}} & \mathbf{c}_{1,2}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is in span $(\mathbf{M}_1)$ and that $\begin{bmatrix} g_2^{Es} & \mathbf{d}_0^{\mathrm{T}} & \mathbf{d}_{1,2}^{\mathrm{T}} & \mathbf{d}_{1,2}^{\mathrm{T}} & \mathbf{d}_{1,3}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is in span $(\mathbf{M}_2)$. That is, for $j = 1, 2$, compute

$$\Psi_1 \leftarrow g_1^{rP_{1,1} + \sum_{i=1,2} \rho_i P_{1,1+i} + \sum_{i,j=1,2} \sigma_{i,j} P_{1,2i+j+1}}$$

$$\Psi_2 \leftarrow g_2^{sP_{2,1} + \sum_{i=1,2} r_i P_{2,1+i} + \sum_{i=1}^{3} \sum_{j=1}^{2} s_{i,j} P_{2,2i+j+1}}$$

∘ set and return

$$C \leftarrow \left(\mathbf{x}, \mathbf{y}, \mathbf{c}(\pi), \mathbf{c}_0, (\mathbf{c}_{1,i})_{i=1}^2, \mathbf{d}_0, (\mathbf{d}_{1,i})_{i=1}^3, \Pi, \Theta, (\Psi_i)_{i=1,2}\right).$$

A ciphertext comprises 14 $\mathbb{G}_1$ elements, 15 $\mathbb{G}_2$ elements and 4 $\mathbb{G}_T$ elements.

$\mathsf{Dec}(sk, C) \to m/\perp$: ∘ verify that the commitments are well-formed, i.e., that

$$e\left(\Psi_1, g_2^{A_1}\right) = e\left(g_1^{Dr}, g_2^{\mathbf{C}_{1,1}}\right) \prod_{i=1,2} e\left(\mathbf{c}_{0,i}, g_2^{\mathbf{C}_{1,1+i}}\right) \prod_{i,j=1,2} e\left(\mathbf{c}_{1,i,j}, g_2^{\mathbf{C}_{1,2i+j+1}}\right)$$

and that

$$e\left(g_1^{A_2}, \Psi_2\right) = e\left(g_1^{\mathbf{C}_{2,1}}, g_2^{Es}\right) \prod_{i=1,2} e\left(g_1^{\mathbf{C}_{2,1+i}}, \mathbf{d}_{0,i}\right) \prod_{i=1,2,3} \prod_{j=1,2} e\left(g_1^{\mathbf{C}_{2,2i+j+1}}, \mathbf{d}_{1,i,j}\right)$$

∘ verify that the opening to $\mathbf{c}(\pi)$ is the smooth projective hash of the tag, i.e., that

$$f\left(\mathbf{c}_0, \begin{bmatrix} 1 \\ g_2 \end{bmatrix}\right) \prod_{i=1,2} f\left(\mathbf{c}_{1,i}, \begin{bmatrix} 1 \\ y_i \end{bmatrix}\right)$$

$$f\left(\begin{bmatrix} 1 \\ g_1 \end{bmatrix}, \mathbf{d}_0\right) \prod_{i=1,2,3} f\left(\begin{bmatrix} 1 \\ x_i \end{bmatrix}, \mathbf{d}_{1,i}\right)$$

$$= f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right) f\left(\Theta_1, \mathbf{v}\right) f\left(\Theta_2, \mathbf{w}\right) \mathbf{c}(\pi)$$

- if both verifications succeed, compute and return $m \leftarrow x_3 x_1^{-\alpha_1} x_2^{-\alpha_2}$, else return $\perp$.

$\mathsf{Vf}(pk, C) \rightarrow b$: do the same verifications as algorithm Dec (they do not require knowledge of sk, only of pk). If they succeed, return 1, else return 0.

$\mathsf{Rand}(pk, C) \rightarrow \hat{C}$: first verify that the ciphertext is valid. Next, re-randomize the tag, and all proofs and commitments, i.e., do

- $\hat{r}, \hat{s} \leftarrow_\$ \mathbb{Z}_p$
- $\hat{\mathbf{x}} \leftarrow \mathbf{x} \begin{bmatrix} g_1^{\hat{r}D} & g_1^{\hat{r}} & g_1^{\hat{r}(\alpha_1 D + \alpha_2)} \end{bmatrix}^{\mathrm{T}}$
- $\hat{\mathbf{y}} \leftarrow \mathbf{y} \begin{bmatrix} g_2^{\hat{s}E} g_2^{\hat{s}} \end{bmatrix}^{\mathrm{T}}$
- re-randomize π, i.e., compute

$$\hat{\pi} \leftarrow \pi \cdot e\left(g_1^{\hat{r}(\beta_1 D + \beta_2)}, g_2\right) \prod_{i=1,2} e\left(g_1^{\hat{r}(F_{1,i} D + F_{2,i})}, y_i\right) e\left(g_1, g_2^{\hat{s}(\gamma_1 E + \gamma_2)}\right) \prod_{i=1,2,3} e\left(x_i, g_2^{\hat{s}(G_{1,i} E + G_{2,i})}\right)$$

- $\hat{r}_{av}, \hat{r}_{bv}, \hat{r}_{aw}, \hat{r}_{bw}, \hat{\rho}_1, \hat{\rho}_2, \hat{\sigma}_{i,j}, \hat{r}_i, \hat{s}_{k,j} \leftarrow_\$ \mathbb{Z}_p$ for $i, j \in \{1, 2\}$ and $k = 1, 2, 3$
- $\hat{\mathbf{c}}(\pi) \leftarrow f\left(\mathbf{a}^{\hat{r}_{av}}, \mathbf{v}\right) f\left(\mathbf{b}^{\hat{r}_{bv}}, \mathbf{v}\right) f\left(\mathbf{a}^{\hat{r}_{aw}}, \mathbf{w}\right) f\left(\mathbf{b}^{\hat{r}_{bw}}, \mathbf{w}\right) \mathbf{c}(\pi)$
- $\hat{\mathbf{c}}_0 \leftarrow \mathbf{c}_0 \mathbf{a}^{\hat{\rho}_1} \mathbf{b}^{\hat{\rho}_2}$
- $\hat{\mathbf{c}}_{1,i} \leftarrow \mathbf{c}_{1,i} \mathbf{a}^{\hat{\sigma}_{i,1}} \mathbf{b}^{\hat{\sigma}_{i,2}}$ for $i = 1, 2$
- $\hat{\mathbf{d}}_0 \leftarrow \mathbf{d}_0 \mathbf{v}^{\hat{r}_1} \mathbf{w}^{\hat{r}_2}$
- $\hat{\mathbf{d}}_{1,i} \leftarrow \mathbf{d}_{1,i} \mathbf{v}^{\hat{s}_{i,1}} \mathbf{w}^{\hat{s}_{i,2}}$ for $i = 1, 2, 3$
- $\hat{\mathbf{T}} \leftarrow_\$ \mathbb{Z}_p^{2\times 2}$
- $\hat{\Theta} \leftarrow \Theta \circ \begin{bmatrix} a_1^{\hat{T}_{1,1}} b_1^{\hat{T}_{2,1}} & a_2^{\hat{T}_{1,1}} b_2^{\hat{T}_{1,2}} g_1^{\hat{r}_1} x_1^{\hat{s}_{1,1}} x_2^{\hat{s}_{2,1}} x_3^{\hat{s}_{3,1}} \\ a_1^{\hat{T}_{1,2}} b_1^{\hat{T}_{2,2}} & a_2^{\hat{T}_{2,1}} b_2^{\hat{T}_{2,2}} g_1^{\hat{r}_2} x_1^{\hat{s}_{1,2}} x_2^{\hat{s}_{2,2}} x_3^{\hat{s}_{3,2}} \end{bmatrix}^{\mathrm{T}}$
- compute

$$\hat{\Pi} \leftarrow \Pi \circ \begin{bmatrix} v_1^{-\hat{T}_{1,1}} w_1^{-\hat{T}_{2,1}} & v_2^{-\hat{T}_{1,1}} w_2^{-\hat{T}_{2,1}} g_2^{\hat{\rho}_1} y_1^{\hat{\sigma}_{1,1}} y_2^{\hat{\sigma}_{2,1}} \\ v_1^{-\hat{T}_{1,2}} w_1^{-\hat{T}_{2,2}} & v_2^{-\hat{T}_{1,2}} w_2^{-\hat{T}_{2,2}} g_2^{\hat{\rho}_2} y_1^{\hat{\sigma}_{1,2}} y_2^{\hat{\sigma}_{2,2}} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} v^{\hat{r}_{av}} w^{\hat{r}_{aw}} \\ v^{\hat{r}_{bv}} w^{\hat{r}_{bw}} \end{bmatrix}$$

- $\hat{\Psi}_1 \leftarrow \Psi_1 \cdot g_1^{\hat{r} P_{1,1} + \sum_{i=1,2} \hat{\rho}_i P_{1,1+i} + \sum_{i,j=1,2} \hat{\sigma}_{i,j} P_{1,2i+j+1}}$

– $\hat{\Psi}_2 \leftarrow \Psi_2 \cdot g_2^{\hat{s}P_{2,1}+\sum_{i=1,2}\hat{r}_i P_{2,1+i}+\sum_{i=1}^{3}\sum_{j=1}^{2}\hat{s}_{i,j}P_{2,2i+j+1}}$

– set and return $\hat{C} \leftarrow \left(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{c}}(\pi),\ \hat{\mathbf{c}}_0,\ (\hat{\mathbf{c}}_{1,i})_{i=1}^{2},\ \hat{\mathbf{d}}_0,\ \left(\hat{\mathbf{d}}_{1,i}\right)_{i=1}^{3},\right.$
$\left.\hat{\Pi},\ \hat{\Theta},\ \left(\hat{\Psi}_{i,j}\right)_{i,j=1}^{2}\right)$.

Threshold Decryption. The scheme of Faonio et al. can be turned into a t-out-of-n scheme by doing a t-out-of-n Shamir share [Sha79] of α_1 and α_2. Before partially decrypting the plaintext, each shareholder first verifies the validity of the ciphertext, which is possible as the scheme is publicly verifiable. If the ciphertext is invalid, the shareholder returns $\perp$. As a result, any $t+1$ shares are sufficient to decrypt ciphertexts, but no information about the plaintexts can be inferred with only t shares.

6.1.7 Smooth Projective Hash Functions

Smooth Projective Hash Functions (SPHFs) [CS02] are hash functions defined over a set $\mathcal{X}$, and which can be evaluated in two ways on a subset $\mathcal{L} \subseteq \mathcal{X}$. An SPHF can be evaluated on $\mathcal{X}$ using a hashing key hk, which can be seen as a private key. On $\mathcal{L}$, it can also be evaluated with a projective key hp, which can be seen a public key, and a witness of membership to $\mathcal{L}$.

Syntax [BBC$^+$13]. A Smooth Projective Hash Function over a language $\mathcal{L} \subseteq \mathcal{X}$ is defined by five algorithms: $\mathsf{Setup}\left(1^\lambda\right) \to pp$ which generates public parameters, $\mathsf{HashKG}(\mathcal{L}) \to hk$ which generates a hashing key for a Language $\mathcal{L}$, $\mathsf{ProjKG}(hk, \mathcal{L}, C) \to hp$ which derives a projective key hp from hk depending on a word $C \in \mathcal{X}$, $\mathsf{Hash}(hk, \mathcal{L}, C) \to \mathfrak{H} \in \mathscr{H}$ which returns a hash value for any word $C \in \mathcal{X}$, and $\mathsf{ProjHash}(hp, \mathcal{L}, C, w) \to \mathfrak{H} \in \mathscr{H}$ which returns a hash value on a word $C \in \mathcal{L}$ given a projective key and a witness w for the membership of C. The public parameters are given as implicit input to all the other algorithms. When $\mathcal{L}$ is the language of ciphertexts of a given message with a certain scheme, the public parameters typically contain the encryption key, and may even include the secret key to efficiently test language membership.

In the definition above due to Gennaro and Lindell [GL03], the projective keys depend on words C. In the original definition of SPHFs [CS02], however, the projective keys are *word-independent*.

Correctness. An SPHF is said to be correct if the hash and the projective hash values match for all words in $\mathcal{L}$, given valid membership witnesses. That is, for all $C \in \mathcal{L}$ with witness w, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$, $hk \leftarrow \mathsf{HashKG}(\mathcal{L})$, $hp \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, C)$, $\mathsf{Hash}(hk, \mathcal{L}, C) = \mathsf{ProjHash}(hp, \mathcal{L}, C, w)$.

Adaptive Smoothness. An SPHF is smooth if its hash values on all $C \in \mathcal{X} \setminus \mathcal{L}$ are statistically indistinguishable from uniformly random values. Katz and Vaikuntanathan introduced [KV11] SPHFs (KV-SPHFs) with word-independent projective keys, and for which smoothness holds even if the words depend on the projective keys. KV-SPHFs are the most flexible kind since the words on which they are evaluated can be chosen even *after* computing and publishing the projective keys. In this sense, they are *adaptive*.

A KV-SPHF is smooth if its hash values on all $C \in \mathcal{X} \setminus \mathcal{L}$ are statistically indistinguishable from uniformly random values, even if C depends on projective keys. Formally, a KV-SPHF is ε-smooth [BBC+13] if, for any map $\mathfrak{f}$ onto $\mathcal{X} \setminus \mathcal{L}$, the following two distributions are ε-close:

$$\{(hp, \mathfrak{H}) \colon \ hk \leftarrow \mathsf{HashKG}(\mathcal{L}), hp \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, \bot), \mathfrak{H} \leftarrow \mathsf{Hash}(hk, \mathcal{L}, \mathfrak{f}(hp))\}$$
$$\{(hp, \mathfrak{H}) \colon \ hk \leftarrow \mathsf{HashKG}(\mathcal{L}), hp \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, \bot), \mathfrak{H} \leftarrow_{\$} \mathscr{H}\}.$$

A KV-SPHF is *perfectly smooth* if it is 0-smooth.

Designated-Verifier Proofs of Membership. SPHFs can be seen as designated-verifier proofs of membership to $\mathcal{L}$ [ACP09, BPV12]. Indeed, to prove that a word C is in $\mathcal{L}$, a designated verifier can generate a key hk and compute a projective key hp which she sends to the prover. Given a membership witness w, the prover evaluates the SPHF on (C, w) with hp and sends the result as a proof to the verifier. To verify the proof, the verifier computes the SPHF on C using hk and accepts if and only if the result matches the proof value. The correctness of the protocol follows from the correctness of the SPHF, and its soundness from the adaptive smoothness of the SPHF.

KV-SPHF for Short Cramer–Shoup Ciphertexts. For a fixed tuple $(p, \mathbb{G} = \langle g \rangle)$, denote by $\mathcal{L}_m^\ell$ the language

$$\left\{C \colon \exists r, \ \ C = (U, E, V) = \left(g^r, h^r m, \left(\gamma \delta^{H(g^r, h^r m, \ell)}\right)^r\right)\right\}.$$

Given a group generator G, $\Gamma \leftarrow \mathsf{G}(p, \mathbb{G} = \langle g \rangle)$ and $m \in G$, the following scheme, due to Abdalla, Benhamouda and Pointcheval [ABP15], is a perfectly smooth KV-SPHF for $\mathcal{L}_m^\ell$.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: generate $sk \leftarrow (\zeta, \alpha, \beta, \alpha', \beta') \leftarrow_{\$} \mathbb{Z}_p^5$. Compute $pk \leftarrow (h, \gamma, \delta) \leftarrow \left(g^\zeta, \ g^\alpha h^\beta, \ g^{\alpha'} h^{\beta'}\right)$. Return (Γ, pk, sk).

$\mathsf{HashKG}\left(\mathcal{L}_m^\ell\right) \to hk$: return $hk \leftarrow (\lambda, \mu, \nu, \theta) \leftarrow_{\$} \mathbb{Z}_p^4$.

$\mathsf{ProjKG}\left(hk, \mathcal{L}_m^\ell, \perp\right) \to hp$: return $hp \leftarrow \left(g^\lambda h^\nu \gamma^\theta,\ g^\mu \delta^\theta\right)$. Note that hp actually depends on neither m nor ℓ.

$\mathsf{Hash}\left(hk, \mathcal{L}_m^\ell, C\right) \to \mathfrak{H}$: compute $\xi \leftarrow H(U, E, \ell)$. Return $\mathfrak{H} \leftarrow U^{\lambda+\mu\xi}\,(E/m)^\nu\, V^\theta$.

$\mathsf{ProjHash}\left(hp, \mathcal{L}_m^\ell, C, r\right) \to \mathfrak{H}$: compute $\xi \leftarrow H(U, E, \ell)$. Return $\mathfrak{H} \leftarrow \left(hp_1 hp_2^\xi\right)^r$.

6.1.8 Key-Derivation Functions

A Key-Derivation Function (KDF) computes pseudorandom keys of appropriate length from a source key material which is not uniformly distributed, or which still has high entropy despite partial adversarial knowledge. The results can then be used as secret keys for cryptosystems.

Syntax. A key-derivation function [Kra10] $\mathsf{KDF}(SKM, XTS, CTX, L) \to K$, takes as input a source key material SKM, an extractor-salt value XTS, some context information CTX and a length L, and returns an L-bit string K.

Security. The security of key-derivation functions can be defined w.r.t. to a specific source of keying material which returns a material SKM and a public piece of information inf.

Definition 6.1.9 ([Kra10, Definition 7])**.** *A key-derivation function* KDF*, which supports salt values from a finite set* Σ*, is secure w.r.t. a source (of key material)* SKM *if for every efficient adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{ll} & (SKM, inf) \leftarrow \mathsf{SKM}; Q \leftarrow \emptyset \\ & XTS \leftarrow_\$ \Sigma \\ & (st, CTX^*, \ell^*) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{KDF}(Q, SKM, XTS, \cdot)}(inf, XTS) \\ & b \leftarrow_\$ \{0,1\} \\ & \textit{if } b = 0 \\ & \quad K^* \leftarrow_\$ \{0,1\}^\ell \\ b = b' : & \textit{else} \\ & \quad K^* \leftarrow \mathsf{KDF}(SKM, XTS, CTX^*, \ell^*) \\ & b' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{KDF}(Q, SKM, XTS, \cdot)}(st, K^*) \\ & \textit{if } CTX^* \in Q \\ & \quad b' \leftarrow_\$ \{0,1\} \\ & \quad \textit{return } (b, b') \\ & \textit{return } (b, b') \end{array}\right]$$

is negligibly close to 1/2, with $\mathcal{O}.\mathsf{KDF}$ *an oracle which, on input* (SKM, XTS)*, replies to a* (CTX, ℓ) *query with* $\mathsf{KDF}(SKM, XTS, CTX, \ell)$ *and then adds* CTX *to* Q*.*

The previous definition is w.r.t. to a specific source, but it can be extended to all sources which return materials with enough entropy even when conditioned on the public information. Formally, for an integer m, a source SKM is a m-entropy source if for all s in the range of key materials and all i in the support of public information returned by SKM, $\Pr[SKM = s|inf = i] \leq 2^{-m}$. A key-derivation function is then said to be *m-entropy secure* if it is secure w.r.t. all m-entropy sources.

Krawczyk's Key-Derivation Function. Krawczyk gave [Kra10, Section 4.2] a secure construction of KDFs from Hash-based Message Authentication Codes [BCK96] (HMAC) which follows the extract-then-expand paradigm.

More precisely, let $\{H_\kappa\}_{\kappa\in\{0,1\}^k}$ be a family of Merkle–Damgård hash functions with k-bit outputs which is based on a family of compression functions $\{h_\kappa\}_\kappa$ (think of SHA-512). Assume (single-keyed) HMAC to be built from NMAC [BCK96], and NMAC to be built from $\{H_\kappa\}_\kappa$. If $\{h_\kappa\}_\kappa$ is a family of pairwise-independent compression functions, and $\{H_\kappa\}_\kappa$ is collision-resistant against linear-size circuits, then [Kra10, Corollary 9] NMAC truncated by c bits is a $\left(m, (n+2)2^{-c/2}\right)$-statistical extractor on n-block inputs. If $\{h_\kappa\}_\kappa$ is modeled as a family of random functions independent from the source, then the same result applies to HMAC.

Moreover, given another family of compression functions $\{g_\kappa\}_\kappa$ (think of SHA-256), if $g_\cdot$ is a Pseudo-Random Function (PRF), and if $\hat{g}\colon (K, m) \to g_m(K)$ is a PRF under a class of affine related-key attacks (defined by the inner and outer pads), then [Bel06, Theorem 3.3, Lemma 5.2] HMAC is a (q_P, ε_P)-PRF for q_P, ε_P explicited in Bellare's paper [Bel06].

Krawczyk's theorem [Kra10, Theorem 1] implies that HMAC is a $\left(q_P, \varepsilon_P + (n+2)2^{-c/2}\right)$-secure KDF w.r.t. to sources with min-entropy at least m. Throughout this chapter, it is assumed that n and c are functions of λ such that $n2^{-c/2}$ is negligible in λ. The construction is described in Algorithm 6.1.2. Therein, given two integers $n \geq d \geq 1$ and $x \in \{0,1\}^n$, $[x]_d$ denotes the sub-string of x consisting of its first d bits.

6.2 Malleable Non-Interactive Proofs

As the construction in Section 6.4 heavily relies on malleability and non-interactive zero-knowledge proofs, this section recalls the definition of proofs which are still sound under "controlled malleability". These proofs allow to compute, from a proof π on a word x, a new proof π' on a transformation $T_x(x)$ of x without the knowledge of a witness for $T_x(x)$, but only if the transformation belongs to a class of "allowed" transformations. The soundness of the proof system can then be defined w.r.t. this class of transformations. In addition to that, the soundness definition can even be extended to consider

Algorithm 6.1.2 Krawczyk's HMAC-based KDF.

Require: (SKM, XTS, CTX, L) and HMAC based on a hash function with output length k
Ensure: $K \in \{0,1\}^L$
  $t \leftarrow \lceil L/k \rceil$
  $d \leftarrow L \bmod k$
  $PRK \leftarrow \mathsf{HMAC}(XTS, SKM)$
  $K(1) \leftarrow \mathsf{HMAC}(PRK, CTX \| 0)$
  for $i = 1$ to $t - 1$ **do**
    $K(i+1) \leftarrow \mathsf{HMAC}(PRK, K(i) \| CTX \| i)$
  end for
  $K \leftarrow K(1) \| \cdots \| [K(t)]_d$
  return K

cases in which proofs are simulatable but remain sound under this controlled malleability.

Chase et al. [CKLM12] gave a definition of proof systems that are extractable under controlled malleability and a generic construction based on signatures and extractable proof systems. This section gives a similar definition which only requires soundness and then a generic construction from signatures and proof system that are only extractable in a sense defined below. The reason is that the EPAD construction in Section 6.4 uses Groth–Sahai proofs from which group elements can be extracted but not exponents.

Syntax of Non-Interactive Proof Systems. Recall from Section 2.6.2 that a non-interactive proof system for a language $\mathcal{L}$ (with corresponding relation $\mathcal{R}$) consists of an algorithm $\mathsf{Setup}\left(1^\lambda\right) \to pp$ which returns public parameters, an algorithm $\mathsf{CRSGen}(pp) \to crs$ that returns a common reference string, an algorithm $\mathsf{Prove}(crs, x, w) \to \pi$ which computes a proof on the input of a word x and of a witness w, and an algorithm $\mathsf{Vf}(crs, x, \pi) \to b \in \{0,1\}$ which returns a bit indicating whether the proof is considered valid.

6.2.1 Transformations

A *transformation* is an efficiently computable function $T := (T_x, T_w) \colon \mathcal{R} \to \mathcal{R}$. A relation $\mathcal{R}$ is said to be *closed* under T if for any $(x, w) \in \mathcal{R}$, $T(x, w) \in R$. Transformation T is then said to be *admissible* for $\mathcal{R}$. A class $\mathcal{T}$ of transformations is *allowable* for $\mathcal{R}$ if for every transformation $T \in \mathcal{T}$, T is admissible for $\mathcal{R}$.

A non-interactive proof system for a relation $\mathcal{R}$ is *malleable* [CKLM12] w.r.t. a class $\mathcal{T}$ of allowable transformations for $\mathcal{R}$ if there exists an algorithm $\mathsf{Eval}(crs, T, x, \pi) \to \pi'$ (word x may further be omitted from the syntax)

which compute a proof π' for $T_x(x)$ from a valid proof π for x, without the knowledge of $T_w(w)$.

6.2.2 Simulation Soundness under Controlled Malleability

In certain cases, it is necessary to be able to simulate proofs while still ensuring that no PPT algorithm can compute new valid proofs for false statements without a trapdoor. This property is commonly known as simulation soundness. However, a proof system cannot a priori be both malleable and simulation sound as malleability allows to compute proofs on transformed words without the knowledge of a witness. Chase et al. [CKLM12] put forth a relaxed version of simulation soundness which allows for malleability while guaranteeing a meaningful form of soundness; namely an adversary cannot compute (without trapdoor) a valid proof for the membership of x to $\mathcal{L}$ if $x \notin \mathcal{L}$ and x is not the image under an admissible transformation (from a pre-determined class) of a word for which it was given a simulated proof. Their definition requires the proof system to be a proof of knowledge (i.e., it requires extractability), but the following definition is rather for proof of statements.

Formal Definition.

Let $(\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{SS\_CRSGen}, \mathsf{SimEval})$ be a non-interactive zero-knowledge proof system for a language $\mathcal{L}$ which is malleable w.r.t. a class $\mathcal{T}$ of allowable transformations for the relation $\mathcal{R}$ relative to $\mathcal{L}$ (the trapdoor CRS generation algorithm is here denoted $\mathsf{SS\_CRSGen}$). The proof system is said to satisfy *Controlled-Malleable (CM) simulation soundness* w.r.t. $\mathcal{T}$ if for all $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{Vf}(crs, x, \pi) = 1 & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ \wedge (x, \pi) \notin Q \wedge x \notin \mathcal{L} & (crs, \tau) \leftarrow \mathsf{SS\_CRSGen}(pp) \\ \wedge \forall (T, x', *) \in \mathcal{T} \times Q, & Q \leftarrow \emptyset \\ \quad x \neq T_x(x') & (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Sim}(Q, crs, \tau, \cdot)}(crs) \end{array}\right]$$

is negligible, with $\mathcal{O}.\mathsf{Sim}$ an oracle which computes simulated proofs π on inputs x and adds (x, π) to Q.

6.2.3 Generic Construction

Let $\mathcal{R}$ be an efficient relation with corresponding language $\mathcal{L}$. Consider a class $\mathcal{T}$ of allowable transformation for $\mathcal{R}$ which contains the identity and for which membership can be efficiently tested. Consider also an existentially unforgeable signature scheme $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Sign}, \mathsf{Vf})$. Let $\hat{\Pi} =$

$(\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf})$ be a sound witness-indistinguishable proof system for the language

$$\{(x := (x_1, x_2), vk) : \exists\,(w, x_1', T, \sigma)\,, (x, w) \in \mathcal{R} \vee \mathsf{SIG.Vf}\,(vk, x_1', \sigma) = 1 \wedge x = T_x\,(x_1', x_2) \wedge T \in \mathcal{T}\}\,.$$

Suppose that it is partially extractable in the sense that there exists a trapdoor-setup algorithm which returns a trapdoor and a CRS indistinguishable from the output of CRSGen; and an extraction algorithm such that no PPT adversary has significant probability of computing a valid proof π on a word (x, vk), where the value $(*, x_1', *, \sigma)$ returned by the extractor on the input of the CRS, the trapdoor and (x, π) is such that $\mathsf{SIG.Vf}(vk, x_1', \sigma) = 0$. Let $\hat{\mathcal{T}}$ be a class of transformations such that for all $T' \in \mathcal{T}$, there exists $\hat{T} \in \hat{\mathcal{T}}$ such that $\hat{T}_x \colon (x, vk) \mapsto (T_x'(x), vk)$ and $\hat{T}_w \colon (w, x_1', T, \sigma) \mapsto (T_w'(w), x_1', T' \circ T, \sigma)$; and suppose that $\hat{\Pi}$ is malleable w.r.t. the class $\hat{\mathcal{T}}$.

Consider then Π, the following proof system inspired by the scheme of Chase et al. [CKLM12, Section 3]:

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $pp_{\hat{\Pi}} \leftarrow \hat{\Pi}.\mathsf{Setup}\left(1^\lambda\right)$ and $pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}\left(1^\lambda\right)$. Set and return $pp \leftarrow (pp_{\hat{\Pi}}, pp_{\mathsf{SIG}})$.

$\mathsf{CRSGen}(pp) \to crs$: run $crs' \leftarrow \hat{\Pi}.\mathsf{CRSGen}\,(pp_{\hat{\Pi}})$ and $(vk, sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$. Set and return $crs \leftarrow (crs', vk)$.

$\mathsf{Prove}(crs, x, w) \to \pi$: run $\pi \leftarrow \hat{\Pi}.\mathsf{Prove}\,(crs', (x, vk),\ (w, \perp))$. Return π.

$\mathsf{Vf}(crs, x, \pi) \to b \in \{0, 1\}$: return $\hat{\Pi}.\mathsf{Vf}\,(crs', (x, vk), \pi)$.

$\mathsf{Eval}(crs, T, x, \pi) \to \pi'$: return $\hat{\Pi}.\mathsf{Eval}\left(crs, \hat{T}, x, \pi\right)$.

Under the above assumptions, Π is complete, zero-knowledge and CM simulation sound w.r.t. $\mathcal{T}$. Indeed, it suffices to define $\mathsf{SS\_CRSGen}$ as CRSGen except that it returns sk as trapdoor, and Sim as an algorithm which signs x with sk and honestly proves the knowledge of $(\perp, x, \mathrm{id}, \sigma)$. The witness indistinguishability of $\hat{\Pi}$ implies that Π is zero-knowledge. The simulation soundness of Π stems from the fact that, since $\hat{\Pi}$ is sound, an adversary can win the game only if it can compute a valid signature on a word x_1' such that $(x_1', *)$ was never queried. However, as $\hat{\Pi}$ is partially extractable, that would contradict the existential unforgeability of SIG.

6.2.4 Strong Derivation Privacy

In addition to the completeness, (CM simulation) soundness and zero-knowledge property for proof systems, a malleable proof system should also satisfy

derivation privacy. It captures the idea that given $(x, w) \in \mathcal{R}$, proofs on a word $T_x(x)$ computed with Prove and $T_w(w)$ as a witness should be indistinguishable from those computed with Eval and a valid proof π for x. Note that re-randomizable malleable proofs necessarily satisfy derivation privacy [CKLM12, Theorem 2.7] (randomize then evaluate). An even *stronger* notion of derivation privacy is that the proof computed with Eval should be indistinguishable from those computed by the zero-knowledge simulator. Following a theorem of Chase et al. [CKLM12, Theorem 3.4], the above construction Π satisfies strong derivation privacy if $\hat{\Pi}$ satisfies derivation privacy.

6.2.5 Groth–Sahai Proofs

Groth and Sahai (GS) designed a practical, non-interactive witness-indistinguishable proof system for a wide class of statements in bilinear groups [GS08]. More precisely, their system allows to prove the existence of values which *simultaneously* satisfy pairing product equations, multi-scalar multiplication equations in the source groups, and quadratic equations in $\mathbb{Z}_p$. That is, given integers $m, n, m', n' \geq 1$, the GS proof system allows to prove that there exists $\mathbf{X} \in \mathbb{G}_1^m$, $\mathbf{Y} \in \mathbb{G}_2^n$, $\mathbf{x} \in \mathbb{Z}_p^{m'}$ and $\mathbf{y} \in \mathbb{Z}_p^{n'}$ which satisfy sets of equations of the form

$$\prod_{i=1}^{n} e\left(A_i, Y_i\right) \prod_{i=1}^{m} e\left(X_i, B_i\right) \prod_{i=1}^{m} \prod_{j=1}^{n} e\left(X_i, Y_i\right)^{\gamma_{ij}} = Z_T$$

$$\prod_{i=1}^{n'} \tilde{A}_i^{y_i} \prod_{i=1}^{m} X_i^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n'} X_i^{\tilde{\gamma}_{ij} y_j} = Z_1$$

$$\prod_{i=1}^{n} Y_i^{a_i} \prod_{i=1}^{m'} \tilde{B}_i^{x_i} \prod_{i=1}^{m'} \prod_{j=1}^{n} Y_j^{\hat{\gamma}_{ij} x_i} = Z_2$$

$$\sum_{i=1}^{n'} \tilde{a}_i y_i + \sum_{i=1}^{m'} \tilde{b}_i x_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} \gamma'_{ij} x_i y_j = z \mod p,$$

with the other values being public. The set of public values for which all sets of equations are satisfiable is further denoted $\mathcal{L}_{\mathrm{GS}}$. Multiscalar multiplication equations in G_2 are further omitted (i.e., $a_i \coloneqq 0, \tilde{B}_i \coloneqq 1_{\mathbb{G}_2}, \hat{\gamma}_{ij} \coloneqq 0, Z_2 \coloneqq 1_{\mathbb{G}_2}$) as there are not of interest in this thesis, and as they are purely symmetric to equations in $\mathbb{G}_1$.

Their construction is given in the Common-Reference String (CRS) model. The CRS is generated in either of two modes: a *soundness mode* in which

case the system is perfectly sound, and a *witness-indistinguishability mode* in which case the system is perfectly witness-indistinguishable. Under standard assumptions over bilinear groups, the two types of CRSs are indistinguishable.

Instantiation under the SXDH Assumption.

Common Reference String. In the instantiation under the SXDH assumption, the GS CRS contains two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1^2$ and two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{G}_2^2$. In a soundness setting, $\mathbf{b} \in \text{span}(\mathbf{a})$ and $\mathbf{w} \in \text{span}(\mathbf{v})$. Given the discrete-logarithm relation between a_2 and a_1, group-element variables can actually be efficiently extracted (which is not the case for $\mathbb{Z}_p$ elements as the discrete-logarithm problem is hard). In a witness-indistinguishable setting, $\mathbf{a}$ and $\mathbf{b}$ are linearly independent ($\mathbf{b} \leftarrow \mathbf{a}^t \begin{bmatrix} 1 & g_2^{-1} \end{bmatrix}^{\mathrm{T}}$ for $t \leftarrow_\$ \mathbb{Z}_p$), and so are $\mathbf{v}$ and $\mathbf{w}$. In either setting, define $\mathbf{d} \coloneqq \mathbf{b} \begin{bmatrix} 1 & g_2 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{u} \coloneqq \mathbf{w} \begin{bmatrix} 1 & g_2 \end{bmatrix}^{\mathrm{T}}$.

Proof Computation. To compute a GS proof of satisfiability of equations of the form above,

- commit to all variables, i.e.,
  - for all $i \in [\![m]\!]$, generate $\rho_i, \sigma_i \leftarrow_\$ \mathbb{Z}_p$
  - $\mathbf{c}(X_i) \leftarrow \begin{bmatrix} 1 & X_i \end{bmatrix}^{\mathrm{T}} \mathbf{a}^{\rho_i} \mathbf{b}^{\sigma_i}$
  - for all $i \in [\![n]\!]$, generate $\rho'_i, \sigma'_i \leftarrow_\$ \mathbb{Z}_p$
  - $\mathbf{c}(Y_i) \leftarrow \begin{bmatrix} 1 & Y_i \end{bmatrix}^{\mathrm{T}} \mathbf{v}^{\rho'_i} \mathbf{w}^{\sigma'_i}$
  - for all $i \in [\![m']\!]$, generate $r_i \leftarrow_\$ \mathbb{Z}_p$
  - $\mathbf{c}(x_i) \leftarrow \mathbf{d}^{x_i} \mathbf{a}^{r_i}$
  - for all $i \in [\![n']\!]$, generate $r'_i \leftarrow_\$ \mathbb{Z}_p$
  - $\mathbf{c}(y_i) \leftarrow \mathbf{u}^{y_i} \mathbf{v}^{r'_i}$
- compute proof elements for each pairing product equation, i.e.,
  - generate $\mathbf{T} \leftarrow_\$ \mathbb{Z}_p^{2\times 2}$
  - $\Theta \leftarrow \begin{bmatrix} \Theta_1 & \Theta_2 \end{bmatrix} =: \begin{bmatrix} 1 & \prod_i A_i^{\rho'_i} \prod_{i,j} X_i^{\gamma_{i,j}\rho'_j} \\ 1 & \prod_i A_i^{\sigma'_i} \prod_{i,j} X_i^{\gamma_{i,j}\sigma'_j} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} a_1^{T_{1,1}} b_1^{T_{1,2}} & a_2^{T_{1,1}} b_2^{T_{1,2}} \\ a_1^{T_{2,1}} b_1^{T_{2,2}} & a_2^{T_{2,1}} b_2^{T_{2,2}} \end{bmatrix}^{\mathrm{T}}$
  - $\Pi \leftarrow \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix} =: \begin{bmatrix} 1 & \prod_i B_i^{\rho_i} \prod_{i,j} Y_j^{\gamma_{i,j}\rho_i} \\ 1 & \prod_i B_i^{\sigma_i} \prod_{i,j} Y_j^{\gamma_{i,j}\sigma_i} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} \mathbf{v}^{\sum_{i,j} \rho_i \gamma_{ij} \rho'_j - T_{1,1}} \mathbf{w}^{\sum_{i,j} \rho_i \gamma_{ij} \sigma'_j - T_{2,1}} \\ \mathbf{v}^{\sum_{i,j} \sigma_i \tilde{\gamma}_{ij} \rho'_j - T_{1,2}} \mathbf{w}^{\sum_{i,j} \sigma_i \gamma_{ij} \sigma'_j - T_{2,2}} \end{bmatrix}$
- for each multi-scalar multiplication equation in $\mathbb{G}_1$,

* generate $\mathbf{T} \leftarrow_\$ \mathbb{Z}_p^2$
* $\Theta \leftarrow \begin{bmatrix} 1 & \prod_i A_i^{r'_i} \prod_{i,j} X_i^{\tilde{\gamma}_{i,j} r'_j} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} a_1^{T_1} b_1^{T_2} & a_2^{T_1} b_2^{T_2} \end{bmatrix}^{\mathrm{T}}$
* $\Pi \leftarrow \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix} =: \begin{bmatrix} \mathbf{u}^{\sum_i b_i \rho_i + \sum_{i,j} \rho_i \tilde{\gamma}_{ij} y_j} \\ \mathbf{u}^{\sum_i b_i \sigma_i + \sum_{i,j} \sigma_i \tilde{\gamma}_{ij} y_j} \end{bmatrix} \circ \begin{bmatrix} \mathbf{v}^{\sum_{i,j} \rho_i \tilde{\gamma}_{ij} r'_j - T_1} \\ \mathbf{v}^{\sum_{i,j} \sigma_i \tilde{\gamma}_{ij} r'_j - T_2} \end{bmatrix}$

In case all $\tilde{\gamma}_{ij}$ are nil, then $\mathbf{T}$ can be set as $\mathbf{0}$, and the proof element Θ can then be restricted to its second component. Besides, if all $b_i = 0$, then Π can be set as $\mathbf{1}$.

– for each quadratic equation in $\mathbb{Z}_p$,

* generate $T \leftarrow_\$ \mathbb{Z}_p$
* $\Theta \leftarrow \mathbf{d}^{\sum_i \tilde{a}_i r'_i + \sum_{i,j} r'_i \gamma'_{ij} y_j} \mathbf{a}^T$
* $\Pi \leftarrow \mathbf{u}^{\sum_i \tilde{b}_i \rho_i + \sum_{i,j} \rho_i \gamma'_{ij} y_j} \mathbf{v}^{\sum_{i,j} \rho_i \gamma'_{ij} r'_j - T}$

In case all γ'_{ij} are nil, then $\mathbf{T}$ can be set as $\mathbf{0}$. Moreover, Θ can also rather be set as $\sum_i \tilde{a}_i r'_i \in \mathbb{Z}_p$ and Π as $\sum_i \tilde{b}_i \rho_i \in \mathbb{Z}_p$, which results in smaller proof elements (the verification algorithm would then first compute $\mathbf{d}^\Theta$ and $\mathbf{u}^\Pi$).

– return all commitments $(\mathbf{c}(X_i))_{i=1}^m, (\mathbf{c}(Y_i))_{i=1}^n, (\mathbf{c}(x_i))_{i=1}^{m'}, (\mathbf{c}(y_i))_{i=1}^{n'}$ and all proof elements. Note that the resulting proof is perfectly *re-randomizable*, and that the proof system therefore satisfies derivation privacy.

Verification. Accept a proof parsed as above if and only if

– for every pairing production equation,

$$\prod_i f\left(\begin{bmatrix} 1 \\ A_i \end{bmatrix}, \mathbf{c}(Y_i)\right) \prod_i f\left(\mathbf{c}(X_i), \begin{bmatrix} 1 \\ B_i \end{bmatrix}\right) \prod_{i,j} f\left(\mathbf{c}(X_i), \mathbf{c}(Y_j)\right)^{\gamma_{ij}}$$
$$= \begin{bmatrix} 1 & 1 \\ 1 & Z_T \end{bmatrix} f\left(\Theta_1, \mathbf{v}\right) f\left(\Theta_1, \mathbf{w}\right) f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right)$$

– for every multi-scalar multiplication equation in $\mathbb{G}_1$,

$$\prod_i f\left(\begin{bmatrix} 1 \\ \tilde{A}_i \end{bmatrix}, \mathbf{c}(y_i)\right) \prod_i f\left(\mathbf{c}(X_i), \begin{bmatrix} 1 \\ b_i \end{bmatrix}\right) \prod_{i,j} f\left(\mathbf{c}(X_i), \mathbf{c}(y_j)\right)^{\tilde{\gamma}_{ij}}$$
$$= f\left(\begin{bmatrix} 1 \\ Z_1 \end{bmatrix}, \mathbf{u}\right) f\left(\Theta, \mathbf{v}\right) f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right).$$

– for every quadratic equation in $\mathbb{Z}_p$,

$$\begin{aligned}&\prod_i f\left(\mathbf{d}^{\tilde{a}_i}, \mathbf{c}(y_i)\right) \prod_i f\left(\mathbf{c}(x_i), \mathbf{u}^{\tilde{b}_i}\right) \prod_{i,j} f\left(\mathbf{c}(x_i), \mathbf{c}(y_j)\right)^{\gamma'_{ij}} \\ =&f\left(\mathbf{d}, \mathbf{u}^z\right) f\left(\Theta, \mathbf{v}\right) f\left(\mathbf{a}, \Pi\right).\end{aligned}$$

Malleability. *Additive transformations* in $\mathbb{Z}_p$ are admissible for the relation of multi-scalar multiplication equations in $\mathbb{G}_1$. That is, for any set $J \subseteq [\![n']\!]$ such that $\gamma_{ij} = 0$ for all $i \in [\![m']\!]$ and $j \in J$, given $(\alpha_i)_{i\in J} \in \mathbb{Z}_p^{|J|}$ and ℓ multi-scalar multiplication equations, the map

$$T_{(\alpha_i)_i}\colon \left((\ldots, Z_{k,1})_{k=1}^{\ell}, ((y_i)_i, \ldots)\right) \mapsto \left(\left(\ldots, Z_{k,1} \prod_i \tilde{A}_{k,i}^{\alpha_i}\right)_k, ((y_i + \alpha_i)_i, \ldots)\right)$$

is admissible. This map is identified with the tuple $(\alpha_i)_{i\in J}$. Note that it is the identity at the coordinates i such that $\alpha_i = 0$.

The GS proof system is malleable w.r.t. to such transformation. More precisely, its algorithm Eval first parses a valid π as above, generates $r''_i \leftarrow_\$ \mathbb{Z}_p$, and computes

– $\mathbf{c}(y_i + \alpha_i) \leftarrow \mathbf{c}(y_i)\mathbf{u}^{\alpha_i}\mathbf{v}^{r''_i}$

– for all $k \in [\![\ell]\!]$, $\Theta_{k,2} \leftarrow \Theta_{k,2} \cdot \prod_i \tilde{A}_{ki}^{r''_i}$.

It then returns $\left((\mathbf{c}(X_i))_{i=1}^m,\ \mathbf{c}(y_i + \alpha_i)_{i=1}^{n'},\ (\Theta_k, \Pi_k)_{k=1}^{\ell}\right)$.

OR Proofs. Define $\mathcal{L}_{\mathrm{OR-GS}}$ as the language of pairs (S_0, S_1) such that $S_0 \in \mathcal{L}_{\mathrm{GS}}$ or $S_1 \in \mathcal{L}_{\mathrm{GS}}$. To prove a statement $(S_0, S_1) \in \mathcal{L}_{\mathrm{OR-GS}}$, following techniques of Groth [Gro06], it suffices to introduce new variables $\chi_0 \in \mathbb{Z}_p$, and $\mathbf{X}_\beta \in \mathbb{G}_1^m$, $\mathbf{Y}_\beta \in \mathbb{G}_2^n$, $\mathbf{x}_\beta \in \mathbb{Z}_p^{m'}$ and $\mathbf{y}_\beta \in \mathbb{Z}_p^{n'}$ for $\beta \in \{0, 1\}$, and then GS prove the satisfiability of the sets of equations

$$\prod_{i=1}^{n_\beta} e\left(A_{\beta,i}, Y_{\beta,i}\right) \prod_{i=1}^{m_\beta} e\left(X_{\beta,i}, B_{\beta,i}\right) \prod_{i=1}^{m_\beta} \prod_{j=1}^{n_\beta} e\left(X_{\beta,i}, Y_{\beta,i}\right)^{\gamma_{\beta,ij}} = Z_{\beta,T}^{\beta\chi_0 + (1-\beta)(1-\chi_0)}$$

$$\prod_{i=1}^{n'_\beta} \tilde{A}_{\beta,i}^{y_{\beta,i}} \prod_{i=1}^{m_\beta} X_{\beta,i}^{b_{\beta,i}} \prod_{i=1}^{m_\beta} \prod_{j=1}^{n'_\beta} X_{\beta,i}^{\tilde{\gamma}_{\beta,ij} y_j} = Z_{\beta,1}^{\beta\chi_0 + (1-\beta)(1-\chi_0)}$$

$$\sum_{i=1}^{n'_\beta} \tilde{a}_{\beta,i} y_{\beta,i} + \sum_{i=1}^{m'_\beta} \tilde{b}_{\beta,i} x_{\beta,i} + \sum_{i=1}^{m'_\beta} \sum_{j=1}^{n'_\beta} \gamma'_{\beta,ij} x_{\beta,i} y_{\beta,j} = z_\beta^{\beta\chi_0 + (1-\beta)(1-\chi_0)} \mod p$$

$$\chi_0^2 - \chi_0 = 0 \mod p$$

for $\beta \in \{0, 1\}$, and with $A_{0,i}$ denoting the variable A_i for S_0 and $A_{1,i}$ the one for S_1, and likewise for the other public values. If $S_0 \in \mathcal{L}_{\mathrm{GS}}$ (with witness

$(\mathbf{X}, \ldots))$, set $\mathbf{X}_0 \leftarrow \mathbf{X}$ and $\mathbf{X}_1 \leftarrow \mathbf{1}$, $\mathbf{x}_0 \leftarrow \mathbf{x}$ and $\mathbf{x}_1 \leftarrow \mathbf{0}$, and similarly for the other variables; and vice versa if $S_1 \in \mathcal{L}_{\mathrm{GS}}$. The last equation implies that $\chi_0 = 0$ or $1 \mod p$, which guarantees that $S_0 \in \mathcal{L}_{\mathrm{GS}}$ or $S_1 \in \mathcal{L}_{\mathrm{GS}}$. To prove the last equation, it suffice to consider χ_0 as both an extra x and an extra y variable such that $x - y = 0 \mod p$ and $x - xy = 0 \mod p$.

Simulation Soundness under Controlled Malleability. The generic construction of a CM simulation proof system of Section 6.2.2 can be applied with Jutla and Roy's signature scheme (Section 6.1.5) to the GS proof system for multi-scalar multiplication equations in $\mathbb{G}_1$; the latter being malleable w.r.t. additive transformations in $\mathbb{Z}_p$.

In more detail, given equations of the form

$$\prod_{i=1}^{n'} \tilde{A}_{k,i}^{y_i} \prod_{i=1}^{m} X_i^{b_{k,i}} \prod_{i=1}^{m} \prod_{j=1}^{n'} X_i^{\tilde{\gamma}_{k,ij} y_j} = Z_{k,1}$$

for $1 \leq k \leq \ell$, let $\mathcal{T}$ denote the class of additive transformations. These transformations are identified with tuples $(\alpha_i)_i$. For a given transformation $(\alpha_i)_i$, let $K \subseteq [\![\ell]\!]$ denote the set of indices such that $Z_{k,1} \neq Z_{k,1} \prod_i \tilde{A}_{k,i}^{-\alpha_i}$. Set μ as $\left(Z_{k,1} \prod_i \tilde{A}_{k,i}^{-\alpha_i}\right)_{k \in K}$. The components of μ are nothing but the public values which are affected by the transformation (i.e., it corresponds to x'_1 in the generic construction). A CM simulation sound GS proof for multi-scalar multiplication equations in $\mathbb{G}_1$ is then a proof that there exists $\mathbf{X}$ and $\mathbf{y}$ such that all ℓ equations are satisfied *or* that there exists μ, $(\alpha_i)_i$ and a Jutla–Roy signature σ such that $\mathsf{Vf}(vk, \mu, \sigma) = 1$ and $T_{(\alpha_i),x}\left(\left(\ldots, Z_{k,1} \prod_i \tilde{A}_{k,i}^{-\alpha_i}\right)_{k=1}^{\ell}\right) = (\ldots, Z_{k,1})_{k=1}^{\ell}$.

Compared to a standard GS proof, a CM-simulation-sound GS proof introduces $|\mu|$ additional variables $\mu_i \in \mathbb{G}_1$, 5 additional $\mathbb{G}_1$ variables and 1 additional $\mathbb{G}_2$ variables for σ, $|\mu|$ additional variables $\alpha_i \in \mathbb{Z}_p$ for the transformation, and 2 additional $\mathbb{Z}_p$ variables x and y (which should satisfy $x - y = x - xy = 0$).

In terms of equations, it introduces 2 pairing-product equations for the verification of the signature, $|\mu|$ multi-scalar multiplication equations in $\mathbb{G}_1$ for the transformation, and 2 quadratic equations in $\mathbb{Z}_p$ for x and y. That means $2(5 + |\mu|)$ $\mathbb{G}_1$ and $2(3 + |\mu|)$ $\mathbb{G}_2$ elements to commit to the additional variables; 4 $\mathbb{G}_1$ and 8 $\mathbb{G}_2$ elements as proof elements for the verification of the signature (which costs $4\mathrm{P}^6 + 4\mathrm{P}^{7+|\mu|}$); 2 $\mathbb{G}_1$, 2 $\mathbb{G}_2$ and 2 $\mathbb{Z}_p$ elements as proof elements for the two equations in $\mathbb{Z}_p$, the of which verification costs 2 $\mathbb{G}_1^1$ + 6 $\mathbb{G}_2^1$ + $8\mathrm{P}^5$; and $|\mu|$ $\mathbb{G}_1$ and $4|\mu|$ $\mathbb{G}_2$ elements for the multi-scalar multiplication equations in $\mathbb{G}_1$ introduced by μ, the verification of each of which costs $2\mathrm{P}^{5+\left|\left\{\mathrm{i}\colon \tilde{\mathrm{A}}_{\mathrm{k,i}} \neq 1_{\mathbb{G}_1}\right\}\right|}$.

6.3 Model for Password-Assisted Decryption

This section introduces Encryption schemes with Password-protected Assisted Decryption (EPAD schemes). As mentioned in the introduction, an EPAD scheme is an encryption scheme which involves three parties, a user, a token and a server. The user, sharing a password with a server, is logging-in from a computer, and her decryption key is shared between the token and the server. The protocol will allow her to decrypt ciphertexts with the help of the token and the server, as soon as each authentication between the three parties (based on the private values mentioned above) succeeds.

6.3.1 Syntax

In a three-party setting with a user $\mathcal{U}$, a token $\mathcal{T}$ and a server $\mathcal{S}$, an EPAD scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: generates public parameters on input a security parameter. These parameters are implicit inputs to all the other algorithms.

$\mathsf{KG}(pp) \to (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}})$: generates a public key, a secret key and shares thereof.

$\mathsf{Enc}(pk, M) \to C$: a probabilistic encryption algorithm.

$\mathsf{Dec}(sk, C) \to M/\perp$: a deterministic decryption algorithm.

$\mathsf{IDec} = \langle \mathsf{U}(pk, p_{\mathcal{U}}, C) \rightleftharpoons \mathsf{T}(sk_{\mathcal{T}}) \rightleftharpoons \mathsf{S}(sk_{\mathcal{S}}, p_{\mathcal{S}}) \rangle \to \langle M/\perp, \perp, \perp \rangle$: an interactive decryption protocol between a user algorithm with input a public key, a user password and a ciphertext; a token algorithm with input a secret-key share; and a server algorithm with input a secret-key share and a server password. The passwords are here treated as bit strings.

In practice, servers usually only hold a function (or transformation, e.g., salted hash) of user passwords. In this case, in the interactive decryption protocol, U should be given the result of that function applied to $p_{\mathcal{U}}$. For notational convenience, it is implicitly assumed that it is already the case.

Correctness. An EPAD scheme is correct if the decryption of an encrypted plaintext, whether by the deterministic decryption algorithm or by the interactive protocol with $p_{\mathcal{U}} = p_{\mathcal{S}}$, results in the plaintext. That is, for all $\lambda \in \mathbb{N}$, all M and all $p_{\mathcal{U}} = p_{\mathcal{S}}$,

$$\Pr\left[\mathsf{IDec} = \langle \mathsf{Dec}(sk, C), \perp, \perp \rangle = \langle M, \perp, \perp \rangle : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp) \\ C \leftarrow \mathsf{Enc}(pk, M) \end{array}\right] = 1.$$

6.3.2 Security Definitions

This section formalizes the security properties expected from EPAD schemes. These properties are Password-protected Indistinguishability under Replayable Chosen-Ciphertext Attacks (P-IND-RCCA), blindness and verifiability.

P-IND-RCCA Security.

Password-protected Indistinguishability under Replayable Chosen-Ciphertext Attacks (P-IND-RCCA) ensures that no efficient adversary can infer any information about a user's plaintext as long as it does not know her password (by corrupting the user's machine or the server) or does not have access to her token. It captures both indistinguishability under replayable chosen-ciphertext attacks and password authentication. The latter means that decryption can only succeed if the user and the server have the same password.

The formal model for P-IND-RCCA security is inspired by the Bellare–Rogaway–Pointcheval (BPR) model for password-based authenticated-key-exchange protocols [BPR00]. It covers the cases of concurrent protocol executions, with potentially many users, tokens and servers. In addition to that, a user may possess several tokens and could be registered on several servers.

Game Overview. The P-IND-RCCA security experiment features an adversary $\mathcal{A}$. After an initial phase to generate parameters, the adversary can request the challenger to generate keys and passwords for the users. The passwords are generated via a password generator PG that returns values in a dictionary D. For simplicity, the passwords are assumed to be uniformly distributed over D, but one could of course consider the min-entropy of the output distribution of PG (in this case, $1/|D|$ should be replaced by this min-entropy in the following statements).

Next, the adversary is given access to several oracles which model different types of attacks (password-related or chosen-ciphertext attacks). $\mathcal{A}$ is also given access to a test oracle which can be called only once but at any time (a definition with several test queries would be equivalent.) It returns an encryption of a randomly chosen message out of two specified by $\mathcal{A}$, under the secret key of a user also chosen by $\mathcal{A}$. Adversary $\mathcal{A}$ is considered successful if it correctly guesses which message was encrypted. An EPAD scheme is then said to be P-IND-RCCA secure if no efficient adversary can win the game with a probability non-negligibly close to $1/2$ plus the maximum advantage the adversary can gain from the trivial online dictionary attacks which consist in guessing passwords (i.e., those should be the only attacks possible).

Initialization & Game Variables. A set of users U is assumed to be fixed. For every $\mathcal{U} \in U$, the set of tokens belonging to $\mathcal{U}$ is denoted by $T[\mathcal{U}]$, and the set of servers with which $\mathcal{U}$ shares a password is denoted $S[\mathcal{U}]$. The set of all tokens is denoted T and the set of all servers S.

During the initialization phase, public parameters for the encryption scheme are generated. Secret-key shares for all tokens and servers are set to $\perp$. Further on, for $id \in T \cup S$, sk_{id} denotes the set of all user key-shares for id.

A finite instance set I for all party algorithms is also assumed to be fixed. Each instance $i \in I$ of the algorithm of party id maintains a state st^i_{id}. A session identifier sid^i_{id}, and partner identities pid^i_0 and pid^i_1 allow to match instances in protocol executions.

A variable $used^i_{id}$ indicates whether an active attack has been performed on the ith instance of the algorithm of party id.

Variables acc^i_{id} and $term^i_{id}$ respectively indicate whether the ith algorithm instance of party id has accepted and terminated. As in the BPR model, acceptance and termination are distinguished. When an instance terminates, it does not output any further message. Nevertheless, it can accept at a certain point of its computation but terminate later. This may occur when an instance confirms its partners, in which case it accepts, and thereafter continues the computation until termination.

A queue Q_{AddU} of added users, i.e., users for which keys and passwords have been generated, and a queue of corrupt users Q_{Corrupt} are also initialized.

At the end of the initialization phase, the encryption parameters, the sets of participants and the user public keys are returned in a public input pin, and the rest is set in a secret input sin. That is, $pin \leftarrow (pp,\ I,\ U,\ T,\ S,\ (pk_{\mathcal{U}})_{\mathcal{U}})$ and $sin \leftarrow \Big(pin, (sk_{\mathcal{T}})_{\mathcal{T}}, (sk_{\mathcal{S}})_{\mathcal{S}}, \Big(st^i_{id}, sid^i_{id}, pid^i_{id,0}, pid^i_{id,1}, acc^i_{id}, term^i_{id}, used^i_{id}\Big)_{i,id},\ Q_{\mathsf{AddU}},\ Q_{\mathsf{Corrupt}}\Big)$. The secret input sin is later made available to all oracles.

Oracles. Throughout the experiment, $\mathcal{A}$ is given access to the oracles detailed below and summarized on Figure 6.1, and which it can query in any order.

- Test_b : returns the encryption with a user public key of one of two messages, all chosen by the adversary. The challenge user identity $\mathcal{U}^*$ is not required to be honest (i.e., the adversary may know her password), but the challenge token $\mathcal{T}^*$ and challenge server $\mathcal{S}^*$ cannot both be corrupt, otherwise the adversary would be able to reconstruct her secret key and trivially win the game. The public key $pk_{\mathcal{U}^*}$ is the one for which $\mathcal{T}^*$ and $\mathcal{S}^*$ hold secret-key shares. For simplicity, the adversary may query this oracle at most once. Note also that as in the

$\mathsf{Init}\left(1^\lambda, U, T, S, I\right)$	$pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ for $(\mathcal{T} \in T)\left\{\text{for } (\mathcal{U} \in U)\ sk_{\mathcal{T}}^{\mathcal{U}} \leftarrow \perp\right\}$ for $(\mathcal{S} \in S)\left\{\text{for } (\mathcal{U} \in U)\ sk_{\mathcal{S}}^{\mathcal{U}} \leftarrow \perp\right\}$ for $(i, id) \in I \times (U \cup T \cup S)$ $st_{id}^i \leftarrow \perp$ $sid_{id}^i \leftarrow pid_{id,0}^i \leftarrow pid_{id,1}^i \leftarrow \perp$ $acc_{id}^i \leftarrow term_{id}^i \leftarrow used_{id}^i \leftarrow \mathsf{FALSE}$ $Q_{\mathsf{AddU}} \leftarrow Q_{\mathsf{Corrupt}} \leftarrow \emptyset$ return (pin, sin)
$\mathsf{AddU}(\mathcal{U}, T[\mathcal{U}], S[\mathcal{U}])$	if $(\mathcal{U} \notin U$ or $\mathcal{U} \in Q_{\mathsf{Corrupt}})$ return $\perp$ for $(\mathcal{S} \in S[\mathcal{U}])\left(p_{\mathcal{U}}, p_{\mathcal{S}}^{\mathcal{U}}\right) \leftarrow \mathsf{PG}(pp)$ // Generate passwords for the user for $(\mathcal{T}, \mathcal{S}) \in T[\mathcal{U}] \times S[\mathcal{U}]$ $(pk_{\mathcal{U}}, sk_{\mathcal{U}}, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp)$ // Generate user keys $\left(sk_{\mathcal{T}}^{\mathcal{U}}, sk_{\mathcal{S}}^{\mathcal{U}}\right) \leftarrow (sk_{\mathcal{T}}, sk_{\mathcal{S}})$ output $pk_{\mathcal{U}}$ $Q_{\mathsf{AddU}} \leftarrow Q_{\mathsf{AddU}} \cup \{\mathcal{U}\}$
$\mathsf{Exec}(\mathcal{U}, i, \mathcal{T}, j, \mathcal{S}, k, C)$	if $(\mathcal{U} \notin Q_{\mathsf{AddU}}$ or $\mathcal{T} \notin T$ or $\mathcal{S} \notin S)$ return $\perp$ if $\left(used_{\mathcal{U}}^i \text{ or } used_{\mathcal{T}}^j \text{ or } used_{\mathcal{S}}^k\right)$ return $\perp$ if $(\mathsf{Dec}(sk_{\mathcal{U}}, C) \in \{M_0, M_1\})$ return replay // Prevent replay attacks $\tau \leftarrow \left\langle \mathsf{U}_i(pk, p_{\mathcal{U}}, C), \mathsf{T}_j\left(sk_{\mathcal{T}}^{\mathcal{U}}\right), \mathsf{S}_k\left(sk_{\mathcal{S}}^{\mathcal{U}}, p_{\mathcal{S}}^{\mathcal{U}}\right)\right\rangle$ // Run an honest protocol execution. return τ

$\mathsf{Send}(id, i, M)$	if $term_{id}^i$ return $\perp$
	$used_{id}^i \leftarrow \mathsf{TRUE}$
	
	if $id = \mathcal{U}^*$ and $M = (\mathcal{T}^*, *, \mathcal{S}^*, *, C)$
	if $(\mathsf{Dec}(sk_{\mathcal{U}^*}, C) \in \{M_0, M_1\})$
	return replay
	if $\mathcal{U}^* \in Q_{\mathsf{Corrupt}}$ or $\mathcal{S}^* \in Q_{\mathsf{Corrupt}}$
	if $(id = \mathcal{T}^*)$ return $\perp$
	// If $\mathcal{A}$ has the password of $\mathcal{U}^*$, reject queries to $\mathcal{T}^*$
	
	if $\mathcal{U}^* \in Q_{\mathsf{Corrupt}}$ and $\mathcal{T}^* \in Q_{\mathsf{Corrupt}}$
	if $(id = \mathcal{S}^*)$ return $\perp$
	/* If $\mathcal{A}$ has the password of $\mathcal{U}^*$ and if $\mathcal{T}^*$ is corrupt,
	reject queries to $\mathcal{S}^*$ */
	
	$\langle m_{out}, acc, term_{id}^i, sid, pid_0, pid_1, st_{id}^i\rangle \leftarrow \langle \mathsf{IDec}\left(id, st_{id}^i, sk_{id}, p_{id}, M\right)\rangle$
	// If $id \in U$, replace sk_{id} with pk_{id}
	// If $id \in T$, replace p_{id} with $\perp$
	
	if acc and $\neg acc_{id}^i$
	$sid_{id}^i \leftarrow sid$; $pid_{id,0/1}^i \leftarrow pid_{0/1}$; $acc_{id}^i \leftarrow acc$
	return $\left(m_{out}, acc, term_{id}^i, sid, pid_0, pid_1, st_{id}^i\right)$

$\mathsf{Corrupt}(id, p)$	if $\left(\exists i \in I\colon used_{id}^i \text{ and } \neg term_{id}^i\right)$ return $\perp$ //Static corruption only $Q_{\mathsf{Corrupt}} \leftarrow Q_{\mathsf{Corrupt}} \cup \{id\}$ if $id \in U$ for $\mathcal{S} \in S[id]$ $p_{\mathcal{S}}^{\mathcal{U}} \leftarrow p[\mathcal{S}]$ // Overwrite the user's passwords return $\left(p_{\mathcal{U}}, \{st_{\mathcal{U}}^i\}_{i \in I}\right)$ else if $(id = \mathcal{T}^*$ and $\mathcal{S}^* \in Q_{\mathsf{Corrupt}})$ or $(id = \mathcal{S}^*$ and $\mathcal{T}^* \in Q_{\mathsf{Corrupt}})$ return $\perp$ // $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt return $(sk_{id}, \{st_{id}^i\}_i)$
$\mathsf{Test}_b\left(\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*, M_0, M_1\right)$	if $(\mathcal{U}^* \notin U$ or $\mathcal{T}^* \notin T[U^*]$ or $\mathcal{S}^* \notin S[U^*])$ return $\perp$ if $\left((\mathcal{T}^*, \mathcal{S}^*) \in Q_{\mathsf{Corrupt}}^2\right)$ return $\perp$ // $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt $C^* \leftarrow \mathsf{Enc}\left(pk_{\mathcal{U}^*}, M_b\right)$ return C^*

Figure 6.1: Oracles for the P-IND-RCCA Security Experiment.

BPR model [BPR00] and the model for distributed session key [BR95], this query is not restricted to be the last query of the adversary.

– AddU : adds an honest user identity. In addition to a user identity $\mathcal{U}$, the adversary specifies a set $T[\mathcal{U}]$ of tokens and a set $S[\mathcal{U}]$ of servers for $\mathcal{U}$. Note that these can be corrupt, except for the challenge identities $\mathcal{U}^*$, $\mathcal{T}^*$, $\mathcal{S}^*$: parties $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt (see the definition of oracle Corrupt). For each server in $S[\mathcal{U}]$, a password $p_{\mathcal{U}}$ and a transformation $p_{\mathcal{S}}^{\mathcal{U}}$ thereof is generated by the password generator PG. Keys and secret-key shares for all token–server pairs of the user are also generated.

– Exec : returns the transcript of an honest (i.e., without the interference of the adversary) decryption-protocol execution on a ciphertext C. Note that Exec queries thereby model offline dictionary attacks among others. The execution is between the ith, jth and kth instances the algorithms of a user $\mathcal{U}$, a token $\mathcal{T}$ and a server $\mathcal{S}$. The notation U_i, T_j and S_k mean that algorithms U, T and S are respectively run with the states $st_{\mathcal{U}}^i$, $st_{\mathcal{T}}^j$ and $st_{\mathcal{S}}^k$. If $\mathsf{Dec}(sk_{\mathcal{U}}, C)$ is one of the challenge messages (with $sk_{\mathcal{U}}$ the secret key for which $\mathcal{T}$ and $\mathcal{S}$ hold shares), the oracle returns a special string replay as in the classical definition of RCCA security (see Sec. 6.1.6). The parties may be corrupt (in which case $\mathcal{A}$ has their states), but $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt.

– Send : the adversary can perform active attacks via this oracle. The adversary can send a message to an algorithm instance (e.g., the kth instance of a server algorithm), all of its choice. The notation $\mathsf{IDec}(id, \cdot)$ respectively stands for $\mathsf{U}(\cdot)$, $\mathsf{T}(\cdot)$ or $\mathsf{S}(\cdot)$ if $id \in U$, T or S. This algorithm then runs the instance on that message. To prompt the ith instance of the algorithm of a user $\mathcal{U}$ to initiate a protocol execution on a ciphertext C with the jth instance of a token $\mathcal{T}$ and the kth instance of a server $\mathcal{S}$, the adversary can query oracle Send on $(\mathcal{U}, i, (\mathcal{T}, j, \mathcal{S}, k, C))$. If such a query decrypt to either of the challenge messages, the oracle returns replay to prevent trivial wins.

  In addition to an output message m_{out}, the algorithm also returns to $\mathcal{A}$ acceptance and termination states acc and $term_{id}^i$, a session identifier sid and partner instances pid_0 and pid_1, and a state st_{id}^i. Identity pid_0 is always assumed to be the party which should receive the next flow of id, i.e., a token identity if $id \in U$, a server identity if $id \in T$ and a token identity if $id \in S$. Variables sid_{id}^i, $pid_{id,0}^i$ and $pid_{id,1}^i$ and acc_{id}^i are updated in case the instance accepts, and all values are revealed to the adversary except the state (which may contain a password or a secret-key share). $\mathcal{A}$ can access this state by corrupting party id.

  If $\mathcal{A}$ knows the password of the challenge user $\mathcal{U}^*$, by corrupting either

that identity or $\mathcal{S}^*$, then queries to $\mathcal{T}^*$ are rejected. It translates the fact that in case of a server breach or if there is a keylogger on the user's machine, then no security can be expected if an attacker also has access to her token.

Likewise, if $\mathcal{U}^*$ is corrupt as well as $\mathcal{T}^*$, queries to $\mathcal{S}^*$ are rejected. It reflects the fact that if a user's token is corrupt, no security can be expected if her password is also leaked. Indeed, in that case, an attacker can use the server which shares a key with the token in order to decrypt the ciphertexts.

– $\mathsf{Corrupt}$: gives the adversary control over all the instances of a party algorithm. In the case of a user, the adversary does not only receives her passwords and the states of all her algorithm instances, but may also overwrite the password held by each of her servers. If the party being corrupted is a token or a server, the adversary receives the states of all of its algorithm instances and also its key shares. Once a Test query has been made with an identity $\mathcal{U}^*$, the corruption of both $\mathcal{T}^*$ and $\mathcal{S}^*$ is not allowed. That is to prevent the adversary from reconstructing her secret key and trivially win the game.

  Only *static corruptions* are here considered, i.e., the adversary cannot corrupt a party of which an algorithm instance is in the middle of a protocol execution. It is expressed by the condition

$$\text{“if } \left(\exists i \in I\colon used_{id}^{i} \text{ and } \neg term_{id}^{i}\right) \text{return } \perp\text{”}.$$

Definition 6.3.3 (P-IND-RCCA)**.** *An EPAD scheme is P-IND-RCCA secure if for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,*

$$\Pr\left[b = b' \colon \begin{array}{l} (pin, sin) \leftarrow \mathsf{Init}\left(1^{\lambda}, U, T, S, I\right) \\ b \leftarrow_{\$} \{0,1\} \\ \mathcal{O} \leftarrow \{\mathsf{Exec}, \mathsf{Send}, \mathsf{Corrupt}, \mathsf{Test}_b\} \\ b' \leftarrow \mathcal{A}^{\mathcal{O}(sin,\cdot)}(pin) \\ \textit{return } (b, b') \end{array}\right]$$

is negligibly close to $1/2 + q_{\mathsf{Send}}/|D|$.

The term $q_{\mathsf{Send}}/|D|$ accounts for online dictionary attacks. An EPAD scheme should guarantee that these are the best attacks possible.

On One-Round Protocols. Note that there cannot exist a one-round protocol secure against offline dictionary attacks. A round is here understood as the set of messages sent between all parties from a message sent by the user to the token to the next response from the token. Were there such a one-rounded protocol, an adversary could intercept the transcript of

an execution of the decryption protocol. In that execution, the user would have had to send sufficient information for the server to verify her password, without having received any prior indication that the server shares this password; sufficient in fact, for the adversary to perform an offline dictionary attack on her password. Therefore, a protocol secure against offline dictionary attacks must consist of at least two rounds. It is not surprising as the user, who initiates the protocol, should verify the server holds the same password as she does. Moreover, the mere fact the user continues the protocol until the end indicates that the server could successfully authenticate itself to the user. On this account, without a throttling mechanism on the user's side, online dictionary attacks against the user's password cannot be prevented.

Blindness.

The blindness property formalizes the idea that neither the token nor the server should be able to infer any information about the ciphertexts the user attempts to decrypt. In its formal definition, the challenge ciphertexts C_0 and C_1 must be either both valid or both invalid, i.e., $(\mathsf{Dec}(sk, C_0) = \perp) = (\mathsf{Dec}(sk, C_1) = \perp)$ should hold, which is a minimal condition to exclude trivial wins. This strong requirement results from the fact token and server are adversarial in the formal definition, and can therefore together reconstruct the entire secret key and password.

Definition 6.3.4 (Blindness)**.** *An EPAD scheme satisfies blindness if for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,*

$$\Pr\left[\, b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp) \\ (st, p_{\mathcal{U}}, C_0, C_1) \leftarrow \mathcal{A}(pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \\ b \leftarrow_{\$} \{0,1\} \\ \textit{if } (\mathsf{Dec}(sk, C_0) = \perp) \neq (\mathsf{Dec}(sk, C_1) = \perp) \\ \quad b' \leftarrow_{\$} \{0,1\} \\ \quad \textit{return } (b, b') \\ \langle *, b' \rangle \leftarrow \langle \mathsf{U}(pk, p_{\mathcal{U}}, C_b), \mathcal{A}(st) \rangle \\ \textit{return } (b, b') \end{array} \right]$$

is negligibly close to 1/2.

Verifiability.

This property captures the idea that the user should accept the result of the decryption protocol only if the token and the server have correctly performed their computations.

Definition 6.3.5 (Verifiability)**.** *An EPAD scheme is verifiable if for all* $\lambda \in \mathbb{N}$*, for every efficient adversary* $\mathcal{A}$*,*

$$\Pr\left[\langle \mathsf{U}\left(pk, p_{\mathcal{U}}, C\right), \mathcal{A}(st)\rangle \notin \{\langle \mathsf{Dec}(sk, C), *\rangle, \langle \perp, *\rangle\} : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp) \\ (st, p_{\mathcal{U}}, C) \leftarrow \mathcal{A}(pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \end{array}\right] = 1.$$

6.4 Construction

In this section, we build an EPAD scheme from the RCCA-secure encryption scheme of Faonio, Fiore, Herranz and Ràfols [FFHR19] (see Section 6.1.6). Similar techniques can a priori be applied to any publicly verifiable, structure-preserving RCCA-secure scheme. We start by showing how users can blind the plaintexts underlying the ciphertexts they want to decrypt. The section then continues with our construction and its efficiency assessment.

6.4.1 Verification of Blinded Ciphertexts

As mentioned in Section 6.3.2, EPAD schemes should satisfy blindness, meaning that even if the token and the server are corrupt, they cannot infer any information about the ciphertexts they are helping a user to decrypt. It is a stringent requirement as a corrupt token and server can reconstruct the secret key, and if the server is actually one associated to the user, it also has her password. Thus given a ciphertext, the user must be able to blind the underlying plaintext using only public information.

On the other hand, EPAD schemes also target an RCCA type of security, so the only type of malleability that should be expected is re-randomization. Therefore, the user cannot a priori blind the plaintext and produce a new valid ciphertext (except with negligible probability) since she does not remember her secret key. Nonetheless, she can provide enough information *auxiliary* to the modified ciphertext which allows the token and the server to verify, *with their secret key shares*, that she correctly blinded the underlying plaintext of a valid ciphertext, and that she knows the blinding factor.

A new algorithm $\mathsf{Blind}(pk, C) \rightarrow \left(\tilde{C}, aux\right)$ is thus introduced. Given a public key and a ciphertext of the scheme of Faonio et al., it essentially adds a one-time pad to the plaintext and gives a Groth–Sahai proof knowledge of pad in some auxiliary information. The new ciphertext can then be verified by another algorithm $\mathsf{BlindVf}\left(sk, \tilde{C}, aux\right) \rightarrow b \in \{0, 1\}$ on the input of a secret-key share and of the auxiliary information. Similar techniques can a priori be applied to other publicly-verifiable schemes.

Formal Description.

The algorithms to blind plaintexts and verify that it was done correctly are given below.

$\mathsf{Blind}(pk, C) \to \left(\tilde{C}, aux\right)$:

- if $\mathsf{Vf}(pk, C) = 0$ return $\perp$
- parse $C = (\mathbf{x},\ \mathbf{y},\ \mathbf{c}(\pi),\ \mathbf{c}_0,\ (\mathbf{c}_{1,i})_{i=1}^2,\ \mathbf{d}_0,\ (\mathbf{d}_{1,i})_{i=1}^3,\ \Pi,\ \Theta,\ (\Psi_i)_{i=1,2})$
- generate a blinding factor $R \leftarrow_\$ \mathbb{G}_1$
- blind the payload part of the ciphertext, i.e., $\tilde{x}_3 \leftarrow x_3 R$
- $\tilde{\mathbf{x}} \leftarrow \begin{bmatrix} x_1 & x_2 & \tilde{x}_3 \end{bmatrix}^{\mathrm{T}}$
- $\tilde{\pi} \leftarrow \pi e\left(R, g_2^{s(G_{1,3}E+G_{2,3})}\right)$
- denote by $\mathbf{c}(\tilde{\pi})$ the matrix $\mathbf{c}(\pi)\left[\tilde{\pi}_{i,j}\right]_{i,j=1,2}$, with $\tilde{\pi}_{i,j} \coloneqq \tilde{\pi}$ if $i = j = 2$ and $1_{\mathbb{G}_T}$ otherwise. It is a commitment to $\tilde{\pi}$ with the same randomness as for $\mathbf{c}(\pi)$
- set $\tilde{C} \leftarrow (\tilde{\mathbf{x}},\ \mathbf{y},\ \mathbf{c}(\pi),\ \mathbf{c}_0,\ (\mathbf{c}_{1,i})_{i=1}^2,\ \mathbf{d}_0,\ (\mathbf{d}_{1,i})_{i=1}^3,\ \Pi,\ \Theta,\ (\Psi_i)_{i=1,2})$. Note that $\mathbf{c}(\pi)$ is *not* replaced by $\mathbf{c}(\tilde{\pi})$, as one would not be able to prove that it is well-formed for the scheme of Faonio et al. is RCCA-secure
- commit to R, i.e.,
  - $\sigma_1, \sigma_2 \leftarrow_\$ \mathbb{Z}_p$; $\mathbf{c}(R) \leftarrow \begin{bmatrix} 1 & R \end{bmatrix}^{\mathrm{T}} \mathbf{a}^{\sigma_1}\mathbf{b}^{\sigma_2}$
- Note that

$$\mathbf{c}(\pi) f\left(\mathbf{c}(R), \begin{bmatrix} 1 \\ g_2^{s(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) = f\left(\mathbf{a}, \begin{bmatrix} 1 \\ g_2^{s\sigma_1(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) f\left(\mathbf{b}, \begin{bmatrix} 1 \\ g_2^{s\sigma_2(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) \mathbf{c}(\tilde{\pi}).$$

  The issue is that without the secret key, one cannot compute $g_2^{s\sigma_i(G_{1,3}E+G_{2,3})}$ for $i = 1, 2$. However, g_2^s and g_2^{sE} are part of the ciphertext, so the user can compute

$$\Sigma \leftarrow \begin{bmatrix} g_2^{s\sigma_1} & g_2^{s\sigma_2} \\ g_2^{s\sigma_1 E} & g_2^{s\sigma_2 E} \end{bmatrix}.$$

  Given Σ and $\mathbf{G}$ (which is part of the secret key), one can then compute $g_2^{s\sigma_1(G_{1,3}E+G_{2,3})}$ and $g_2^{s\sigma_1(G_{1,3}E+G_{2,3})}$

- $aux \leftarrow (\mathbf{c}(R), \Sigma)$. It is essentially a designated GS proof intended for any party who knows $\mathbf{G}$
- return $\left(\tilde{C}, aux\right)$.

$\mathsf{BlindVf}\left(sk, \tilde{C}, aux\right) \rightarrow b \in \{0,1\}$:

- verify that commitments $\mathbf{c}_0, (\mathbf{c}_{1,i})_{i=1}^2, \mathbf{d}_0, (\mathbf{d}_{1,i})_{i=1}^3$, are well-formed as in algorithm Dec
- compute $\Sigma_1 \leftarrow \Sigma_{1,1}^{G_{2,3}} \Sigma_{2,1}^{G_{1,3}}$ and $\Sigma_2 \leftarrow \Sigma_{1,2}^{G_{2,3}} \Sigma_{2,2}^{G_{1,3}}$ (this requires the secret key)
- verify with the corrective terms Σ_1 and Σ_2 that the opening to $\mathbf{c}(\pi)$ is the smooth projective hash of the tag, i.e., that

$$
\begin{aligned}
& f\left(\mathbf{c}_0, \begin{bmatrix} 1 \\ g_2 \end{bmatrix}\right) \prod_{i=1,2} f\left(\mathbf{c}_{1,i}, \begin{bmatrix} 1 \\ y_i \end{bmatrix}\right) f\left(\begin{bmatrix} 1 \\ g_1 \end{bmatrix}, \mathbf{d}_0\right) \prod_{i=1,2,3} f\left(\begin{bmatrix} 1 \\ \tilde{x}_i \end{bmatrix}, \mathbf{d}_{1,i}\right) \\
= & f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right) f\left(\Theta_1, \mathbf{v}\right) f\left(\Theta_2, \mathbf{w}\right) \mathbf{c}(\pi) f\left(\mathbf{c}(R), \begin{bmatrix} 1 \\ g_2^{s(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) \\
& f\left(\mathbf{a}, \begin{bmatrix} 1 \\ \Sigma_1^{-1} \end{bmatrix}\right) f\left(\mathbf{b}, \begin{bmatrix} 1 \\ \Sigma_2^{-1} \end{bmatrix}\right)
\end{aligned}
$$

- if both verifications succeed, return 1, else return 0.

6.4.2 Main Construction

This section presents the main construction of an EPAD scheme which is further denoted $\mathcal{E}$. Recall from Section 6.3 that a secure scheme must be at least two rounds. The protocol, which consists of two round, is therefore round optimal.

Building Blocks. The construction uses as building blocks

- the RCCA-secure encryption scheme scheme of Faonio et al. (Section 6.1.6) denoted $\mathcal{E}_{\mathsf{rcca}}$
- the short Cramer–Shoup encryption scheme in $\mathbb{G}_1$ (Section 6.1.6) with hash-function family $\mathcal{H}_{\mathrm{PCA}}$ to encrypt passwords. It is further denoted $\mathcal{E}_{\mathsf{pca}}$
- Groth's one-time signature scheme (see Section 6.1.4), further denoted OTS, with hash-function family $\mathcal{H}_{\mathrm{OTS}}$

- the KV-SPHF for short Cramer–Shoup ciphertexts (Section 6.1.7)
- a (single-keyed) Hash-based Message Authentication Code [BCK96] (HMAC), further denoted MAC, with $\{g_\kappa\}_\kappa$ as family of compression functions (SHA-256 in practice)
- Krawczyk's KDF (Section 6.1.8) denoted KDF with
  - for the extraction phase, HMAC based on a family $\{H_\kappa\}_\kappa$ of Merkle–Damgård hash functions with $\{h_\kappa\}_\kappa$ as underlying family of compression functions (SHA-512 in practice), and
  - for the expansion phase, HMAC with $\{g_\kappa\}_\kappa$ as family of compression functions (SHA-256 in practice)
- the SXDH-based simulation-sound Groth–Sahai proof system which is controllably malleable w.r.t. additive transformations in $\mathbb{Z}_p$ (Section 6.2.5), and which uses Jutla and Roy's scheme (further denoted SIG) as underlying signature scheme (Section 6.1.5). The proof system is further denoted SS_GS.

Construction Overview.

The main steps of the scheme are as follows.

Setup & Key Generation. The parameters include parameters for the scheme of Faonio et al., for the SPHF for Elgamal ciphertexts, and for the simulation-sound GS proof system.

Encryption & Decryption Algorithms. These are the same as the ones of the scheme of Faonio et al.

Interactive Decryption. During the interactive decryption protocol IDec, the parties proceed as follows.

- At the beginning of the protocol, the user and the server essentially do a one-round Password-Authenticated Key Exchange (PAKE) with short Cramer–Shoup encryption of their passwords and KV-SPHF evaluations on them; following techniques of Benhamouda et al. [BBC+13]. The underlying idea is to enable each party to implicitly check via the projective keys that the ciphertext of the other party encrypts the same password as hers (see Section 6.1.7).

  The encrypted passwords are bound to the ongoing session via the projective keys sent during the PAKE and used as labels for the encrypted passwords. It prevents replay attacks since the corresponding hashing keys are needed to recover the PAKE key.

The key obtained later serves two purposes. It is first used as key material for a KDF of which the output is used to authenticate the respective next flows of the server and then the user, thereby making sure that the other party holds the same password. Its second use is to mask the partial decryption of the server so that only a party who has the same password as the server can later remove it and retrieve the plaintext.

In parallel, the token and the server verify that the other party knows a share of the secret key, and the user checks that the token and the server can together reconstruct the secret key. The user therefore makes sure that the server knows *both* a secret-key share, and her password if their PAKE outputs are the same (which she ascertains with the MAC).

Note that, to bind its proof to the ongoing session, the token also signs the proof (and the input from the user) with a one-time scheme. The verification key is used as label for an encryption (with a key different from the one used to encrypt the passwords) of the partial user public key relative to the token share. As only the server can also compute the partial user public key of the token, it can check (given the decryption key) that the token is the party who computed the ciphertext, and that the one-time verification key was not altered.

On the other hand, the server need not do the same as no computation involving the token share is done before the server must prove knowledge of the hashing key corresponding to the projective key sent in the first round. As the projective key is authenticated together with the server proof via the MAC, and as the smoothness of the SPHF guarantees that the hashing key can be recovered with only negligible probability, the server proof is also bound to the ongoing session.

– After the PAKE, the user re-randomizes her ciphertext and blinds the underlying plaintext (as in Section 6.4.1) so that even if the token and the server are corrupt, they cannot infer any information about the ciphertext they help her decrypt. She also authenticates the blinded, re-randomized ciphertext with a key derived with the KDF from the PAKE key as key material. She then sends the ciphertext and the tag to the token.

– The token verifies that the user correctly blinded the plaintext of a valid ciphertext, and that she knows the blinding factor, before forwarding to the server what it just received. This verification could actually be done *in parallel* of the server computation, but before partially decrypting the ciphertext. It is to make sure that no attacker can obtain partial decryption from the token on

invalid ciphertexts and possibly infer information about its share.

- The server verifies the authenticity of the flow via the MAC and then performs the same verifications as the token. If they succeed, it partially decrypts the ciphertext and masks it with the PAKE key. The server then sends the partially decrypted (and masked) ciphertext to the token, along with a proof that she decrypted the ciphertext with the key share of which she proved knowledge *in the first round* (i.e., verification is done w.r.t. the GS commitments to that share and which was were in the first round), and that it masked the result with the PAKE key from the first round. Although the token does *not* know the password shared by the user and the token, it is convinced that the mask is really the PAKE key from the first round. That is because the token verifies the server proof w.r.t. the encrypted passwords and the projective keys that were sent in the first round, and this is absolutely crucial to prevent man-in-the-middle attacks between the token and the server.
- After verifying the proof, the token finishes the decryption and uses the malleability of GS proofs w.r.t. additive transformations to compute, from the server proof, a proof that it and the server both correctly performed their computations. It then sends the decrypted, though blinded (by the user) and masked, plaintext and the proof of correct computation.
- The user verifies the proof, and if it is correct, removes her blinding factor, removes the mask with the PAKE key and can then recover the plaintext.

The proof of security of the scheme requires to be able to guarantee soundness even after giving the adversary simulated proofs, which is why the GS proof system must be CM simulation sound w.r.t. additive transformations. It is due to the fact that secret keys are kept across different sessions.

Formal Description.

The decryption protocol is given on Figure 6.4.2. Each message sent is assumed to be prepended with a session identifier and the identities of the two partner instances. It is assumed that an algorithm aborts if it receives an ill-formed message or if a verification fails, and that it erases all its temporary variables (which include its randomness) once it terminates. The proofs from the token and the server are outlined below.

Notation. Let $[\cdot]\colon \mathbb{G}_1 \to \{0,1\}^\ell$ be an injection, namely the bit representation of $\mathbb{G}_1$ elements. Integers in $\{0,\ldots,p-1\}$ are identified with their

binary representations in $\{0,1\}^\ell$. For three integers $i \in \{1,2\}$, j and k, whenever computational costs in a bilinear structure are considered, $\mathbb{G}_i^j$ denotes a j-exponentiation in $\mathbb{G}_i$ and P^k denotes the computation of product k pairing values. As for communication costs, $j\mathbb{G}_i$ denotes j elements in $\mathbb{G}_i$ and $j\mathbb{Z}_p$ denotes j elements in $\mathbb{Z}_p$.

In the following, SS_GS is a Simulation-Sound Groth Sahai proof system with two main algorithms, a Prove algorithm that generates valid proofs for a statement with respect to the secret used, and Vf that checks if a proof is valid with respect to a statement. An extra algorithm Eval allows to further refine a proof by using extra witnesses.

OTS is a one time signature, with a KG key generation algorithm, a Sign signing algorithm, and a verification Vf algorithm checking the consistency of the signature with respect to its public key.

$\mathcal{E}_{\mathsf{pca}}$ is a plaintext checkable encryption scheme, while $\mathcal{E}_{\mathsf{rcca}}$ is a randomizable CCA encryption scheme allowing access to extra algorithms: a randomization Rand algorithm, and Blind/BlindVf that respectively allow to blind the payload in a ciphertext, and to check consistency to know whether the decryption algorithm will return a value different from $\perp$.

HashKG, ProjKG, Hash, ProjHash are the four algorithms constituting an SPHF, to generate keys to allow to implicitly prove the validity of a statement with respect to a given witness.

Parameters. Given two families $\mathcal{H}_{\mathrm{PCA}}$ and $\mathcal{H}_{\mathrm{OTS}}$ of hash functions from $\{0,1\}^*$ to $\{0,1\}^\ell$, to generate public parameters,

- generate a bilinear structure $\Gamma \leftarrow (p, \mathbb{G}_1 = \langle g_1\rangle, \mathbb{G}_2 = \langle g_2\rangle, \mathbb{G}_T, e) \leftarrow_{\$} \mathsf{G}\left(1^\lambda\right)$. Let $\ell \leftarrow \lfloor \log_2 p \rfloor + 1$ be the bit length of p
- select $H_{\mathrm{PCA}} \leftarrow_{\$} \mathcal{H}_{\mathrm{PCA}}$ and $H_{\mathrm{OTS}} \leftarrow_{\$} \mathcal{H}_{\mathrm{OTS}}$
- generate a salt value $XTS \leftarrow_{\$} \{0,1\}^\ell$ for the KDF
- generate keys for the short Cramer–Shoup encryption scheme in $\mathbb{G}_1$, i.e.,

  Decryption key $dk = (\zeta, \alpha, \beta, \alpha', \beta')$

  Encryption key $ek = (h_1, \gamma, \delta) = \left(g_1^\zeta,\ g_1^\alpha h_1^\beta,\ g_1^{\alpha'} h_1^{\beta'}\right)$
- generate parameters for $\mathcal{E}_{\mathsf{rcca}}$, i.e., GS parameters $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1^2$; $\mathbf{v}, \mathbf{w} \in \mathbb{G}_2^2$ in soundness mode
- generate parameters for SS_GS, i.e., GS parameters $\tilde{\mathbf{a}}, \tilde{\mathbf{b}} \in \mathbb{G}_1^2$; $\tilde{\mathbf{v}}, \tilde{\mathbf{w}} \in \mathbb{G}_2^2$ in soundness mode and a pair of keys $(vk, sk) \leftarrow \mathsf{SIG.KG}(\Gamma, 2)$
- return $pp \leftarrow crs \leftarrow \left(\Gamma; \mathbf{a}, \mathbf{b}; \mathbf{v}, \mathbf{w}; \tilde{\mathbf{a}}, \tilde{\mathbf{b}}; \tilde{\mathbf{v}}, \tilde{\mathbf{w}}; H_{\mathrm{PCA}}, ek;\ H_{\mathrm{OTS}}; vk; XTS\right)$.

Passwords. Passwords are elements of $\mathbb{G}_1$; $1_{\mathbb{G}_1}$ is not valid password.

Key Generation. To generate keys for $\mathcal{E}$, first run $(pk_{\mathcal{U}}, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathcal{E}_{\mathsf{rcca}}.\mathsf{KG}(pp)$. The secret-key shares $sk_{\mathcal{T}}/sk_{\mathcal{S}} \leftarrow \big(\alpha_{1,\mathcal{T}/\mathcal{S}},\ \alpha_{2,\mathcal{T}/\mathcal{S}},\ \mathbf{G},\ A_1,\ A_2,\ \mathbf{C}_1,\ \mathbf{C}_2\big)$, are such that $\alpha_{i,\mathcal{T}} \leftarrow_{\$} \mathbb{Z}_p$ and $\alpha_{i,\mathcal{S}} = \alpha_i - \alpha_{i,\mathcal{T}}$ for $i = 1, 2$. Generate also another pair $(ek', dk') \leftarrow \mathcal{E}_{\mathsf{pca}}.\mathsf{KG}(p, g_1, \mathbb{G}_1, H)$. Note that $pk_{\mathcal{U}}$ is of the form $\big(\Gamma, pk_1 \coloneqq g_1^D,\ \ pk_2 \coloneqq g_1^{\alpha_1 D + \alpha_2}, \ldots\big)$. The token is given ek', and the server dk'. The whole token share (cf. the syntax of Section 6.3) is then $(sk_{\mathcal{T}}, ek')$ and the whole server share is $(sk_{\mathcal{S}}, dk')$.

Acceptance and Termination. The U instance accepts after verifying tag $\tau_{\mathcal{S}}$, and terminates after returning M. The T instance accepts after receiving $\big(\tilde{C}_M, aux, \tau_{\mathcal{U}}\big)$, and terminates after sending $M_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$ to the user. The S instance accepts after verifying $\tau_{\mathcal{U}}$, and terminates after sending $M_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$ to the token.

First Proofs $\tilde{\pi}_{\mathcal{T}}$ from the Token. In the first round, the token proves with $\mathsf{SS\_GS}$ to the server that it knows the other share of the user secret key. Proof $\tilde{\pi}_{\mathcal{T}}$ then consists of 20 $\mathbb{G}_1$ element, 26 $\mathbb{G}_2$ elements and 2 $\mathbb{Z}_p$ elements, and verifying it costs $4\mathbb{G}_1^1 + 4\mathbb{G}_2^1 + 2\mathrm{P}^4 + 6\mathrm{P}^6 + 8\mathrm{P}^5 + 4\mathrm{P}^8$.

First Proofs $\tilde{\pi}_{\mathcal{S}}$ from the Server. Leveraging the malleability of $\mathsf{SS\_GS}$, the server computes a proof of knowledge of the complete user secret key.

Second Proof $\pi_{\mathcal{S}}$ from the Server. After the server partially decrypts the ciphertext and masks the result with an SPH, it proves that it correctly performed its computation. In particular, the server proves the server proves that secret-key shares it used to partially decrypt are the ones to which it committed *in the first round*. It also proves that the password it encrypted in the first round is the same that is used to computed the SPH mask.

Equations. Let $\xi \leftarrow H(U, E, hp_{\mathcal{S}})$. The server must prove knowledge of $p_{\mathcal{S}}$, $hk_{\mathcal{S}} \coloneqq (\lambda, \mu, \nu, \theta)$ and $r_{\mathcal{S}}$ such that

$$
\begin{aligned}
hp_{\mathcal{S},1} &= g_1^{\lambda} h_1^{\nu} \gamma^{\theta} \\
hp_{\mathcal{S},2} &= g_1^{\mu} \delta^{\theta} \\
C_{\mathcal{S}} = (U, E, V) &= \left(g_1^{r_{\mathcal{S}}}, h_1^{r_{\mathcal{S}}} p_{\mathcal{S}}, \left(\gamma \delta^{\xi}\right)^{r_{\mathcal{S}}}\right) \\
M_{\mathcal{S}} \tilde{x}_3^{-1} &= \tilde{x}_1^{-\alpha_{1,\mathcal{S}}} \tilde{x}_2^{-\alpha_{2,\mathcal{S}}} U^{\lambda + \mu\xi} E^{\nu} \left(1/p_{\mathcal{S}}\right)^{\nu} V^{\theta} \left(hp_{\mathcal{U},1} hp_{\mathcal{U},2}^{\xi}\right)^{r_{\mathcal{S}}}.
\end{aligned}
$$

With $\mathsf{SS\_GS}$, proof $\pi_{\mathcal{S}}$ consists of 38 $\mathbb{G}_1$ elements, 42 $\mathbb{G}_2$ elements and 2 $\mathbb{Z}_p$ elements. Verifying it costs $4\mathbb{G}_1^1 + \mathbb{G}_2^2 + 2\mathrm{P}^3 + 4\mathrm{P}^4 + 4\mathrm{P}^5 + 6\mathrm{P}^6 + 6\mathrm{P}^7 + 4\mathrm{P}^9 + 8\mathrm{P}^{10} + 2\mathrm{P}^{13}$. Note that the token verifies w.r.t. to the commitments $\mathbf{c}(\alpha_{i,\mathcal{S}})$ it infers from the first proof $\tilde{\pi}_{\mathcal{S}}$ and its secret-key share.

Figure 6.2: Decryption Protocol.

Second Proof $\pi_{\mathcal{T}}$ from the Token. Upon receiving $\pi_{\mathcal{S}}$ from the server, the token decryption algorithm first verifies it. If $\pi_{\mathcal{S}}$ is correct, the token algorithm uses the malleability of GS proofs to compute a proof that decryption was done with the full secret key of which knowledge was proved in the first round.

Correctness & Security.

This section shows that $\mathcal{E}$ is correct and satisfies the security properties defined in Section 6.3.

The P-IND-RCCA security and the verifiability of $\mathcal{E}$ rely of the following assumptions (see also Section 6.1.8):

- $\{h_\kappa\}_\kappa$ is pairwise-independent
- $\{H_\kappa\}_\kappa$ is collision-resistant against linear-size circuits
- $g.$ is a secure Pseudo-Random Function (PRF)
- $\hat{g}\colon (K, M) \to g_M(K)$ is a secure PRF under a class of affine related-key attacks defined by the inner and outer pads [Bel06, Lemma 5.2].

Assuming that the SXDH assumption over G holds, that $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, $g.$ and $\hat{g}.$ satisfy the assumptions above, that $\mathcal{H}_{\mathrm{PCA}}$ is second-preimage resistant and that $\mathcal{H}_{\mathrm{OTS}}$ is collision-resistant, $\mathcal{E}$ is P-IND-RCCA secure. Moreover, $\mathcal{E}$ satisfies blindness under the SXDH assumption over G. Lastly, $\mathcal{E}$ is verifiable if $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, $g.$ and $\hat{g}.$ satisfy the assumptions above and if the SXDH assumption over G holds.

Theorem 6.4.3 (Correctness)**.** *$\mathcal{E}$ is correct.*

Proof. This immediately follows from the correctness and completeness of each building block of the interactive-decryption protocol. □

Theorem 6.4.4 (P-IND-RCCA)**.** *Assuming that the SXDH assumption over G holds, that $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, $g.$ and $\hat{g}.$ satisfy the assumptions above, that $\mathcal{H}_{\mathrm{PCA}}$ is second-preimage resistant and that $\mathcal{H}_{\mathrm{OTS}}$ is collision-resistant, $\mathcal{E}$ is P-IND-RCCA secure.*

Proof. Let $\mathcal{A}$ be an adversary for the P-IND-RCCA game which makes at most q_{Exec} Exec queries and at most q_{Send} Send queries. A message is subsequently said to be *oracle-generated* if it was computed by the challenger as a response to a Send query and it was not altered. Otherwise, the message is said to be *adversarially generated*. If an adversarially generated message is given as an input to an algorithm which simply forwards it (possibly after some verifications), the message is still considered adversarially generated. Recall that only static corruptions are considered. However, $\mathcal{A}$ can

of course modify the messages sent between the parties. Recall also that all instances are assumed to erase their temporary variables (which include their randomness) upon termination.

Further distinguish the following cases:

1. $\mathcal{A}$ never makes a Send query to an $\mathcal{S}$ instance on a valid tuple (i.e., which passes all verification) $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that the pair $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated but $(C_{\mathcal{T}}, ovk)$ is adversarially generated, although $\mathcal{T}$ and $\mathcal{S}$ are honest.

2. $\mathcal{A}$ makes a Send query of the above form.

In the first case, the main idea is to define a game which is indistinguishable from the real one, but in which the adversary cannot modify the messages computed by honest parties within a session; unless it knows their passwords in the case of user and server identities. Moreover, the oracle-generated messages in this latter game are independent of the passwords, so as long as a user and one of her servers are honest, $\mathcal{A}$ can only guess their common password, and it can only do so with probability at most $|D|^{-1}$. Winning the 1-out-of-2 IND-RCCA security of $\mathcal{E}_{\mathsf{rcca}}$ can then be reduced to winning that game.

To this end, consider the sequence of games hereafter. It starts by modifying how Exec queries are handled, and then continues with Send queries.

Game 0. This is the real P-IND-RCCA game.

Game 1. In this game, the challenger generates a trapdoor $\mathsf{SS\_GS}$ CRS, i.e., a signing key sk which allows to simulate proofs, as well as the discrete-logarithm relations between $\tilde{a}_1$ and $\tilde{a}_2$, and between $\tilde{b}_1$ and $\tilde{b}_2$, which allow to extract committed group elements. As the resulting parameters are perfectly indistinguishable from honest parameters, Game 1 and Game 0 are perfectly indistinguishable.

Game 2. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger simulates $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$. Note that since $\mathsf{SS\_GS}$ satisfies perfect derivation privacy, proofs in the previous game (leveraging its malleability) are perfectly indistinguishable from proofs computed with the full secret key. By the zero-knowledge property of $\mathsf{SS\_GS}$, these latter are indistinguishable from simulates proofs of the full secret key.

The zero-knowledge property of $\mathsf{SS\_GS}$ implies that $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Exec}})$, with the latter denoting the supremal advantage of any PPT adversary in distinguishing real proofs from simulated ones.

Note that even if $\mathcal{S}$ is corrupt at the beginning of the protocol and $\mathcal{A}$ thus knows the witness of the proofs, it still cannot distinguish real

proofs from simulated ones since $\mathsf{SS\_GS}$ is zero-knowledge. Moreover, as temporary variables of the token algorithm instance are erased upon termination, $\mathcal{A}$ cannot distinguish real proofs from simulated ones by corrupting $\mathcal{S}$ after the execution either.

Game 3. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger computes $H_{\mathcal{U}} = H_{\mathcal{S}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p\mathcal{U}}, \mathcal{C}_{\mathcal{S}}\right) \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p\mathcal{S}}, \mathcal{C}_{\mathcal{U}}\right)$. Game 3 from Game 2 are perfectly indistinguishable by correctness of the SPHF.

Game 4. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger now computes $\mathcal{C}_{\mathcal{U}}$ as $\mathsf{Enc}^{hp_{\mathcal{U}}}\left(ek, 1_{\mathbb{G}_1}\right)$ and $\mathcal{C}_{\mathcal{S}}$ as $\mathsf{Enc}^{hp_{\mathcal{S}}}\left(ek, 1_{\mathbb{G}_1}\right)$ (recall that $1_{\mathbb{G}_1}$ is assumed not to be a valid password).

The indistinguishability of Game 4 from Game 3 stems from the IND-CPA security (implied by the IND-PCA security) of $\mathcal{E}_{\mathsf{pca}}$ which relies on the SXDH assumption. The distinguishing advantage of $\mathcal{A}$ is at most $q_{\mathsf{Exec}}\varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\mathrm{ind-cpa}}$, with $\varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\mathrm{ind-cpa}}$ denoting the supremal advantage of any efficient adversary in the IND-CPA game with scheme $\mathcal{E}_{\mathsf{pca}}$.

Game 5. The challenger now answers Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$ by choosing $H_{\mathcal{U}} = H_{\mathcal{S}}$ uniformly at random. This game is perfectly indistinguishable from the previous one by the smoothness property of the SPHF.

Game 6. In this game, the challenger also saves the short-Cramer–Shoup decryption key dk. Game 6 is perfectly indistinguishable from Game 5.

Game 7. To answer Send queries to an $\mathcal{S}$ instance on $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ and on $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$, the challenger simulates $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$. The challenger also simulates $\tilde{\pi}_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$ to answer queries to $\mathcal{T}$ instances. The same arguments as in Game 2 imply that $\mathcal{A}$ can distinguish Game 7 from Game 6 with an advantage of at most $\varepsilon_{\mathsf{SS\_GS}}^{zk}(q_{\mathsf{Send}})$.

Game 8. The challenger now answers Send queries on $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ to an $\mathcal{S}$ instance as follows:

- if $\mathsf{OTS.Vf}\left(ovk, \left(C_{\mathcal{U}}, hp_{\mathcal{U}}, \tilde{\pi}_{\mathcal{T}}\right), \sigma_{\mathcal{T}}\right) \neq 1$ or $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) \neq pk_2 pk_1^{-\alpha_{1,\mathcal{S}}} g_1^{-\alpha_{2,\mathcal{S}}}$ or $\mathsf{GS.Vf}\left(crs, pk_{\mathcal{U}}, \alpha_{i,\mathcal{S}}, \tilde{\pi}_{\mathcal{T}}\right) \neq 1$, return $\perp$
- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated and the tuple of the Send query also is, compute $H_{\mathcal{S}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p\mathcal{U}}, \mathcal{C}_{\mathcal{S}}\right) \cdot \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p\mathcal{S}}, \mathcal{C}_{\mathcal{U}}\right)$ (the challenger knows $hk_{\mathcal{U}}$ since $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is honestly generated). In this case, this game is perfectly indistinguishable from the previous one by the correctness of the SPHF

- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ and $(C_{\mathcal{T}}, ovk)$ are oracle-generated (i.e., the tuple $(*, ovk, *, C_{\mathcal{T}}, *)$ was computed as an answer to a Send query) but the tuple is adversarially generated, if $\mathcal{T}$ is honest, abort as $\mathcal{A}$ contradicted the strong one-time security of OTS. If $\mathcal{T}$ is corrupt, then reply with $\perp$ if $\mathcal{U}$ is corrupt; if $\mathcal{U}$ is honest, compute $H_{\mathcal{S}} \leftarrow \mathsf{Hash}\,(hk_{\mathcal{U}},\ \mathcal{L}_{p_{\mathcal{U}}}, \mathcal{C}_{\mathcal{S}}) \cdot \mathsf{Hash}\,(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, \mathcal{C}_{\mathcal{U}})$.

  Denoting by $\varepsilon_{\mathsf{OTS}}$ the supremal advantage of any efficient adversary in the strong one-time security game, $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $|I|\varepsilon_{\mathsf{OTS}}$ (the reduction algorithm must guess the $\mathcal{T}$ instance for which it sets the one-time signing key as that of the challenger)
- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated but $(C_{\mathcal{T}}, ovk)$ is adversarially generated,
  - if $\mathcal{T}$ is corrupt then
    - if $(\mathcal{U}, \mathcal{T}, \mathcal{S}) = (\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*)$ and $\mathcal{U}^*$ is corrupt, reply with $\perp$ thereby following the definition of oracle Send
    - else, compute $H_{\mathcal{S}} \leftarrow \mathsf{Hash}\,(hk_{\mathcal{U}},\ \mathcal{L}_{p_{\mathcal{U}}}, \mathcal{C}_{\mathcal{S}}) \cdot \mathsf{Hash}\,(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, \mathcal{C}_{\mathcal{U}})$
  - if $\mathcal{T}$ is honest, then $\mathcal{S}$ is necessarily corrupt as $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated and $(C_{\mathcal{T}}, ovk)$ is adversarially generated (cf. conditions of case 1). Compute $H_{\mathcal{S}}$ as in the real game
- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is adversarially generated and $\mathcal{U}$ and $\mathcal{S}$ are honest, check whether $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{hp_{\mathcal{U}}}(dk, C_{\mathcal{U}}) = p_{\mathcal{S}}^{\mathcal{U}}$. If so, i.e., $\mathcal{A}$ correctly guessed $p_{\mathcal{S}}^{\mathcal{U}}$, abort and return 1 indicating that $\mathcal{A}$ won the game (which increases the advantage of $\mathcal{A}$ in this game). If not, generate $H_{\mathcal{S}}$ uniformly at random, and the perfect smoothness of the SPHF implies the perfect indistinguishability from the previous game
- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is adversarially generated and $\mathcal{U}$ or $\mathcal{S}$ is corrupt,
  - if $(\mathcal{U}, \mathcal{T}, \mathcal{S}) = (\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*)$ and $\mathcal{T}^*$ is corrupt, return $\perp$. By definition of oracle Send, Game 8 is perfectly indistinguishable from Game 7
  - if $\mathcal{T}$ is honest, compute $H_{\mathcal{S}}$ as in the real game.

The advantage of $\mathcal{A}$ in the previous game is therefore upper-bounded by its advantage in Game 8 plus $q_{\mathsf{Send}}|I|\varepsilon_{\mathsf{OTS}}$.

Game 9. The challenger now answers Send on $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ to a $\mathcal{U}$ instance as follows:

- if $\mathsf{SS\_GS.Vf}\,(crs, pk_{\mathcal{U}}, \tilde{\pi}_{\mathcal{S}}) \neq 1$, return $\perp$

- if $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is oracle-generated, compute $H_{\mathcal{U}} \leftarrow \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, \mathcal{C}_{\mathcal{U}}) \cdot \mathsf{Hash}(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, \mathcal{C}_{\mathcal{S}})$ (the challenger knows $hk_{\mathcal{S}}$ as $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is honestly generated). In this case, Game 9 is perfectly indistinguishable from Game 8 by the correctness of the SPHF
- if $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is adversarially generated and $\mathcal{S}$ and $\mathcal{U}$ are honest, check whether $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{hp_{\mathcal{S}}}(dk, C_{\mathcal{S}}) = p_{\mathcal{U}}$. If so (i.e., $\mathcal{A}$ correctly guessed $p_{\mathcal{U}}$), abort and return 1. If not, choose $H_{\mathcal{U}}$ uniformly at random, and the perfect smoothness of the SPHF implies the perfect indistinguishability from the previous game
- if the tuple is adversarially generated and $\mathcal{S}$ or $\mathcal{U}$ is corrupt, compute $H_{\mathcal{U}}$ as in the real game.

The advantage of $\mathcal{A}$ in the previous game is thus upper-bounded by its advantage in this game.

From this game onwards, the randomness used to encrypt passwords is necessary to compute neither $H_{\mathcal{U}}$ nor $H_{\mathcal{S}}$ in case $\mathcal{U}$ and $\mathcal{S}$ are honest.

Game 10. In this game, the challenger answers Send queries to

- an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}},\ C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}},\ hp_{\mathcal{U}})$ is oracle-generated, and which are oracle-generated if $\mathcal{T}$ is honest, by generating $H_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$ in case $\mathcal{U}$ is honest
- a $\mathcal{U}$ instance on oracle-generated tuples $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$ by setting $H_{\mathcal{U}} \leftarrow H_{\mathcal{S}}$, the latter being the value of the partner $\mathcal{S}$ instance of the $\mathcal{U}$ in the current session

A lemma by Katz and Vaikuntanathan [KV11, Lemma 1] states that SPH values are computationally indistinguishable from uniformly random group elements even if hashing keys and ciphertexts are used several times and if projective keys made public, under the assumption that the encryption scheme is IND-PCA secure (they actually prove it in the case of IND-CCA security, but the application of the lemma only makes use of plaintext-check queries) and that the SPHF is statistically smooth. The lemma then entails that $\mathcal{A}$ can distinguish Game 10 from Game 9 with an advantage of at most $2q_{\mathsf{Send}}^2 \varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\mathrm{ind-pca}}(q_{\mathsf{Send}})$, with $\varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\mathrm{ind-pca}}(q_{\mathsf{Send}})$ denoting the supremal advantage of any efficient adversary which makes at most q_{Send} plaintext-check queries in the IND-PCA game with scheme $\mathcal{E}_{\mathsf{pca}}$.

Game 11. For prompting queries $(\mathcal{T}, *, \mathcal{S}, *, *)$ to a $\mathcal{U}$ instance and Send queries to an $\mathcal{S}$ instance on oracle-generated tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}},\ C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is also oracle-generated, if $\mathcal{U}$ and $\mathcal{S}$ are honest, the challenger computes $C_{\mathcal{S}}$ as $\mathsf{Enc}^{hp_{\mathcal{U}}}(ek, 1_{\mathbb{G}_1})$ and $C_{\mathcal{U}}$ as

$\mathsf{Enc}^{hp_{\mathcal{S}}}(ek, 1_{\mathbb{G}_1})$. Distinguishing Game 11 from Game 10 can then be reduced to winning the IND-PCA game for $\mathcal{E}_{\mathsf{pca}}$.

Note that for any Send query on $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ to an $\mathcal{S}$ instance, if the tuple is oracle-generated and $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ also is, the reduction algorithm already knows the password encrypted in $C_{\mathcal{U}}$ and can then do the same test as the challenger of Game 8. Similarly for Send queries on tuples $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ to $\mathcal{U}$ instances. In case $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ or $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is adversarially generated, the reduction can make a plaintext-check queries.

It follows that $\mathcal{A}$ can distinguish Game 11 from Game 10 with an advantage at most $2q_{\mathsf{Send}}\varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})$. Note also that from this game on, the messages computed by instances of $\mathcal{U}$ and $\mathcal{S}$ are independent of the passwords if they are both honest, $\mathcal{A}$ can then guess their common password with probability at most $|D|^{-1}$ at each Send query.

Game 12. For Send queries to an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated, and which are oracle-generated if $\mathcal{T}$ is honest, the challenger generates $K_{\mathcal{S}}$ uniformly at random in case $\mathcal{U}$ is honest. For a Send query to a partner $\mathcal{U}$ instance on an oracle-generated tuple $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$, the challenger sets $K_{\mathcal{U}} \leftarrow K_{\mathcal{S}}$.

Distinguishing this game from the previous can be reduced to the security of KDF w.r.t. uniformly random sources. Note that since the distribution of the source is independent of the adversary, generating the salt value XTS before the end of the PAKE does not raise any issue in the reduction. Indeed, the reduction algorithm can generate a uniformly random source value at the beginning of the reduction, submit it to the KDF-security-game challenger, then receive back a uniform salt value before generating the other parameters for $\mathcal{E}$.

Denoting by $\varepsilon_{\mathsf{KDF}}(0)$ the supremal advantage of any efficient adversary which makes no oracle query in the KDF security game with scheme KDF, adversary $\mathcal{A}$ can distinguish Game 12 from Game 11 with an advantage at most $|I|^2\varepsilon_{\mathsf{KDF}}(0)$ (the reduction algorithm guesses the instances of $\mathcal{U}$ and $\mathcal{S}$ for which it sets the keys as the key returned by the KDF challenger).

Game 13. For a Send query on an adversarially generated tuple $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ to a $\mathcal{U}$ instance, abort the protocol if $\mathcal{S}$ and $\mathcal{U}$ are honest. Moreover, if the tuple is oracle-generated in a different session, abort the protocol. Likewise, for a Send query on a an adversarially generated tuple $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$ to an $\mathcal{S}$ instance, if $\mathcal{U}$ and $\mathcal{S}$ are honest, abort the protocol. Besides, if the tuple is oracle-generated in a different session, abort the protocol.

Distinguishing Game 13 from Game 12 can be reduced to the security of MAC. It follows $\mathcal{A}$ can distinguish Game 13 from Game 12 with an advantage of at most $|I|^2\varepsilon^{\mathrm{prf}}_{\mathsf{MAC}}(1)$, with $\varepsilon^{\mathrm{prf}}_{\mathsf{MAC}}(1)$ being the supremal advantage of any efficient adversary which makes at most one oracle query in the PRF game with scheme MAC (the reduction algorithm guesses the $\mathcal{U}$ and $\mathcal{S}$ instance for which it sets the common MAC key as the challenge one).

Game 14. For a Send query to a $\mathcal{T}$ instance on pair $(M_{\mathcal{S}}, \pi_{\mathcal{S}})$ which is either adversarially generated or oracle-generated in a different session, abort if $\mathcal{U}$ and $\mathcal{S}$ are honest. Indeed, the word for which $\pi_{\mathcal{S}}$ is a proof is generated anew for each session since $hk_{\mathcal{S}}$ is always freshly generated (and $hp_{\mathcal{S}}$ is thus not the image of a previous projective key under an additive transformation). It follows that

- if $(M_{\mathcal{S}}, \pi_{\mathcal{S}})$ is oracle generated in a different session, the verification can only succeed with negligible probability if $\mathsf{SS\_GS}$ is sound. In this case, $\mathcal{A}$ can distinguish this game from the previous with an advantage of at most $\varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}})$, with the latter denoting the supremal advantage of any efficient adversary which makes at most q_{Send} queries in the CM-simulation-soundness game for $\mathsf{SS\_GS}$
- if $(M_{\mathcal{S}}, \pi_{\mathcal{S}})$ is adversarially generated, the CM simulation soundness of $\mathsf{SS\_GS}$ guarantees that the commitments $\mathbf{c}(\lambda)$, $\mathbf{c}(\mu)$, $\mathbf{c}(\nu)$, $\mathbf{c}(\theta)$ satisfy

$$f\left(g_1, \mathbf{c}(\lambda)\right) f\left(h_1, \mathbf{c}(\nu)\right) f\left(\gamma, \mathbf{c}(\theta)\right) = f\left(hp_1, \mathbf{u}\right) f\left(\Theta_{hp_{\mathcal{S},1}}, \mathbf{v}\right)$$

and

$$f\left(g_1, \mathbf{c}(\mu)\right) f\left(\delta, \mathbf{c}(\theta)\right) = f\left(hp_2, \mathbf{u}\right) f\left(\Theta_{hp_{\mathcal{S},2}}, \mathbf{v}\right)$$

(with proof elements excerpted from $\pi_{\mathcal{S}}$). However, these commitments can then be used to contradict the perfect smoothness of the KV-SPHF for short Cramer–Shoup ciphertext. Indeed, given $M \in \mathbb{G}_1$ and $C \coloneqq (U, E, V)$ such that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{hp_{\mathcal{S}}}(dk, C) \neq M$, these commitments and $hp_{\mathcal{S}}$ can be used to distinguish $\mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_M, C)$ from $\mathfrak{H} \leftarrow_{\$} \mathbb{G}_1$ simply by testing whether

$$\begin{aligned} &f\left(U, \mathbf{c}(\lambda)\right) f\left(U^{\xi}, \mathbf{c}(\mu)\right) f\left(E/M, \mathbf{c}(\nu)\right) f\left(V, (\theta)\right) \\ =&f\left(\mathfrak{H}, \mathbf{u}\right) f\left(\Theta_{hp_{\mathcal{S},1}} \Theta^{\xi}_{hp_{\mathcal{S},2}}, \mathbf{v}\right). \end{aligned}$$

The perfect smoothness of the KV-SPHF for short Cramer–Shoup ciphertexts implies that it can only occur with probability $1/p$.

Therefore, in this case, the adversary can distinguish this game from the previous one with an advantage of at most

$$q_{\mathsf{Send}}\left(\varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}}) + p^{-1}\right)$$

(the algorithm for the reduction to the perfect smoothness of the SPHF must guess the query for which it sets the projective key as $hp_{\mathcal{S}}$).

$\mathcal{A}$ can then distinguish this game from the previous one with an advantage of at most $q_{\mathsf{Send}}\left(\varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}}) + p^{-1}\right)$.

In Game 14, once the challenge tuple $(\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*)$ is defined, all values received by $\mathcal{U}^*$, $\mathcal{T}^*$ and $\mathcal{S}^*$ instances up to $(M_{\mathcal{S}^*}, \pi_{\mathcal{S}^*})$ included are either all oracle-generated in the same session or replied to with $\perp$ as long as these identities are honest. In particular, if $\mathcal{U}^*$ is corrupt, Send queries to $\mathcal{T}^*$ instances are rejected (by definition of oracle Send) and $\mathcal{A}$ thus cannot make successful Send queries to $\mathcal{S}^*$ instances anymore as long as $\mathcal{T}^*$ is honest; and if $\mathcal{U}^*$ and $\mathcal{T}^*$ are corrupt, Send queries to $\mathcal{S}^*$ instances are rejected anyway. Likewise, if $\mathcal{S}^*$ is corrupt, Send queries to $\mathcal{T}^*$ instances are rejected (recall that $\mathcal{S}^*$ and $\mathcal{T}^*$ cannot both be corrupt) and $\mathcal{A}$ cannot make successful Send queries to $\mathcal{S}^*$ instances anymore. Winning the 1-out-of-2 RCCA game for $\mathcal{E}_{\mathsf{rcca}}$ can then reduced to winning Game 14 as follows.

At the beginning of the game, the reduction algorithm, further denoted $\mathcal{B}$, guesses the identities $\mathcal{U}^*$, $\mathcal{T}^*$ and $\mathcal{S}^*$. If $\mathcal{A}$ later makes its Test query on a different tuple of identities, $\mathcal{B}$ simply aborts and sends to the challenger a bit chosen uniformly at random.

Recall that $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt by definition of P-IND-RCCA game. On this account, first suppose that $\mathcal{S}^*$ is honest throughout the game. $\mathcal{B}$ then sets the public key as the public key of $\mathcal{U}^*$ and asks for a secret key share that it sets as the share of $\mathcal{T}^*$ (which is given to $\mathcal{A}$ in case of corruption). The other share is then implicitly the share of party $\mathcal{S}^*$.

For Exec queries and Send queries on oracle-generated tuples of the form $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}^*}\right)$ to $\mathcal{S}^*$ instances, since $\tilde{C}_M$ is computed by $\mathcal{B}$, the latter can make decryption queries to the RCCA challenger on the original ciphertext C_M, receive back $x_3 x_1^{-\alpha_{1,\mathcal{S}^*}} x_2^{-\alpha_{1,\mathcal{S}^*}}$, and multiply it by $RH_{\mathcal{S}^*}$, with R denoting the blinding factor generated by algorithm Blind.

$\mathcal{B}$ also simulates with the trapdoor proofs $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$.

For query Test, algorithm $\mathcal{B}$ simply forwards (M_0, M_1) to the RCCA challenger. After the test query, for Exec queries as above, if C_M decrypts to M_0 or M_1, the RCCA challenger answers with replay and so does $\mathcal{B}$.

For a prompting Send query on $(\mathcal{T}^*, *, \mathcal{S}^*, *, C_M)$ to a $\mathcal{U}^*$ instance, the reduction makes a decryption query to the RCCA challenger on C_M. If it decrypts to M_0 or M_1, the reduction algorithm receives replay and forwards it to $\mathcal{A}$.

The condition "if $(\mathsf{Dec}(sk_{\mathcal{U}}, C) \in \{M_0, M_1\})$ return replay" guarantees that $\mathcal{B}$ can answer with $\perp$ Exec queries and prompting Send queries on C^*, and perfectly emulate the Game-14 challenger.

Moreover, recall that in case 1), $\mathcal{A}$ never makes a Send query on a valid tuple $(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*}, ovk, \tilde{\pi}_{\mathcal{T}^*}, C_{\mathcal{T}^*}, \sigma_{\mathcal{T}^*})$ such that $(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*})$ is oracle-generated but $(C_{\mathcal{T}^*}, ovk)$ is adversarially generated, although $\mathcal{T}^*$ and $\mathcal{S}^*$ are honest. It means that if $\mathcal{U}^*$ is honest, $\mathcal{A}$ can obtain partial decryption from $\mathcal{S}^*$ only with oracle-generated tuples, and the condition mentioned above ensures that $\mathcal{S}^*$ never has to make a decryption query on a ciphertext which decrypts to M_0 or M_1. If $\mathcal{U}^*$ is corrupt and $\mathcal{T}^*$ is honest, $\mathcal{B}$ never has to make such a decryption query either as $\mathcal{A}$ would never submit a valid tuple in the first flow. If $\mathcal{U}^*$ and $\mathcal{T}^*$ are both corrupt, then $\mathcal{B}$ can simply answer Send queries with $\perp$.

As $\mathcal{B}$ perfectly emulates the Game-14 challenger, it wins the RCCA game with at least the same advantage as $\mathcal{A}$ in that game.

In case $\mathcal{T}^*$ is honest throughout the game, the reduction is similar except that $\mathcal{B}$ now sets the key share it gets as the share of $\mathcal{S}^*$. Note that in Game 14, all valid pairs $(M_{\mathcal{S}^*}, \pi_{\mathcal{S}^*})$ submitted to $\mathcal{T}^*$ instances are oracle-generated in the same session if $\mathcal{U}^*$ and $\mathcal{S}^*$ are honest. If either of them is corrupt, all Send queries to $\mathcal{T}^*$ instances are rejected. Consequently, $\mathcal{B}$ never has to make a decryption query on a ciphertext that decrypts to M_0 or M_1.

It follows that the advantage of $\mathcal{A}$ in the P-IND-RCCA game in case 1) is at most

$$\begin{aligned}
&\varepsilon^{zk}_{\mathsf{SS_GS}}(q_{\mathsf{Exec}}) + \varepsilon^{zk}_{\mathsf{SS_GS}}(q_{\mathsf{Send}}) + q_{\mathsf{Exec}}\varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}} + |I|\varepsilon_{\mathsf{OTS}} \\
&+ q_{\mathsf{Send}}\left(|D|^{-1} + \varepsilon^{\mathrm{sound}}_{\mathsf{SS_GS}}(q_{\mathsf{Send}}) + p^{-1} + 2(q_{\mathsf{Send}}+1)\varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})\right) \\
&+ |I|^2\left(\varepsilon_{\mathsf{KDF}}(0) + \varepsilon^{\mathrm{prf}}_{\mathsf{MAC}}(1)\right) + |U||T||S|\mathsf{Adv}^{\mathrm{ind-rcca}}_{\mathsf{G},\mathcal{E}_{\mathsf{rcca}}}(\lambda).
\end{aligned}$$

In the second case, $\mathcal{T}$ and $\mathcal{S}$ are honest in the query that distinguishes the two cases. The major argument is that only $\mathcal{T}$ and $\mathcal{S}$ can compute $pk_1^{\alpha_{1,\mathcal{T}}} g_1^{\alpha_{2,\mathcal{T}}}$ and that only $\mathcal{S}$ holds the decryption dk'. Therefore, an adversary cannot distinguish $C_{\mathcal{T}}$ from encryption of a dummy message which contains no information about the token share. Moreover, as long as $\mathcal{U}$ and $\mathcal{S}$ are honest, $H_{\mathcal{S}}$ is indistinguishable from a uniformly random value and $M_{\mathcal{S}}$ thus contains no information about the shares either. For the adversary to compute a valid tuple from that point, it has to compute a valid proof on the token share after only getting zero-knowledge ones. This valid proof can then be used to contradict the 1-out-of-2 security of $\mathcal{E}_{\mathsf{rcca}}$.

To prove it formally, consider the following sequence of games.

Game 0–5. These are the same as in the previous case.

Game 6. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger computes, the challenger generates $K_{\mathcal{U}} \leftarrow K_{\mathcal{S}}$ uniformly at random. $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $|I|\varepsilon_{\mathsf{KDF}}(0)$.

Game 7. In this game, to answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger generates $M_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$. This game is perfectly indistinguishable from the previous one as $H_{\mathcal{S}}$ is uniformly random in the latter.

Game 8. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger now computes $C_{\mathcal{T}}$ as $\mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}(ek', 1_{\mathbb{G}_1})$ (and verifies that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) = 1_{\mathbb{G}_1}$). The indistinguishability from the previous games stems from the IND-CPA security of $\mathcal{E}_{\mathsf{pca}}$ (which is implies by its IND-PCA security). Therefore, $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $q_{\mathsf{Exec}}\varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}}$, with $\varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}}$ denoting the supremal advantage of any efficient adversary in the IND-CPA game with scheme $\mathcal{E}_{\mathsf{pca}}$.

Game 9. To compute $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$ to answer Send queries to $\mathcal{S}$ instances, the challenger now simulates them with the trapdoor. Likewise, the challenger simulates $\tilde{\pi}_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$ to answers Send queries to $\mathcal{T}$ instances. The zero-knowledge property of $\mathsf{SS\_GS}$ implies that $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}})$.

Game 10. The challenger now answers Send queries to an $\mathcal{S}$ instance on $(C_{\mathcal{U}},\ hp_{\mathcal{U}}, ovk,\ \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated by computing $H_{\mathcal{S}} \leftarrow \mathsf{Hash}(hk_{\mathcal{U}},\ \mathcal{L}_{p_{\mathcal{U}}}, \mathcal{C}_{\mathcal{S}}) \cdot \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}},\ \mathcal{C}_{\mathcal{U}})$. This game is perfectly indistinguishable from the previous one by the correctness of the SPHF.

Game 11. The challenger now answers Send to a $\mathcal{U}$ instance on a tuple $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ such that $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is oracle-generated by computing $H_{\mathcal{U}} \leftarrow \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, \mathcal{C}_{\mathcal{U}}) \cdot \mathsf{Hash}(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, \mathcal{C}_{\mathcal{S}})$. This game is perfectly indistinguishable from the previous one by the correctness of the SPHF.

Game 12. In this game, the challenger answers Send queries to

- an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that the couple $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated by generating $H_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$ in case $\mathcal{U}$ is honest
- a $\mathcal{U}$ instance on oracle-generated tuples $(\tilde{\pi}_{\mathcal{S}},\ C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$ by setting $H_{\mathcal{U}} \leftarrow H_{\mathcal{S}}$, the latter being the value of the partner $\mathcal{S}$ instance of the $\mathcal{U}$ in the current session

Katz and Vaikuntanathan's lemma [KV11, Lemma 1] implies that $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $2q_{\mathsf{Send}}^2 \varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\mathrm{ind-pca}}(q_{\mathsf{Send}})$.

Game 13. For Send queries to an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated, the challenger generates $K_{\mathcal{S}}$ uniformly at random in case $\mathcal{U}$ is honest. For a Send query to a partner $\mathcal{U}$ instance on an oracle-generated tuple $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$, the challenger sets $K_{\mathcal{U}} \leftarrow K_{\mathcal{S}}$.

Distinguishing this game from the previous can be reduced to the security of KDF w.r.t. uniformly random sources.

Denoting by $\varepsilon_{\mathsf{KDF}}(0)$ the supremal advantage of any efficient adversary which makes no oracle query in the KDF security game with scheme KDF, this game can be distinguished from the previous one with an advantage of at most $|I|^2 \varepsilon_{\mathsf{KDF}}(0)$.

Game 14. In this game, if $\mathcal{U}$ is honest, the challenger answers Send queries to $\mathcal{S}$ instances on $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$ tuples by generating $M_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$. This game is perfectly indistinguishable from the previous one as $H_{\mathcal{S}}$ is uniformly random in the latter.

Game 15. To answer Send queries to $\mathcal{T}$ instances on oracle-generated pairs $(C_{\mathcal{U}}, hp_{\mathcal{U}})$, the challenger now generates $K \leftarrow_{\$} \mathbb{G}_1$ and computes $C_{\mathcal{T}}$ as $\mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}(ek', K)$. To answer Send queries to the partner $\mathcal{S}$ instance in this session on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, the challenger verifies that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) = K$. The indistinguishability from the previous game follows from the IND-PCA security of $\mathcal{E}_{\mathsf{pca}}$. Indeed, for queries to $\mathcal{S}$ instances as above, if $(C_{\mathcal{T}}, ovk)$ is oracle-generated (i.e., $(*, ovk, *, C_{\mathcal{T}}, *)$ was computed as an answer to a Send query), then the reduction need not make a decryption query as it computed it itself. If $(C_{\mathcal{T}}, ovk)$ is adversarially generated, i.e., $C_{\mathcal{T}}$ was computed with a label different from ovk, the reduction algorithm can then make a decryption query.

Distinguishing this game from the previous one can thus be done with an advantage of at most $q_{\mathsf{Send}} \varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\mathrm{ind-pca}}(q_{\mathsf{Send}})$.

Game 16. To answer Send queries to $\mathcal{T}$ instances on oracle-generated pairs $(C_{\mathcal{U}}, hp_{\mathcal{U}})$, the challenger now generates $K, K' \leftarrow_{\$} \mathbb{G}_1$ and computes $C_{\mathcal{T}}$ as $\mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}(ek', K)$. To answer Send queries to the partner $\mathcal{S}$ instance in this session on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, if $(C_{\mathcal{T}}, ovk)$ is oracle-generated in the same session, the challenger skips the verification, otherwise the challenger verifies that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) = K'$. Once again, indistinguishability from the previous game follows

from the IND-PCA security of $\mathcal{E}_{\mathsf{pca}}$. Adversary $\mathcal{A}$ can thus distinguish this game from the previous one with an advantage of at most $q_{\mathsf{Send}}\varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})$.

Game 17. For Send queries to $\mathcal{S}$ instances on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, if ovk and $\pi_{\mathcal{T}}$ are oracle-generated but in different sessions, return $\perp$. Denoting by $\varepsilon_{\mathsf{OTS}}$ the supremal advantage of any efficient adversary in the strong one-time security game, $\mathcal{A}$ can distinguish this game from the previous with an advantage of at most $q_{\mathsf{Send}}|I|\varepsilon_{\mathsf{OTS}}$ (the reduction algorithm must guess the $\mathcal{T}$ instance for which it sets the one-time signing key as that of the challenger).

Game 18. For Send' queries to $\mathcal{S}$ instances on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, if ovk is adversarially generated and $\pi_{\mathcal{T}}$ is oracle-generated, the challenger returns $\perp$. Adversary $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $1/p$. Indeed, if $\pi_{\mathcal{T}}$ is oracle-generated, there exists a K' as defined in Game 16 which is independent of all the other messages computed by the challenger and of all the inputs from $\mathcal{A}$.

Winning the 1-out-of-2 RCCA game can then be reduced to winning the last game as follows. At the beginning of the game, the reduction algorithm, denoted $\mathcal{B}$, guesses again the identities $\mathcal{U}^*$, $\mathcal{T}^*$ and $\mathcal{S}^*$. (If $\mathcal{A}$ later makes its Test query on a different tuple of identities, $\mathcal{B}$ simply aborts and sends to the challenger a bit chosen uniformly at random.) $\mathcal{B}$ then sets the public key as the public key of $\mathcal{U}^*$ and asks for a secret key share that it sets as the share of $\mathcal{S}^*$. The other share is then implicitly the share of $\mathcal{T}^*$.

Recall that if $\mathcal{A}$ corrupts $\mathcal{U}^*$, algorithm $\mathcal{B}$ can reply to Send queries to $\mathcal{T}^*$ instances with $\perp$ and perfectly emulate the challenger of the last game.

Whenever $\mathcal{A}$ makes its first Send query to an $\mathcal{S}^*$ instance on a valid tuple $(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*}, ovk, \tilde{\pi}_{\mathcal{T}^*}, C_{\mathcal{T}^*}, \sigma_{\mathcal{T}^*})$ such that $(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*})$ is oracle-generated but $(C_{\mathcal{T}^*}, ovk)$ is adversarially generated, ovk and $\pi_{\mathcal{T}^*}$ are necessarily adversarially generated by definition of the challenger of the last game. By CM simulation soundness of $\mathsf{SS\_GS}$ w.r.t. additive transformations (which holds under the existential unforgeability of SIG), either (i) $f\left(pk_1, \mathbf{c}(\alpha_{1,\mathcal{T}^*})\right) f\left(g_1, \mathbf{c}(\alpha_{2,\mathcal{T}^*})\right) = f\left(pk_2 pk_1^{-\alpha_{1,\mathcal{S}^*}} g_1^{-\alpha_{2,\mathcal{S}^*}}, \mathbf{u}\right) f\left(\Theta, \mathbf{v}\right)$, or there exists an oracle-generated proof (respectively by an $\mathcal{S}^*$ instance or by a $\mathcal{T}^*$ instance) $(\mathbf{c}_1, \mathbf{c}_2, \Theta')$ such that (ii-a) $f\left(pk_1, \mathbf{c}_1\right) f\left(g_1, \mathbf{c}_2\right) = f\left(pk_2, \mathbf{u}\right) f\left(\Theta', \mathbf{v}\right)$ or (ii-b) $f\left(pk_1, \mathbf{c}_1\right) f\left(g_1, \mathbf{c}_2\right) = f\left(pk_2\ pk_1^{-\alpha_{1,\mathcal{S}^*}} g_1^{-\alpha_{2,\mathcal{S}^*}}, \mathbf{u}\right) f\left(\Theta', \mathbf{v}\right)$ and a tuple $(\mathbf{c}(\zeta_1), \mathbf{c}(\zeta_2), \Theta'')$, with $(\zeta_1, \zeta_2) \in \mathbb{Z}_p^2$ representing a transformation, such that

$$f\left(pk_1, \mathbf{c}(\zeta_1)\right) f\left(g_1, \mathbf{c}(\zeta_2)\right) = f\left(pk_2 pk_1^{-\alpha_{1,\mathcal{S}^*}}\ g_1^{-\alpha_{2,\mathcal{S}^*}}, \mathbf{u}\right) f\left(\Theta'', \mathbf{v}\right).$$

It follows that for $i = 1, 2$ either $\mathbf{c}(\alpha_{i,\mathcal{T}^*})$ or $\mathbf{c}(\zeta_i)$ is a commitment to $\alpha_{i,\mathcal{T}^*}$; and assume without loss of generality that it is the former. $\mathcal{B}$ then computes

$\mathbf{c}(\alpha_i) \leftarrow \mathbf{c}(\alpha_{i,\mathcal{T}^*})\mathbf{u}^{\alpha_{i,\mathcal{S}^*}}$ for $i = 1, 2$, i.e., commitments to α_i with the same randomness used to compute $\mathbf{c}(\alpha_{i,\mathcal{T}^*})$.

If $\mathcal{A}$ makes a Test query on the guessed identities, $\mathcal{B}$ simply forwards (M_0, M_1) to the RCCA challenger, and receives back a challenge ciphertext C^* which encrypts M_b for $b \leftarrow_\$ \{0, 1\}$. (If the guess was incorrect, $\mathcal{B}$ aborts its interaction with $\mathcal{A}$ and sends a uniformly random bit to the challenger.) Note that $x_3^* M_b^{-1} = x_1^{*\alpha_1} x_2^{*\alpha_2}$ and that $\mathrm{dlog}_{pk_1} x_1^* = \mathrm{dlog}_{g_1} x_2^* = \mathrm{dlog}_{pk_2} x_3^*$. Therefore, $f\left(x_1^*, \mathbf{c}(\alpha_1)\right) f\left(x_2^*, (\alpha_2)\right) = f\left(x_3^* M_b^{-1}, \mathbf{u}\right) f\left(\Theta, \mathbf{v}\right)$. Algorithm $\mathcal{B}$ can then test the previous equality with M_0 and M_1 and win the 1-out-of-2 RCCA game with at least the same advantage as $\mathcal{A}$ in the selective P-IND-RCCA game.

Consequently, in case 2), $\mathcal{A}$ wins the P-IND-RCCA game with an advantage of at most

$$\begin{aligned}
&\varepsilon^{zk}_{\mathsf{SS_GS}}(q_{\mathsf{Exec}}) + \varepsilon^{zk}_{\mathsf{SS_GS}}(q_{\mathsf{Send}}) + q_{\mathsf{Exec}}\varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}} + |I|\varepsilon_{\mathsf{OTS}} + 2|I|^2\varepsilon_{\mathsf{KDF}}(0) \\
&+ 2q_{\mathsf{Send}}(q_{\mathsf{Send}} + 1)\varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}}) + \varepsilon^{\mathrm{sound}}_{\mathsf{SS_GS}}(q_{\mathsf{Send}}) \\
&+ |U||T||S|\mathsf{Adv}^{\mathrm{ind-rcca}}_{\mathsf{G},\mathcal{E}_{\mathsf{rcca}}}(\lambda).
\end{aligned}$$

□

Theorem 6.4.5 (Blindness)**.** *$\mathcal{E}$ satisfies blindness under the SXDH assumption over G.*

Proof. The blindness property of $\mathcal{E}$ can be proved as follows via a sequence of indistinguishable games starting from the real game and ending with a game in which the advantage of any adversary is nil.

Game 0. This is the real game.

Game 1. The challenger generates GS parameters $\mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{w}$ in witness-indistinguishability mode. This CRS is computationally indistinguishable from the one in the previous game under the SXDH assumption over G.

Game 2. Instead of computing $\mathbf{c}(R)$ and Σ as algorithm Blind, the challenger simulates those values with the GS proof-system simulator. Further denote the resulting algorithm as $\mathsf{SimBlind}$. Game 2 is perfectly indistinguishable from Game 1 since the simulation is perfect.

Game 3. In this game, the challenger runs $\mathsf{SimBlind}$ on $\mathsf{Enc}\left(pk, \mathsf{Dec}(sk, C_b)\right)$. Game 3 is perfectly indistinguishable from Game 2 since the scheme of Faonio et al. is perfectly unlikable.

Game 4. Instead of running algorithm $\mathsf{SimBlind}$ on $\mathsf{Enc}\left(pk, \mathsf{Dec}(sk, C_b)\right)$, the challenger runs it on $\mathsf{Enc}\left(pk, K\right)$ for $K \leftarrow_\$ \mathbb{G}_T$. Game 4 is perfectly

indistinguishable from Game 3 since $\mathsf{SimBlind}$ computes $\tilde{x}_3$ as $x_3 R$ for $R \leftarrow_{\$} \mathbb{G}_T$, which entirely re-randomizes $\mathsf{Dec}(sk, C_b)$ in Game 3.

Note that in Game 4, the advantage of any adversary is nil.

It follows that $\mathcal{E}$ satisfies blindness under the SXDH assumption over G. □

Theorem 6.4.6 (Verifiability)**.** *$\mathcal{E}$ is verifiable if $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, g. and $\hat{g}$. satisfy the assumptions above and if the SXDH assumption over G holds.*

Proof. Suppose that there exists an efficient adversary $\mathcal{A}$ which wins the verifiability game with a non-negligible probability, i.e., it returns $p_{\mathcal{U}}$ and C such that the honest execution of U on $(pk, p_{\mathcal{U}}, C)$ with $\mathcal{A}$ results in a value different from $\mathsf{Dec}(sk, C)$ and $\perp$. In the event in which $\mathcal{A}$ wins the game, then either $\mathsf{Dec}(dk, C_{\mathcal{S}}) = p_{\mathcal{U}}$ or not. If so, then there exists an algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as sub-routine and contradicts the perfect soundness of the GS proof system. If $\mathsf{Dec}(dk, C_{\mathcal{S}}) \neq p_{\mathcal{U}}$, then there exists an algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as sub-routine and wins the MAC game with non-negligible probability.

In the first case, the correctness of the SPHF guarantees that $\mathcal{H}(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}})\,\mathsf{ProjHash}(hp_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{U}}}, \perp, r_{\mathcal{U}}) = \mathcal{H}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}})\,\mathsf{ProjHash}(hp_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{S}}}, \perp, r_{\mathcal{S}})$. Therefore, if the value returned by U at the end of the protocol is different from both $\mathsf{Dec}(sk, C)$ and $\perp$, adversary $\mathcal{A}$ necessarily contradicted the soundness of $\mathsf{SS\_GS}$ for to the language

$$
\begin{aligned}
\{(ek, pk_{\mathcal{U}}, C_{\mathcal{U}}, C_{\mathcal{S}},\ hp_{\mathcal{U}}, \tilde{C}_M, M_{\mathcal{S}}) \colon \exists\,(\alpha_i, hk_{\mathcal{S}}, p_{\mathcal{S}}, r_{\mathcal{S}})\,,\\
pk_2 &= pk_1^{\alpha_1} g_1^{\alpha_2}\\
hp_{\mathcal{S}} &= \mathsf{ProjKG}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, \perp)\\
C_{\mathcal{S}} &= \mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{hp_{\mathcal{S}}}(ek, p_{\mathcal{S}}; r_{\mathcal{S}})\\
M_{\mathcal{S}}\tilde{x}_3^{-1} &= \tilde{x}_1^{-\alpha_1}\tilde{x}_2^{-\alpha_2}\mathcal{H}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}})\\
&\quad \cdot\mathsf{ProjHash}(hp_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{S}}}, \perp, r_{\mathcal{S}})\}\,,
\end{aligned}
$$

which is impossible under the existential unforgeability of Jutla and Roy's signature, which relies on the SXDH assumption.

In the second case, i.e., if $\mathsf{Dec}(dk, C_{\mathcal{S}}) \neq p_{\mathcal{U}}$, the verifiability of $\mathcal{E}$ can be reduced to the security of the MAC through a sequence of games as below.

Game 0. This is the real game.

Game 1. In this game, the challenger replaces $H_{\mathcal{U}}$ with a uniformly random value. By the smoothness of the SPHF, Game 1 is perfectly indistinguishable from Game 0.

Game 2. The challenger now generates a random key $K_{\mathcal{U}}$ instead of computing it with the KDF. The indistinguishability of Game 2 from Game

1 can then be reduced to the security of KDF w.r.t. uniformly random source.

Note that as the distribution of the source is independent of the adversary, generating the salt value XTS before the end of the PAKE does not raise any issue in the reduction. Indeed, the reduction algorithm can generate a uniformly random source value at the beginning of the reduction, submit it to the KDF-security-game challenger, then receive back a uniform salt value before generating the other parameters for $\mathcal{E}$.

Therefore, under the assumptions on the compression functions which imply the security of Krawczyk's KDF, an efficient adversary can distinguishing Game 2 from Game 1 with advantage at most $\varepsilon_{\mathsf{KDF}}(q_{\mathsf{Send}})$, with $\varepsilon_{\mathsf{KDF}}(q_{\mathsf{Send}})$ denoting the supremal advantage of any efficient adversary which makes at most q_{Send} queries in the KDF security game with scheme KDF.

As $\mathcal{A}$ must compute a tuple $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}}, \tau_{\mathcal{S}})$ such that $\mathsf{MAC}\,(K_{\mathcal{U}}, (\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}})) = \tau_{\mathcal{S}}$ to win Game 0 without any prior MAC computation by U, it does so with non-negligible probability in Game 2. It follows that $\mathcal{A}$ can then be run as sub-routine to win the MAC game with non-negligible probability. Once again, under the assumptions on the compression functions (which imply the security of HMAC), an efficient adversary can only do so with negligible probability; a contradiction. It follows that such an adversary $\mathcal{A}$ cannot exist and $\mathcal{E}$ is thus verifiable. □

Efficiency.

Table 6.1 sums up the communication cost of the decryption protocol.

	$\mathsf{U} \rightleftharpoons \mathsf{T}$	$\mathsf{T} \rightleftharpoons \mathsf{S}$
1st Flow	$5\mathbb{G}_1$	$31\mathbb{G}_1 + 26\mathbb{G}_2 + 4\mathbb{Z}_p$
2nd Flow	$25\mathbb{G}_1 + 26\mathbb{G}_2 + 3\mathbb{Z}_p$	$25\mathbb{G}_1 + 26\mathbb{G}_2 + 3\mathbb{Z}_p$
3rd Flow	$16\mathbb{G}_1 + 19\mathbb{G}_2 + 4\mathbb{G}_T$ $+1\mathbb{Z}_p$	$16\mathbb{G}_1 + 19\mathbb{G}_2 + 4\mathbb{G}_T$ $+1\mathbb{Z}_p$
4th Flow	$39\mathbb{G}_1 + 42\mathbb{G}_2 + 1\mathbb{Z}_p$	$39\mathbb{G}_1 + 42\mathbb{G}_2 + 1\mathbb{Z}_p$

Table 6.1: Communication Cost of the Decryption Protocol.

On Adaptive Corruptions.

The main reason why the decryption protocol is not secure against adaptive corruptions is that the short Cramer–Shoup encryption is "fully committing", meaning that there is only one valid opening (i.e., message–randomness pair) for each commitment (i.e., ciphertext). However, in one of the intermediate games in the proof of P-IND-RCCA security, the challenger computes commitments to dummy passwords for honest parties. It means that if the adversary corrupts an honest party right after she has sent her commitment, the challenger cannot return to the adversary a valid opening which contains the actual password of that party.

To overcome this hurdle, one could instead use an equivocable commitment scheme which supports adaptively smooth KV-SPHFs. As an equivocable commitment scheme allows to compute a valid opening to any commitment given a trapdoor, such a scheme together with a KV-SPHF should make the protocol secure against adaptive corruptions. Blazy and Chevalier's commitment scheme and its associated KV-SPHF [BC16] precisely satisfy these conditions. The commitment scheme relies on the SXDH assumption, and although its size is constant in the bit length of the message (i.e., 4 $\mathbb{G}_1$ elements and 2 $\mathbb{G}_2$ elements for each commitment), it is still larger than that of the short Cramer–Shoup encryption scheme. The latter was then chosen for efficiency reasons but at the expense of adaptive corruptions.

Note also that if $\mathcal{U}^*$ is corrupted right after the end of the PAKE (i.e., at the end of the first round), then the adversary could get from the server instance a partial decryption on the challenge ciphertext with the third flow (even if the $\mathcal{U}^*$ instance was prompted on a different ciphertext). To prevent this, the token could again compute a one-time signature as in the first flow, but also include C_M in the label of the encryption. Doing ensures the server that the third flow went through the token, which would not be possible in case of corruption of $\mathcal{U}^*$ as token queries are then rejected.

On Composability.

The Section-6.3 definitions are game-based and do not guarantee composability. However, a universal-composability [Can01] functionality for EPAD schemes could be defined in the same vein as for PAKE [CHK$^+$05] and RCCA-secure encryption schemes [CKN03] considered together. Yet, to achieve composability, and as for PAKE protocols, it would be necessary for the UC simulator (in case the adversary correctly guesses the password of an honest party) to be able to compute correct SPH values on *invalid* words C (encrypting $1_{\mathbb{G}_1}$) with the sole knowledge of a projective key hp from the adversary and not of the corresponding hashing key hk.

Benhamouda et al. [BBC$^+$13] introduced trapdoor SPHFs exactly for this purpose and gave a construction for the original Cramer–Shoup en-

cryption scheme which is computationally and adaptively smooth under the SXDH assumption. It could readily be adapted to the short version of their scheme, though each projective key would contain one more $\mathbb{G}_1$ element. Alternatively, one could also use structure-preserving SPHFs which were introduced by Blazy and Chevalier [BC16].

Mitigating Server Breaches.

EPAD schemes have so far been defined only w.r.t. a single server. Nevertheless, password theft from server databases is common in practice, and users even tend to use the same password for several services. It means that not only the confidentiality of user messages is threatened if a single server is compromised (and if her token is corrupt), but also potential other services.

To mitigate the impact of server breaches, a potential solution is to use threshold cryptography [Sha79, DF90]. User passwords are then encrypted in a database and the decryption key is shared between $n \geq 1$ servers so that no information about the passwords is leaked if at most a number t of them are corrupt. Nevertheless, any $t + 1$ servers should be able to recover user passwords. Blazy, Chevalier and Vergnaud [BCV16, Section 5] proposed an efficient protocol for threshold PAKE [MSJ06] from SPHFs which is based on this idea. It allows a user and a gateway that interacts with $t + 1$ of servers to agree on a common high-entropy key if the user password and the encrypted password match, without revealing any information about the passwords. However, if they do not, the keys obtained by each party are independent and uniformly random. In consequence, the security of the key exchange is guaranteed so long as at most t servers are corrupt.

The EPAD scheme in Section 6.3 can then be turned into a scheme with n servers and which withstand the corrupt of t of them (i.e., a t-out-of-n scheme) as follows. During the set-up phase, the user encrypts her password and sends the encrypted password and a t-out-of-n secret-key share to each server. The $\mathcal{E}_{\mathsf{rcca}}$ secret key is $t + 1$-out-of-$n + 1$ shared between the n servers and the token so that any $t + 1$ servers and the token can decrypt user ciphertexts. During the decryption protocol, each of the participating servers plays in parallel the role of the gateway of Blazy, Chevalier and Vergnaud's protocol, and uses the resulting keys to authenticate the second flow in the protocol of Figure 6.4.2. In the last flow, each server would then have to prove that it partially decrypted the ciphertext with its $\mathcal{E}_{\mathsf{rcca}}$ key share and masked it with the key resulting from the threshold PAKE. As the threshold PAKE is based on SPHFs, the same techniques (as in Section 6.3) leveraging malleability apply.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Chapter 5 presented zone encryption, a cryptographic primitive tailored to the needs of C-ITSs which can enhance their privacy properties. When used in combination with the group signatures from Chapter 4 augmented with attributes, vehicles send 216-Byte authentication tokens, while needing only a single constant-size credential per epoch. This is an important improvement over the limited pseudonym pool sizes and their expensive reloading protocols that are currently proposed for deployment. Secondly, our zone encryption scheme enables efficient encryption of position beacon messages, protecting their content from eavesdroppers.

Moreover, all variants of group signatures and the multi-signature scheme presented in Chapter 4 constitute contributions of independent interest with applications beyond the scope of vehicle-to-vehicle communication.

Despite these improvements, these schemes should still be used with some care. Actively participating eavesdroppers can still receive all communication, so minimizing the information contained in CAMs for the particular envisaged applications remains crucial. Besides, authentication tokens leak the identity of the issuers, i.e., vehicles are only anonymous among other vehicles with the same issuers. This is easily circumvented at an organizational level by letting all vehicles use the same (e.g., country-wide) issuers. If that is not possible, technical solutions would involve delegatable credentials [BCC+09], but their tokens are too long to be used in C-ITS.

It should be stressed that there are further privacy limitations inherent to V2X communication, as vehicles can be tracked by other means than CAMs. One could, e.g., fingerprint radio transmitters and antennas, use side-channel analysis, or even cameras and image processing to track vehicles [TCMDS11]. However, the possibility of such attacks does not mean that privacy for vehicular communication should be entirely forgone. In fact, similar arguments can be made for most applications, e.g., users' online ac-

tivities which can be fingerprinted through the hardware they use. Still, efficient privacy-preserving protocols are still sought after instead abandoning the idea of online privacy altogether.

Chapter 6 tackled public-key encryption in a context in which secure storage is unavailable. It introduced encryption with password-protected assisted decryption, a new primitive which allows users to decrypt ciphertexts with the help of an untrusted token and of a server with which they share a password, while maintaining the privacy of users. The security of this primitive was formalized via game-based definitions, and an efficient, provably-secure construction was proposed.

7.2 Future Work

The main construction of group signatures in Chapter 4 can only tolerate the corruption of less than half of the issuers and openers. In contrast, the construction based on multi-signatures does not support threshold issuance but can tolerate the corruption of all but one issuer. Hence the question of whether it is possible to construct as efficient group signatures with both threshold issuance and opening, but with higher thresholds than half of the authorities.

Besides, as explained in Section 4.3.2 and contrarily to what could be hoped for, the anonymity of the scheme still requires some issuers to be honest and likewise, some openers must be honest to guarantee traceability. It would thus be interesting to build a scheme, perhaps based on different ideas, that can overcome these shortcomings while maintaining signatures as short.

Concerning zone encryption, the main hurdle for a practical deployment remains the key-agreement strategy among vehicles that minimizes zone clustering. Designing strategies and testing them via simulations of real-world scenarios would then be an important step for privacy in V2V communication.

As for encryption with password-assisted decryption, defining its security in the universal-composability framework [Can01], with adaptive corruptions, would provide stronger security guarantees and would be an interesting contribution on its own, although that may come at the price of efficiency.

Part II

Zero-Knowledge Arguments and Randomness Certification

Chapter 8

Introduction

The focus of this second part is on general-purpose cryptographic primitives that are also critical to privacy-preserving applications. Chapter 9 gives zero-knowledge arguments for a wide class of cryptographic problems, namely those that can be represented by Diophantine equations. As for Chapter 10, it addresses the fundamental question of randomness in the context of key generation, which is a cornerstone of any cryptographic construction.

8.1 Diophantine Satisfiability

A *Diophantine equation* is a multi-variate polynomial equation with integer coefficients, and it is satisfiable if it has a solution with all unknowns taking integer values. Davis, Putnam, Robinson and Matiyasevich [Mat70] showed that any computational problem can be modeled as finding a solution of such equations, thereby proving that the general Diophantine satisfiability problem is undecidable and giving a negative answer to Hilbert's tenth problem. For instance, several classical NP-problems such as 3-SAT, Graph 3-colorability or Integer Linear Programming can be readily encoded as Diophantine equations. Several cryptographic problems such as proving knowledge of an RSA signature, that a committed value is non-negative or that encrypted votes are honestly shuffled by a mix-net, can also be encoded as Diophantine equations.

Efficient zero-knowledge arguments of knowledge of solutions to Diophantine equations, if a solution is known to a party, can thus be useful for many practical cryptographic tasks; and doing so requires to do zero-knowledge proofs on committed integers.

8.1.1 Prior Work

Integer Commitments. Fujisaki and Okamoto [FO97] presented the first efficient integer commitment scheme and also suggested a zero-knowledge protocol for verifying multiplicative relations over committed values. Such a commitment scheme allows to commit to any $x \in \mathbb{Z}$ in a group of unknown order, with a Pedersen-like commitment scheme. This makes the security analysis more intricate since division modulo the unknown group order cannot be performed in general. As an evidence that this setting is error-prone, it was shown by Michels that the Fujisaki–Okamoto proof system was flawed. Damgård and Fujisaki [DF02] later proposed a statistically-hiding and computationally binding integer commitment scheme under standard assumptions in a hidden-order group $\mathbb{G}$ with an efficient argument of knowledge of openings to commitments, and arguments of multiplicative relations over committed values. This primitive gives rise to a (honest-verifier) zero-knowledge proof of satisfiability of a Diophantine equation with M multiplications over $\mathbb{Z}$ that requires $\Omega(M)$ integer commitments and requires $\Omega(M)$ proofs of multiplicative relations [DF02, Lip03]. These complexities have not been improved since then.

Circuit Satisfiability over $\mathbb{Z}_p$. Similarly, it is possible to design a zero-knowledge proof of satisfiability of an arithmetic circuit over $\mathbb{Z}_p$ using Pedersen's commitment scheme [Ped92] in a group $\mathbb{G}$ of public prime order p. An immediate solution is to use the additive homomorphic properties of Pedersen's commitment and zero-knowledge protocols for proving knowledge of the contents of commitments and for verifying multiplicative relations over committed values [Sch91, CS97].

For an arithmetic circuit with M multiplication gates, this protocol requires $\Omega(M)$ commitments and $\Omega(M)$ arguments of multiplication consistency and has a communication complexity of $\Omega(M)$ group elements. In 2009, Groth [Gro09] proposed a sub-linear size zero-knowledge arguments for statements involving linear algebra and used it to reduce this communication complexity to $O\left(\sqrt{M}\right)$ group elements. This breakthrough initiated a decade of progress for zero-knowledge proofs for various statements (see e.g., [BG13, GK15, BCC$^+$16, BBB$^+$18] and references therein). It culminated with the argument system "*Bulletproofs*" proposed by Bünz, Bootle, Boneh, Poelstra, Wuille and Maxwell [BBB$^+$18] which permits to prove the satisfiability of such an arithmetic circuit with communication complexity $O(\log(M))$ and round complexity $O(\log(M))$. The corner stone of their protocol is an argument that two committed vectors satisfy an inner-product relation. It has logarithmic communication and round complexity in the vector length; and its security only relies on the discrete-logarithm assumption and does not require a trusted setup.

Circuit satisfiability over any finite field is an NP-complete problem so

the "*Bulletproofs*" argument system has a widespread application. However, as mentioned above, in many cryptographic settings, it is desirable to prove statements such as "the committed value x is a valid RSA signature on a message m for an RSA public key (N, e)". In this case, the prover has to convince the verifier that $x^e = H(m) \bmod N$, or in other words that there exists an integer k such that $x^e + kN = H(m)$ where this equality holds over the integers for $|k| \leq N^{e-1}$ and H is some cryptographic hash function. In order to use directly an argument of satisfiability of an arithmetic circuit to prove the knowledge of a pair (x, k) which satisfies this equation, one needs to use a group $\mathbb{G}$ a prime order p with $p > N^e$ (and to additionally prove that $x < N$ and $k < N^e$). For a large e, this approach results in a proof with prohibitive communication complexity.

Moreover, in various settings, such as the Integer-Linear-Programming problem, there is no *a priori* upper-bound on the sizes of the integer solutions during setup when p is defined. Being able to argue on integers instead of residue classes modulo a fixed prime integer then becomes necessary. Besides, generic reductions to circuit satisfiability over prime-order fields for some simple problems naturally defined over the integers may return circuits with a very large number of multiplication gates and even the "*Bulletproofs*" argument system could produce large proofs. Modeling computational problems using Diophantine equations is more versatile, and a succinct argument system for Diophantine satisfiability thus has many potential applications.

8.2 Public-Key Generation with Verifiable Randomness

Cryptographic protocols are commonly designed under the assumption that the protocol parties have access to perfect (i.e., uniform) randomness. However, random sources used in practical implementations rarely meet this assumption and provide only a stream of bits with a certain "level of randomness".

The quality of the random numbers directly determines the security strength of the systems that use them. Following preliminary work by Juels and Guajardo [JG02] and Corrigan-Gibbs, Mu, Boneh and Ford [CMBF13], Chapter 10 revisits the problem of proving that a cryptographic user algorithm has selected and correctly used a truly random seed in the generation of her cryptographic public–secret key pair.

8.2.1 Related Work.

A prominent example that the use of randomness in public-key cryptography (and especially in key-generation protocols) is error-prone is the recent randomness failure known as the *ROCA vulnerability* [NSS+17]. This weak-

ness allows a private key to be recovered efficiently from the public key only (in factoring-based cryptography). The flawed key-generation algorithm selects specific prime numbers as part of the private key instead of generating uniformly random primes and many certified devices were shown vulnerable (e.g., Estonian and Slovakian smartcards and standard cryptographic libraries). This kind of weaknesses is not new as in 2012, Lenstra, Hughes, Augier, Bos, Kleinjung and Wachter [LHA+12] did a sanity check of factoring-based public keys collected on the web. They showed that a significant percentage of public keys (0.5%) share a common prime factor, and this fact was explained [HDWH12] by the generation of these low entropy keys during booting. Since cryptographic failures due to weak randomness can be dire [NSS+17, LHA+12, HDWH12], designers should build schemes that can withstand deviations of the random sources from perfect randomness.

Following seminal works by Simmons on the threat of covert channels (also called subliminal channels) in cryptography [Sim83], the concept of *kleptography* was proposed by Young and Yung [YY97]. It models the fact that an adversary may subvert cryptographic algorithms by modifying their implementations in order to leak secrets using for instance covert channels present in randomized algorithms. Several sources have recently revealed that cryptographic algorithms have effectively been subverted to undermine the security of users. This raises the concern of guaranteeing a user's security even when she may be using a compromised machine or algorithm. Motivated by the (in)famous potential backdoor on the Dual Elliptic Curve Deterministic Random Bit Generator (Dual EC DRBG) [CMG+16], Bellare, Paterson, and Rogaway [BPR14] initiated a formal analysis of kleptographic attacks on symmetric key encryption algorithms. For factoring-based public-key cryptography, in light of the known shortcomings of implemented key generators, a line of research has focused on proving that RSA moduli satisfy certain properties [GMR98, CM99, AP18], or on attesting that RSA prime factors were generated with a specified prime generator [BFGN17]. This line of work is only concerned with the structure of the keys, not with the fact that they are generated with enough entropy. Juels and Guajardo [JG02] suggested as early as in 2002 an approach for users to prove to another party (which is typically a trusted certificate authority or CA) that her public–secret key pair was generated honestly using proper randomness. In their setting, the CA provides an additional source of randomness in an interactive process, and the user algorithm proves that it has not weakened, whether intentionally or unintentionally, the key-generation procedure[1]. The security goal of such a primitive is threefold.

1. **Maintain User Privacy:** if the user uses a randomness source with high entropy, then an adversary (possibly the CA himself) has no additional information on the secret-key compared to a key generated

[1]This notion is very similar to the more recent cryptographic reverse firewalls [MS15].

by the real key-generation algorithm on uniform randomness.

2. **Improve Randomness Quality:** if the user *or* the CA use a randomness source with high entropy, then, an adversary (other than the CA) has no additional information on the secret-key compared to a key generated by the real key-generation algorithm on uniform randomness.

3. **Resist Information Exfiltration:** the generated public key leaks no information whatsoever to the outer world. In particular, a faulty user algorithm cannot use it to convey any information. In this sense, the CA certifies to the end user, that she can securely use the generated key.

A malicious user can obviously publish her secret key, but the problem we tackle is different: we want the CA to only certify keys that he knows to have been generated with high-entropy randomness and without covert channels.

Juels and Guajardo proposed a formal security model for *verifiable random key generation* with the goal to achieve these three security objectives. Their model is unfortunately not strong enough to capture real-world threats since

- it is restricted to public-key cryptosystems where a given public key corresponds to a *unique* secret key (and cannot be used for many recent schemes);

- it considers only a stand-alone or independent key-generation instances (and therefore does not prevent attacks such as the one considered in [LHA+12, HDWH12] where several public-keys are generated with correlated randomness sources);

- it only bounds the distance that a dishonest user algorithm can generate a given key to that of an honest algorithm executing the key generation protocol.

As a simple example, consider the problem of generating an ElGamal public key g^x in a group $\mathbb{G} = \langle g \rangle$ of prime order p. Juels and Guajardo outlined a protocol for generating such a key with verifiable randomness. The natural idea to generate a public-key g^x in this (illusorily) simple setting is to share the secret key x as $x = x_U + x_{CA} \bmod p$ where x_U denotes the user randomness and x_{CA} denotes the CA randomness. However, this protocol fails to achieve (3) as the user algorithm can choose x_U to match a specify value after seeing x_{CA}. To overcome this issue, a simple idea would be to make the user first commit to x_U and then prove its knowledge. However, the hiding and zero-knowledge properties of commitment schemes and proof systems inherently rely on perfect randomness, which the user algorithm is assumed not to have at its disposal.

Juels and Guajardo also proposed a protocol for the generation of RSA keys where the distance in (3) increases by a factor which is polynomial in the security parameter λ (assuming some number-theoretic conjecture). Therefore, their protocol does not rule out the existence of covert channels with $O(\log \lambda)$ bit capacity. Their model was reconsidered by Corrigan-Gibbs, Mu, Boneh and Ford [CMBF13] in a weaker setting that guarantees (1) and (2) but not (3), and does not even prevent a malicious user algorithm from generating malformed keys.

8.3 Results

8.3.1 Diophantine Satisfiability

Chapter 9 provides the first succinct argument for the satisfiability of Diophantine equations with a communication complexity and a round complexity that grows logarithmically in the size of the polynomial equation. It is statistical honest-verifier zero-knowledge and is extractable under standard computational assumptions over hidden-order groups such as RSA groups or ideal-class groups.

Integer Commitments. Section 9.2 introduces a new computationally hiding and binding commitment scheme that allows to commit to vectors of integers. It is close to Damgård and Fujisaki's seminal proposal, but has much smaller parameters. Denoting by λ the security parameter and letting $2^{b_{\mathbb{G}}}$ be an upper bound on the group order, the version of our scheme which allows to commit to n integers at once has parameters consisting of $O(b_{\mathbb{G}} + \log n)$ bits instead of $\Omega\left(nb_{\mathbb{G}} \cdot \mathrm{polylog}(\lambda)\right)$ as with the generalized version of Damgård and Fujisaki's scheme.

Damgård and Fujisaki's commitment scheme, for $n = 1$, is a variant of Pedersen's commitment in a hidden-order group $\mathbb{G}$: given two group elements $g, h \in \mathbb{G}$, the commitment to an integer value $x \in \mathbb{Z}$ is $C = g^x h^r$, where r is an integer of appropriate size. The hiding property of their scheme crucially relies on the fact that $g \in \langle h \rangle$, which is not always guaranteed as the group may not be cyclic. Damgård and Fujisaki's proposed a Schnorr-type [Sch91] protocol to prove such statements, but their challenge set is restricted to $\{0, 1\}$ to guarantee soundness under the assumptions on the group. Their protocol must then be repeated logarithmically many times to achieve negligible soundness, and the resulting parameters are large. The situation is worse when n is large as commitments are computed as $g_1^{x_1} \cdots g_n^{x_n} h^r$ and a proof for each g_i must be computed.

The scheme in Section 9.2 is based on the observation that proving that $g^2 \in \langle h^2 \rangle$ can be done more efficiently in a single protocol run under the assumptions on the group. Our commitments are thus computed as

$(g^x h^r)^2 \in \mathbb{G}$. We further such how to aggregate the proofs of several such statements to reduce the size of our parameters when n is large.

Succinct Inner-Product Arguments on Integers. Section 9.3 presents a succinct argument that two integer vectors committed with our scheme satisfy an inner-product relation. That is, an argument of knowledge of vectors $\mathbf{a}$ and $\mathbf{b} \in \mathbb{Z}^n$ (and of a randomness $r \in \mathbb{Z}$) that open a commitment C and such that $\langle \mathbf{a}, \mathbf{b} \rangle = z$ given a public integer z. Succinct here means that the communication complexity of the prover is of order $O(\ell + \log(n) b_{\mathbb{G}})$, where ℓ is the bit length of the largest witness (see the corresponding section for a discussion on the verification time). The complexity is measured in bits as during the protocol, the prover sends logarithmically many group elements and three integers, but these latter could be arbitrarily large.

The argument of Bünz et al. [BBB+18] for inner-product relations over $\mathbb{Z}_p$ is not applicable to integers as their proof of extractability relies on the generalized discrete-logarithm assumption for which there is no equivalent in hidden-order groups that may not even be cyclic, and on the invertibility of elements in $\mathbb{Z}_p^*$ since it requires to solve linear systems over $\mathbb{Z}_p$. Besides, their argument is not zero-knowledge and is on vectors committed with the non-hiding version of Pedersen's scheme (i.e., with nil randomness). Therefore, whenever it is used as a sub-protocol in another one, techniques specific to the larger protocol must always be used to guarantee that it is zero-knowledge. del Pino, Seiler and Lyubashevsky [dLS19] later solved this issue by adapting the argument of Bünz et al. in prime-order groups to make it perfectly honest-verifier zero-knowledge with the full-fledged Pedersen's scheme.

The protocol in Section 9.3 uses halve-then-recurse techniques similar to those of Bünz et al. for the Section-9.2.2 commitment scheme in hidden-order groups and thus allows to succinctly argue on integers, but only uses the integrality of $\mathbb{Z}$ as a ring since one cannot invert modulo the unknown order. (Note that these techniques are themselves inspired by the recursive inner-product argument of Bootle et al. [BCC+16].) In particular, we prove that even though one cannot a priori solve in $\mathbb{Z}$ the linear system of Bünz et al. required to prove the extractability of their protocol, one can instead solve a "relaxed" system in $\mathbb{Z}$. Then, under the assumptions on the hidden-order group, we show that the solution to the relaxed system is enough to extract a representation of the commitment in the public bases. In groups with public prime orders, the assumption that discrete-logarithm relations are hard to compute allows to conclude that this representation of the commitment actually leads to a valid witness, but this assumption is not a priori translatable to hidden-order groups. Instead, we prove that a similar assumption in the subgroup generated by a randomly sampled element is weaker than the assumptions on the group, and that suffices to prove the

extractability of the protocol. The details of these technical challenges are outlined in Section 9.3.1.

Furthermore, as the group order is unknown to all parties, the argument is only statistically honest-verifier zero-knowledge. To ensure this property, the randomness range of the prover is carefully adapted to allow for simulatability without knowledge of a witness.

Succinct Arguments for Multi-Integer Commitments. Section 9.4 then gives several succinct protocols related to multi-integer commitments. These protocols are important building blocks in our Diophantine satisfiability argument system but may also find applications in other settings.

We show how to succinctly argue knowledge of an opening $(a_1, \ldots, a_n, r)$ to a commitment. The protocol has bit communication complexity $O(\ell + \log(n)b_{\mathbb{G}})$. This argument system is based on the same halve-then-recurse techniques as in Section 9.3. We also propose a protocol that allows to aggregate arguments of knowledge of openings to m such commitments for any $m \geq 2$. With the same notation, the bit communication complexity of this aggregated protocol is only $O(\ell + \log(n)b_{\mathbb{G}} + \log(m))$ (i.e., the number of group elements does not increase with m). It is worth mentioning that the techniques used in it can be applied to additively-homomorphic commitment schemes in public-order groups.

We also show how to obtain short parameters for our new integer-vector commitment scheme and the inner-product argument with communication complexity $O(\ell + \log(n)b_{\mathbb{G}} + \log(m))$ bits, i.e., arguments that group elements $g_1, \ldots, g_m, h$ for $m \geq 2$ satisfy $g_i^2 \in \langle h^2 \rangle$ for $i \in \{1, \ldots, m\}$ with communication complexity $O(b_{\mathbb{G}} + \log m)$ bits instead of $O(mb_{\mathbb{G}})$ bits obtained by repeating m times the protocol for one group element. Finally, we show how to succinctly argue knowledge of the same vector of integers in $\mathbb{Z}^n$ committed with our schemes using m different bases for any $m \geq 2$ (and different randomness).

Succinct Arguments for Diophantine Equations. Section 9.5 presents our succinct protocol to argue satisfiability of Diophantine equations. The approach therein is inspired by Skolem's method [Sko50] which consists in reducing the degree of the polynomial by introducing new variables to obtain a new polynomial of degree at most 4, in such a way that the satisfiability of one polynomial implies that of the other. Tailoring Skolem's method to the problem of arguing satisfiability, we show how to reduce the satisfiability of any polynomial in $\mathbb{Z}[x_1, \ldots, x_\nu]$ of total degree δ with μ monomials to the existence of vectors $\mathbf{a}_L = \begin{bmatrix} a_{L,1} & \cdots & a_{L,n} \end{bmatrix}$, $\mathbf{a}_R = \begin{bmatrix} a_{R,1} & \cdots & a_{R,n} \end{bmatrix}$ and $\mathbf{a}_O = \begin{bmatrix} a_{O,1} & \cdots & a_{O,n} \end{bmatrix}$ in $\mathbb{Z}^n$, for $n \leq \nu\lfloor \log \delta \rfloor + (\delta - 1)\mu$, such that $a_{O,i} = a_{L,i}a_{R,i}$ for all $i \in \{1, \ldots, n\}$, and that satisfy $1 \leq Q \leq 1 + 2\nu(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu$

linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q,$$

where $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q} \in \mathbb{Z}^n$ and $c_q \in \mathbb{Z}$ for all $q \in \{1, \ldots, Q\}$. The reduction is constructive as it allows to infer the vectors and the constraints directly from the original polynomial.

Bootle et al. [BCC+16] then Bünz et al. [BBB+18] gave an argument system for proving knowledge of vectors in $\mathbb{Z}_p$ (instead of $\mathbb{Z}$) that satisfy such constraints. They use this protocol to argue for the satisfiability of arithmetic circuits over $\mathbb{Z}_p$. Our argument shares similarities with theirs, but again there are key technical differences that arise from the fact that $\mathbb{Z}$ is not a field. Indeed, as one cannot invert nor reduce integers modulo the unknown orders of the bases, we use different techniques notably to prevent the integers involved in the argument from increasing too much, and to ensure consistency between the variables in the entry-wise product and those in the linear constraints. Guaranteeing this latter consistency requires to construct new polynomials for the argument that do not involve inverting integers. Besides, one cannot use their commitment-key switching technique which consists in interpreting g^a as a commitment to xa to the base $g^{x^{-1}}$ in groups of public prime order. Finally, extra precaution must be taken to guarantee the zero-knowledge property as integers are not reduced modulo p and may carry information about the witness. These challenges and the ways we overcome them are described in details in Section 9.5.2.

As a result, the communication complexity of the Diophantine-satisfiability argument in Section 9.5 has a communication-complexity of

$$O\left(\delta\ell + \min(\nu, \delta) \log\left(\nu + \delta\right) b_{\mathbb{G}} + H\right)$$

bits, if the absolute value of all the polynomial coefficients is upper-bounded by 2^H for some integer H. In contrast, the overall communication complexity using Damgård and Fujisaki's multiplication argument is upper-bounded by $O\left(\binom{\nu+\delta}{\delta}\left(\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right) H + b_{\mathbb{G}}\right)\right)$ and lower-bounded by $\Omega\left(\binom{\nu+\delta}{\delta}(\ell + b_{\mathbb{G}})\right)$.

Applications. Lastly, Section 9.6 resents several applications of the Diophantine-satisfiability argument in Section 9.5. It provides explicit reductions to Diophantine satisfiability for the following problems:

- argument of knowledge of a (possibly committed) RSA e-th root in $\mathbb{Z}_N$ of some public value with in $O\left(\log(\log(e))b_{\mathbb{G}}\right)$ bits. This has application to credential systems when combined with proofs of non-algebraic statements [CGM16];
- argument of knowledge of $O\left(\log(\log p)b_{\mathbb{G}}\right)$ bits for ECDSA signatures with a prime p, and of $O\left(\log(\log q)b_{\mathbb{G}} + \log(\log p)\right)$ bits for DSA signatures with primes p and q. The signed message is public, but can be

committed if the argument is combined with proofs of non-algebraic statements [CGM16];

- argument that two committed lists of integers of length n are permutations of each other with $O(\ell + \log(n) b_{\mathbb{G}})$ bits;
- argument of satisfiability of a 3-SAT Boolean formula with m clauses and n variables with $O(\log(n+m) b_{\mathbb{G}})$ bits;
- argument of satisfiability of an Integer-Linear-Programming problem of the form $\mathbf{x} \in \mathbb{N}^n$ and $\mathbf{A}\mathbf{x}^{\mathrm{T}} \geq \mathbf{b}^{\mathrm{T}}$, for $\mathbf{A} \in \mathbb{Z}^{m \times n}$ and $\mathbf{b} \in \mathbb{Z}^m$, with $O(\ell + \log(4n+3m)\ b_{\mathbb{G}} + \log \|\mathbf{A}\|_\infty + \log \|\mathbf{b}\|_\infty)$ bits.

8.3.2 Public-Key Generation with Verifiable Randomness

Chapter 10 revisits the verifiable key-generation primitive and presents the first strong security models and efficient, provably secure constructions.

Game-Based Security Model. Section 10.2 presents a game-based model that covers concurrent protocol executions with different instances of protocol algorithms. It is inspired by the Bellare-Pointcheval-Rogaway (BPR) model for authenticated key exchange [BPR00]. The communication between the user and the CA is assumed to be carried over an insecure channel. Messages can be tapped and modified by an adversary, and the communication between the user and the CA is asynchronous. The adversary is split into two algorithms: (1) the *sampler* which provides the randomness sources to the user and the CA (for multiple instances of the protocol) and (2) the *distinguisher* which tries to gain information from the generated public key. The protocol is deemed secure if the distinguisher is unable to do so assuming that the entropy of either random source is high enough.

The main difficulty in defining the security model for this primitive is to formalize the third security goal. A dishonest user algorithm can indeed always execute several instances of the protocol with the CA until she obtains a public-key which has some specific property which allows to exfiltrate information. This is similar to the "halting attack" subliminal channel [Des96] and cannot be avoided, at least without adding a third party to the model as in access-control encryption [DHO16, IPV10]. This *narrow-band* subliminal channel is taken into consideration in the security model while capturing the fact that in a secure protocol, this should be the only possible covert channel for a dishonest user algorithm. In practical applications, this covert channel can be prevented easily if the CA charges an important fee for a user that performs too many key generation procedures, or if an increasing time-gating mechanism for repeating queries is introduced.

This model does not suffer from the shortcomings of the model proposed from [JG02] as it allows for multiple dependent runs of the protocol and

captures the resistance to exfiltration of information (with only the narrow-band subliminal channel from the "halting attack"). It guarantees security with concurrent sessions and is thus much stronger than security consdered in cryptographic reverse firewalls [MS15].

Generic Protocol for Probabilistic Circuits. Section 10.3.1 then presents a generic approach for key generation based on (families of) probabilistic circuits and we prove its security in the security model of Section 10.2. It relies on two-source randomness extractors, pseudo-random-function families and extractable commitments with associated zero-knowledge proofs. Since two-party computation (2PC) protocols rely on perfect randomness, a generic 2PC protocol cannot be used in this setting; moreover, such a protocol guarantees privacy and correctness, but it does not guarantee that a user cannot influence the result (and thus requirement (3)).

Efficient Protocol for RSA Keys. Section 10.3.4 also proposes a new generic protocol for factoring-based cryptography and proves it secure in the same model. It relies on classical cryptographic tools (namely commitments, pseudo-random functions (PRFs) and zero-knowledge proofs). Section 10.4 provides an instantiation based on the Dodis–Yampolskiy PRF [DY05] in the group of quadratic residue modulo a safe prime which outputs group elements. The main technical difficulty is to convert the outputs of this PRF into integers while proving that the RSA prime factors are outputs of the PRF. In the process, Section 10.4 proposes a new efficient zero-knowledge proof system for the so-called *double discrete logarithm problem* (in pubic-order groups). A double discrete logarithm of an element $y \neq 1_{\mathbb{G}}$ in a cyclic group $\mathbb{G}$ of prime order p with respect to bases $g \in \mathbb{G}$ and $h \in \mathbb{Z}_p^*$ (generators of $\mathbb{G}$ and $\mathbb{Z}_p^*$ respectively) is an integer $x \in \{0, \ldots, p-1\}$ such that $y = g^{h^x}$. Stadler introduced this computational problem for verifiable secret-sharing [Sta96] and it was used to design numerous cryptographic protocols (e.g., group signatures [CS97], blind signatures [ASM10], e-cash systems [CG07] and credential systems [CGM16]). All these constructions rely on a proof system proposed by Stadler which has $\Omega(\log p)$ computational and communication complexity (in terms of group elements). Our new proof system outputs proofs with only $O(\log \log p)$ group elements and permits an efficient instantiation of our generic protocol for factoring-based cryptography. As a by-product, the new protocol can be used directly in all the aforementioned applications in a prime, public-order setting to exponentially reduce their communication complexity. It relies on techniques similar to those presented in Section 9.5.1.

Chapter 9

Succinct Diophantine-Satisfiability Arguments

Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk.

— Leopold Kronecker

This chapter presents the first succinct honest-verifier zero-knowledge argument for the satisfiability of Diophantine equations with a communication complexity and a round complexity that grows logarithmically in the size of the polynomial equation. The security of the argument relies on standard assumptions on generators of hidden-order groups. As the argument requires to commit to integers, it introduces a new integer-commitment scheme (in the random-oracle model) that has much smaller parameters than Damgård and Fujisaki's scheme. The last section of the chapter shows how to succinctly argue knowledge of solutions to several NP-complete problems and cryptographic problems by encoding them as Diophantine equations.

Contents

9.1 Preliminaries

This section introduces introduces preliminary material to this chapter.

9.1.1 Non-interactive Commitments in the Random-Oracle Model

This section defines, in exact-security terms, commitment schemes in the random-oracle model, i.e., in a model in which the scheme algorithms (and the adversary) are given access to a random oracle. The reason is that the algorithms may have to check non-interactive proofs computed with a random oracle before carrying on with their computation. Therefore, the number of random-oracle queries made by an adversary can affect the security of the scheme.

Formally, a (non-interactive) commitment scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^{\lambda}\right) \to pp$: generates public parameters on the input of a security parameter 1^{λ}. These parameters are implicit inputs to the other algorithms.

$\mathsf{KG}\,(pp) \to ck$: computes a commitment key on the input of public parameters. The parameters and the commitment key further define a message space denoted $\mathcal{X}_{pp,ck}$.

$\mathsf{Com}\,(ck, x) \to (C, d)$: computes a commitment C to a value x and an opening or decommitment information d on the input of a commitment key ck. It is further assumed that if $x \notin \mathcal{X}_{pp,ck}$, then the algorithm returns $\perp$.

$\mathsf{ComVf}(ck, C, x, d) \to b \in \{0,1\}$**:** deterministically returns a bit indicating whether the decommitment d is valid (bit 1) for C and x w.r.t. key ck, or not (bit 0). It is assumed that if $C = \perp$ or if $x \notin \mathcal{X}_{pp,ck}$, then it returns 0.

A commitment scheme is *correct* if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$, for all $ck \leftarrow \mathsf{KG}\,(pp)$ and all $x \in \mathcal{X}_{pp,ck}$,

$$\Pr\left[\mathsf{ComVf}(ck, C, x, d) = 1 \colon (C, d) \leftarrow \mathsf{Com}(ck, x)\right] = 1.$$

Give a random oracle $\mathcal{H}$, a commitment scheme is $(T, q_{\mathcal{H}}, \varepsilon)$-*hiding* ($(q_{\mathcal{H}}, \varepsilon)$-statistically hiding) if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs it time at most $T(\lambda)$ (computationally unbounded, i.e., $T(\lambda) = \infty$) and makes at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$,

$$\left|\Pr\left[b = b' \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ (ck, x_0, x_1, st) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp) \\ b \leftarrow_{\$} \{0,1\} \\ (C, d) \leftarrow \mathsf{Com}(ck, x_b) \\ b' \leftarrow \mathcal{A}(st, C) \\ \text{for } e \in \{0,1\} \\ \quad \text{if } x_e \in \mathcal{X}_{pp,ck} \text{ and } x_{1-e} \notin \mathcal{X}_{pp,ck} \\ \quad\quad b' \leftarrow_{\$} \{0,1\} \end{array}\right] - 1/2\right| \leq \varepsilon(\lambda).$$

Note that in the definition, the adversary is the party computing the commitment key. One could consider a weaker variant of the definition with *trusted* key generation in which the key is necessarily honestly generated.

Give a random oracle $\mathcal{H}$, a commitment scheme is $(T, q_{\mathcal{H}}, \varepsilon)$-*binding* if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$ and makes at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$,

$$\Pr\left[\begin{array}{c} \mathsf{ComVf}(ck, C, x_i, d_i) = 1 \\ \wedge\, x_0 \neq x_1 \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ ck \leftarrow \mathsf{KG}\,(pp) \\ (C, (x_i, d_i)_{i=0,1}) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp, ck) \end{array}\right] \leq \varepsilon(\lambda).$$

Discussion. The syntax above separates the commitment-key generation algorithm from the setup algorithm, although these are often tacitly combined (as in Section 2.5.3), especially for commitments in public-order groups. The main reason is that doing so allows to define the hiding property for schemes even when the keys are possibly *invalid*. This question does not arise

for schemes with keys that are elements of a prime-order group $\mathbb{G} = \langle g \rangle$ (e.g., Pedersen's scheme [Ped92]) since any element $h \in \mathbb{G}^*$ is a valid commitment key. However, when the scheme is defined over an unknown-order group $\mathbb{G}$ which may not be cyclic, and that keys are elements of the *subgroup* generated by an element (as it is the case for Damgård–Fujisaki commitments recalled in Section 9.2.1), say h, there may not be an efficient way to test whether another element $g \in \mathbb{G}$ is in $\langle h \rangle$. Computing a commitment with an invalid key may then not guarantee that the commitment is hiding. That is why the definition of the hiding property allows the key to be adversarially generated so that if the definition is satisfied, commitments computed with a potentially invalid key do not reveal information about the committed values.

On the other hand, the definition of the binding property does not consider adversarially generated keys. To understand why, it might be helpful to rather think of an interactive commitment protocol between Alice and Bob. Bob generates a key and sends it to Alice, and Alice commits to a value that she later opens to Bob. It now becomes clear that Alice's committed value should remain hidden before she opens the commitment, and so even if she does not trust Bob's key. Yet, Bob, who needs to ensure that Alice does not later open to a value different from the committed one, is the party who computed the key and thus need not verify that it is valid. The situation is the same with Pedersen's scheme, as its binding property relies on the fact that the discrete-logarithm relation between g and h is unknown to the party who computes the commitment.

9.2 Integer Commitments

This section recalls a scheme due to Damgård and Fujisaki which allows to commit to integers[1]. Then comes a new integer-commitment scheme with parameters smaller than those of Damgård and Fujisaki's scheme, and which are also more efficient to compute. For the version of our scheme which allows to commit to n integers, the parameters are of $O(b_{\mathbb{G}} + \log n)$ bits instead of $\Omega(nb_{\mathbb{G}} \log P)$ as with the generalized version of Damgård and Fujisaki's scheme, where $2^{b_{\mathbb{G}}}$ is an upper bound on the group order.

9.2.1 Damgård–Fujisaki Commitments

The Damgård–Fujisaki commitment scheme [FO97, DF02], parameterized by a group generator G, consists of the following algorithms:

[1] Couteau, Peters and Pointcheval [CPP17] proved that in the case of RSA groups (with Blum integers), the security of Damgård and Fujisaki's scheme is provable under (a variant of) the RSA assumption instead of the strong RSA assumption. This also holds for our scheme. However, this result does not concern generic hidden-order groups.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$, generate $h \leftarrow_\$ \mathbb{G}$ and return $(\mathbb{G}, P, h)$. Recall that these parameters are implicit inputs to all the other algorithms.

$\mathsf{KG}(pp) \to ck$: generate $\alpha \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+\lambda}]\!]$ ($2^{b_\mathbb{G}}$ is an upper bound on $\mathrm{ord}(\mathbb{G})$), compute and return $g \leftarrow h^\alpha$.

$\mathsf{Com}(g, x \in \mathbb{Z}) \to (C, d)$: generate $r \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+\lambda}]\!]$, compute $C \leftarrow g^x h^r$, set $d \leftarrow (r, 1_\mathbb{G})$ and return (C, d).

$\mathsf{ComVf}\,(g, C, x, d) \to b \in \{0, 1\}$: parse d as $(r, \tilde{g})$. If $C = g^x h^r \tilde{g}$ and $\tilde{g}^2 = 1_\mathbb{G}$, return 1, else return 0.

Equivalently, the commitment-algorithm could simply set the decommitment information d to r, and the commitment-verification would return 1 if the equality $C^2 = \left(g^x h^d\right)^2$ holds and 0 otherwise. The squaring in the verification is due to the fact that the small-order assumption does not exclude the possibility to efficiently compute square roots of unity, and they thus relaxed the verification equation to allow for sound argument of knowledge of openings to commitments. In other words, the scheme would still be binding without the squaring in the verification equation, and the relaxation is simply an artifact to allow for sound arguments.

More precisely, suppose that the verification were not relaxed, i.e., that it would only check that $C = g^x h^d$. Two accepting transcripts (D, e_1, z_1, t_1) and (D, e_2, z_2, t_2) of a standard Schnorr-type argument of knowledge of an opening would imply that $C^{e_1-e_2} = g^{z_2-z_1} h^{t_2-t_1}$. Assuming $e_1, e_2 \in [\![0; P-1]\!]$, $e_1 \neq e_2$, and that $e_1 - e_2$ divides $z_2 - z_1$ and $t_2 - t_1$ (Damgård and Fujisaki showed that this latter event occurs with probability negligibly close to $1/2$ under the assumptions on the group generator), the previous equality would imply that $\left(g^{(z_2-z_1)/(e_1-e_2)} h^{(t_2-t_1)/(e_1-e_2)} C^{-1}\right)^{e_1-e_2} = 1_\mathbb{G}$, and the small-order assumption would only allow to conclude that $C^2 = \left(g^{(z_2-z_1)/(e_1-e_2)} h^{(t_2-t_1)/(e_1-e_2)}\right)^2$. The trivial attack in which an adversary computes C as $g^x h^d \tilde{g}$ with $\tilde{g} \in \mathbb{G}$ such that $\tilde{g}^2 = 1_\mathbb{G}$ would then not be excluded by the protocol.

Properties. Damgård and Fujisaki's scheme is correct, is computationally binding under the strong-root and the μ-assumption, and is statistically hiding. Note that hiding property crucially relies on the fact that $g \in \langle h \rangle$. To guarantee the statistical hiding property of the scheme *without trusted key generation*, the party which computes g is then also required to compute a non-interactive proof that $g \in \langle h \rangle$. The commitment algorithm would then verify the proof and proceed as above if it is valid, and otherwise return $\perp$. Damgård and Fujisaki proposed to compute such a proof with a

Schnorr-type protocol (made non-interactive via the Fiat–Shamir heuristic) with $\{0,1\}$ as challenge set; i.e., given $\alpha \in \mathbb{Z}$ such that $g = h^\alpha$, the prover generates $k \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+2\lambda}]\!]$ and sends $f \leftarrow h^k$ to the verifier, this later chooses and sends to the prover a challenge $c \leftarrow_\$ \{0,1\}$, the prover replies with $z \leftarrow k - c\alpha$ (computed in $\mathbb{Z}$), and the verifier accepts if and only if $h^z g^c = f$. To attain a soundness error of at most $1/P$, the proof must then be repeated at least $\lceil \log P \rceil$ times. With the Fiat–Shamir heuristic, each proof consists of (c, z), and the total proof in the public parameters then consists of $\lceil \log P \rceil (b_\mathbb{G} + 2\lambda + 2) = \Omega(b_\mathbb{G} \log P)$ bits (recall that P is super-polynomial in λ, e.g., $\lambda^{\log \lambda}$).

9.2.2 A new Integer-Commitment Scheme

This section introduces a novel integer-commitment scheme that is close to Damgård and Fujisaki's scheme, but with an argument (rather than a proof) of only $O(b_\mathbb{G})$ (with b such that $\mathrm{ord}(\mathbb{G}) \leq 2^{b_\mathbb{G}}$) bits in non-trusted keys, and the argument only requires a single protocol run to reach the same soundness error. As the soundness of the protocol relies on computational assumptions on the group generator, the scheme is only computationally hiding, whereas Damgård and Fujisaki's cut-and-choose protocol is perfectly sound (the prover is not assumed to be computationally bounded) but inefficient.

Formally, let G be a group generator and let $FS.\Pi^{\mathcal{H}}$ be a Fiat–Shamir non-interactive argument system with random oracle $\mathcal{H}$ for the language $\left\{g \in \mathbb{G}, \ell \in \mathbb{N}^* : \ \exists \alpha \in [\![0; 2^\ell]\!], g = h^\alpha\right\}$, given parameters $(\mathbb{G}, P, h, 1)$ (integer 1 is just to indicate that there is only one group element g in the word for which the proof is computed) and the empty string as CRS. The proof of the hiding property will require the protocol to satisfy culpable soundness w.r.t. the language $\sqrt{\langle h^2 \rangle}$. The scheme, parameterized by G and further denoted $\mathscr{C}$, consists of the following algorithms.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$, generate $h \leftarrow_\$ \mathbb{G}$ and return $(\mathbb{G}, P, h)$. Recall that these parameters are implicit inputs to all the other algorithms.

$\mathsf{KG}(pp) \to ck$: generate $\alpha \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+\lambda}]\!]$, compute $g \leftarrow h^\alpha$ and a proof $\pi \leftarrow FS.\Pi^{\mathcal{H}}.\mathsf{Prove}((\mathbb{G}, P, h, 1), (g, b_\mathbb{G} + \lambda), \alpha)$, and return (g, π).

$\mathsf{Com}\left((g, \pi), x \in \mathbb{Z}\right) \to (C, d)$: if $FS.\Pi^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, h, 1), (g, b_\mathbb{G} + \lambda), \pi) = 0$, then return $\perp$; else generate $r \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+\lambda}]\!]$, compute $C \leftarrow (g^x h^r)^2$, set $d \leftarrow r$ and return (C, d).

$\mathsf{ComVf}\left((g, \pi), C, x, d\right) \to b \in \{0,1\}$: if $C^2 = \left(g^x h^d\right)^4$ return 1, else return 0.

Comparison with Damgård–Fujisaki Commitments. As for Damgård and Fujisaki's commitments, the squaring in the verification equation (compared to the computation of commitments) is again to later allow for sound arguments of knowledge of openings. The main difference compared to Damgård and Fujisaki's commitments is that commitments are computed as $(g^x h^r)^2$ instead of $g^x h^r$. It is simply due to the fact that π only guarantees that $g^2 \in \langle h^2 \rangle$, not that $g \in \langle h \rangle$, hence the power 2 in the computation of commitments to ascertain that they are hiding. However, only requiring that $g^2 \in \langle h^2 \rangle$ instead of $g \in \langle h \rangle$ is precisely what allows to have much smaller arguments that can be computed in a single protocol run.

Correctness & Security.

We now prove the correctness and the security of the commitment scheme $\mathscr{C}$.

Theorem 9.2.3. *$\mathscr{C}$ is correct.*

Proof. The correctness of $\mathscr{C}$ immediately follows from its definition. □

Theorem 9.2.4. *Assuming $FS.\Pi^{\mathcal{H}}$ to be $\left(T, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}, \sqrt{\langle h^2 \rangle}\right)$-sound, scheme $\mathscr{C}$ is $\left(T, q_{\mathcal{H}}, 2^{-\lambda+1} + \varepsilon^{\mathrm{snd}}\right)$-hiding.*

Proof. Unless the adversary can contradict the culpable soundness of $FS.\Pi^{\mathcal{H}}$, the key (g, π) it returns is such that $g^2 = h^{2\alpha}$ for some $\alpha \in \mathbb{Z}$. As the distribution of $(g^x h^r)^4 = h^{4(\alpha x + r)}$ for $r \leftarrow_{\$} [\![0; 2^{b_{\mathbb{G}}+\lambda}]\!]$ is the same as the distribution of h^{4z} for $z \leftarrow_{\$} [\![\alpha x \bmod \mathrm{ord}\,(h^4)\,; \alpha x \bmod \mathrm{ord}\,(h^4) + 2^{b_{\mathbb{G}}+\lambda}]\!]$, which is at a statistical distance of at most $2^{-\lambda+1}$ from the distribution of h^{4z} for $z \leftarrow_{\$} [\![0; \mathrm{ord}\,(h^4) - 1]\!]$, the claim follows. □

Theorem 9.2.5. *For any $T \colon \mathbb{N}^* \to \mathbb{N}^*$, scheme $\mathscr{C}$ is $\left(T, q_{\mathcal{H}}, \varepsilon^{\mathrm{strg}} + \varepsilon^{\mathrm{zk}} + 2^{-\lambda}\right)$-binding if $FS.\Pi^{\mathcal{H}}$ is $\left(T, T_{\mathsf{Sim}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{zk}}\right)$-zero-knowledge and if the $(T_{\mathsf{Sim}} + O(T + b_{\mathbb{G}}), \varepsilon^{\mathrm{strg}})$-strong-root assumption holds on G.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ running in time T and contradicts the binding property of the scheme with probability at least ε. Consider then an algorithm $\mathcal{B}$ which runs a trapdoor setup algorithm which is exactly as $\mathsf{Setup}\left(1^{\lambda}\right)$ (in particular, it computes h^{α} for $\alpha \leftarrow_{\$} [\![0; 2^{b_{\mathbb{G}}+\lambda}]\!]$, which requires at most $2(b_{\mathbb{G}} + \lambda + 1)$ group operations with the square-and-multiply algorithm), except that it simulates an argument of knowledge of α with the simulator of $FS.\Pi^{\mathcal{H}}$. The runtime of this trapdoor setup algorithm is then of order $T_{\mathsf{Sim}} + O(b_{\mathbb{G}} + \lambda) = T_{\mathsf{Sim}} + O(b_{\mathbb{G}})$ (since $b_{\mathbb{G}} = \Omega(\lambda)$ by assumption). Adversary $\mathcal{A}$ can then distinguish $\mathcal{B}$ from the challenger of the binding game with an advantage of at most $\varepsilon^{\mathrm{zk}}$, so $\mathcal{A}$ contradicts the binding property of the scheme with probability at least $\varepsilon - \varepsilon^{\mathrm{zk}}$ on the

input of parameters returned by algorithm $\mathcal{B}$. Given two pairs of integers $(x,d),(x',d')$ and a group element C such that $x \neq x'$ and $C^2 = \left(g^x h^d\right)^4 = \left(g^{x'} h^{d'}\right)^4$, the equality $h^{4(\alpha(x-x')+d-d')} = 1_{\mathbb{G}}$ holds. Let $0 \leq \rho < \operatorname{ord}(h)$ denote the unique integer such that $\alpha = \operatorname{ord}(h)\lfloor \alpha / \operatorname{ord}(h) \rfloor + \rho$. Note that the distribution of (x,d) and (x',d') only depends on ρ as $\mathcal{A}$ is only given g (and not α) as input. Since $\lfloor \alpha / \operatorname{ord}(h) \rfloor$ is uniformly distributed over a set of size at least 2^λ, the probability that $\alpha(x-x')+d-d' = 0$ is at most $2^{-\lambda}$. Lemma 9.2.6 then shows that denoting by $T_{\mathcal{B}}$ the running time of $\mathcal{B}$ and setting $n \coloneqq 4\left(\alpha(x-x')+d-d'\right)$, the inequality $\varepsilon - \varepsilon^{\mathrm{zk}} - 2^{-\lambda} \leq \varepsilon^{\mathrm{strg}}$ holds under the $(T_{\mathcal{B}} + O(\log n), \varepsilon^{\mathrm{strg}})$-strong-root assumption. Remark that the integers returned by $\mathcal{A}$ are necessarily less than $2^{O(T)}$ as an algorithm running in time T can return a value of at most $O(T)$ bits (the algorithm might be using an alphabet different from $\{0,1\}$, hence the big O). Therefore, $\log n = O(T + b_{\mathbb{G}})$. Since $T_{\mathcal{B}} = T + T_{\mathsf{Sim}} + O(b_{\mathbb{G}})$, the claim follows. □

Lemma 9.2.6. *Consider the problem (depending on λ) of computing a value $n \in \mathbb{Z}^*$ such that $h^n = 1$ on the input of $(\mathbb{G},P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and of a group element $h \leftarrow_{\$} \mathbb{G}$. If there exists an algorithm $\mathcal{A}$ that solves this problem in time T with probability at least ε, then there exists an algorithm that solves the strong-root problem in time at most $T + O(\log n)$ with probability at least ε.*

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the lemma. Consider an algorithm $\mathcal{B}$ which, on the input of $(\mathbb{G},P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and of a group element $h \leftarrow_{\$} \mathbb{G}$, runs $\mathcal{A}$ as a subroutine on the input of $(\mathbb{G},P)$ and h. If $\mathcal{A}$ returns an integer $n \in \mathbb{Z}^*$ such that $h^n = 1$ note that $h^{|n|+1} = h$. Algorithm $\mathcal{B}$ then computes $|n|+1$ (which can be done in time at most $O(\log n)$) and returns $(h, |n|+1)$. □

Argument System $FS.\Pi^{\mathcal{H}}$.

It only remains to provide a protocol $FS.\Pi^{\mathcal{H}}$ to argue knowledge of an integer $\alpha \in \mathbb{Z}$ such that $g^2 = h^{2\alpha}$, which is sufficient for the commitment scheme to be computationally hiding. We first give an interactive protocol Π for the language $\left\{ g \in \mathbb{G}, \ell \in \mathbb{N}^* \colon \exists \alpha \in [\![0; 2^\ell]\!], g = h^\alpha \right\}$ given parameters $(\mathbb{G},P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and that satisfies culpable soundness w.r.t. $\sqrt{\langle h^2 \rangle}$, and then apply the Fiat–Shamir heuristic to obtain $FS.\Pi^{\mathcal{H}}$.

In more detail, the (interactive) protocol Π is as follows: the prover generates $k \leftarrow_{\$} [\![0; 2^{\ell+\lambda}P]\!]$, computes $t \leftarrow h^k$ and sends t to the verifier; the verifier chooses $c \leftarrow_{\$} [\![0; P-1]\!]$ and sends it to the prover; the prover then replies with $r \leftarrow k - c\alpha$, and the verifier accepts if and only if $h^r g^c = t$. With the Fiat–Shamir heuristic, the proof consists of (c,r), i.e., $2\lfloor \log P \rfloor + \ell + \lambda + 3$

bits. For $\ell = b_{\mathbb{G}} + \lambda$, that is $2\lfloor \log P \rfloor + b_{\mathbb{G}} + 2\lambda + 3 = O(b_{\mathbb{G}})$ bits (recall that $P \leq 2^{b_{\mathbb{G}}}$ and $b_{\mathbb{G}} = \Omega(\lambda)$).

Properties. We now show that Π is complete, statistically honest-verifier zero-knowledge and satisfies culpable extractability w.r.t. $\sqrt{\langle h^2 \rangle}$ under the assumptions introduced in Section 2.4.7.

Theorem 9.2.7. Π *is complete.*

Proof. It immediately follows from the definition of Π. □

Theorem 9.2.8. Π *is* $\left(O\left(\left(\ell + \lambda + \log P\right) T_{\mathbb{G}}\right), 2^{-\lambda+1}\right)$*-statistically honest-verifier zero-knowledge.*

Proof. To simulate such a proof conditioned on a challenge c, it suffices to generate $k \leftarrow_{\$} [\![0; 2^{\ell+\lambda} P]\!]$, compute $t \leftarrow h^k g^c$ and set $r \leftarrow k$. As the distribution of $\left(h^{k+c\alpha}, c, k\right)$ for $k \leftarrow_{\$} [\![0; 2^{\ell+\lambda} P]\!]$ is at a $2^{-\lambda+1}$ statistical distance from the distribution of $\left(h^k, c, k - c\alpha\right)$ for $k \leftarrow_{\$} [\![0; 2^{\ell+\lambda} P]\!]$, the protocol is honest-verifier $2^{-\lambda+1}$-statistically zero-knowledge. The simulator simply generates $k \leftarrow_{\$} [\![0; 2^{\ell+\lambda} P]\!]$ and $c \leftarrow_{\$} [\![0; P-1]\!]$, computes $h^k g^c$ and returns $\left(h^k g^c, c, k\right)$. As $h^k g^c$ can be computed in $O(\ell + \lambda + \log P)$ group operations with classical sliding-window algorithms [Ava05], the simulator runs in time $O\left(\left(\ell + \lambda + \log P\right) T_{\mathbb{G}}\right)$. □

Theorem 9.2.9. Π *is* $\left(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu, \sqrt{\langle h^2 \rangle}\right)$*-extractable with* $T_{\mathcal{E}} \coloneqq \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O(T_{\mathsf{Prove}^*} \log P)$*, and* $T_{\mathcal{A}}$ *and* T_{Prove^*} *such that* $T_{\mathcal{A}} + \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O\left(T_{\mathsf{Prove}^*} T_{\mathbb{G}} \log P + \log^2 P\right) \leq \min(T^{\mathrm{strg}}, T^{\mathrm{ord}})$.

Proof. To prove that the protocol satisfies culpable extractability (and thus culpable soundness), notice that given two valid transcripts (t, c, r) and (t, c', r') such that $c \neq c'$, the equality $h^{r-r'} = g^{c'-c}$ holds. Assume without loss of generality that $c' > c$, and let $d \coloneqq \gcd(r - r', c' - c)$. The extended Euclidean algorithm applied to $r - r'$ and $c' - c$ returns, in time $O\left(\log(r - r') \log(c' - c)\right)$, integers u and v such that $d = u(r - r') + v(c' - c)$ and $|u|, |v| \leq \max\left(|r - r'|, |c' - c|\right)/d$.

- If $d = c' - c$, then $\left(h^{(r-r')/(c'-c)} g^{-1}\right)^{c'-c} = 1$ since $h^{r-r'} = g^{c'-c}$. The fact that $0 < c' - c < P$ and the small-order assumption therefore implies that $h^{2(r-r')/(c'-c)} = g^2$, i.e., $(r-r')/(c'-c)$ is a valid (culpable) witness.
- If $d < c' - c$, since the equality $h^{r-r'} = g^{c'-c}$ implies that $h^d = \left(g^u h^v\right)^{(c'-c)}$, $\left(\left(g^u h^v\right)^{(c'-c)/d} h^{-1}\right)^d = 1$ although $d < c' - c < P$. The

small-order assumption thus implies that $\tilde{g}^2 = 1$ for $\tilde{g} \coloneqq (g^u h^v)^{(c'-c)/d} h^{-1}$. If $\tilde{g} = 1_{\mathbb{G}}$, then $(g^u h^v, (c'-c)/d)$ is a solution to the strong-root problem. If, $\tilde{g} \neq 1_{\mathbb{G}}$, further distinguish two sub-cases:

- if $(c'-c)/d$ is odd, then $\tilde{g}^{(c'-c)/d} = \tilde{g}$ and thus $h = (g^u h^v \tilde{g})^{(c'-c)/d}$, i.e., $(g^u h^v \tilde{g}, (c'-c)/d)$ is a solution to the strong-root problem
- if $(c'-c)/d$ is even, then the low-dyadic-valuation assumption implies that $\operatorname{ord}\left((g^u h^v)^{(c'-c)/d}\right)$ is odd, which is impossible if $\operatorname{ord}(h)$ is P-rough (and necessarily odd) as $\operatorname{ord}(\tilde{g}h) = 2\operatorname{ord}(h)$ in this case.

Note that since the absolute value of integers returned by the algorithm Prove^* of a prover $(\mathcal{A}, \mathsf{Prove}^*)$ are necessarily less than $2^{O(T_{\mathsf{Prove}^*})}$, computing the witness $(r-r')/(c'-c)$ in case $d = c'-c$ can be done in time $O(T_{\mathsf{Prove}^*} \log P)$. Besides, in any of the cases in which there is a reduction to the strong-root problem, computing $(c'-c)/d$ can be done in time $O\left(\log^2 P\right)$ as $d \leq c'-c < P$, and computing $g^u h^v \tilde{g}$ can be done in $O\left(\max\left(T_{\mathsf{Prove}^*}, \log P\right)\right)$ group operations using the square-and-multiply algorithm, i.e., in time $O\left(\max\left(T_{\mathsf{Prove}^*}, \log P\right) T_{\mathbb{G}}\right)$ (recall that $T_{\mathbb{G}}$ is the bit complexity of performing group operations), after u and v have been computed in time $O\left(T_{\mathsf{Prove}^*} \log P\right)$ with the extended Euclidean algorithm. The solution to the strong-root problem can thus be computed in time $O\left(T_{\mathsf{Prove}^*} T_{\mathbb{G}} \log P + \log^2 P\right)$ given two accepting transcripts.

To obtain two valid transcripts with distinct challenges, it suffices to run the proving algorithm once, rewind it to the computation step right after it sent its first message and run it anew on fresh verifier randomness. If both runs are valid and the challenges are distinct, then return the two transcripts, otherwise restart. The expected runtime of this procedure is at most $\left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*}$ if ε denotes the success probability of the prover.

Define then an extractor $\mathcal{E}$ as an algorithm that runs the previous procedure and returns the witness $(r-r')/(c'-c)$ in case $c'-c \mid r-r'$, and otherwise returns $\perp$. The running time of $\mathcal{E}$ is then $\left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O(T_{\mathsf{Prove}^*} \log P)$ in expectation.

Moreover, for T^{strg} and T^{ord} such that $T_{\mathcal{A}} + \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O\left(T_{\mathsf{Prove}^*} T_{\mathbb{G}} \log P + \log^2 P\right) \leq \min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)$, the previous case analysis shows that a valid witness cannot be extracted with probability at $\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu$ under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G. The theorem thus follows. □

Arguing Knowledge of Openings.

As for Damgård and Fujisaki's commitments, one can efficiently argue knowledge of openings, i.e., of integers x and r such that a given commitment C satisfies $C^2 = (g^x h^r)^4$.

The protocol imposes an upper bound of ℓ on the bit length of the witness, with ℓ being part of the (public) word. It is simply to adapt the randomness range of the prover (and of the honest-verifier zero-knowledge simulator) to ensure that the protocol remains statistically honest-verifier zero-knowledge; and ℓ can be arbitrarily large. The protocol does *not* guarantee that the largest absolute value in the extracted witness is at most ℓ bits long [2]. In technical terms, the protocol is for the relation $\left\{\left(C \in \mathbb{G}, \ell \in \mathbb{N}^*; x, r \in [\![0; 2^\ell]\!]\right) : C^2 = (g^x h^r)^4\right\}$ that satisfies culpable extractability for the relation $\Sigma \coloneqq \left\{(C \in \mathbb{G}, \ell \in \mathbb{N}^*; x, r \in \mathbb{Z}) \colon C^2 = (g^x h^r)^4\right\}$.

More precisely, consider the problem of arguing in zero-knowledge knowledge of integers x and r such that $C^2 = (g^x h^r)^4$ and $|x|, |r| \leq 2^\ell$, for a group element C chosen by the prover and public bases h and g, and a public proof π that $g \in \sqrt{\langle h^2 \rangle}$. The prover first verifies π and aborts if it is invalid. The prover generates $y, s \leftarrow_\$ [\![0; P2^{\ell+\lambda}]\!]$, computes and sends $D \leftarrow (g^y h^s)^2$ to the verifier. The verifier then chooses $e \leftarrow_\$ [\![0; P-1]\!]$, sends it to the prover, and this latter replies with $z \leftarrow y - ex$ and $t \leftarrow s - er$ (computed in $\mathbb{Z}$). The verifier then accepts if and only if $(g^z h^t)^2 C^e = D$. Further denote the protocol by $\Pi_{\mathscr{C}}$.

Theorem 9.2.10. *$\Pi_{\mathscr{C}}$ is complete.*

Proof. This fact immediately follows from the definition of $\Pi_{\mathscr{C}}$. □

Theorem 9.2.11. *$\Pi_{\mathscr{C}}$ is $\left(O\left((\ell + \lambda + \log P)\, T_{\mathbb{G}}\right), q_{\mathcal{H}}, 2^{-\lambda+1} + \varepsilon^{\mathrm{snd}}\right)$-statistically honest-verifier zero-knowledge if $\Pi^{\mathcal{H}}$ is $\left(T, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}, \sqrt{\langle h^2 \rangle}\right)$-sound.*

Proof. Unless the adversary can contradict the culpable soundness of $\Pi^{\mathcal{H}}$, there exists $\alpha \in \mathbb{Z}$ such that $g^2 = h^{2\alpha}$. Assuming $|x|$ and $|r|$ to be at most ℓ bits long, for $y, s \leftarrow_\$ [\![0; P2^{\ell+\lambda}]\!]$, the distribution of the tuple $\left(h^{2(\alpha y + s)}, e, y - ex, s - er\right)$ is at a statistical distance of at most $2^{-\lambda+1}$ from the distribution of the tuple $\left(h^{2(\alpha(y+ex)+(s+ex))}, e, y, s\right) = \left((g^y h^s)^2 C^e, e, y, s\right)$. Since $(g^y h^s)^2 C^e$ can be computed in $O(\ell + \lambda + \log P)$ group operations with classical sliding-window algorithms [Ava05], the statement follows. □

Theorem 9.2.12. *$\Pi_{\mathscr{C}}$ is $\left(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*}, \varepsilon^{\mathrm{ext}}, \Sigma\right)$-extractable for any $T_{\mathcal{A}}$ and T_{Prove^*} such that $T_{\mathcal{A}} + b_{\mathbb{G}} \log(P) T_{\mathsf{Prove}^*} T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$, with $\varepsilon^{\mathrm{ext}} \coloneqq \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right) + \varepsilon^{\mathrm{zk}}_{\Pi^{\mathcal{H}}}$.*

[2] To prove such statements using hidden-order groups, Lipmaa's range argument [Lip03], corrected by Couteau, Peters and Pointcheval [CPP17], is suitable.

Proof. To prove culpable extractability, note that from two accepting transcripts (D, e_1, z_1, t_1) and (D, e_2, z_2, t_2) (with the same $D \in \mathbb{G}$) such that $e_1 \neq e_2$, the equality $C^{e_2-e_1} = \left(g^{z_1-z_2}h^{t_1-t_2}\right)^2$ follows. If $(e_2 - e_1)$ divides $(z_1 - z_2)$ and $(t_1 - t_2)$, then the low-order assumption implies that $C = \left(g^{(z_1-z_2)/(e_2-e_1)}h^{(t_1-t_2)/(e_2-e_1)}\right)^2 \tilde{g}_C$ for some group element $\tilde{g}_C$ such that $\tilde{g}_C^2 = 1$, i.e., $((z_1 - z_2)/(e_2 - e_1), (t_1 - t_2)/(e_2 - e_1)$ is a valid witness for C. Besides, one can show that $e_1 - e_2$ does not divide $z_2 - z_1$ or $t_2 - t_1$ with probability at most $\left(1/2 - 2^{-\lambda} - (1 - \mu)\right)^{-1}\left(\varepsilon^{\text{ord}} + \varepsilon^{\text{strg}} + 1 - \mu\right)$ under the $(T^{\text{strg}}, \varepsilon^{\text{strg}})$-strong-root assumption, the $\left(T^{\text{ord}}, \varepsilon^{\text{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, and assuming that the running time of the prover $(\mathcal{A}, \mathsf{Prove}^*)$ is such that $T_{\mathcal{A}} + \log(P) b_{\mathbb{G}} T_{\mathsf{Prove}^*} T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\text{strg}}, \ T^{\text{ord}}\right)\right)$. This part is a special case of Lemma 9.3.7 and is similar to a part of the extractability proof of the protocol proposed by Damgård and Fujisaki to argue knowledge of openings to their commitments.

To obtain two valid transcripts with distinct challenges, it suffices to run the proving algorithm once, rewind it to the computation step right after it sent its first message and run it anew on fresh verifier randomness. If both runs are valid and the challenges are distinct, then return the two transcripts, otherwise restart. If a prover $(\mathcal{A}, \mathsf{Prove}^*)$ convince the verifier with probability at least ε, the expected runtime of this procedure is at most $\left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*}$. Define then an extractor $\mathcal{E}$ which first simulates a proof π that $g \in \sqrt{\langle h^2 \rangle}$ with $FS.\Pi^{\mathcal{H}}.\mathsf{Sim}$ and then runs the previous procedure. The theorem then follows. □

Multi-Integer Commitments.

The above commitments can be generalized to vectors of integers just like Damgård–Fujisaki commitments (as Couteau, Peters and Pointcheval [CPP17] did). That is to say, the scheme can be extended to commit to several integers at once.

Formally, let G be a group generator and suppose that there exists a non-interactive argument system $FS.\Pi^{\mathcal{H}}$ with random oracle $\mathcal{H}$ for the language $\left\{g_1, \ldots, g_n \in \mathbb{G}, \ell \in \mathbb{N}^* \colon \exists \alpha_1, \ldots, \alpha_n \in \left[\!\left[0; 2^\ell\right]\!\right], \forall i \in [\![n]\!] \ g_i = h^{\alpha_i}\right\}$ given parameters $(\mathbb{G}, P, h, n)$ and the empty string as CRS.

$\mathsf{Setup}\left(1^\lambda, n\right) \to pp$: run $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$, generate $h \leftarrow_{\$} \mathbb{G}$ and return $(\mathbb{G}, P, h, n)$.

$\mathsf{KG}(pp) \to ck$: generate $\alpha_i \leftarrow_{\$} \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$ for $i \in [\![n]\!]$, compute $g_i \leftarrow h^{\alpha_i}$ and $\pi \leftarrow FS.\Pi^{\mathcal{H}}.\mathsf{Prove}\left((\mathbb{G}, P, h, n), (\mathbf{g}, b_{\mathbb{G}} + \lambda), (\alpha_i)_{i=1}^n\right)$, and return $(\mathbf{g}, \pi)$.

$\mathsf{Com}\left((\mathbf{g}, \pi), x_1, \ldots, x_n \in \mathbb{Z}\right) \to (C, d)$: if $FS.\Pi^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, h, n), (\mathbf{g}, b_{\mathbb{G}} + \lambda),$

$\pi) = 0$ return $\perp$; generate $r \leftarrow_{\$} [\![0; 2^{b_{\mathbb{G}}+\lambda}]\!]$, compute $C \leftarrow (\prod_{i=1}^{n} g_i^{x_i} h^r)^2$, set $d \leftarrow r$ and return (C, d).

$\mathsf{ComVf}((\mathbf{g}, \pi), C, x_1, \ldots, x_n, d) \rightarrow b \in \{0,1\}$: if $C^2 = \left(\prod_i g_i^{x_i} h^d\right)^4$ return 1, else return 0.

The only missing component is an interactive protocol Π that satisfies culpable soundness w.r.t. $\{g_1, \ldots, g_n \in \mathbb{G} \colon \exists \alpha_1, \ldots, \alpha_n \in \mathbb{Z}, \forall i \in [\![n]\!]\, g_i^2 = h^{2\alpha_i}\}$. A possible solution is to run n times in parallel the protocol from the case $n = 1$ for each of the α_i values. However, they achieve an overall $2^{-\lambda}$ statistical distance from n simulated arguments, the range of the prover's randomness in the Section–9.2.2 protocol must be multiplied by n so that each argument is $2^{-\lambda} n^{-1}$-zero-knowledge. A better solution is to use the protocol of Section 9.4.3, which results in arguments of $O(b_{\mathbb{G}} + \log n)$ bits. This should be compared to the $\Omega(n b_{\mathbb{G}} \log P)$-bit parameters of the generalized Damgård–Fujisaki commitments.

9.3 Succinct Inner-Product Arguments on Integers

This section gives a statistically honest-verifier zero-knowledge, logarithmic-size inner-product argument on integers committed in hidden-order groups with the scheme from Section 9.2.2. That is, an argument of knowledge of vectors $\mathbf{a}$ and $\mathbf{b} \in \mathbb{Z}^n$, and of a randomness $r \in \mathbb{Z}$ such that $C^2 = \left(\mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} f^r\right)^4$ and $\langle \mathbf{a}, \mathbf{b} \rangle = z$ given public bases $\mathbf{g}$ and $\mathbf{h}$, a public commitment C and a public integer z; and the bit-communication complexity of the protocol is logarithmic in of order $O(\ell + \log n b_{\mathbb{G}})$ where ℓ is an upper-bound on the bit length of the largest integer witness and $2^{b_{\mathbb{G}}}$ an upper-bound on the order of the group.

9.3.1 Formal Description

This section formalizes the protocol and gives precise statements as to the properties it satisfies.

Relations.

The protocol is a honest-verifier zero-knowledge argument for

$$\mathcal{R} \coloneqq \left\{ (C \in \mathbb{G}, z \in \mathbb{Z}, \ell \in \mathbb{N}^*; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n, r \in \mathbb{Z}) : C^2 = \left(\mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} f^r\right)^4 \right.$$
$$\left. \wedge \langle \mathbf{a}, \mathbf{b} \rangle = z \wedge \left\| \begin{bmatrix} \mathbf{a} & \mathbf{b} & r \end{bmatrix} \right\|_\infty < 2^\ell \right\}$$

given parameters $(\mathbb{G}, P, f, n)$ with $f \in \mathbb{G}$ and $n \in \mathbb{N}^*$, and $(\mathbf{g}, \mathbf{h}, \pi_{crs}) \in \mathbb{G}^{2n} \times \{0,1\}^*$ as CRS.

The relation imposes the largest value (in absolute value) in the witness $\begin{bmatrix}\mathbf{a} & \mathbf{b} & r\end{bmatrix}$ to be at most ℓ bits long, with ℓ being part of the (public) word. As for the argument of knowledge of openings in Section 9.2.2, it is again to adapt the randomness range of the prover and of the honest-verifier zero-knowledge simulator to make sure that the protocol remains statistically honest-verifier zero-knowledge; and ℓ can be arbitrarily large. However, the protocol does not necessarily return a witness with integers of at most ℓ bits in absolute value. In other words, the protocol satisfies culpable extractability w.r.t. the relation

$$\Sigma \coloneqq \left\{(C \in \mathbb{G}, z \in \mathbb{Z}, \ell \in \mathbb{N}^*; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n, r \in \mathbb{Z}) : C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}f^r\right)^4 \right.$$
$$\left. \wedge \langle \mathbf{a}, \mathbf{b}\rangle = z\right\}.$$

The argument for $\mathcal{R}$ is actually reduced to a logarithm-size argument (given on Figure 9.2) for the following relation in which the inner product is also committed:

$$\mathcal{R}' \coloneqq \left\{(C \in \mathbb{G}, \ell \in \mathbb{N}^*; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n, r \in \mathbb{Z}) : C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}e^{\langle \mathbf{a},\mathbf{b}\rangle}f^r\right)^4 \right.$$
$$\left. \wedge \left\|\begin{bmatrix}\mathbf{a} & \mathbf{b} & r\end{bmatrix}\right\|_\infty < 2^\ell\right\}$$

given parameters $(\mathbb{G}, P, f, n)$ with $f \in \mathbb{G}$ and $n \in \mathbb{N}^*$, and $(\mathbf{g}, \mathbf{h}, e, \pi_{crs}) \in \mathbb{G}^{2n+1} \times \{0,1\}^*$ as CRS. Again, the protocol does not guarantee that the extracted witness satisfies the bounds on its bit length – denote by Σ' the relation defined as $\mathcal{R}'$ without the restriction on the size of the witness.

During the reduction, the verifier chooses a base $e \in \langle f \rangle$ and proves to the prover that e is in $\sqrt{\langle f^2 \rangle}$, which guarantees to the prover that the commitment Ce^{2z} remains hiding. (As explained in Section 9.2, this precaution is not needed in groups of public prime orders.) However, since the protocol in Section 9.2.2 is only honest-verifier, and the extractability of the argument system partly relies on the fact that the prover does not know a discrete-logarithm relation between e and f, the verifier must compute a non-interactive argument with a random oracle. In other words, the extractability of the argument relies on the zero-knowledge property of the protocol in Section 9.2.2. Moreover, the CRS of the protocol includes a proof that $\mathbf{g}$ and $\mathbf{h}$ are in $\sqrt{\langle f^2 \rangle}^n$, and the argument is only guaranteed to be honest-verifier zero-knowledge if it is indeed the case; that is, the zero-knowledge property of the argument relies on the soundness of the protocol. This mirroring in the properties of two protocols is simply due to the fact that at the beginning of the inner-product argument, the prover becomes the verifier of the protocol for $\mathbf{g}, \mathbf{h} \in \sqrt{\langle f^2 \rangle}^n$.

Main Insights.

The goal is to have a protocol for $\mathcal{R}'$ in which the prover sends only $2\lceil \log n \rceil + 2$ group elements and three integers of at most $O(\ell + b_{\mathbb{G}} + \log(n)\log(P))$ bits. The main idea is to have the prover first send a constant number of commitments that depend on the witness vectors (which are in $\mathbb{Z}^n$), so that the verifier can thereafter choose *integer* linear combinations (defined by an integer x) of the witness vectors that are of length $n/2$ (to ease the explanation, further assume n to be a power of 2 in this section). These new vectors then serve as witness for a new commitment derived from the original commitment on which the proof is computed, the commitments sent by the prover and x; in bases of length $n/2$ and determined by the original bases and x. The prover and the verifier can thus recursively run the protocol with vectors of length $n/2$. After $\log n$ recursive calls, the vectors are of length 1, and the parties run a protocol that two committed integers a and b satisfy $ab = z$ for a public z.

In more detail, given $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$ and $r \in \mathbb{Z}$ such that $C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} e^{\langle \mathbf{a}, \mathbf{b} \rangle} f^r\right)^4$, the prover first sends commitments $U \leftarrow \left(\mathbf{g}_1^{\mathbf{a}_2}\mathbf{h}_2^{\mathbf{b}_1} e^{\langle \mathbf{a}_2, \mathbf{b}_1 \rangle} f^{s_u}\right)^2$ and $V \leftarrow \left(\mathbf{g}_2^{\mathbf{a}_1}\mathbf{h}_1^{\mathbf{b}_2}\ e^{\langle \mathbf{a}_1, \mathbf{b}_2 \rangle} f^{s_v}\right)^2$, for s_u and s_v with uniform distribution over an integer set large enough for the commitments to be hiding. The verifier chooses $x \leftarrow_{\$} [\![0; P-1]\!]$, sends it to the prover, and this latter computes $\mathbf{a}' \leftarrow \mathbf{a}_1 + x\mathbf{a}_2$, $\mathbf{b}' \leftarrow x\mathbf{b}_1 + \mathbf{b}_2$ and $t \leftarrow s_v + rx + s_u x^2$. Note that all these operations are performed in $\mathbb{Z}$ and do not require to invert any integer. Now note that

$$\left((\mathbf{g}_1^x \circ \mathbf{g}_2)^{\mathbf{a}'} (\mathbf{h}_1 \circ \mathbf{h}_2^x)^{\mathbf{b}'} e^{\langle \mathbf{a}', \mathbf{b}' \rangle} f^t\right)^4 = \left(U^{x^2} C^x V\right)^2,$$

which means that the prover and verifier can run the protocol again with $\mathbf{g}_1^x \circ \mathbf{g}_2$ and $\mathbf{h}_1 \circ \mathbf{h}_2^x$ as bases and $\mathbf{a}'$ and $\mathbf{b}'$ (all of size $n/2$ instead of n) as witness for $U^{x^2} C^x V$.

To understand how a witness consisting of integer vectors can be extracted, suppose that one can obtain three transcripts $\left(U, V, x_j, \mathbf{a}'_j, \mathbf{b}'_j, t'_j\right)_{i=1}^3$ such that

$$\left((\mathbf{g}_1^{x_j} \circ \mathbf{g}_2)^{\mathbf{a}'_j} (\mathbf{h}_1 \circ \mathbf{h}_2^{x_j})^{\mathbf{b}'_j} e^{\langle \mathbf{a}'_j, \mathbf{b}'_j \rangle} f^{t_j}\right)^4 = \left(U^{x_j^2} C^{x_j} V\right)^2$$

for all $j \in [\![3]\!]$. The goal is to find a representation of C in the bases $\mathbf{g}$, $\mathbf{h}$, e and f. To do so, consider the linear system:

$$\mathbf{X} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ for } \mathbf{X} \coloneqq \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \text{ and indeterminate } \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix}.$$

It does not necessarily have a solution in $\mathbb{Z}^3$ (and this is the first major difference with Bulletproofs in groups with public prime orders). However,

denoting by $\mathrm{adj}(\mathbf{X})$ the adjugate matrix of $\mathbf{X}$ (which is in $\mathbb{Z}^{3\times 3}$), the column vector

$$\nu_C \coloneqq \mathrm{adj}(\mathbf{X}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ satisfies } \mathbf{X}\nu_C = \mathbf{X}\,\mathrm{adj}(\mathbf{X}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \det(\mathbf{X}) \\ 0 \end{bmatrix}$$

since $\mathbf{X}\,\mathrm{adj}(\mathbf{X}) = \det(\mathbf{X})\mathbf{I}_3$. Therefore, via linear combinations with coefficient determined by ν_C, one can obtain $\mathbf{a}_C, \mathbf{b}_C \in \mathbb{Z}^n$ and $z_C, r_C \in \mathbb{Z}$ such that $C^{2\det\mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}_C}\mathbf{h}^{\mathbf{b}_C}e^{z_C}f^{r_C}\right)^4$. If the challenges x_1, x_2, x_3 are pairwise distinct, then $\det\mathbf{X} \neq 0$, and Lemma 9.3.7 shows that under the assumptions on the group generator, $2\det\mathbf{X}$ must divide (with overwhelming probability) $4z_C$, $4r_C$ and each of the components of $4\mathbf{a}_C$, $4\mathbf{b}_C$. Therefore, up to a relabeling of $2\mathbf{a}_C/\det\mathbf{X}$ and so on, one can extract $\mathbf{a}_C, \mathbf{b}_C \in \mathbb{Z}^n$ and $z_C, r_C \in \mathbb{Z}$ such that $C = \left(\mathbf{g}^{\mathbf{a}_C}\mathbf{h}^{\mathbf{b}_C}e^{z_C}f^{r_C}\right)^2 \tilde{g}_C$ for $\tilde{g}_C \in \mathbb{G}$ that satisfies $\tilde{g}_C^2 = 1_{\mathbb{G}}$.

Nonetheless, it is not yet certain that $z_C = \langle \mathbf{a}_C, \mathbf{b}_C \rangle$. To guarantee it, it suffices to extract similar representations for U and V, and replacing U, C and V by those representations in the equality

$$\left((\mathbf{g}_1^x \circ \mathbf{g}_2)^{\mathbf{a}'}\,(\mathbf{h}_1 \circ \mathbf{h}_2^x)^{\mathbf{b}'}\,e^{\langle \mathbf{a}', \mathbf{b}' \rangle} f^t\right)^4 = \left(U^{x^2}C^xV\right)^2$$

for any $x \in \{x_1, x_2, x_3\}$. This leads to a discrete-logarithm relation $1_{\mathbb{G}} = \mathbf{g}_1{}^{p_{\mathbf{g}_1}(x)}\mathbf{g}_2{}^{p_{\mathbf{g}_2}(x)}\mathbf{h}_1{}^{p_{\mathbf{h}_1}(x)}\mathbf{h}_2{}^{p_{\mathbf{h}_2}(x)}e^{p_e(x)}f^{p_f(x)}$ with $p_{\mathbf{g}_1}, p_{\mathbf{g}_2}, p_{\mathbf{h}_1}, p_{\mathbf{h}_2}, p_e, p_f$ polynomials in $\mathbb{Z}[x]$ of degree at most 2. Lemma 9.3.8 essentially states that it is hard to find discrete-logarithm relations in the subgroup generated by a group element $f \leftarrow_{\$} \mathbb{G}$ (this is the second main difference with Bulletproofs in groups with public prime orders). It thus implies that if the bases are all in $\langle f \rangle$ with exponents chosen uniformly at random over a large integer set, these polynomials must all be zero (with overwhelming probability) when evaluated at x; and $p_{\mathbf{g}_1}, p_{\mathbf{h}_2}$ and p_e together lead to an integer polynomial of degree 4, with leading coefficient $z_C - \langle \mathbf{a}_c, \mathbf{b}_C \rangle$, that must be nil when evaluated at x. Therefore, starting with five accepting transcripts instead of three entails that this polynomial of degree 4 must be nil and thus $z_C = \langle \mathbf{a}_c, \mathbf{b}_C \rangle$, i.e., $\mathbf{a}_C, \mathbf{b}_C \in \mathbb{Z}^n, r_C \in \mathbb{Z}$ is a valid witness for C.

As for the zero-knowledge property of the scheme, the ranges of s_u and s_v at each of the $\log n$ recursion step are chosen so that the statistical distance of (U, V) to a pair of uniform values in $\langle f^2 \rangle$ is at most $\left(\log(n)2^\lambda\right)^{-1}$. It then remains to compute an upper-bound on the bit length of the witness at the lass step of the protocol so that the randomness of the prover can be chosen from a set of which the bit length is λ times larger. The calculation is detailed in the proof of Theorem 9.3.5.

Protocol Algorithms.

The argument system for relation $\mathcal{R}$ is further denoted Π. It uses as building blocks a group generator G and the Fiat–Shamir non-interactive variant $FS.\tilde{\Pi}^{\mathcal{H}}$ with a random oracle $\mathcal{H}$ of a protocol $\tilde{\Pi}$ for the language

$$\left\{(\mathbf{g},\mathbf{h}) \in \mathbb{G}^{2n} \colon \exists \alpha,\beta \in \mathbb{Z}^{2n}, \forall i \in [\![n]\!]\, g_i = f^{\alpha_i} \wedge h_i = f^{\beta_i}\right\}$$

given parameters $(\mathbb{G}, P, f, 2n)$ and the empty string as CRS. $\tilde{\Pi}^{\mathcal{H}}$ is later assumed to satisfy culpable soundness w.r.t. the language

$$\left\{(\mathbf{g},\mathbf{h}) \in \mathbb{G}^{2n} \colon \exists \alpha,\beta \in \mathbb{Z}^{2n}, \forall i \in [\![n]\!]\, g_i^2 = f^{2\alpha_i} \wedge h_i^2 = f^{2\beta_i}\right\}.$$

The protocol algorithms are then as follows:

- $\Pi.\mathsf{Setup}\left(1^\lambda, n \in \mathbb{N}^*\right)$ runs $(\mathbb{G}, P') \leftarrow \mathsf{G}\left(1^\lambda\right)$, computes $P \coloneqq \left\lfloor P'^{1/3} \right\rfloor$ (the power $1/3$ is to ensure extractability under the assumptions on the group generator), generates $f \leftarrow_\$ \mathbb{G}$ and returns $pp \leftarrow (\mathbb{G}, P, n, f)$ as public parameters.
- $\Pi.\mathsf{CRSGen}(pp)$ generates $\alpha_i, \beta_i \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+2\lambda}]\!]$ for $i \in [\![n]\!]$, computes $g_i \leftarrow f^{\alpha_i}$, $h_i \leftarrow f^{\beta_i}$ and $\pi_{crs} \leftarrow FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Prove}\left((\mathbb{G}, P, f, 2n), (\mathbf{g},\mathbf{h}), \alpha, \beta\right)$, and returns $(\mathbf{g},\mathbf{h},\pi_{crs})$.
- $\Pi.\mathsf{Prove}$ and $\Pi.\mathsf{Vf}$ are as on Figure 9.1. They run as sub-routines the proving and verification algorithms of a protocol Π' for relation $\mathcal{R}'$.
  - $\tilde{\Pi}.\mathsf{Setup}\left(1^\lambda, n \in \mathbb{N}^*\right)$ is the same as $\Pi.\mathsf{Setup}\left(1^\lambda, n \in \mathbb{N}^*\right)$.
  - $\tilde{\Pi}.\mathsf{CRSGen}(pp)$ computes $\mathbf{g}$ and $\mathbf{h}$ as $\Pi.\mathsf{CRSGen}(pp)$, and also $e \leftarrow f^\gamma$ for $\gamma \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+2\lambda}]\!]$. It then computes a proof $\pi_{crs} \leftarrow FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Prove}\left((\mathbb{G}, P, f,\ 2n+1), (\mathbf{g},\mathbf{h},e), \alpha, \beta, \gamma\right)$ that the bases g_i, h_i for $i \in [\![n]\!]$ and e are in $\sqrt{\langle f^2 \rangle}$, and eventually returns the tuple $(\mathbf{g},\mathbf{h},e,\pi_{crs})$
  - $\Pi'.\mathsf{Prove}$ and $\Pi'.\mathsf{Vf}$ are as on Figure 9.2, except that $\Pi'.\mathsf{Prove}$ first verifies the CRS and aborts if it is invalid. In the presentation on Figure 9.2, the CRS does not include the proof that it is well-formed and the prover does not perform any preliminary verification (i.e., the CRS is assumed to be honestly generated). The reason is that $\Pi.\mathsf{Prove}$ already does this verification; if Π' were to be used as a stand-alone protocol, those verification would be necessary. $\Pi'.\mathsf{Prove}$ and $\Pi'.\mathsf{Vf}$ additionally take as input a variable i which keeps track of the recursion depth during the protocol execution to adjust the randomness of the prover.

$\mathcal{P}(n, f, \mathbf{g}, \mathbf{h}, \pi_{crs}, C, z, \ell; \mathbf{a}, \mathbf{b}, r)$ $\quad$ $\mathcal{V}(n, f, \mathbf{g}, \mathbf{h}, \pi_{crs}, C, z, \ell)$

$$C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}f^r\right)^4 \wedge \langle \mathbf{a}, \mathbf{b}\rangle = z \wedge \left\|\begin{bmatrix}\mathbf{a} & \mathbf{b} & r\end{bmatrix}\right\|_\infty < 2^\ell$$

---

if $FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, f, 2n), (\mathbf{g}, \mathbf{h}), \pi_{crs}) = 0$
then return $\perp$

$\alpha \leftarrow_\$ [\![0; 2^{b+2\lambda}]\!]$; $e \leftarrow f^\alpha$
$\pi \leftarrow FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Prove}((\mathbb{G}, P, f, 1), e, \alpha)$

$\xleftarrow{e,\pi}$

if $FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, f, 1), e, \pi) = 0$
then return $\perp$
run the protocol on Figure 9.2 on input $\left(1, n, f, \mathbf{g}, \mathbf{h}, e, C_1 \coloneqq Ce^{2z}, \ell; \mathbf{a}, \mathbf{b}, r\right)$

Figure 9.1: Inner-Product Argument on Integers.

$\mathcal{P}(i, n, f, \mathbf{g}, \mathbf{h}, e, C_i, \ell; \mathbf{a}, \mathbf{b}, r)$ $\quad$ $\mathcal{V}(i, n, f, \mathbf{g}, \mathbf{h}, e, C_i, \ell)$

$$C_i^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}e^{\langle \mathbf{a}, \mathbf{b}\rangle}f^r\right)^4 \wedge \left\|\begin{bmatrix}\mathbf{a} & \mathbf{b} & r\end{bmatrix}\right\|_\infty < 2^\ell$$

---

if $n = 1$

$\alpha, \beta \leftarrow_\$ [\![0; 2^{\ell+\lambda}P^i]\!]$
$s \leftarrow_\$ [\![0; 2(i-1)2^{b_\mathbb{G}+\lambda}]\!]$
$t \leftarrow_\$ \left[\!\!\left[0; 2(i-1)2^{b_\mathbb{G}+2\lambda}P^{i+2} + 2^{\ell+\lambda}(P-1)^{i+1}\right]\!\!\right]$
Replace $i-1$ by 1 if $i = 1$
$\Gamma \leftarrow \left(g^\alpha h^\beta e^{\alpha b + a\beta} f^s\right)^2$
$\Delta \leftarrow \left(e^{\alpha\beta} f^t\right)^2$

$\xrightarrow{\Gamma,\Delta}$

$x_i \leftarrow_\$ [\![0; P-1]\!]$

$\xleftarrow{x_i}$

$a' \leftarrow \alpha + ax_i$
$b' \leftarrow \beta + bx_i$
$u \leftarrow t + sx_i + rx_i^2$

$\xrightarrow{a',b',u}$

$\left(g^{x_i a'} h^{x_i b'} e^{a'b'} f^u\right)^4 \stackrel{?}{=} \left(C_i^{x_i}\Gamma^{x_i}\Delta\right)^2$

else

$s_u, s_v \leftarrow_\$ [\![0; 2(\lceil \log n \rceil + i - 1)2^{b_\mathbb{G}+\lambda}]\!]$
$U_i \leftarrow \left(\mathbf{g}_1^{\mathbf{a}_2}\mathbf{h}_2^{\mathbf{b}_1}e^{\langle \mathbf{a}_2, \mathbf{b}_1\rangle}f^{s_u}\right)^2$
$V_i \leftarrow \left(\mathbf{g}_2^{\mathbf{a}_1}\mathbf{h}_1^{\mathbf{b}_2}e^{\langle \mathbf{a}_1, \mathbf{b}_2\rangle}f^{s_v}\right)^2$

$\xrightarrow{U_i,V_i}$

$x_i \leftarrow_\$ [\![0; P-1]\!]$

$\xleftarrow{x_i}$

$\mathbf{a}' \leftarrow \mathbf{a}_1 + x_i\mathbf{a}_2$
$\mathbf{b}' \leftarrow x_i\mathbf{b}_1 + \mathbf{b}_2$
$t \leftarrow s_v + rx_i + s_u x_i^2$

recurse on $\left(i+1, \lceil n/2 \rceil, f, \mathbf{g}_1^{x_i} \circ \mathbf{g}_2, \mathbf{h}_1 \circ \mathbf{h}_2^{x_i}, e, C_{i+1} \coloneqq U_i^{x_i^2} C_i^{x_i} V_i, \ell; \mathbf{a}', \mathbf{b}', t\right)$

Figure 9.2: Argument for Relation $\mathcal{R}'$.

Prover-Communication Complexity.

Throughout the protocol, the prover sends $2n' + 2$ group elements (with $n' = \lceil \log n \rceil$), two integers ($a'$ and b') less than $2^{\ell} P^{n'}$ in absolute value and an integer (u) less than $\left(2n' 2^{b_{\mathbb{G}}+\lambda} P^{n'+3} + 2^{\ell}(P-1)^{n'+2}\right)\left(1 + 2^{\lambda}\right)$ in absolute value. The bit communication complexity of the prover is then of order $O\left(\ell + \log(n)(b_{\mathbb{G}} + \log P) + \lambda + \max\left(\log\log n + b_{\mathbb{G}} + \lambda, \ell\right)\right)$. Since $\log P \leq b_{\mathbb{G}} = \Omega(\lambda)$, that is $O\left(\ell + \log(n) b_{\mathbb{G}} + \max\left(\log\log n + b_{\mathbb{G}}, \ell\right)\right)$, or even $O\left(\ell + \log(n) b_{\mathbb{G}}\right)$ bits (n is here assumed to be greater than 1).

Verification via a Single Multi-Exponentiation.

As described on Figure 9.2, the verifier computes a new commitment $U_i^{x_i^2} C_i^{x_i} V_i$, and new vectors $\mathbf{g}_1^{x_i} \circ \mathbf{g}_2$ and $\mathbf{h}_1 \circ \mathbf{h}_2^{x_i}$ at each recursion step i. In total, the verifier then has to compute $n' \coloneqq \lceil \log n \rceil$ 3-exponentiations with exponents less than P^2 and two $\lceil n 2^{-i} \rceil$-exponentiations with exponents less than P for $i = 0, \ldots, n' - 1$. At the last stage of the protocol, the verifier also has to check that $\left(g^{x_{n'+1} a'} h^{x_{n'+1} b'} e^{a' b'} f^{u}\right)^4 = \left(C_{n'+1}^{x_{n'+1}^2} \Gamma^{x_{n'+1}} \Delta\right)^2$, i.e., a 7-exponentiation with exponents (in absolute value) less than the bit length of the largest exponent.

Alternatively, the verifier could simply generate the challenges after receiving the U_i and V_i values, delay its verification to the last stage of the protocol and then do a single multi-exponentiation. As shown below, this multi-exponentiation is a $(2n + 2n' + 5)$-exponentiation, which results in computational savings in practice since computing a k-exponentiation with ℓ-bit exponents requires ℓ group operations with a pre-computed table of 2^k group elements following classical sliding-window methods [Ava05], which is much faster than computing k separate single exponentiations with ℓ-bit exponents (which requires $k\ell$ group operations with a single group element in memory) and multiplying the result. Of course, if n is large, then the pre-computation might be prohibitively long with the standard multi-exponentiation method, in which case one would rather split the multi-exponentiation in small batches of appropriate size. In any case, delaying the verification until the last step already has the benefit of eliminating latency in the verification.

To reduce the verification to a single multi-exponentiation, the commitment at the last stage and the bases g and h must be expressed in terms of the challenges x_i and of the initial vectors $\mathbf{g}$ and $\mathbf{h}$ alone.

Ultimate Commitment.

Given that $C_{i+1} \coloneqq U_i^{x_i^2} C_i^{x_i} V_i$ for all $i \in [\![n']\!]$, the commitment $C_{n'+1}$ at the last step of the protocol is equal to

$$U_{n'}^{x_{n'}^2} \prod_{i=1}^{n'-1} U_i^{x_i^2 x_{i+1} \cdots x_{n'}} C^{x_1 \cdots x_{n'}} \prod_{i=1}^{n'-1} V_i^{x_{i+1} \cdots x_{n'}} V_{n'}.$$

Expression for g and h.

It now remains to express the bases g and h at the last step of the recursion in terms of g_i, h_i for $i \in [\![n]\!]$ and x_j for $j \in [\![n']\!]$.

First assume n to be a power of 2 and let $n' \coloneqq \log n$. In this case,

$$g = \prod_{i=1}^{n} g_i^{\prod_{j \in S_i} x_j} \quad \text{and} \quad h = \prod_{i=1}^{n} h_i^{\prod_{j \in [\![n]\!] \setminus S_i} x_j}$$

with

$$S_i \coloneqq \left\{ j \in [\![n']\!] : n' + 1 - j\text{th bit of } i - 1 \text{ is } 0 \right\}.$$

In case n is not a power of 2, finding an explicit expression for the exponents x_j to which g_i is raised is much more intricate. For instance, in case $n = 5$, $g = g_1^{x_1 x_2 x_3} g_2^{x_1 x_3} g_3^{x_2 x_3} g_4^{x_3} g_5$. The above expression is no longer true since, for example, the exponent g_5 is 1 although the first and second bits of 4 are nil and the exponent of g_4 is x_3 even though the first bit of 3 is 1.

Explicit Expression for the Final Bases. First consider the case of g. Notice that the g_i elements that are raised to the power x_1 in the expression of g are such that $1 \leq i \leq \lfloor n/2 \rfloor$, and the elements that are raised to the power x_2 are such that $1 \leq i \leq \lfloor \lceil n/2 \rceil /2 \rfloor$ or $\lfloor n/2 \rfloor + 1 \leq i \leq \lfloor n/2 \rfloor + \lfloor \lceil n/2 \rceil /2 \rfloor$. Likewise, the elements that are raised to the power x_3 are such that

$$\begin{gathered} 1 \leq i \leq \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor \\ \lfloor \lceil n/2 \rceil /2 \rfloor + 1 \leq i \leq \lfloor \lceil n/2 \rceil /2 \rfloor + \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor \\ \lfloor n/2 \rfloor + 1 \leq i \leq \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor \\ \lfloor n/2 \rfloor + \lfloor \lceil n/2 \rceil /2 \rfloor + 1 \leq i \leq \lfloor n/2 \rfloor + \lfloor \lceil n/2 \rceil /2 \rfloor + \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor. \end{gathered}$$

In general, the elements g_i that are raised to the power x_j (for $j \geq 2$) are such that $1 + k * \lfloor \lceil n2^{j-2} \rceil /2 \rfloor \leq i \leq 1 + k * \lfloor \lceil n2^{j-2} \rceil /2 \rfloor + \lfloor \lceil n2^{-j+1} \rceil /2 \rfloor$ for $0 \leq k \leq j - 2$, with $k * \lfloor \lceil n2^{j-2} \rceil /2 \rfloor$ defined as $k \lfloor \lceil n2^{j-2} \rceil /2 \rfloor$ if k is odd and $k/2 * \lfloor \lceil n2^{j-3} \rceil /2 \rfloor$ if k is even and greater than 0 (if $k = 0$, it is simply

0). That is, if $v_2(k)$ denotes the dyadic valuation of k, $k * \lfloor \lceil n2^{j-2} \rceil /2 \rfloor = k2^{-v(k)} \left\lfloor \left\lceil n2^{j-2-v(k)} \right\rceil /2 \right\rfloor$.

As for $\mathbf{h}$, the elements h_i that are raised to the power x_j (for $j \geq 2$) in the expression of h are such that i is in the complement of the union of the above intervals. □

Another possibility is to expand, $\mathbf{g}$ and $\mathbf{h}$ to vectors $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ of size $2^{n'}$ by inserting $1_{\mathbb{G}}$ at specific positions so that the result of the folding procedure applied $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are the same as the result of the procedure applied to $\mathbf{g}$ and $\mathbf{h}$ (i.e., g and h). This reduces the multi-exponentiation to the case in which the size of the vectors is a power of 2.

Reduction to Powers of 2. The idea is to expand, as does Algorithm SPAN, $\mathbf{g}$ and $\mathbf{h}$ to vectors $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ of size $2^{n'}$ by inserting $1_{\mathbb{G}}$ at specific positions so that the result of the folding procedure applied $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are the same as the result of the procedure applied to $\mathbf{g}$ and $\mathbf{h}$ (i.e., g and h). This reduces the multi-exponentiation to the case in which the size of the vectors is a power of 2.

As a first step, the following lemma shows that the size of the vectors of group elements at the $(i+1)$th step of the protocol, which is $\lceil \lceil \lceil n/2 \rceil /2 \rceil \cdots /2 \rceil$ (i times), is actually equal to $\lceil n2^{-i} \rceil$.

Lemma 9.3.2. *For any two integers $n \geq 1$ and $i \geq 0$, $\lceil \lceil n2^{-i+1} \rceil /2 \rceil = \lceil n2^{-i} \rceil$.*

Proof. Let w denote the Hamming weight of n. As $n \geq 1$, $w \geq 1$ as well. Then, write n as $2^{\nu_1} + \cdots + 2^{\nu_w}$, with $0 \leq \nu_1 < \cdots < \nu_w = \lfloor \log n \rfloor$, and thus $n2^{-i+1} = 2^{\nu_1 - i + 1} + \cdots + 2^{\nu_w - i + 1}$.

If $i > \lfloor \log n \rfloor = \nu_w$, then $0 < \lceil n2^{-i+1} \rceil \leq 2$, $\lceil \lceil n2^{-i+1} \rceil /2 \rceil = 1$, and $0 < n2^{-i} = 2^{\nu_1 - i} + \cdots + 2^{\nu_w - i} < 1$, so $\lceil n2^{-i} \rceil = 1$.

If $i \leq \lfloor \log n \rfloor$, the set $\{1 \leq k \leq w \colon \nu_k > i - 1\}$ is non-empty as it always contains w. Let ℓ denotes its minimum. Rewrite then $n2^{-i+1}$ as $2^{\nu_1 - i + 1} + \cdots + 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$. Note that by definition, $\nu_\ell \geq i$.

If $\ell = 1$, then $\lceil n2^{-i+1} \rceil = 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$ and $\lceil \lceil n2^{-i+1} \rceil /2 \rceil = 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i} = n2^{-i} = \lceil n2^{-i} \rceil$. If $\ell > 1$, further distinguish two cases:

- if $\nu_{\ell-1} = i - 1$, then $\lceil n2^{-i+1} \rceil = 2 + 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$ and $\lceil \lceil n2^{-i+1} \rceil /2 \rceil = \lceil n2^{-i+1} \rceil /2 = 1 + 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i}$. Besides, $n2^{-i} = 2^{\nu_1 - i} + \cdots + 2^{-1} + 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i}$, and therefore $\lceil n2^{-i} \rceil = 1 + 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i} = \lceil \lceil n2^{-i+1} \rceil /2 \rceil$

- if $\nu_{\ell-1} > i - 1$, then $\lceil n2^{-i+1} \rceil = 1 + 2^{\nu_{\ell-1} - i + 1} + 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$ and $\lceil \lceil n2^{-i+1} \rceil /2 \rceil = 1 + 2^{\nu_{\ell-1} - i} + \cdots + 2^{\nu_w - i} = \lceil n2^{-i} \rceil$.

□ □

Algorithm 9.3.3 Span

Require: integer $n \geq 1$, vector $\mathbf{g} \in \mathbb{G}^n$
Ensure: vector $\tilde{\mathbf{g}} \in \mathbb{G}^{2^{\lceil \log n \rceil}}$
  if HammingWeight$(n) = 1$ **then** // n is a power of 2
    return g
  end if
  $i \leftarrow \text{Lsb}(n)$ // least significant bit of n counting from 1
  $I \leftarrow \left\{ \lceil n2^{-i} \rceil + j2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\}$
  while $i < \lfloor \log n \rfloor$ **do**
    $i \leftarrow i + 1$
    if $\lfloor (n \bmod 2^i)\, 2^{-i+1} \rfloor \bmod 2 = 0$ **then** // the ith bit of n is 0
      $I \leftarrow I \cup \left\{ \lceil n2^{-i} \rceil + j2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\}$
    end if
  end while
  $\tilde{\mathbf{g}} \leftarrow \mathbf{1}_{\mathbb{G}}^{2^{\lceil \log n \rceil}}$
  Head $\leftarrow 1$
  for $i = 1$ to $2^{\lceil \log n \rceil}$ **do**
    if $i \in I$ **then**
      continue // expand $\mathbf{g}$ by one element by inserting $1_{\mathbb{G}}$ at position i
    end if
    $\tilde{\mathbf{g}}[i] \leftarrow \mathbf{g}[\text{Head}]$
    Head $\leftarrow$ Head $+ 1$
  end for
  return $\tilde{\mathbf{g}}$

Now, to prove that the result of the folding procedure applied $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are g and h, notice that it suffices to show that for any integers $n \geq 2$ and x, and any vector $\mathbf{g} \in \mathbb{G}^n$,

$$\text{Span}\left(\lceil n/2 \rceil, \text{Fold}(n, x, \mathbf{g})\right) = \text{Fold}\left(2^{\lceil \log n \rceil}, x, \text{Span}(n, \mathbf{g})\right).$$

Indeed, in the case of the vector $\mathbf{g}$ (of size n, and recall that $n' \coloneqq \lceil \log n \rceil$) at the initial protocol step and of the last base g for instance, if the above statement is true, then

Algorithm 9.3.4 Fold

Require: integers $n \geq 2$ and x, vector $\mathbf{g} \in \mathbb{G}^n$
Ensure: vector $\tilde{\mathbf{g}} \in \mathbb{G}^{\lceil n/2 \rceil}$
  $\tilde{\mathbf{g}} \leftarrow \mathbf{g}_1^x \circ \mathbf{g}_2$
  return $\tilde{\mathbf{g}}$

$$\begin{aligned} g &= \textsc{Span}(1, g) \\ &= \textsc{Span}\left(1, \textsc{Fold}\left(2, x_{n'}, (\cdots \textsc{Fold}\left(n, x_1, \mathbf{g}\right) \cdots)\right)\right) \\ &= \textsc{Fold}\left(2, x_{n'}, \left(\cdots \textsc{Fold}\left(2^{n'}, x_1, \textsc{Span}\left(n, \mathbf{g}\right)\right) \cdots\right)\right) \\ &= \textsc{Fold}\left(2, x_{n'}, \left(\cdots \textsc{Fold}\left(2^{n'}, x_1, \tilde{\mathbf{g}}\right) \cdots\right)\right). \end{aligned}$$

To prove that $\textsc{Span}\left(\lceil n/2 \rceil, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}\left(2^{\lceil \log n \rceil}, x, \textsc{Span}(n, \mathbf{g})\right)$ for any integers $n \geq 2$ and x, and any vector $\mathbf{g} \in \mathbb{G}^n$, distinguish the following cases:

- if n is a power of 2, then $n/2$ also is and $\textsc{Span}\left(\lceil n/2 \rceil, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}(n, x, \mathbf{g}) = \textsc{Fold}\left(n, x, \textsc{Span}\left(n, \mathbf{g}\right)\right)$
- if n is even and not a power of 2, note that running $\textsc{Span}$ on $(n/2, \mathbf{g}_1^x \circ \mathbf{g}_2)$ results in a vector of size $2^{\lceil \log(n/2) \rceil}$ with $1_{\mathbb{G}}$ at positions in

  $$\left\{ \left\lceil n 2^{-1} 2^{-i} \right\rceil + j 2^{\lceil \log(n/2) \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\}$$

  for $\textsc{Lsb}(n/2) \leq i \leq \lfloor \log(n/2) \rfloor$, i.e., for integers i such that $\textsc{Lsb}(n) \leq i + 1 \leq \lfloor \log n \rfloor$.

  On the other hand,

  $$\begin{aligned} &\left\{ \left\lceil n 2^{-i} \right\rceil + j 2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\} \\ =&\left\{ \left\lceil n 2^{-1} 2^{-i+1} \right\rceil + j 2^{\lceil \log(n/2) \rceil - i + 2} : 0 \leq j < 2 \cdot 2^{(i-1)-1} \right\} \end{aligned}$$

  for any $\textsc{Lsb}(n) \leq i \leq \lfloor \log n \rfloor$ such that the ith bit of n is 0. Therefore, applying $\textsc{Fold}$ to $\left(2^{\lceil \log n \rceil}, \textsc{Span}\left(n, \mathbf{g}\right)\right)$ results in a vector of size $2^{\lceil \log n \rceil - 1} = 2^{\lceil \log(n/2) \rceil}$ with $1_{\mathbb{G}}$ inserted in $\mathbf{g}_1^x \circ \mathbf{g}_2$ at positions

  $$\left\{ \left\lceil n 2^{-1} 2^{-i} \right\rceil + j 2^{\lceil \log(n/2) \rceil - i + 1} : 0 \leq j < 2^{(i-1)-1} \right\}$$

  for $\textsc{Lsb}(n) \leq i \leq \lfloor \log n \rfloor$. A change of variables $i \leftarrow i - 1$ then allows to conclude that $\textsc{Span}\left(n/2, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}\left(n, x, \textsc{Span}\left(n, \mathbf{g}\right)\right)$
- if n is odd, write n as $2^0 + \cdots + 2^{\ell_1} + 2^{\nu_2} + \cdots + 2^{\nu_2 + \ell_2} + \cdots + 2^{\nu_k + \ell_k}$ for $k > 1$ and $0 \leq \ell_1 < \nu_2 - 1 < \cdots \leq \nu_2 + \ell_2 < \cdots < \nu_k - 1 < \cdots \leq \nu_k + \ell_k$. Then, $\lceil n/2 \rceil = 2^{\ell_1} + 2^{\nu_2 - 1} + \cdots + 2^{\nu_k + \ell_k - 1}$. One the one hand, running algorithm $\textsc{Span}$ on input $\lceil n/2 \rceil$ and $\mathbf{g}_1^x \circ \mathbf{g}_2 = \begin{bmatrix} g_1 & \cdots & g_{\lceil n/2 \rceil - 1} & 1_{\mathbb{G}} \end{bmatrix}^x \circ \begin{bmatrix} g_{\lceil n/2 \rceil} & \cdots & g_n \end{bmatrix}$ returns a vector of size $2^{\lceil \log \lceil n/2 \rceil \rceil}$ with $1_{\mathbb{G}}$ inserted at positions in

  $$\left\{ \left\lceil \lceil n/2 \rceil 2^{-i} \right\rceil + j 2^{\lceil \log \lceil n/2 \rceil \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\}$$

for $i = \ell_1 + 1, \ldots, \nu_2 - 1, \ldots, \ell_{k-1} + 1, \ldots, \nu_k - 1$.

On the other hand, running algorithm SPAN on $(n, \mathbf{g})$ inserts $1_{\mathbb{G}}$ in $\mathbf{g}$ at positions in $\left\{\lceil n2^{-i} \rceil + j2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1}\right\}$ for $i = 1, \ell_1 + 2, \ldots, \nu_2, \ldots, \ell_{k-1}+2, \ldots, \nu_k$. Consequently, running FOLD on $\left(2^{\lceil \log n \rceil}, x, \text{SPAN}(n, \mathbf{g})\right)$ results in a vector of size $2^{\lceil \log n \rceil - 1} = 2^{\lceil \log \lceil n/2 \rceil \rceil}$ with $1_{\mathbb{G}}$ inserted in $\begin{bmatrix} g_1 & \cdots & g_{\lceil n/2 \rceil - 1} & 1_{\mathbb{G}} \end{bmatrix}^x \circ \begin{bmatrix} g_{\lceil n/2 \rceil} & \cdots & g_n \end{bmatrix}$ at positions in

$$\left\{\left\lceil \left\lceil n2^{-i} \right\rceil 2^{-1} \right\rceil + j2^{\lceil \log n \rceil - (i-1)} : 0 \leq j < 2^{(i-1)-1}\right\}$$

for $i = \ell_1 + 2, \ldots, \nu_2, \ldots, \ell_{k-1} + 2, \ldots, \nu_k$. Moreover, Lemma 9.3.2 implies that $\lceil \lceil n2^{-i} \rceil 2^{-1} \rceil = \lceil n2^{-i-1} \rceil = \lceil \lceil n2^{-1} \rceil 2^{-i} \rceil$, and since $\lceil \log \lceil n/2 \rceil \rceil = \lceil \log n \rceil - 1$, the change of variables $i \leftarrow i - 1$ shows that

$$\text{SPAN}\left(\lceil n/2 \rceil, \text{FOLD}(n, x, \mathbf{g})\right) = \text{FOLD}\left(2^{\lceil \log n \rceil}, x, \text{SPAN}(n, \mathbf{g})\right).$$

□

Verification Efficiency.

In case n is a power of 2 (the previous paragraph shows that the problem can always be reduced to this case), the verifier then only has to check that

$$\left(\prod_{i=1}^{n} g_i^{\prod_{j \in S_i} x_j}\right)^{4x_{n'+1}a'} \left(\prod_{i=1}^{n} h_i^{\prod_{j \in [\![n]\!] \setminus S_i} x_j}\right)^{4x_{n'+1}b'} e^{4a'b'} f^{4u}$$
$$= \left(U_{n'}^{x_{n'}} \prod_{i=1}^{n'-1} U_i^{x_i x_{i+1} \cdots x_{n'}} C^{x_1 \cdots x_{n'}} \prod_{i=1}^{n'-1} V_i^{x_{i+1} \cdots x_{n'}} V_{n'}\right)^{2x_{n'+1}^2} \Gamma^{2x_{n'+1}} \Delta^2,$$

i.e., do a $(2n + 2n' + 5)$-exponentiation with exponents (in absolute value) less than

$$4 \max\left(2^{\ell} P^{2n'+1}, \overbrace{2^{2\ell} P^{2n'}}^{|a'b'|<}, \overbrace{\left(2n' 2^{b_{\mathbb{G}}+\lambda} P^{n'+1} + 2^{\ell}(P-1)^{n'+2}\right)\left(1 + 2^{\lambda}\right)}^{|u|<}\right).$$

Verification thus requires $O(\ell + b_{\mathbb{G}} + \log(n)\log(P))$ group operations $(n \geq 2)$.

9.3.3 Completeness and Security

This section proves that Π is complete, honest-verifier zero-knowledge and extractable. The proof of extractability of Π is based on the proof of extractability of Π' and on Lemma 9.3.8. The proof of extractability of Π' is itself based on Lemma 9.3.8 and Lemma 9.3.7, and Lemma 9.3.7 relies on Lemma 9.3.6.

Theorem 9.3.4 (Completeness)**.** Π *is complete.*

Proof. The completeness of the protocol immediately follows from the completeness of protocol $\Pi^{\mathcal{H}}$, from the fact that each step $i \in [\![n']\!]$,

$$\begin{aligned}
&\left((\mathbf{g}_1^{x_i} \circ \mathbf{g}_2)^{\mathbf{a}'} (\mathbf{h}_1 \circ \mathbf{h}_2^{x_i})^{\mathbf{b}'} e^{\langle \mathbf{a}', \mathbf{b}' \rangle} f^t\right)^4 \\
&= (\mathbf{g}_1^{x_i} \circ \mathbf{g}_2)^{4(\mathbf{a}_1 + x_i \mathbf{a}_2)} (\mathbf{h}_1 \circ \mathbf{h}_2^{x_i})^{4(x_i \mathbf{b}_1 + \mathbf{b}_2)} e^{4\left(x^2 \langle \mathbf{a}_2, \mathbf{b}_1 \rangle + x \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle\right)} \\
&\quad f^{4(s_v + r x_i + s_u x_i^2)} \\
&= \mathbf{g}_1{}^{4x_i(\mathbf{a}_1 + x_i \mathbf{a}_2)} \mathbf{g}_2{}^{4(\mathbf{a}_1 + x_i \mathbf{a}_2)} \mathbf{h}_1{}^{4(x_i \mathbf{b}_1 + \mathbf{b}_2)} \mathbf{h}_2{}^{4x_i(x_i \mathbf{b}_1 + \mathbf{b}_2)} e^{4\left(x^2 \langle \mathbf{a}_2, \mathbf{b}_1 \rangle + x \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle\right)} \\
&\quad f^{4(s_v + r x_i + s_u x_i^2)} \\
&= \left(U_i^{x_i^2} C_i^{x_i} V_i\right)^2,
\end{aligned}$$

and from the completeness of the last step of the protocol (i.e., $i = n' + 1$). The second equality stem from the fact that for any two vectors $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n$ and any vector $\mathbf{k} \in \mathbb{Z}^n$, $(\mathbf{G} \circ \mathbf{H})^{\mathbf{k}} = \mathbf{G}^{\mathbf{k}} \mathbf{F}^{\mathbf{k}}$. □

Theorem 9.3.5 (Honest-Verifier Zero-Knowledge)**.** *If* $\tilde{\Pi}$ *is* $\left(T_{\tilde{\Pi}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}_{\tilde{\Pi}}\right)$*-sound, then protocol* Π *is* $\left(T_{\tilde{\Pi}}, O\left(\left(\ell + b_{\mathbb{G}} + \log(P)\log(n)\right) T_{\mathbb{G}}\right), q_{\mathcal{H}}, 2^{-\lambda+2} + \varepsilon^{\mathrm{snd}}_{\tilde{\Pi}}\right)$*-honest-verifier zero-knowledge.*

Proof. If the initial length n of the vectors is at least 2, note that at each step $i \in [\![n']\!]$, the length of the vectors is $n_i \coloneqq \lceil n 2^{-i+1} \rceil$, so $\lceil \log n_i \rceil + i - 1 = n'$. Since $e^2 \in \langle f^2 \rangle$ (the verifier is honest), and since the group elements in the parameters are also in $\langle f^2 \rangle$ unless the adversary can contradict the soundness of $\tilde{\Pi}$, the pair (U_i, V_i) is at a $2^{-\lambda+1}/n'$ statistical distance from a pair (f^{2s_u}, f^{2s_v}) for $s_u, s_v \leftarrow_{\$} [\![0; 2n' 2^{b_{\mathbb{G}}+\lambda}]\!]$, and the tuple $(U_i, V_i)_{i=1}^{n'}$ is thus at a statistical distance of at most $2^{-\lambda+1}$ from a $2n'$-tuple with components generated as f^{2s} for $s \leftarrow_{\$} [\![0; 2n' 2^{b_{\mathbb{G}}+\lambda}]\!]$.

At the last step of the protocol (i.e., for $i = n' + 1$), note that $|a|, |b| \leq 2^{\ell} P^{n'+1}$ and that the absolute value of the randomness r in the commitment $C_{n'+1}$ is less than $2n' 2^{b_{\mathbb{G}}+\lambda} P^{n'+1} + 2^{\ell} (P-1)^{n'}$. Indeed, if $n = 1$ the statement is immediate and if $n \geq 2$, at the first step of the protocol $i = 1$, $\max \mathbf{a}' = \max \mathbf{a}_1 + x_1 \max \mathbf{a}_2 \leq 2^{\ell} P$. If $\min \mathbf{a}_2 < 0$, then $\min \mathbf{a}' = \min \mathbf{a}_1 + x_1 \min \mathbf{a}_2 \geq -2^{\ell} P$, and if $\min \mathbf{a}_2 \geq 0$, then $\min \mathbf{a}' = \min \mathbf{a}_1 \geq -2^{\ell} \geq -2^{\ell} P$; and similarly for $\mathbf{b}'$. A simple induction then shows that $|a|, |b| \leq 2^{\ell} P^{n'+1}$. Besides, the randomness of the commitment C_1 is at most 2^{ℓ} in absolute value, and under the assumption that the randomness in C_i is at most $2n' 2^{b_{\mathbb{G}}+\lambda} P^i + 2^{\ell} (P-1)^{i-1}$ in absolute value for $i \in [\![n']\!]$,

the randomness in C_{i+1} is less than

$$\begin{aligned}
&2n'2^{b_{\mathbb{G}}+\lambda} + \left(2n'2^{b_{\mathbb{G}}+\lambda}P^i + 2^\ell\,(P-1)^{i-1}\right)(P-1) + 2n'2^{b_{\mathbb{G}}+\lambda}(P-1)^2 \\
&= 2n'2^{b_{\mathbb{G}}+\lambda}\left(1 + P^i(P-1) + (P-1)^2\right) + 2^\ell\,(P-1)^i \\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}\left(P^i(P-1) + P^2\right) + 2^\ell\,(P-1)^i \\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}P^{i+1} + 2^\ell\,(P-1)^i\,.
\end{aligned}$$

Moreover, it is also greater than $-2^\ell(P-1)^i$ since the randomness s_u and s_v are always positive. The statement then holds for all $i \in [\![n'+1]\!]$.

It follows that at the last step of the protocol, (assuming $n' \geq 1$,)

$$\begin{aligned}
sx_{n'+1} + rx_{n'+1}^2 &\leq 2n'2^{b_{\mathbb{G}}+\lambda} + \left(2n'2^{b_{\mathbb{G}}+\lambda}P^{n'+1} + 2^\ell\,(P-1)^{n'}\right)(P-1)^2 \\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}\left(1 + P^{n'+1}(P-1)^2\right) + 2^\ell(P-1)^{n'+2} \\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}P^{n'+3} + 2^\ell(P-1)^{n'+2}.
\end{aligned}$$

Consequently, for $\alpha, \beta \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda}P^{n'+1}\right]\!\right]$, $s \leftarrow_{\$} [\![0; 2\max(n',1)2^{b_{\mathbb{G}}+\lambda}]\!]$ and $t \leftarrow_{\$} [\![0;\ 2\max(n',1)2^{b_{\mathbb{G}}+2\lambda}P^{n'+3} + 2^{\ell+\lambda}\,(P-1)^{n'+2}]\!]$, the distribution of

$$\left(\overbrace{\left(g^\alpha h^\beta e^{\alpha b+a\beta} f^s\right)^2}^{\Gamma}, \overbrace{\left(g^{x_{n'+1}a'} h^{x_{n'+1}b'} e^{a'b'} f^u\right)^2 C_{n'+1}^{-x_{n'+1}^2}\Gamma^{-x_{n'+1}}}^{\Delta=\left(e^{\alpha\beta} f^t\right)^2}, x_{n'+1}, \overbrace{\alpha + ax_{n'+1}}^{a'}, \overbrace{\beta + bx_{n'+1}}^{b'}, \overbrace{t + sx_{n'+1} + rx_{n'+1}^2}^{u}\right)$$

is at a statistical distance of at most $2^{-\lambda}$ from the distribution of

$$\left(\left(g^\alpha h^\beta e^{\alpha b+a\beta} f^s\right)^2, \left(g^{x_{n'+1}\alpha} h^{x_{n'+1}\beta} e^{\alpha\beta} f^t\right)^2 C_{n'+1}^{-x_{n'+1}^2}\Gamma^{-x_{n'+1}}, x_{n'+1}, \alpha, \beta, t\right),$$

which is itself at a statistical distance of at most $2^{-\lambda}$ from the distribution of

$$\left(f^{2s}, \left(g^{x_{n'+1}\alpha} h^{x_{n'+1}\beta} e^{\alpha\beta} f^t\right)^2 C_{n'+1}^{-x_{n'+1}^2} f^{-2sx_{n'+1}}, x_{n'+1}, \alpha, \beta, t\right).$$

Recall also that

$$C_{n'+1} = U_{n'}^{x_{n'}^2} \prod_{i=1}^{n'-1} U_i^{x_i^2 x_{i+1}\cdots x_{n'}} C^{x_1\cdots x_{n'}} \prod_{i=1}^{n'-1} V_i^{x_{i+1}\cdots x_{n'}} V_{n'}.$$

Therefore, the transcript $\left((U_i, V_i, x_i)_{i=1}^{n'}, \Gamma, \Delta, x_{n'+1}, a', b', u\right)$ of an honest protocol execution is at a statistical distance of at most $2^{-\lambda+2}$ from a tuple

$$\left((U_i, V_i, x_i)_{i=1}^{n'}, f^{2s}, \left(g^{x_{n'+1}\alpha} h^{x_{n'+1}\beta} e^{\alpha\beta} f^t\right)^2 C_{n'+1}^{-x_i^2} f^{-2sx_{n'+1}}, x_{n'+1}, \alpha, \beta, t\right)$$

with, for all $i \in [\![n'+1]\!]$, $x_i \leftarrow_{\$} [\![0; P-1]\!]$, $s_{u,i}, s_{v,i} \leftarrow_{\$} [\![0; 2n' 2^{b_{\mathbb{G}}+\lambda}]\!]$, $U_i \leftarrow g^{s_{u,i}}$, $V_i \leftarrow g^{s_{v,i}}$, $\alpha, \beta \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda} P^{n'+1}\right]\!\right]$, $s \leftarrow_{\$} [\![0; 2\max(n',1) 2^{b_{\mathbb{G}}+\lambda}]\!]$, $t \leftarrow_{\$} \left[\!\left[0; 2\max(n',1) 2^{b_{\mathbb{G}}+2\lambda} P^{n'+3} + 2^{\ell+\lambda} (P-1)^{n'+2}\right]\!\right]$ and $C_{n'+1}$ computed as above. Note the latter distribution is independent of the witness. Besides, each U_i and V_i can be computed in $O(b_{\mathbb{G}}+\lambda+\log\log n)$ group operations and the group element $\left(g^{x_{n'+1}\alpha} h^{x_{n'+1}\beta} e^{\alpha\beta} f^t\right)^2 C_{n'+1}^{-x_i^2} f^{-2sx_{n'+1}}$ can be computed in

$$O\left(\ell + b_{\mathbb{G}} + \lambda + \log(P)\log(n) + \log\log n\right) = O(\ell + b_{\mathbb{G}} + \log(P)\log(n))$$

group operations ($\log P \leq b_{\mathbb{G}} = \Omega(\lambda)$). A simulated transcript can thus be computed in time $O\left((\ell + b_{\mathbb{G}} + \log(P)\log(n))\, T_{\mathbb{G}}\right)$, hence the theorem. □

Lemma 9.3.6. *Let n be a natural integer and let $a_0, \ldots, a_n, b$ and N be integers, with $N \geq 1$. Assuming that the a_i integers are not all nil modulo N, the number of tuples $(x_0, \ldots, x_n) \in \mathbb{Z}_N^{n+1}$ such that $a_0x_0 + \cdots + a_nx_n + b = 0 \bmod N$ is either 0 or $N^n \gcd(a_0, \ldots, a_n, N)$.*

Proof. The lemma can be proved by induction on n as follows. For $n = 0$, note that the equation $a_0x_0 + b = 0 \bmod N$ has no solution if $\gcd(a_0, N) \nmid b$. If $\gcd(a_0, N) \mid b$, let a_0' be such that $a_0 = \gcd(a_0, N)a_0'$, and define b' and N' similarly. Integers a_0' and N' are then coprime, and let u and v be integers such that $a_0'u + N'v = 1$. Consider now the equation $a_0'x_0 + b' = 0 \bmod N'$. Multiplying it by u shows that $x_0 = -b'u \bmod N'$. It follows that the integers x_0 solutions to the equation $a_0'x_0 + b' = 0 \bmod N'$ are of the form $-b'u + kN'$ for $k \in \mathbb{Z}$. Besides, for two integers k_0 and k_1, $-b'u + k_0N' = -b'u + k_1N' \bmod N$ if and only if $\gcd(a_0, N) \mid k_1 - k_0$. Therefore, if $\gcd(a_0, N) \mid b$, the solutions x_0 to the equation $a_0x_0 + b = 0 \bmod N$ are $-b'u + kN'$ for $0 \leq k \leq \gcd(a_0, N)$.

Now suppose the statement to be true for $n \geq 0$ and consider the equation $a_0x_0 + \cdots + a_{n+1}x_{n+1} + b = 0 \bmod N$ for some integers $a_0, \ldots, a_{n+1}, b$ and $N \geq 1$. For a fixed value $x_{n+1} \in \mathbb{Z}_N$, as in the case $n = 0$, there is no solution if $\gcd(a_0, \ldots, a_n, N) \nmid a_{n+1}x_{n+1} + b$; and if $\gcd(a_0, \ldots, a_n, N) \mid a_{n+1}x_{n+1} + b$, then the induction hypothesis implies that there are $N^n \gcd(a_0, \ldots, a_n, N)$ solutions in $\mathbb{Z}_N$. It now remains to determine the number of values $x_{n+1} \in \mathbb{Z}_N$ such that $a_{n+1}x_{n+1} + b = 0 \bmod \gcd(a_0, \ldots, a_n, N)$. To this end, let u_{n+1} and v_{n+1} be integers such that $a_{n+1}u_{n+1} + \gcd(a_0, \ldots, a_n, N)v_{n+1} = \gcd(a_0, \ldots, a_{n+1}, N)$. As in the case $n = 0$, there is no solution if $\gcd(a_0, \ldots,$

$a_{n+1}, N) \nmid b$, and if $\gcd(a_0, \ldots, a_{n+1}, N) \mid b$, then the solutions to the equation $a_{n+1}x_{n+1} + b = 0 \bmod \gcd(a_0, \ldots, a_n, N)$ are exactly $x_{n+1} = -bu_{n+1} + k_{n+1}d'$ for $0 \leq k_{n+1} < \gcd(a_0, \ldots, a_{n+1}, N)$ and $d' \in \mathbb{Z}$ such that $\gcd(a_0, \ldots, a_n, N) = \gcd(a_0, \ldots, a_{n+1}, N)d'$. The values $x_{n+1} \in \mathbb{Z}_N$ for which the equation $a_0x_0 + \cdots + a_{n+1}x_{n+1} + b = 0 \bmod N$ is satisfied in case $\gcd(a_0, \ldots, a_{n+1}, N) \mid b$ are then exactly $-bu_{n+1} + k_{n+1}d' + k\gcd(a_0, \ldots, a_n, N)$ for $0 \leq k_{n+1} < \gcd(a_0, \ldots, a_{n+1}, N)$ and $0 \leq k < N'$, where $N' \in \mathbb{Z}$ is such that $N = \gcd(a_0, \ldots, a_n, N)N'$. Therefore, the number of solutions to the equation $a_0x_0 + \cdots + a_{n+1}x_{n+1} + b = 0 \bmod N$ is either 0 or $N^n \gcd(a_0, \ldots, a_n, N) \gcd(a_0, \ldots, a_{n+1}, N)N' = N^{n+1}\gcd(a_0, \ldots, a_{n+1}, N)$. The statement of the lemma is then true for all $n \in \mathbb{N}$. □

Lemma 9.3.7. *Consider the problem (depending on λ) of computing, on input $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and $f \leftarrow_\$ \mathbb{G}$ and $(f^{x_i})_{i=0}^n$ (for integers $x_i \leftarrow_\$ [\![0; 2^{2b_\mathbb{G}+\lambda}(n+1)]\!]$) an element $C \in \mathbb{G}$ and integers $a_0, \ldots, a_n, b, \delta$ such that $1 < |\delta| < P$, δ does not divide b or at least one of the a_i integers, and $C^\delta = f_0^{a_0} \cdots f_n^{a_n} f^b$.*

Under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, the probability that any probabilistic algorithm running in time T solves this problem is at most $\left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$, if T is such that $(n+1)\max(\log(n+1), 1)\log(P)b_\mathbb{G}TT_\mathbb{G} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$.

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the lemma and assume without loss of generality that $\delta > 0$ (if $\delta < 0$, raise the equality to the power -1). The equality $C^\delta = f_0^{a_0} \cdots f_n^{a_n} f^b$ implies that $C^\delta = f^{\sum_i a_i x_i + b}$. The goal is to show that in case δ does not divide $\sum_i a_i x_i + b$, algorithm $\mathcal{A}$ can be used to violate the assumptions on generator G; and to show that conditioned on the event in which $\mathcal{A}$ solves the problem, the probability that δ divides $\sum_i a_i x_i + b$ is at most $1/2 + 2^{-\lambda} + (1-\mu)$.

More precisely, if δ does not divide $\sum_i a_i x_i + b$, let $d \coloneqq \gcd\left(\delta, \sum_i a_i x_i + b\right)$ and $u, v \in \mathbb{Z}$ such that $d = u\delta + v\left(\sum_i a_i x_i + b\right)$. Then, $f^d = \left(f^u C^v\right)^\delta$, i.e., $\left(\left(f^u C^v\right)^{\delta/d} f^{-1}\right)^d = 1_\mathbb{G}$. Since $1 \leq d < \delta < P$ by assumption, the small-order assumption over G implies that the element $\tilde{g} \coloneqq \left(f^u C^v\right)^{\delta/d} f^{-1}$ is such that $\tilde{g}^2 = 1_\mathbb{G}$ with probability at least $\varepsilon^{\mathrm{ord}}$. If $\tilde{g} = 1_\mathbb{G}$ and $d > 1$, then $\left(\left(f^u C^v\right)^{\delta/d}, d\right)$ is a solution to the strong-root problem. Otherwise,

- if δ/d is odd, then $\tilde{g}^{\delta/d} = \tilde{g}$ and therefore, $(f^u C^v \tilde{g}, \delta/d)$ is a solution to the strong-root problem
- if δ/d is even, then the low-dyadic-valuation assumption on orders implies that $\mathrm{ord}\left(\left(f^u C^v\right)^{\delta/d}\right)$ is odd, which is impossible if $\mathrm{ord}(f)$ is P-rough (and thus odd) since $\mathrm{ord}(f\tilde{g}) = 2\,\mathrm{ord}(f)$ in this case.

Consequently, δ does not divide $\sum_i a_i x_i + b$ with probability at most $\varepsilon^{\text{ord}} + \varepsilon^{\text{strg}} + 1 - \mu$.

Since $|a_i|, |b| \leq 2^{O(T)}$, $\sum_i a_i x_i + b$ can be computed in time $O\left((n+1)T(b_{\mathbb{G}} + \log(n+1))\right)$. Then, u and v can be computed in time $O((T + b_{\mathbb{G}} + \log(n+1)) \log P)$ with the extended Euclidean algorithm as $|\sum_i a_i x_i + b| \leq n(n+1)2^{O(T)}2^{2b_{\mathbb{G}}+\lambda} + 2^{O(T)}$ and $|\delta| \leq P$; and u and v are such that $|u|, |v| \leq \max\left(|\delta|, |\sum_i a_i x_i + b|\right)/d$. Besides, computing δ/d can be done in time $O\left(\log^2 P\right)$ and then $f^u C^v \tilde{g}$ in

$$O\left(\max\left(T + b_{\mathbb{G}} + \log(n+1), \log P\right)\right) = O\left(T + b_{\mathbb{G}} + \log(n+1)\right)$$

group operations since $P \leq 2^{b_{\mathbb{G}}}$. The solution to the strong-root problem can thus be computed in time

$$O\left((n+1)(b_{\mathbb{G}} + \log(n+1))T + (T + b_{\mathbb{G}} + \log(n+1))\log(P)T_{\mathbb{G}}\right),$$

after the bases $f_0, \ldots, f_n$ have been computed in $O((n+1)\max(\log(n+1), 1)b_{\mathbb{G}})$ group operations.

It remains to show that δ divides $\sum_i a_i x_i + b$ with probability at most $1/2 + 2^{-\lambda} + 1 - \mu$ conditioned on the event in which $\mathcal{A}$ solves the problem. To do so, consider the event in which it occurs. Let p and j respectively be a prime and a positive integer such that p^j divides δ and p^j does not divide b or at least one of the a_i integers. Such p and j necessarily exist for an assumption of the lemma is that δ does not divide b or at least one of the a_i integers. Note that p^j cannot divide all the a_i integers as it would otherwise divide b as well, since it divides $\sum_i a_i x_i + b$. Moreover, if μ-assumption that there are many rough-order elements in the groups generated by G holds, p does not divide $\text{ord}(f)$. Therefore, if the μ-assumption holds, p^j does not divide $a_i \,\text{ord}(f)$ for some $i \in [\![0; n]\!]$.

For $i \in [\![0; n]\!]$, let $0 \leq \rho_i < \text{ord}(f)$ be the unique integer such that $x_i = \text{ord}(f)\lfloor x_i / \text{ord}(f) \rfloor + \rho_i$, and note that $f^{x_i} = f^{\rho_i}$. Then, $\sum_i a_i x_i + b = \sum_i a_i \,\text{ord}(f) \lfloor x_i / \text{ord}(f) \rfloor + \sum_i a_i \rho_i + b = 0 \bmod p^j$ and $a_i \,\text{ord}(f) \neq 0 \bmod p^j$ for some $i \in [\![0; n]\!]$. Lemma 9.3.6 shows that the equation $\sum_i A_i X_i + B = 0 \bmod p^j$ with $A_i \coloneqq a_i \,\text{ord}(f)$ and $B \coloneqq \sum_i a_i \rho_i + b$ has at most $p^{jn} \gcd\left(a_0 \,\text{ord}(f), \ldots, a_n \,\text{ord}(f), p^j\right)$ solutions, and the integer $\gcd(a_0 \,\text{ord}(f), \ldots, a_n \,\text{ord}(f), p^j)$ is at most p^{j-1} since $a_i \,\text{ord}(f) \neq 0 \bmod p^j$ for some $i \in [\![0; n]\!]$. However, the variables $X_i \coloneqq \lfloor x_i / \text{ord}(f) \rfloor$ are identically distributed and independent of the values returned by $\mathcal{A}\left(\mathbb{G}, P, f, f^{\rho_0}, \ldots, f^{\rho_n}\right)$; and their distribution is at a statistical distance of at most $\text{ord}(f)2^{-2b_{\mathbb{G}}-\lambda}(n+1)^{-1} \leq 2^{-b_{\mathbb{G}}-\lambda}(n+1)^{-1}$ from the uniform distribution over

$$\left[\!\left[0; \left\lfloor (n+1)2^{2b_{\mathbb{G}}+\lambda} / \,\text{ord}(f) \right\rfloor\right]\!\right] \supseteq \left[\!\left[0; (n+1)2^{b_{\mathbb{G}}+\lambda}\right]\!\right].$$

Besides, if a variable X is uniformly distributed over the set $[\![0; (n+1)2^{b_{\mathbb{G}}+\lambda}]\!]$, then the distribution of $X \bmod p^j$ is at a statistical of at most $p^j 2^{-b_{\mathbb{G}}-\lambda}(n+$

$1)^{-1} \leq (P-1)2^{-b_{\mathbb{G}}-\lambda}(n+1)^{-1}$ from the uniform distribution over $\mathbb{Z}_{p^j}$. The distribution of the random vector $(X_0 \bmod p^j, \ldots, X_n \bmod p^j)$ is then at a statistical of at most $P2^{-b_{\mathbb{G}}-\lambda} \leq 2^{-\lambda}$ from the uniform distribution over $\mathbb{Z}_{p^j}^{n+1}$. Consequently, the equation $\sum_i a_i x_i + b = 0 \bmod p^j$ can then be satisfied with probability at most $2^{-\lambda} + p^{j(n+1)-1}/\left(p^j\right)^{n+1} \leq 1/2 + 2^{-\lambda}$ and thus, δ divides $\sum_i a_i x_i + b$ with probability at most $1/2 + 2^{-\lambda} + 1 - \mu$.

In summary, denoting by ε the probability that $\mathcal{A}$ solves the problem of the statement of the lemma, $\varepsilon \leq \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu + \left(1/2 + 2^{-\lambda} + 1 - \mu\right)\varepsilon$, which is equivalent to $\varepsilon \leq \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. □

Lemma 9.3.8 (Discrete-Logarithm Relations)**.** *Let n be a non-negative integer. Consider the problem (depending on λ) of computing, on the input of $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^{\lambda}\right)$ and of group elements $f \leftarrow_{\$} \mathbb{G}$ and $(f^{x_i})_{i=0}^{n}$ (for $x_i \leftarrow_{\$} [\![0; 2^{2b_{\mathbb{G}}+\lambda}\ (n+1)]\!]$), integers $a_0, \ldots, a_n$, b such that $f_0^{a_0} \cdots f_n^{a_n} f^b = 1_{\mathbb{G}}$ although at least one of $a_0, \ldots, a_n, b$ is non-zero. Under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, the probability that any probabilistic algorithm running in time at most T solves this problem is at most*

$$\varepsilon^{\mathrm{strg}} + \max\left(2^{-b_{\mathbb{G}}-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right)$$

if T is such that $(n+1)\max(\log(n+1), 1)\log(P) b_{\mathbb{G}} T T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$.

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the lemma and denote the probability that it solves the problem by ε. If $a_0 = \cdots = a_n = 0$, then $b \neq 0$ by assumption and Lemma 9.2.6 shows that since $f^b = 1_{\mathbb{G}}$, there exists an algorithm that solves the strong-root problem in time at most $T + O(\log b)$ with probability at least ε, and since $b = 2^{O(T)}$, $\varepsilon \leq \varepsilon^{\mathrm{strg}}$. Now turn to the case in which $a_i \neq 0$ for some $i \in [\![0; n]\!]$. If $n = 0$, then $f^{a_0 x_0 + b} = 1_{\mathbb{G}}$ by assumption. Writing x_0 as $x_0 = \mathrm{ord}(f)\lfloor x_0/\mathrm{ord}(f)\rfloor + \rho_0$ for $0 \leq \rho_0 < \mathrm{ord}(f)$, the random variable $X_0 \coloneqq \lfloor x_0/\mathrm{ord}(f)\rfloor$ is independent of the values returned by $\mathcal{A}(\mathbb{G}, P, f, f^{\rho_0})$, and is at a statistical distance of at most $\mathrm{ord}(f)2^{-2b_{\mathbb{G}}-\lambda} \leq 2^{-b_{\mathbb{G}}-\lambda}$ from the uniform distribution over $\left[\!\left[0; \left\lfloor 2^{2b_{\mathbb{G}}+\lambda}/\mathrm{ord}(f)\right\rfloor\right]\!\right] \supseteq [\![0; 2^{b_{\mathbb{G}}+\lambda}]\!]$. However, for $A_0 \coloneqq a_0\,\mathrm{ord}(f)$ and $B \coloneqq a_0\rho_0 + b$, the equation $A_0 X_0 + B = 0$ in $\mathbb{Z}$ has no solution if $A_0 \nmid B$ and exactly one otherwise. Therefore, the probability that $a_0 x_0 + b = 0$ in $\mathbb{Z}$ is at most $2^{-b_{\mathbb{G}}-\lambda+1}$, and there exists an algorithm that solves the strong-root problem in time at most $O(T)$ with probability at least $\varepsilon - 2^{-b-\mathbb{G}-\lambda+1}$, so $\varepsilon \leq \varepsilon^{\mathrm{strg}} + 2^{-b_{\mathbb{G}}-\lambda+1}$.

If $n > 0$, it suffices to prove that the probability that $f_0^{a_0} \cdots f_n^{a_n} f^b = 1_{\mathbb{G}}$ and $\sum_i a_i x_i + b = 0$ is at most $\left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. Then, in case $f^{\sum_i a_i x_i + b} = 1_{\mathbb{G}}$ and $\sum_i a_i x_i + b \neq 0$, Lemma 9.2.6 shows that this probability is at most $\varepsilon^{\mathrm{strg}}$. This then would imply that

$$\varepsilon \leq \varepsilon^{\mathrm{strg}} + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right).$$

Suppose that $\sum_i a_i x_i + b = 0$ (which implies that $f_0^{a_0} \cdots f_n^{a_n} f^b = 1_{\mathbb{G}}$). Let $d \coloneqq \gcd(a_0, \ldots, a_n)$ and note that d necessarily divides b. Besides, $\sum_i a_i x_i + b = 0$ if and only if $\sum_i (a_i/d) x_i + (b/d) = 0$ and therefore, $f_0^{a_0/d} \cdots f_n^{a_n/d} f^{b/d} = 1_{\mathbb{G}}$ with $\gcd(a_0/d, \ldots, a_n/d) = 1$. However, $1_{\mathbb{G}}^2 = 1_{\mathbb{G}} = f_0^{a_0/d} \cdots f_n^{a_n/d} f^{b/d}$ although the integers a_i/d cannot all be even as they are coprime. Lemma 9.3.7 then implies that $\sum_i a_i x_i + b = 0$ with probability at most $\left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. □

Theorem 9.3.9 (Extractability of Π')**.** *Under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, protocol Π' (with honest CRS generation) is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon^{\mathrm{ext}}, \Sigma')$-extractable for any $T_{\mathcal{A}}$ and T_{Prove^*} such that $T_{\mathcal{A}} + n \log(n+1) \log(P) b_{\mathbb{G}} T_{\mathsf{Prove}^*} T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$, with*

$$T_{\mathcal{E}} = O\left(n b_{\mathbb{G}} T_{\mathbb{G}} + n^{\log 5 + \log \alpha} \log(n+1) \log(P) T_{\mathsf{Prove}^*} / \varepsilon\right)$$

for any $\alpha > (1 - 5/P)^{-2}$ assuming that $\varepsilon_{\mathcal{A},\mathsf{Prove}^*} \geq 5n^{\log \alpha}/((\alpha - 1)P)$, and with

$$\begin{aligned}\varepsilon^{\mathrm{ext}} &= \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} \\ &\quad + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right) \\ &\quad + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right).\end{aligned}$$

Proof. To prove that the system satisfies witness-extended emulation, it suffices to show that at the $(n'+1)$th step of the protocol, a witness for commitment $C_{n'+1}$ can be extracted by rewinding the prover, and then that for i from $n'+1$ down to 2, given a witness for C_i, a witness for C_{i-1} can again be extracted by rewinding the prover sufficiently many times. One a witness for the initial commitment C_1 is extracted, it is then possible to generate a transcript with the same distribution as in the interaction with the honest verifier since the protocol is public coin.

Henceforward, and until the end of this proof, the underscript i indicating the current step of the protocol will be omitted as the specific step should always be clear from the context. The underscripts in what follows

rather indicate several challenges at the *same* protocol step. It might be convenient for the reader to think of these latter as underscripts j; for instance, at the ith step of the protocol, x_1, x_2 and x_3 are actually challenges $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$, i.e., $x_{i,j}$ for $j = 1, 2, 3$.

Now consider the case $i = n' + 1$ (recall that $n' \coloneqq \lceil \log n \rceil$). Given five transcripts $\left(\Gamma, \Delta, x_j, a'_j, b'_j, u_j\right)_{j=1}^{5}$ such that $\left(g^{x_j a'_j} h^{x_j b'_j} e^{a'_j b'_j} f^{u_j}\right)^4 = \left(C^{x_j^2} \Gamma^{x_j} \Delta\right)^2$ for all $j \in [\![5]\!]$, the goal is to extract a representation of C in the bases g, h, e and f. To do so, consider the linear system

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

with unknowns ν_1, ν_2 and ν_3. Denote by $\mathbf{X}$ the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix}$. It is a Vandermonde matrix, and its determinant is thus $(x_3 - x_2)(x_3 - x_1)(x_2 - x_1)$. Even if $\det \mathbf{X} \neq 0$, this system may not have a solution in $\mathbb{Z}$, but it does in $\mathbb{Q}$. Moreover, Cramer's rule implies that

$$\nu_1 \det \mathbf{X} = \det \begin{bmatrix} 0 & 1 & 1 \\ 0 & x_2 & x_3 \\ 1 & x_2^2 & x_3^2 \end{bmatrix}, \ \nu_2 \det \mathbf{X} = \det \begin{bmatrix} 1 & 0 & 1 \\ x_1 & 0 & x_3 \\ x_1^2 & 1 & x_3^2 \end{bmatrix} \text{ and}$$

$$\nu_3 \det \mathbf{X} = \det \begin{bmatrix} 1 & 1 & 0 \\ x_1 & x_2 & 0 \\ x_1^2 & x_2^2 & 1 \end{bmatrix}.$$

It follows that

$$\nu_1 \overbrace{(x_3 - x_1)(x_2 - x_1)}^{\delta_1}(x_3 - x_2) = \overbrace{1}^{\gamma_1}(x_3 - x_2)$$

$$\nu_2 \overbrace{(x_3 - x_2)(x_1 - x_2)}^{\delta_2}(x_3 - x_1) = \overbrace{1}^{\gamma_2}(x_3 - x_1)$$

$$\nu_3 \overbrace{(x_2 - x_3)(x_1 - x_3)}^{\delta_3}(x_2 - x_1) = \overbrace{1}^{\gamma_3}(x_2 - x_1)$$

and that

$$\det \mathbf{X} \sum_j \delta^j \gamma_j x_j^2 = \delta \det \mathbf{X}, \ \sum_j \delta^j \gamma_j x_j = 0 \ \text{ and } \ \sum_j \delta^j \gamma_j = 0$$

for $\delta \coloneqq \delta_1 \delta_2 \delta_3$ and $\delta^j \coloneqq \delta / \delta_j$ for $j \in [\![3]\!]$; and δ, δ^j and γ_j are in $\mathbb{Z}$ for all $j \in [\![3]\!]$. Besides, note that $|\delta| \leq P^3 \leq P'$. In other words, the linear system

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \delta \end{bmatrix}$$

has a (unique) solution in $\mathbb{Z}$ if $\det \mathbf{X} \neq 0$, and it is $\begin{bmatrix} \delta^1\gamma_1 \\ \delta^2\gamma_2 \\ \delta^3\gamma_3 \end{bmatrix}$. Therefore, assuming x_1, x_2 and x_3 to be pairwise distinct, one can extract, via linear combinations of the responses, integers a_C, b_C, z_C and r_C such that $C^{2\delta} = \left(g^{a_C} h^{b_C} e^{z_C} f^{r_C}\right)^4$, and $\delta \neq 0$; and the bit length of the linear coefficients is of order $O(\log P)$. However, g can be expressed in terms of $g_1, \ldots, g_n$ and $x_1, \ldots, x_n$, and the exponent of g_n in the expression of g is 1. Similarly, h can be expressed in terms of $h_1, \ldots, h_n$ and $x_1, \ldots, x_n$, and the exponent of h_1 in the expression of h is 1. Lemma 9.3.7 then implies that if $|\delta| > 1$, 2δ divides $4a_C$, $4b_C$, $4z_C$ and $4r_C$ with probability at least $1 - \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. The small-order assumption then implies that $C = \left(g^{a_C/\delta} h^{b_C/\delta} e^{z_C/\delta} f^{r_C/\delta}\right)^2 \tilde{g}_C$ for some group element $\tilde{g}_C$ such that $\tilde{g}_C^2 = 1_{\mathbb{G}}$, i.e., up to a relabeling of the integers $2a_C/\delta$, $2b_C/\delta$, $2z_C/\delta$ and $2r_C/\delta$, there exist a group element $\tilde{g}_C$ and integers a_C, b_C, z_C and r_C such that $C = \left(g^{a_C} h^{b_C} e^{z_C} f^{r_C}\right)^2 \tilde{g}_C$ and $\tilde{g}_C^2 = 1_{\mathbb{G}}$.

Likewise, by considering the linear systems $\mathbf{X} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ and $\mathbf{X} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, with probability at least the same probability as above, there exist group elements $\tilde{g}_\Gamma$ and $\tilde{g}_\Delta$ and integers a_Γ, a_Δ, b_Γ, b_Δ, z_Γ, z_Δ, r_Γ and r_Δ such that $\Gamma = \left(g^{a_\Gamma} h^{b_\Gamma} e^{z_\Gamma} f^{r_\Gamma}\right)^2 \tilde{g}_\Gamma$, $\Delta = \left(g^{a_\Delta} h^{b_\Delta} e^{z_\Delta} f^{r_\Delta}\right)^2 \tilde{g}_\Delta$ and $\tilde{g}_\Gamma^2 = \tilde{g}_\Delta^2 = 1_{\mathbb{G}}$.

Consequently, for any $(x, a', b', u) \in \left\{ \left(x_j, a'_j, b'_j, u_j\right)_{j=1}^5 \right\}$,

$$\left(g^{xa'} h^{xb'} e^{a'b'} f^u\right)^4 = \left(g^{a_C} h^{b_C} e^{z_C} f^{r_C}\right)^{4x^2} \left(g^{a_\Gamma} h^{b_\Gamma} e^{z_\Gamma} f^{r_\Gamma}\right)^{4x} \left(g^{a_\Delta} h^{b_\Delta} e^{z_\Delta} f^{r_\Delta}\right)^4,$$

Furthermore, Lemma 9.3.8 entails that

$$\begin{aligned} a'x &= a_C x^2 + a_\Gamma x + a_\Delta \\ b'x &= b_C x^2 + b_\Gamma x + a_\Delta \\ a'b' &= z_C x^2 + z_\Gamma x + z_\Delta \\ u &= r_C x^2 + r_\Gamma x + r_\Delta \end{aligned}$$

unless one can find a non-trivial discrete-logarithm relation in $\langle f \rangle$. That is because the exponent of g_n is 1 in the expression of g in terms of the initial group elements $g_1, \ldots, g_n$ and of the challenges $x_1, \ldots, x_{n'}$, and since

the exponent of h_1 in the expression of h is 1 in terms of $h_1, \ldots, h_n$ and of $x_1, \ldots, x_{n'}$. Multiplying the third equality by x^2 then implies that

$$\left(a_C x^2 + a_\Gamma x + a_\Delta\right)\left(b_C x^2 + b_\Gamma x + a_\Delta\right) - z_C x^4 + z_\Gamma x^3 + z_\Delta x^2 = 0,$$

i.e., the above polynomial in $\mathbb{Z}[x]$ is of degree 4 and has at least 5 integer roots if the challenges $x_1, \ldots, x_5$ are pairwise distinct. It is thus the zero polynomial, $a_C b_C = z_C$ and (a_C, b_C, r_C) is a valid culpable witness for C.

It now remains to show that for i from n' down to 1, given a witness for the commitment at the $i+1$th step, a witness for the commitment at the ith step can be extracted. To this end, consider five transcripts $\left(U, V, x_j, \mathbf{a}'_j, \mathbf{b}'_j, t'_j\right)_{i=1}^{5}$ such that

$$\left((\mathbf{g}_1{}^{x_j} \circ \mathbf{g}_2)^{\mathbf{a}'_j} (\mathbf{h}_1 \circ \mathbf{h}_2{}^{x_j})^{\mathbf{b}'_j} e^{\langle \mathbf{a}'_j, \mathbf{b}'_j \rangle} f^{t_j}\right)^4 = \left(U^{x_j^2} C^{x_j} V\right)^2$$

for all $j \in [\![5]\!]$. The objective is to find an expression for C in the bases $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{h}_1$, $\mathbf{h}_2$, e and f.

As in the case $i = n'+1$, by considering the linear systems $\mathbf{X}\begin{bmatrix}\nu_1\\ \nu_2\\ \nu_3\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix}$, $\mathbf{X}\begin{bmatrix}\nu_1\\ \nu_2\\ \nu_3\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$ and $\mathbf{X}\begin{bmatrix}\nu_1\\ \nu_2\\ \nu_3\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix}$, one can extract integers $\delta_U, \delta_C, \delta_V$, $z_U, z_C, z_V, r_U, r_C, r_V$, and integer vectors $\mathbf{a}_U, \mathbf{a}_C, \mathbf{a}_V, \mathbf{b}_U, \mathbf{b}_C, \mathbf{b}_V \in \mathbb{Z}^{\lceil n 2^{-i+2} \rceil}$ such that

$$\begin{aligned} U^{2\delta_U} &= \left(\mathbf{g}^{\mathbf{a}_U} \mathbf{h}^{\mathbf{b}_U} e^{z_U} f^{r_U}\right)^4 \\ C^{2\delta_C} &= \left(\mathbf{g}^{\mathbf{a}_C} \mathbf{h}^{\mathbf{b}_C} e^{z_C} f^{r_C}\right)^4 \\ V^{2\delta_V} &= \left(\mathbf{g}^{\mathbf{a}_V} \mathbf{h}^{\mathbf{b}_V} e^{z_V} f^{r_V}\right)^4. \end{aligned}$$

Moreover, if the challenges are pairwise distinct, $\delta_U, \delta_C, \delta_V \neq 0$. Lemma 9.3.7 and the small-order assumption imply that one can actually extract integers z_U, z_C, z_V, r_U, r_C, r_V, and integer vectors $\mathbf{a}_U, \mathbf{a}_C, \mathbf{a}_V$, $\mathbf{b}_U, \mathbf{b}_C, \mathbf{b}_V \in \mathbb{Z}^{\lceil n 2^{-i+2} \rceil}$ such that

$$\begin{aligned} U &= \left(\mathbf{g}^{\mathbf{a}_U} \mathbf{h}^{\mathbf{b}_U} e^{z_U} f^{r_U}\right)^2 \tilde{g}_U \\ C &= \left(\mathbf{g}^{\mathbf{a}_C} \mathbf{h}^{\mathbf{b}_C} e^{z_C} f^{r_C}\right)^2 \tilde{g}_C \\ V &= \left(\mathbf{g}^{\mathbf{a}_V} \mathbf{h}^{\mathbf{b}_V} e^{z_V} f^{r_V}\right)^2 \tilde{g}_V \end{aligned}$$

for some $\tilde{g}_U, \tilde{g}_C, \tilde{g}_V \in \mathbb{G}$ that satisfy $\tilde{g}_U^2 = \tilde{g}_C^2 = \tilde{g}_V^2 = 1_{\mathbb{G}}$.

Therefore, for any $(x, \mathbf{a}', \mathbf{b}', t') \in \left\{ \left(x_j, \mathbf{a}'_j, \mathbf{b}'_j, t'_j \right)_{i=1}^{5} \right\}$,

$$1_{\mathbb{G}} = \mathbf{g}_1{}^{4\left(x\mathbf{a}' - \mathbf{a}_{U,1}x^2 - \mathbf{a}_{C,1}x - \mathbf{a}_{V,1}\right)} \mathbf{g}_2{}^{4\left(\mathbf{a}' - \mathbf{a}_{U,2}x^2 - \mathbf{a}_{C,2}x - \mathbf{a}_{V,2}\right)} \mathbf{h}_1{}^{4\left(\mathbf{b}' - \mathbf{b}_{U,1}x^2 - \mathbf{b}_{C,1}x - \mathbf{b}_{V,1}\right)}$$
$$\mathbf{h}_2{}^{4\left(x\mathbf{b}' - \mathbf{b}_{U,2}x^2 - \mathbf{b}_{C,2}x - \mathbf{b}_{V,2}\right)} e^{4\left(\langle \mathbf{a}', \mathbf{b}' \rangle - z_U x^2 - z_C x - z_V\right)} f^{4\left(t' - r_U x^2 - r_C x - r_V\right)}.$$

Lemma 9.3.8 then implies that

$$x\mathbf{a}' = \mathbf{a}_{U,1}x^2 + \mathbf{a}_{C,1}x + \mathbf{a}_{V,1} \tag{9.1}$$
$$\mathbf{a}' = \mathbf{a}_{U,2}x^2 + \mathbf{a}_{C,2}x + \mathbf{a}_{V,2} \tag{9.2}$$
$$\mathbf{b}' = \mathbf{b}_{U,1}x^2 + \mathbf{b}_{C,1}x + \mathbf{b}_{V,1} \tag{9.3}$$
$$x\mathbf{b}' = \mathbf{b}_{U,2}x^2 + \mathbf{b}_{C,2}x + \mathbf{b}_{V,2} \tag{9.4}$$
$$\langle \mathbf{a}', \mathbf{b}' \rangle = z_U x^2 + z_C x + z_V \tag{9.5}$$
$$t' = r_U x^2 + r_C x + r_V \tag{9.6}$$

since the expressions of the components of $\mathbf{g}_1, \mathbf{g}_2, \mathbf{h}_1, \mathbf{h}_2$ in terms of the challenges $x_1, \ldots, x_{i-1}$ (if $i \geq 2$) and of the initial bases $g_1, \ldots, g_n, h_1, \ldots, h_n$ are such that for each component, there exists an initial basis with 1 as an exponent. Indeed, if $i = 1$, then the statement is true by definition. If $i \geq 2$, denote by $\mathbf{g}(i)$ and $\mathbf{h}(i)$ the vectors at step i. Then, $\mathbf{g}(i) = \mathbf{g}(i-1)_1^{x_{i-1}} \circ \mathbf{g}(i)_2$ and $\mathbf{h}(i) = \mathbf{h}(i-1)_1 \circ \mathbf{h}(i-1)_2^{x_{i-1}}$, which shows that if the statement is true for $i-1$, then it is true for i and the statement is then true for all $i \in [\![n' + 1]\!]$.

Equations 9.1 and 9.2 imply that

$$\mathbf{a}_{U,2}x^3 + (\mathbf{a}_{C,2} - \mathbf{a}_{U,1})\, x^2 + (\mathbf{a}_{V,2} - \mathbf{a}_{C,1})\, x - \mathbf{a}_{V,1} = 0. \tag{9.7}$$

Similarly, Equations 9.3 and 9.4 entail that

$$\mathbf{b}_{U,1}x^3 + (\mathbf{b}_{C,1} - \mathbf{b}_{U,2})\, x^2 + (\mathbf{b}_{V,1} - \mathbf{b}_{C,2})\, x - \mathbf{b}_{V,2} = 0. \tag{9.8}$$

Besides, from Equations 9.2, 9.3 and 9.5,

$$\left\langle \mathbf{a}_{U,2}x^2 + \mathbf{a}_{C,2}x + \mathbf{a}_{V,2}, \mathbf{b}_{U,1}x^2 + \mathbf{b}_{C,1}x + \mathbf{b}_{V,1} \right\rangle - z_U x^2 + z_C x + z_V = 0, \tag{9.9}$$

and the coefficient of x in this polynomial is

$$\langle \mathbf{a}_{C,2}, \mathbf{b}_{V,1} \rangle + \langle \mathbf{a}_{V,2}, \mathbf{b}_{C,1} \rangle - z_C.$$

If the 5 challenges are pairwise distinct, the polynomials in Equations 9.7, 9.8 and 9.9 are necessarily nil as they are of degree at most 4. Therefore, $\mathbf{b}_{V,1} = \mathbf{b}_{C,2}$, $\mathbf{a}_{V,2} = \mathbf{a}_{C,1}$ and $z_C = \langle \mathbf{a}_C, \mathbf{b}_C \rangle$. That is, $(\mathbf{a}_C, \mathbf{b}_C, r_C)$ is a valid witness for C.

Given a prover $(\mathcal{A}, \mathsf{Prove}^*)$ with success probability at least ε, define then $\mathcal{E}$ as an algorithm which first generates bases $\mathbf{g}, \mathbf{h}, e$ as in the real

protocol (which requires $O(nb_{\mathbb{G}})$ group operations). It continues by running the challenge-tree generator of del Pino, Seiler and Lyubashevsky's forking lemma [dLS19] (corrected in Section 9.3.3) with blackbox access to Prove^*. Since the extractor needs 5 transcripts at each recursion step, their challenge-tree generator runs in time $O\left(n^{\log 5+\log \alpha}\log(n+1)T_{\mathsf{Prove}^*}/\varepsilon\right)$ for any $\alpha > (1-5/P)^{-2}$, assuming that the prover succeeds with probability at least $5n^{\log\alpha}/((\alpha-1)P)$. Then, algorithm $\mathcal{E}$ can extract a valid witness in time $O(n\log(n+1)\log(P)T_{\mathsf{Prove}^*})$ unless the extraction fails at one of the protocol steps, since a witness at each recursion step can be computed via linear combinations with coefficients of $O(\log P)$ bits of integer vectors of length n derived from integers returned by Prove^* (so at most $2^{O(T_{\mathsf{Prove}^*})}$ in absolute value). Extraction fails with probability at most

$$\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+\max\left(2^{-b-\lambda+1},\left(1/2-2^{-\lambda}-(1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+1-\mu\right)\right)$$
$$+\left(1/2-2^{-\lambda}-(1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+1-\mu\right),$$

hence the theorem. □

Corollary 9.3.10 (Extractability of Π)**.** *Under the* $(T^{\mathrm{strg}},\varepsilon^{\mathrm{strg}})$*-strong-root assumption, the* $\left(T^{\mathrm{ord}},\varepsilon^{\mathrm{ord}}\right)$*-small-order assumption, the low-dyadic-valuation assumption and the* μ*-assumption over* G*, protocol* Π *is* $(T_{\mathcal{A}},T_{\mathsf{Prove}^*},T_{\mathcal{E}},q_{\mathcal{H}},\varepsilon^{\mathrm{ext}},\ \Sigma)$*-extractable for any* $T_{\mathcal{A}}$ *and* T_{Prove^*} *such that*

$$T_{\mathcal{A}}+n\log(n+1)\log(P)b_{\mathbb{G}}T_{\mathsf{Prove}^*}T_{\mathbb{G}}\leq\Omega\left(\min\left(T^{\mathrm{strg}},T^{\mathrm{ord}}\right)\right),$$

with $T_{\mathcal{E}}=3T_{\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}}+2T_{\Pi'.\mathcal{E}}+O((b+\log n+\ell)T_{\mathbb{G}})$ *and*

$$\varepsilon^{\mathrm{ext}}=\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+\varepsilon^{\mathrm{zk}}_{\tilde{\Pi}}+\left(1/2-2^{-\lambda}-(1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+1-\mu\right)$$
$$+\max\left(2^{-b-\lambda+1},\left(1/2-2^{-\lambda}-(1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+1-\mu\right)\right),$$

if $FS.\tilde{\Pi}^{\mathcal{H}}$ *is* $\left(T_{\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}},q_{\mathcal{H}},\varepsilon^{\mathrm{zk}}_{\tilde{\Pi}^{\mathcal{H}}}\right)$*-statistically honest-verifier zero-knowledge.*

Proof. Given integers α,α',r,r' and integer vectors $\mathbf{a}$, $\mathbf{a}'$, $\mathbf{b}$, $\mathbf{b}'$ such that $\left(Cf^{2\alpha z}\right)^2=\left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}f^{\alpha\langle\mathbf{a},\mathbf{b}\rangle}f^{r}\right)^4$ and $\left(Cf^{2\alpha' z}\right)^2=\left(\mathbf{g}^{\mathbf{a}'}\mathbf{h}^{\mathbf{b}'}f^{\alpha'\langle\mathbf{a}',\mathbf{b}'\rangle}f^{r'}\right)^4$, it follows that

$$1_{\mathbb{G}}=\left(\mathbf{g}^{\mathbf{a}-\mathbf{a}'}\mathbf{h}^{\mathbf{b}-\mathbf{b}'}f^{\alpha(\langle\mathbf{a},\mathbf{b}\rangle-z)}f^{\alpha'(\langle\mathbf{a}',\mathbf{b}'\rangle-z)}f^{r-r'}\right)^4.$$

Lemma 9.3.8 shows that if $\alpha,\alpha'\leftarrow_{\$}[\![0;2^{b+2\lambda}]\!]$, then $\mathbf{a}=\mathbf{a}'$, $\mathbf{b}=\mathbf{b}'$ and $z=\langle\mathbf{a},\mathbf{b}\rangle$ unless one can find a non-trivial discrete-logarithm relation in $\langle f\rangle$.

with probability at least

$$1 - \varepsilon^{\mathrm{strg}} - \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right).$$

Consider then an algorithm $\Pi.\mathcal{E}$ that generates bases as in the real protocol but simulates π_{crs} and π with $FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}$ (the time to generate the bases is already included in $T_{\Pi'.\mathcal{E}}$). It generates $\alpha \leftarrow_{\$} [\![0; 2^{b+2\lambda}]\!]$, computes $Cf^{2\alpha z}$ and runs $\Pi'.\mathcal{E}$ with Prove^* as subroutine from the computation step after this latter has verified π, on the inputs of protocol Π'. It then obtains an integer r and integer vectors $\mathbf{a}$ and $\mathbf{b}$ such that $\left(Cf^{2\alpha z}\right)^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} f^{\alpha\langle \mathbf{a},\mathbf{b}\rangle} f^r\right)^4$. It generates anew $\alpha' \leftarrow_{\$} [\![0; 2^{b+2\lambda}]\!]$, computes $Cf^{2\alpha' z}$, simulates π and runs $\Pi'.\mathcal{E}$ as before a second time. Since $\Pi'.\mathcal{E}$ cannot extract a valid witness for Σ' with probability at most

$$\begin{aligned}\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right) \\ + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right),\end{aligned}$$

the analysis above then shows that $\Pi.\mathcal{E}$ cannot extract a valid witness for Σ with probability at most

$$\begin{aligned}&\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}} \\ &\quad + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right) \\ &\quad + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right).\end{aligned}$$

Note that the bounds on the failure probability of $\Pi'.\mathcal{E}$ already accounted for the probability of finding non-trivial discrete-logarithm relations in $\langle f \rangle$.

As $z = \langle \mathbf{a}, \mathbf{b} \rangle$ and the components of $\mathbf{a}$ and $\mathbf{b}$ are of absolute value less than 2^ℓ, $|z| \leq n2^{2\ell}$ and $Cf^{2\alpha z}$ and $Cf^{2\alpha' z}$ can both be computed in $O(b_{\mathbb{G}} + \ell + \log n)$ group operations. Therefore, $\Pi.\mathcal{E}$ runs in time $3T_{\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}} + 2T_{\Pi'.\mathcal{E}} + O((b + \log n + \ell)T_{\mathbb{G}})$ and the theorem follows. □

Challenge-Tree Generators.

For any prover $(\mathcal{A}, \mathsf{Prove}^*)$ with success probability at least $\varepsilon > 0$, the challenge-tree generator of del Pino, Seiler and Lyubashevsky's forking lemma [dLS19] is claimed to run in time $O\left(n^{\log m + \log \alpha} \log(n)\ T_{\mathsf{Prove}^*}/\varepsilon\right)$ for any $\alpha > (1 - 5/P)^{-2}$, with m denoting the arity of the tree that must be generated. However, we explain that their analysis does not apply to their algorithm, and show how to modify their algorithm to obtain the claimed bound.

More precisely, the task of their algorithm is to generate a rooted tree of height $n' = \lceil \log n \rceil$ (the number of recursion steps) in which each node at height $1 \leq i \leq n' - 1$ has m_i children ($m = \max m_i$), and to label each node with a challenge so that no two siblings in the tree have the same label. Moreover, the executions of Prove^* with the challenges on the paths from the root to each leaf must be accepting. These conditions must all be met for the extractor of the Bulletproofs to be bale extract a witness at each recursion step.

To generate such challenge trees, Bootle et al. [BCC+16] had proposed a recursive tree-finder algorithm that, given fixed challenges $x_1, \ldots, x_{i-1}$, generates a challenge x_i uniformly at random at each recursion step $i > 1$ and makes a recursive call with $x_1, \ldots, x_i$. A challenge is then appended to the tree if the recursive call returns a non-empty tree. At the last recursion step (i.e., at each leaf of the tree), the algorithm runs the protocol with the generated challenge and returns the challenge (as single node) only if the execution is successful. At each recursion step $i < n'$, the algorithm repeats the procedure of generating fresh challenges and making recursive calls as many times as necessary to obtain m_i challenges that eventually lead to a success.

The main issue with their algorithm as pointed out by del Pino, Seiler and Lyubashevsky is that for challenges $x_1, \ldots, x_{i-1}$, the prover may only have negligible success probability if the first $i - 1$ challenges are fixed to $x_1, \ldots, x_{i-1}$, even though the overall success probability of the prover might be non-negligible if all challenges are generated uniformly at random. As a consequence, in case such $x_1, \ldots, x_{i-1}$ is chosen, the tree-finder algorithm may run for an arbitrarily long time.

To resolve this issue, del Pino, Seiler and Lyubashevsky's idea was then to estimate the probability that a certain x_i leads to a success probability that is not too small compared to the success probability of the prover with fixed $x_1, \ldots, x_{i-1}$ *before appending* x_i *to the tree*, and then recursively call the tree-finder algorithm. More precisely, if $\varepsilon_i(x_1, \ldots, x_{i-1})$ denotes the success probability of the prover with the first $i - 1$ challenges fixed to $x_1, \ldots, x_{i-1}$ if $i > 1$ (set $\varepsilon_1 = \varepsilon$), their idea is to generate a challenge x_i that is distinct from its siblings and estimate whether $\varepsilon_{i+1}(x_1, \ldots, x_i) \geq \varepsilon_i(x_1, \ldots, x_{i-1})/\alpha$ for some constant $\alpha > 1$ (to conduct their analysis, α must actually be greater than $(1 - 5/P)^{-2}$). To perform this estimation for a generated x_i, the tree-finder algorithm runs the protocol with the first i challenges fixed to $x_1, \ldots, x_i$ a certain number $T_i = O\left(n^{\log m + \log \alpha} T_{\mathsf{Prove}^*}/\varepsilon\right)$ of times and counts the number of successful executions. Challenge x_i is then appended to the tree only if this number is higher than a certain threshold. If the threshold is not reached, the algorithm generates a fresh challenge x_i.

Using probabilistic methods, they attempt to show that the probability that with this number T_i of protocol runs, for a certain challenge x_i, (1)

the probability that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ and that the threshold is reached is high, and (2) the probability that $\varepsilon_{i+1} < \varepsilon_i/\alpha$ although the threshold is reached is low. The second part lead to a proof that the probability that no node in the final tree is labeled with a challenge x_i such that $\varepsilon_{i+1} < \varepsilon_i/\alpha$ is high.

However, an issue in the proof of the first step is in the use of heavy-row arguments: the authors claimed that the probability that x_i is such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ is at least $1 - 1/\alpha$, which is not the guaranteed by heavy-row arguments. Instead, heavy-row arguments guarantee that the probability that x_i is such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ *conditioned on the event in which* x_i *leads to a success* is at least $1 - 1/\alpha$. Without the conditioning, the best known bound is only $\varepsilon_i(1 - 1/\alpha)$.

We thus propose to modify their tree-finder algorithm as follows: the algorithm does not generate a challenge x_i and start counting, but instead generates x_i and runs the protocol until the end. If the execution is not successful, then the algorithm generates a fresh challenge x_i. If the execution is successful, then the algorithm proceeds with the counting. This way, when the algorithm starts counting, x_i is already known to lead to a success and their running-time analysis applies. The expected running time to generate a value x_i that leads to a success is $T_{\mathsf{Prove}^*}/\varepsilon_i$.

Now recall that a challenge x_i is only appended to the tree if it subsequently leads to a high number of successes. The analysis of del Pino, Seiler and Lyubashevsky then continued by showing that, in expectation, a constant of x_i must be drawn before appending one to the tree, assuming that only x_i such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ are chosen. Therefore, the expected running time until our algorithm appends a challenge x_i to the tree is of order $O((1/\varepsilon_i + T_i)T_{\mathsf{Prove}^*})$ in expectation; and since $\varepsilon_i \geq \varepsilon/\alpha^{i-1}$, $1/\varepsilon_i = o(T_i)$, and $O((1/\varepsilon_i + T_i)T_{\mathsf{Prove}^*}) = O(T_i)$. Their analysis of the expected running time of the whole tree-finder algorithm in case only challenges such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ are chosen then shows that the expected time of our algorithm is of order $O\left(n^{\log m + \log \alpha} \log(n+1) T_{\mathsf{Prove}^*}/\varepsilon\right)$; and our algorithm certainly returns a challenge-tree with transcripts that are all accepting, not just with probability 1/4 as theirs.

9.4 Succinct Arguments for Multi-Integer Commitments

This section gives several succinct protocols related to multi-integer commitments. The first protocol allows to succinctly argue knowledge of an opening to a multi-integer commitment, and is based on the same halve-then-recurse techniques as in Section 9.3. Then comes a protocol that allows to aggregate arguments of knowledge of openings to several commitments (and the techniques used in it can be applied to additively-homomorphic commitment scheme in public-order groups). With the same aggregation techniques, we

then show how to obtain short parameters for the commitment scheme in Section 9.2.2 and the inner-product argument in Section 9.3. Finally, we show how to succinctly argue knowledge of the same opening to several commitments in different bases.

9.4.1 Succinct Arguments of Openings

It is worth noting that the halving techniques used in protocol Π' can also be used to argue knowledge of openings to the multi-integer commitments presented in Section 9.2.2. To argue knowledge of integers $a_1, \ldots, a_n, r$ (with $n \geq 2$) of absolute value less than 2^ℓ and such that $C^2 = (\mathbf{g}^{\mathbf{a}} h^r)^4$ for positive integers n and ℓ, group elements $h, g_1, \ldots, g_n, C \in \mathbb{G}$ and a proof π_{crs} that $\mathbf{g} \in \sqrt{\langle h^2 \rangle}^n$, the prover starts by verifying π_{crs} and aborts if it is invalid. Next, she computes $U \leftarrow \mathbf{g}_1{}^{\mathbf{a}_2} h^{s_U}$ and $V \leftarrow \mathbf{g}_2{}^{\mathbf{a}_1} h^{s_V}$ for s_U and s_V uniformly random over the same range as in protocol Π', and sends them to the verifier. The verifier chooses $x \leftarrow_\$ [\![0; P-1]\!]$, sends it to the prover, and this latter computes $\mathbf{a}' \leftarrow \mathbf{a}_1 + x\mathbf{a}_2$. They then recurse (without verifying π_{crs}) with $\lceil n/2 \rceil$ as new vector-length, $\mathbf{g}_1^x \circ \mathbf{g}_2$ and h as new bases, $U^{x^2} C^x V$ as commitment, and the prover uses $\mathbf{a}'$ and $s_v + rx + s_U x^2$ as new witness. When the vector length is 1, they run the protocol in Section 9.2.2.

As in the inner-product protocol, the bit communication complexity of the prover is of the order of $O(\ell + \log(n) b_{\mathbb{G}})$ bits.

This should compared with the complexity of the straightforward protocol which consists in adapting the protocol in Section 9.2.2 with vector of length 1 to the case of vectors of length n. The bit complexity of this latter protocol is of order $O(b_{\mathbb{G}} + n(\ell + \lambda + \log P))$, and $O(n(\ell + \lambda + \log P))$ with the Fiat–Shamir heuristic.

9.4.2 Aggregating Arguments of Openings to Integer Commitments

This section shows how to aggregate several arguments of knowledge of openings to integer commitments, i.e., given m commitments $C_1, \ldots, C_m$, how to argue *at once* knowledge of integer vectors $\mathbf{a}_j \in \mathbb{Z}^n$ and integers r_j such that $C_j^2 = (\mathbf{g}^{\mathbf{a}_j} h^{r_j})^4$ for all $j \in [\![m]\!]$. These techniques can also be applied to Pedersen's commitments in public-order groups, in which case the size of the proof is constant in m (the only effect of m is on the extraction probability).

More formally, the protocol is for the relation

$$\Big\{(\mathbf{C} \in \mathbb{G}^m, \ell \in \mathbb{N}^*; \mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}^n, \mathbf{r} \in \mathbb{Z}^m) : \forall j \in [\![m]\!]\ C_j^2 = (\mathbf{g}^{\mathbf{a}_j} h^{r_j})^4 \\ \wedge \left\| \begin{bmatrix} \mathbf{a}_j & r_j \end{bmatrix} \right\|_\infty < 2^\ell \Big\},$$

given parameters $(\mathbb{G}, P, h, n, m)$ and $(\mathbf{g}, \pi_{crs}) \in \mathbb{G}^n \times \{0,1\}^*$ as CRS. The protocol satisfies culpable extractability w.r.t. the language define similarly but without the bounds on the witness.

The construction is generic in the sense that it builds upon any protocol for a single commitment. Of course, aggregating arguments is only interesting if it leads to savings in terms of communication size and computational costs compared to m parallel executions of the protocol for a single inner product. Applied to the protocol in Section 9.3.1, the number of group elements sent by the prover in the aggregated argument is $2\lceil\log(n)\rceil + 1$ instead of $m(2\lceil\log n\rceil + 1)$ group elements for m parallel executions of the protocol, but the last two integers in the aggregated argument are mP times larger. Moreover, the verification of the aggregated argument requires a single multi-exponentiation instead of m, but with exponents that are mP times larger than for m separate multi-exponentiations.

Protocol.

The main building block is a protocol Π for the relation in Section 9.4.2. At the beginning of the protocol, the verifier chooses integers $\xi_1, \ldots, \xi_m \leftarrow_\$ [\![0; P-1]\!]$ and sends them to the prover. The prover and the verifier then compute $C_1^{\xi_1} \cdots C_m^{\xi_m}$. With $(\mathbb{G}, P, h, n)$ as parameters, and $\mathbf{g}$ and the proof that $\mathbf{g} \in \sqrt{\langle h^2 \rangle}^n$ as CRS, the parties run Π on the input of $C_1^{\xi_1} \cdots C_m^{\xi_m}$ and $\ell + \lfloor\log(mP)\rfloor + 1$ as maximum length (if ℓ is the maximum bit length of the integers in the witness); and the prover uses $\xi_1\mathbf{a}_1 + \cdots + \xi_m\mathbf{a}_m \in \mathbb{Z}^n$ and $\xi_m r_1 + \cdots + \xi_m r_m \in \mathbb{Z}$ as witness.

The underlying idea is simple: if the prover indeed knows openings to all commitments, then this latter should be able to open random linear combination of $C_1, \ldots, C_m$, and this is enough to convince the verifier as the prover can guess the combination $\xi_1, \ldots, \xi_m$ in advance with only negligible probability.

Alternatively, the verifier could send a single integer $\xi \leftarrow_\$ [\![0; P-1]\!]$, which would define a vector $\Xi \coloneqq \begin{bmatrix} 1 & \xi & \cdots & \xi^{m-1} \end{bmatrix}$. Although this would reduce the communication from the verifier to the prover, the integers in the witness would be P^m times larger instead of mP times. In case optimizing the communication size from the prover is more important (e.g., for non-interactive arguments with the Fiat–Shamir heuristic, although the computational cost of the prover increases), the first variant is preferable. It should be noted that in public-order groups, one should rather favor the second variant since the integers can always be reduced modulo the group order.

Completeness and Security.

The protocol is complete by construction. Moreover, if Π is $(T, T_{\mathsf{Sim}}, \varepsilon)$-honest-verifier zero-knowledge, then so is the protocol.

As for extractability, note that for any matrix $\Xi \in \mathbb{Z}^{m\times m}$, denoting by $\mathbf{e}_j \in \mathbb{Z}^m$ the canonical row vector with 1 at position j and 0 elsewhere for $j \in [\![m]\!]$, and by $\mathrm{adj}(\Xi)$ the adjugate matrix of Ξ, the vector $\mathbf{x}_j \coloneqq \mathbf{e}_j \,\mathrm{adj}(\Xi)$ satisfies $\mathbf{x}_j\Xi = \mathbf{e}_j \,\mathrm{adj}(\Xi)\Xi = \det(\Xi)\mathbf{e}_j$ since $\mathrm{adj}(\Xi)\Xi = \det(\Xi)\mathbf{I}_m$. Assuming that for all $i \in [\![m]\!]$, the equality $\left(C_1^{\xi_{i,1}} \cdots C_m^{\xi_{i,m}}\right)^2 = (\mathbf{g}^{\mathbf{a}_i} h^{r_i})^4$ holds for some integer vectors $\mathbf{a}_i \in \mathbb{Z}^m$ and integers r_i, it follows that $C_j^{2\det\Xi} = \left(\mathbf{g}^{\sum_{i=1}^m \mathbf{x}_{j,i}\mathbf{a}_i} h^{\sum_{i=1}^m \mathbf{x}_{j,i} r_i}\right)^4$ for all $j \in [\![m]\!]$.

Consider then an extractor which first generates $\Xi_1 \leftarrow_{\$} [\![0; P-1]\!]^m$. If $\Xi_1 = 0_{\mathbb{Z}^m}$, then it generates a new vector Ξ_1 and otherwise runs the protocol with Prove^* with Ξ_1 as first message from the verifier. If the protocol execution fails, then the extractor rewinds the Prove^* to the beginning of the protocol and generates a fresh vector Ξ_1.

Note that conditioned on the event in which $\Xi_1 \neq 0_{\mathbb{Z}^m}$, Prove^* convinces the verifier with probability at least $\varepsilon - P^{-m}$. Moreover, conditioned on the first message Ξ_1, a heavy-row argument implies that with probability at least $1/2$, the row vector Ξ_1 is such that the verifier of the sub-protocol Π is convinced with probability at least $(\varepsilon - P^{-m})/2$ on input $C_1^{\xi_{1,1}} \cdots C_m^{\xi_{1,m}}$.

The extractor then runs $\Pi.\mathcal{E}$ on Prove^* from the computation step after it has received Ξ_1. If $\Pi.\mathcal{E}$ does not return a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $(\varepsilon - P^{-m})/2$, the extractor generates a new vector Ξ_1 and proceeds as before. Denote by H (as in "heavy") the event in which Ξ_1 is such that the verifier of Π is convinced with probability at least $(\varepsilon - P^{-m})/2$, and by T the event in which $\Pi.\mathcal{E}$ returns a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $(\varepsilon - P^{-m})/2$ for a given $C_1^{\xi_{1,1}} \cdots C_m^{\xi_{1,m}}$ (Markov's inequality implies that this event occurs with probability at most $1/2$). The extractor returns a value in the event $H \cap T$ and $\Pr[T \cap H] = \Pr[T|H]\Pr[H] \geq 1/4$. Therefore, the generator restarts from the generation of Ξ_1 an expected number of times at most 4 and at each repetition, its running time is at most $\tilde{\varepsilon}^{-1}(1 - P^{-m})^{-1} T_{\mathsf{Prove}^*} + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)$, with $\tilde{\varepsilon} \coloneqq \varepsilon - P^{-m}$ and $T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)$ the expected running time of $\Pi.\mathcal{E}$ given a prover that succeeds with probability at least $\tilde{\varepsilon}/2$. The term T_{Prove^*} comes from the fact that the generator must determine whether Ξ_1 leads to a success. The expected running time is at most $4\left(\tilde{\varepsilon}^{-1}(1 - P^{-m})^{-1} T_{\mathsf{Prove}^*} + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\right)$.

Now, for $i \in \{2, \ldots, m\}$, the extractor generates $\Xi_i \leftarrow_{\$} [\![0; P-1]\!]^m$. If $\Xi_1, \ldots, \Xi_i$ are linearly dependent over $\mathbb{Q}$, then the extractor generates a new vector $\Xi_i \leftarrow_{\$} [\![0; P-1]\!]^m$. Slinko [Sli01, Corollary 2] proved that this event occurs with probability at most P^{-m+i-1}. It follows that for fixed

$\Xi_1, \ldots, \Xi_{i-1}$, conditioned on the event in which $\Xi_1, \ldots, \Xi_i$ are linearly independent, Prove^* convinces the verifier with probability at least $\varepsilon - P^{-m+i-1}$. The extractor proceeds as in the case $i = 1$ but with $P^{-m+i-1} \leq P^{-1}$ instead of P^{-m}. Note, however, that the extractor must determine whether $\Xi_1, \ldots, \Xi_i$ are linearly independent, and it can do so by computing its rank, i.e., the rank of a matrix with coefficient in $[\![0; P-1]\!]$. Using Bareiss algorithm [Bar68] which requires $O(m^3)$ arithmetic operations on integers less than $O\left(m^{m/2} P^m\right)$, i.e., it can be done in time $O\left(m^5 (\log m + \log P)^2\right)$. Besides, the extractor must determine whether Ξ_i leads to a success, which takes time at most T_{Prove^*}. At any step $i \in [\![m]\!]$, the expected running time is thus at most

$$4\left(\tilde{\varepsilon}^{-1} O\left(\left(1 - P^{-1}\right)^{-1}\left(m^5 (\log m + \log P)^2 + T_{\mathsf{Prove}^*}\right)\right) + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\right)$$

for $\tilde{\varepsilon} \coloneqq \varepsilon - P^{-1}$.

Consequently, the extractor can obtain vectors $\mathbf{a}_i \in \mathbb{Z}^n$ and integers r_i such that $\left(C_1^{\xi_{i,1}} \cdots C_m^{\xi_{i,m}}\right)^2 = \left(\mathbf{g}^{\mathbf{a}_i} h^{r_i}\right)^4$ for all $i \in [\![m]\!]$ and for some $\Xi \in \mathbb{Z}^{m \times m}$ such that $\det \Xi \neq 0$ in expected time at most

$$4m\left(O\left(\tilde{\varepsilon}^{-1}\left(1 - P^{-1}\right)^{-1}\left(m^5 (\log m + \log P)^2 + T_{\mathsf{Prove}^*}\right)\right) + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\right).$$

Lemma 9.3.7 then shows that with probability at least

$$1 - \left(1/2 - 2^{-\lambda} - (1 - \mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right),$$

$\det \Xi$ divides $2\sum_{i=1}^m \mathbf{x}_{i,j} \mathbf{a}_i$ and $2\sum_{i=1}^m \mathbf{x}_{i,j} r_i$ for all $j \in [\![m]\!]$ under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, if $T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)$ plus the time to compute $2\sum_{i=1}^m \mathbf{x}_{i,j} \mathbf{a}_i$ and $2\sum_{i=1}^m \mathbf{x}_{i,j} r_i$, which is denoted T, is such that $n(b_{\mathbb{G}} + \log n) T T_{\mathbb{G}} + (T + b_{\mathbb{G}} + \log n) \log(P) T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$. The coefficient of $\mathrm{adj}(\Xi)$ are of order $O(mP^m)$ in absolute value, and the components of $\mathbf{a}_i$ and r_i are at most $2^{O(T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2))}$ in absolute value. Therefore, $2\sum_{i=1}^m \mathbf{x}_{i,j} \mathbf{a}_i$ and $2\sum_{i=1}^m \mathbf{x}_{i,j} r_i$ can be computed in time

$$O\left(nm T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\ (\log m + m \log P)\right).$$

9.4.3 Shorter Parameters for Integer Commitments

The keys of the multi-integer commitment scheme in Section 9.2.2 include a proof that that they were well-formed, i.e., that if the scheme allows to commit to m integers, then the group elements $g_1, \ldots, g_m$ in the key are all in $\sqrt{\langle h^2 \rangle}$ for $h \leftarrow_{\$} \mathbb{G}$. One can of course run m times in parallel the protocol

in Section 9.2.2, which would result in arguments of $O(mb_{\mathbb{G}})$ bits. Alternatively, one could use the same techniques as in Section 9.4.2 to aggregate these arguments and obtain a single argument of $O(b_{\mathbb{G}} + \log m)$ bits.

In more detail, the protocol is a proof system for the language

$$\left\{g_1, \ldots, g_m \in \mathbb{G}, \ell \in \mathbb{N}^* \colon \exists \alpha_1, \ldots, \alpha_m \in \left[\!\left[0; 2^\ell\right]\!\right], \forall i \in [\![m]\!]\, g_i = h^{\alpha_i}\right\},$$

given parameters $(\mathbb{G}, P, h, m)$ and the empty string as CRS. It guarantees that $g_i \in \sqrt{\langle h^2 \rangle}$ for all $i \in [\![m]\!]$. At the beginning of the protocol, the verifier chooses integers $\xi_1, \ldots, \xi_m \leftarrow_\$ [\![0; P-1]\!]$ and sends them to the prover. The prover and the verifier compute $g_1^{\xi_1} \cdots g_m^{\xi_m}$, and run the protocol of Section 9.2.2 (i.e., for the case $m = 1$) on the input of $g_1^{\xi_1} \cdots g_m^{\xi_m}$ and $\ell + \lfloor \log(mP) \rfloor + 1$ as maximum bit length; and the prover uses $\xi_1 x_1 + \cdots + \xi_m x_m \in \mathbb{Z}$ as witness. With the Fiat–Shamir heuristic, the proof then consists of $2\lfloor \log P \rfloor + \ell + \lfloor \log(mP) \rfloor + \lambda + 4$ bits. For $\ell = b_{\mathbb{G}} + \lambda$, that is $O(b_{\mathbb{G}} + \log m)$ bits (recall that $P \leq 2^{b_{\mathbb{G}}}$ and $b_{\mathbb{G}} = \Omega(\lambda)$). The same arguments as in Section 9.4.2 imply that this protocol is complete, statistically honest-verifier zero-knowledge and extractable.

Similarly, the CRS of the inner-product protocol in Section 9.3.1 includes an argument that the bases $\mathbf{g}$ and $\mathbf{h} \in \mathbb{G}^n$ (for some $n \in \mathbb{N}^*$) are in $\sqrt{\langle f^2 \rangle}^n$ for $f \leftarrow_\$ \mathbb{G}$. The same technique can then be used to obtain an argument of $O(b_{\mathbb{G}} + \log n)$ bits instead of $O(nb_{\mathbb{G}})$ bits.

9.4.4 Succinct Base-Switching Arguments

This section shows how to succinctly argue knowledge of an integer vector $\mathbf{a} \in \mathbb{Z}^n$ and of integers $r_1, \ldots, r_m$ such that $(\mathbf{a}, r_1), \ldots, (\mathbf{a}, r_m)$ respectively open to commitments $C_1, \ldots, C_m$ w.r.t. bases $(\mathbf{g}_1, h), \ldots, (\mathbf{g}_m, h) \in \mathbb{G}^{n+1}$. Said otherwise, the same vector $\mathbf{a}$ is committed in m different bases. Formally, the protocol is for the relation

$$\left\{(\mathbf{C} \in \mathbb{G}^m, \ell \in \mathbb{N}^*; \mathbf{a} \in \mathbb{Z}^n, \mathbf{r} \in \mathbb{Z}^m) : \forall j \in [\![m]\!]\; C_j^2 = (\mathbf{g}_j{}^{\mathbf{a}} h^{r_j})^4 \right.$$
$$\left. \wedge \left\| \begin{bmatrix} \mathbf{a} & \mathbf{r} \end{bmatrix} \right\| < 2^\ell \right\},$$

given parameters $(\mathbb{G}, P, h, n, m)$ and $(\mathbf{g}_1, \ldots, \mathbf{g}_m, \pi_{crs}) \in \mathbb{G}^{m \times n} \times \{0,1\}^*$ as CRS, and π_{crs} is an argument that each of the components of all the $\mathbf{g}_j$ vectors are in $\sqrt{\langle h^2 \rangle}$.

The idea underlying the protocol is again to use linear combinations to reduce $C_1, \ldots, C_m$ to a single group element, and run the succinct argument for multi-integer openings on the linear combination of the basis. More precisely, the verifier generates $\xi \leftarrow_\$ [\![0; P-1]\!]^m$ and sends it to the prover. Both parties compute $C \leftarrow \mathbf{C}^\xi = C_1^{\xi_1} \cdots C_m^{\xi_m}$ and $\mathbf{g} \leftarrow \mathbf{g}_1^{\xi_1} \circ \cdots \circ \mathbf{g}_m^{\xi_m}$. They then run the protocol for multi-integer openings on the input of C and $\mathbf{g}$,

and the prover uses $\mathbf{a}$ and $\xi_1 r_1 + \cdots + \xi_m r_m$ as witness. The new witness is then at most mP times larger than the original one.

The completeness and the zero-knowledge property of the scheme are immediate. Concerning its extractability, notice that if for all $i \in [\![m]\!]$ one can obtain vectors $\mathbf{a}_i \in \mathbb{Z}^m$ and integers r_j such that $\mathbf{C}^{2\xi_i} = \left(g_1^{\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m^{\xi_{i,m}}\right)^{4\mathbf{a}_i} h^{4r_i}$, with $\xi_1, \ldots, \xi_m$ vectors in $[\![0; P-1]\!]^m$ that are linearly independent over $\mathbb{Q}$, then one can first compute $\mathbf{x}_j \coloneqq \mathbf{e}_j \operatorname{adj}(\Xi)$, where $\mathbf{e}_j \in \mathbb{Z}^m$ denotes the canonical row vector with 1 at position j and 0 elsewhere for all $j \in [\![m]\!]$, Ξ denotes the matrix with rows $\xi_1, \ldots, \xi_m$ and $\operatorname{adj}(\Xi)$ its adjugate. Then, since $\mathbf{x}_j \Xi = \det(\Xi)\mathbf{e}_j$, it follows that for all $j \in [\![m]\!]$,

$$\begin{aligned}
C_j^{2\det\Xi} &= \prod_{i=1}^{m} \left(\mathbf{g}_1{}^{\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m{}^{\xi_{i,m}}\right)^{4\mathbf{x}_{j,i}\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j} \\
&= \prod_{i=1}^{m} \left(\mathbf{g}_1{}^{\mathbf{x}_{j,i}\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m{}^{\mathbf{x}_{j,i}\xi_{i,m}}\right)^{4\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j} \\
&= \left(\mathbf{g}_1{}^{\sum_i \mathbf{x}_{j,i}\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m{}^{\sum_i \mathbf{x}_{j,i}\xi_{i,m}}\right)^{4\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j} \\
&= \mathbf{g}_j{}^{4\det(\Xi)\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j}.
\end{aligned}$$

The second and third equalities respectively rely on the fact that for any two vectors $\mathbf{e}, \mathbf{f} \in \mathbb{G}^n$ and any vector $\mathbf{c} \in \mathbb{Z}^n$, $(\mathbf{e} \circ \mathbf{f})^{\mathbf{c}} = \mathbf{e}^{\mathbf{c}}\mathbf{f}^{\mathbf{c}}$, and for any two integers c, d, $\mathbf{e}^c \circ \mathbf{e}^d = \mathbf{e}^{cd}$. Using rewinding techniques similar to those in the extractability proof the aggregated arguments then shows that the protocol is extractable.

9.5 Succinct Argument for Diophantine Equations

This section gives a succinct argument to argue satisfiability of Diophantine equations. Although Davis, Putnam, Robinson and Matiyasevich [Mat70] showed that there does not exist an algorithm that can decide whether any Diophantine equation has a solution (thereby giving a negative answer to Hilbert's tenth problem), one can argue in zero-knowledge knowledge of a solution, if a solution is known to the prover, which convinces the verifier that the equation is satisfiable.

Damgård and Fujisaki gave [DF02, Section 4.2] a protocol to argue, given three commitments C_1, C_2, C_3 computed with their scheme, knowledge of openings x_1, x_2, x_3 such that $x_3 = x_1 x_2$. Therefore, to show the satisfiability of an ν-variate polynomial $\sum_{\mathbf{i} \in \mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1} \cdots x_\nu^{i_\nu}$ of total degree δ using their scheme, if the polynomial can be computed in $M(\nu, \delta)$ multiplications, then one would have to compute $2M(\nu, \delta) + 1$ integer commitments and compute $M(\nu, \delta)$ multiplication-consistency arguments. As Damgård and Fujisaki's scheme is additively homomorphic, the verifier can verify addition itself.

Computing a monomial $x_1^{i_1} \cdots x_\nu^{i_\nu}$ can be done in at most $\delta - 1$ multiplications since the polynomial is of total degree δ. Without any further restriction on the polynomial than its number of variables ν and its total degree δ, the best bound on the number of multiplications (between variables) one can give is $\delta - 1$ as δ could be less than ν, and all i_k at most 1. Evaluating an ν-variate polynomial of total degree δ thus *a priori* requires $(\delta - 1)\binom{\nu+\delta}{\delta}$ multiplications as such a polynomial has at most $\binom{\nu+\delta}{\delta}$ monomials. This can be improved to $\binom{\nu+\delta}{\delta} - \nu - 1 \leq \binom{\nu+\delta}{\delta}$ multiplications by evaluating all possible monomials (even those which may have coefficient 0) recursively by increasing degree and storing the previous evaluations. There exist more efficient methods for specific polynomials (e.g., recursive Horner's method or summation for polynomials with a small numbers of monomials of large degree) but no better upper-bound on the number of multiplications is known for generic polynomials.

Consider a prover that wants to argue the satisfiability of a (generic) ν-variate polynomial of total degree δ with integer coefficients whose absolute value is upper-bounded by 2^H for some integer H. The communication complexity of the arguments of the first multiplication gates are of order $\Omega(\log P + \ell + b_{\mathbb{G}})$ if ℓ denotes the maximum bit length of any coordinate in the solution. Since the total degree of the polynomial is δ, the bit length of the witness at the maximum-depth multiplication gates can be as large as $\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right) H$ and the communication complexity of the argument of the satisfiability of the Diophantine equation (i.e., the proof that the polynomial actually evaluates to 0) is $\Omega\left(\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right) H + b_{\mathbb{G}}\right)$. The overall communication complexity with Damgård and Fujisaki's scheme is therefore upper-bounded by $O\left(\binom{\nu+\delta}{\delta}\left(\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right) H + b_{\mathbb{G}}\right)\right)$ and lower-bounded by $\Omega\left(\binom{\nu+\delta}{\delta}(\ell + b_{\mathbb{G}})\right)$ for generic polynomials.

This section shows how to argue the satisfiability of Diophantine equations with a communication complexity of order $O\left(\delta\ell + \min(\nu, \delta)\log\left(\nu + \delta\right)\right.$ $\left. b_{\mathbb{G}} + H\right)$.

9.5.1 Arguments via Polynomial-Degree Reductions

Our approach to argue for Diophantine satisfiability is different and is inspired by Skolem's method [Sko50]. The idea is to give a systematic method to turn any polynomial equation to another of degree at most 4 by increasing the number of variables so that the satisfiability of one polynomial implies that of the other. The resulting polynomial is such that its satisfiability is equivalent to the satisfiability (over the integers) of a Hadamard product of the form $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$ and of linear equations with the entries of $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ as indeterminate. The length of these latter vectors is the number of variables in the resulting polynomial, and if the original polynomial is ν-variate and of total degree at most δ, then the new polynomial has at most

$\nu\lfloor\log\delta\rfloor + (\delta-1)\mu$ variables, where $\mu \leq \binom{\nu+\delta}{\delta}$ is the number of monomials in the original polynomial.

On this account, if one can argue for the satisfiability of such Hadamard products and linear constraints, then one can argue for the satisfiability of the original polynomial. In the protocol given in Section 9.5.2, the prover only sends logarithmically many group elements in the length of the vectors in the Hadamard product, and a constant number of integers. The bit length of those integers is upper-bounded by $O\left(\delta\ell + b_{\mathbb{G}} + \min(\nu,\delta)\log\left(\nu+\delta\right)\log P + H\right)$ if the bit length of the witness is upper-bounded by ℓ and the bit length of each coefficient of the polynomial is at most H.

Reducing Arbitrary Polynomials to Polynomials of Degree at most 4.

We now give a systematic procedure to reduce any Diophantine equation into an equation of degree at most 4 of which the satisfiability can be reduced to the satisfiability of a Hadamard product and linear constraints; and the Hadamard product and the constraints can be read immediately from the resulting polynomial. The presentation is gradual as it starts with ν-variate affine equations, proceeds with ν-variate Diophantine equations in which the degree in each variable is at most 1, further tackles univariate polynomials of arbitrary degree and then considers arbitrary Diophantine equations. The method applies to every multivariate integer polynomial, but for specific polynomials, more astute techniques could lead to a smaller number of new variables and/or constraints.

Step 1–Affine Equations. Given an integer polynomial $a_1x_1+\cdots+a_\nu x_\nu+b \in \mathbb{Z}[x_1,\ldots,x_\nu]$, set $\mathbf{a}_O \leftarrow \begin{bmatrix} x_1 & \cdots & x_\nu \end{bmatrix}$ and for all $i \in [\![\nu]\!]$, set $\mathbf{a}_{L,i} = 1$ and $\mathbf{a}_{R,i} = x_i$. The equation $a_1x_1 + \cdots + a_\nu x_\nu + b = 0$ is satisfied if and only if $\left\langle \begin{bmatrix} a_1 & \cdots & a_\nu \end{bmatrix}, \mathbf{a}_O \right\rangle = -b$ and $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$. Note that no variable or linear constraint was added to the system of equations.

Step 2–Restricted Diophantine Equations. Consider an integer polynomial $\sum_{\mathbf{i}\in\mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1}\cdots x_\nu^{i_\nu} \in \mathbb{Z}[x_1,\ldots,x_\nu]$ of total degree δ such that $a_{\mathbf{i}} \neq 0_{\mathbb{Z}} \implies \mathbf{i} \in \{0,1\}^\nu$, i.e., the polynomial is of degree at most 1 in each variable. For all $\mathbf{i} \in \mathbb{N}^\nu \setminus \{0_{\mathbb{N}^\nu}\}$ such that $a_{\mathbf{i}} \neq 0_{\mathbb{Z}}$, let $\{j_1,\ldots,j_{w(\mathbf{i})}\}$ be the subset of $[\![\nu]\!]$ such that $j_1 < \cdots < j_{w(\mathbf{i})}$ and $i_{j_1} = \cdots = i_{w(\mathbf{i})} = 1$, with $w(\mathbf{i})$ denoting the Hamming weight of $\mathbf{i}$ (which is necessarily less than δ). If $w(\mathbf{i}) > 1$, introduce new variables

$$u_{\mathbf{i},1} \leftarrow x_{j_1}x_{j_2}, \quad u_{\mathbf{i},2} \leftarrow u_{\mathbf{i},1}x_{j_3}, \quad \ldots, \quad u_{\mathbf{i},w(\mathbf{i})-1} \leftarrow u_{\mathbf{i},w(\mathbf{i})-2}x_{j_{w(\mathbf{i})}},$$

with the convention that $u_{\mathbf{i},0} \coloneqq x_{j_1}$. Note that $\sum_{\mathbf{i}\in\mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1}\cdots x_\nu^{i_\nu} = 0$

if and only if

$$\sum_{\substack{\mathbf{i}\in\mathbb{N}^\nu:\, a_\mathbf{i}\neq 0_\mathbb{Z}\\ w(\mathbf{i})>1}} \sum_{k=1}^{w(\mathbf{i})-1} \left(u_{\mathbf{i},k} - u_{\mathbf{i},k-1}x_{j_{k+1}}\right)^2 + \left(\sum_{\mathbf{i}\in\mathbb{N}^\nu} \mathbf{a}_\mathbf{i} u_{\mathbf{i},w(\mathbf{i})-1}\right)^2 = 0,$$

with the convention that $u_{\mathbf{0}_{\mathbb{N}^\nu},-1} = 1$. This latter polynomial is of degree 4, and the equation is satisfied if and only if the linear equation $\sum_{\mathbf{i}\in\mathbb{N}^\nu} \mathbf{a}_\mathbf{i} u_{\mathbf{i},w(\mathbf{i})-1} = 0$ is as well as the constraints $u_{\mathbf{i},k} - u_{\mathbf{i},k-1}x_{j_{k+1}} = 0$. Set then

$$\begin{aligned}
\mathbf{a}_L &\leftarrow \begin{bmatrix} x_{j_1} & u_{\mathbf{i},1} & \cdots & u_{\mathbf{i},w(\mathbf{i})-2}\end{bmatrix}\\
\mathbf{a}_R &\leftarrow \begin{bmatrix} x_{j_2} & x_{j_3} & \cdots & x_{j_{w(\mathbf{i})}}\end{bmatrix}\\
\mathbf{a}_O &\leftarrow \begin{bmatrix} u_{\mathbf{i},1} & u_{\mathbf{i},2} & \cdots & u_{\mathbf{i},w(\mathbf{i})-1}\end{bmatrix},
\end{aligned}$$

and introduce the linear constraints $\mathbf{a}_{L,i+1} - \mathbf{a}_{O,i} = 0$ for all $i \in \{1,\ldots,w(\mathbf{i})-2\}$. The procedure introduces at most $\delta-1$ new variables and $\delta-2$ new linear constraints per monomial, and since there are at most $\binom{\nu+\delta}{\delta}$ monomials in an ν-variate polynomial of total degree δ, that is at most $(\delta-1)\binom{\nu+\delta}{\delta}$ variables and $(\delta-2)\binom{\nu+\delta}{\delta}$ constraints.

Step 3–Univariate Polynomials. Given a polynomial $Z = a_0 + a_1x + \cdots + a_\delta x^\delta \in \mathbb{Z}[x]$ of degree $\delta \geq 2$, introduce variables

$$u_1 \leftarrow x^2, \quad u_2 \leftarrow u_1^2, \quad \ldots, \quad u_{\lfloor\log\delta\rfloor} \leftarrow u_{\lfloor\log\delta\rfloor-1}^2.$$

Now notice that $a_0 + a_1x + \cdots + a_\delta x^\delta = 0$ if and only if

$$\left(u_1 - x^2\right)^2 + \sum_{i=2}^{\lfloor\log\delta\rfloor} \left(u_i - u_{i-1}^2\right)^2 + \left(Z'(x,u_1,\ldots,u_{\lfloor\log\delta\rfloor})\right)^2 = 0,$$

where $Z'(x,u_1,\ldots,u_{\lfloor\log\delta\rfloor})$ is $\lfloor\log\delta\rfloor+1$-variate integer polynomial in which the degree of each variable is at most 1, i.e., if and only if $Z'(x,u_1,\ldots,u_{\lfloor\log\delta\rfloor}) = 0$ and the constraints $u_1 - x^2 = 0$ and $u_{i+1} - u_i^2 = 0$ are satisfied.

Since

$$\sum_{i=0}^{\delta} a_i x^i = a_0 + \sum_{k=0}^{\lfloor\log\delta\rfloor} \sum_{i=2^k}^{2^{k+1}-1} a_i x^i = a_0 + \sum_{k=0}^{\lfloor\log\delta\rfloor} \sum_{i=2^k}^{2^{k+1}-1} a_i x^{i_0} u_1^{i_1} \cdots u_{k-1}^{i_{k-1}} u_k,$$

where $i_0,\ldots,i_{k-1}$ is the binary decomposition of i and $a_i \coloneqq 0$ for $i > \delta$, this give an explicit expression for Z'.

Set then the vectors $\mathbf{a}_L \leftarrow \mathbf{a}_R \leftarrow \begin{bmatrix} x & u_1 & \cdots & u_{\lfloor \log \delta \rfloor - 1} \end{bmatrix}$ and $\mathbf{a}_O \leftarrow \begin{bmatrix} u_1 & u_2 & \cdots & u_{\lfloor \log \delta \rfloor} \end{bmatrix}$, and introduce constraints

$$\mathbf{a}_{L,i+1} - \mathbf{a}_{O,i} = \mathbf{a}_{R,i+1} - \mathbf{a}_{O,i} = 0$$

for all $i \in [\![\lfloor \log \delta \rfloor - 1]\!]$.

As the second step shows that the satisfiability of Z' can be reduced to a Hadamard product and linear constraints, the satisfiability of Z can be reduced to a Hadamard product and linear constraints. This procedure introduces $\lfloor \log \delta \rfloor$ new variables and $2\,(\lfloor \log \delta \rfloor - 1)$ new linear constraints. It is important for Step 4 to remark that the number of monomial of Z' is at most the same as the number of monomials in Z.

Step 4–Arbitrary Diophantine Equations. For any integer polynomial $Z = \sum_{\mathbf{i} \in \mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1} \cdots x_\nu^{i_\nu} \in \mathbb{Z}[x_1, \ldots, x_\nu]$ (for $\nu \geq 2$) of total degree δ, apply Step 3 to Z considering it as a polynomial in $\mathbb{Z}[x_2, \ldots, x_\nu][x_1]$, i.e., a polynomial in x_1 with coefficients in $\mathbb{Z}[x_2, \ldots, x_\nu]$. Let Z' be the resulting polynomial with coefficients in $\mathbb{Z}[x_2, \ldots, x_\nu]$ and of degree at most 1 in each variable as in Step 3. Repeat Step 3 with Z' and variable x_2. After Step 3 has been repeated for each $x_1, \ldots, x_\nu$, at most $\nu \lfloor \log \delta \rfloor$ new variables and $2\nu(\lfloor \log \delta \rfloor - 1)$ new linear constraints have been introduced, the resulting polynomial is of degree at most 1 in all variables and has coefficients in $\mathbb{Z}$. Concerning its total degree, note that during the process, for each monomial $x_1^{i_1} \cdots x_\nu^{i_\nu}$, the term $x_k^{i_k}$ is replaced by at most one variable if $i_k \leq 2$ and by the product of $\log i_k + 1 \leq i_k$ variables if ß$_k > 2$ for all $k \in [\![\nu]\!]$, so the total degree remains at most δ. Now apply then Step 2 to the resulting polynomial.

In summary, the procedure reduces the satisfiability of any polynomial in $\mathbb{Z}[x_1, \ldots, x_\nu]$ of total degree δ with μ monomials ($\mu \leq \binom{\nu+\delta}{\delta}$ necessarily) to the satisfiability of a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ integer vectors of length at most $\nu \lfloor \log \delta \rfloor + (\delta - 1)\mu$, and Q linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for all $q \in [\![Q]\!]$ with $Q \leq 1 + 2\nu(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu$ and with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q}$ integer vectors and $c_q \in \mathbb{Z}$. The coefficients of the linear constraints introduced by the procedure are in $\{-1, 0, 1\}$, except for one of which the coefficients are the coefficients of the original polynomial.

Example. As a simple illustration of the procedure, consider the polynomial $2x^3 + xy - 1$. The procedure introduces new variables $u \leftarrow x^2$, $v \leftarrow xy$

and $w \leftarrow ux$, and the equation $2x^3 + xy - 1 = 0$ is satisfiable if and only if $(u - x^2)^2 + (v - xy)^2 + (w - ux)^2 + (2w + v - 1)^2 = 0$ also is, which allows to write a Hadamard product and linear constraints which are satisfiable if and only if this latter equation is.

Diophantine Equations as Circuits.

It is worth noting that any polynomial in $\mathbb{Z}[x_1, \ldots, x_\nu]$ can naturally be viewed as an arithmetic circuit with integer inputs, and addition gates correspond to addition between two integers and similarly for multiplication gates. Nevertheless, different circuits can compute the same polynomial. For instance, the polynomial $xy + 2y = y(x + 2)$ can be computed by multiplying x and y, multiplying y by 2 and adding the result, or by adding 2 to x and multiplying the result by y. The fewer multiplication gates there are in a circuit that represents a polynomial, the smaller the communication cost of the protocol for its satisfiability will be. In any case, one can always use the circuit directly inferred by its representation as sum of monomials.

Bootle, Cerulli, Chaidos, Groth and Petit [BCC$^+$16, Appendix A] described a procedure to turn any arithmetic circuit over $\mathbb{Z}_p$ into a circuit with only multiplication gates together with a list of linear constraints to ensure consistency between the outputs of a multiplication gate and the inputs of the gates at the next depth level of the circuit. The procedure replaces addition gates and multiplication by a constant with linear constraints and only retains multiplication gates. It ensures that the new circuit and the constraints are satisfiable if and only if the original circuit is satisfiable.

More precisely, their procedure converts any arithmetic circuit with n multiplication gates into a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ in $\mathbb{Z}_p^n$, and $Q \leq 2n$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for $q \in [\![Q]\!]$, with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q} \in \mathbb{Z}_p^n$ and $c_q \in \mathbb{Z}_p$. The vectors $\mathbf{a}_L$, $\mathbf{a}_R$ respectively denote the vectors of left and right inputs to the multiplication gates, and $\mathbf{a}_O$ the vector of outputs.

Unfortunately, their procedure cannot be directly use for circuits over the integers as it requires to re-write a linear equation $\mathbf{y} = \mathbf{A}\mathbf{x}$ (with indeterminate $\mathbf{x}$) as an equation $\mathbf{z} = \mathbf{A}'\mathbf{x}$, with $\mathbf{A}'$ in reduced row-echelon form obtained from $\mathbf{A}$ via Gaussian elimination. Computing $\mathbf{A}'$ may thus possibly require to invert entries of $\mathbf{A}$. Yet, for any matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, there exist [Art10, Theorem 14.4.6] invertible integer matrices $\mathbf{Q} \in GL_m(\mathbb{Z})$ and $\mathbf{P} \in GL_n(\mathbb{Z})$ such that $\mathbf{Q}^{-1}\mathbf{A}\mathbf{P}$ is a matrix of the form $\mathrm{diag}(d_1, \ldots, d_k, 0, \ldots, 0)$, with all d_i positive integers such that $d_1 \mid d_2 \mid \cdots \mid d_k$. Given this observation, the equation $\mathbf{y} = \mathbf{A}\mathbf{x}$ can be re-written as $\mathbf{z} = \mathbf{A}'\mathbf{x}$, with $\mathbf{A}'$ of the previous form, i.e., $z_i = d_i x_i$ for $i \in [\![k]\!]$, and the circuit cannot be satisfied

if $z_i \neq 0$ for any $k > i$. By introducing new variables $z_i \coloneqq d_i x_i$, and increasing the number of constraints to include the constraints $z_i - d_i x_i = 0$, one could probably proceed as they did and have at most $Q \leq 3n$ constraints to satisfy.

The issue with using this procedure to argue for Diophantine satisfiability is that one cannot readily infer the constraints from the initial polynomial and one must always determine them on a case-by-case basis. Besides, if one uses the circuit directly inferred by the monomials of the polynomial without introducing new variables to decrease its degree (which would amount to modifying the circuit), computing x_1^δ for instance requires $\delta - 1$ multiplications instead of $\lfloor \log \delta \rfloor$ as with our method.

9.5.2 Protocol

Section 9.5.1 shows how to reduce the satisfiability of any polynomial in $\mathbb{Z}[x_1, \ldots, x_\nu]$ of total degree δ with μ monomials ($\mu \leq \binom{\nu+\delta}{\delta}$ necessarily) to the satisfiability of a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ integer vectors of length at most $\nu\lfloor \log \delta \rfloor + (\delta - 1)\mu$, and $1 + 2\nu(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for all $q \in [\![Q]\!]$, with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q}$ integer vectors and $c_q \in \mathbb{Z}$.

To argue for Diophantine satisfiability, it thus suffices to give a protocol protocol such relations. The following protocol is actually for more general relations in which variables of the polynomial can be committed (with the scheme in Section 9.2), which allows to argue on committed values while saving the cost of encoding the commitment scheme as an integer polynomial. More precisely, the protocol is for the relation

$$\begin{aligned}\Big\{\Big(\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}^{Q\times n}, \mathbf{W}_V \in \mathbb{Z}^{Q\times m}, \mathbf{V} \in \mathbb{G}^m, \mathbf{c} \in \mathbb{Z}^Q, \ell \in \mathbb{N}^*;\\ \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}^n, \mathbf{v}, \rho \in \mathbb{Z}^m\Big) :\\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}}\\ \wedge \forall i \in [\![m]\!]\, V_i^2 = (e^{v_i} f^{\rho_i})^4 \wedge \left\| \begin{bmatrix} \mathbf{a}_L & \mathbf{a}_R & \mathbf{a}_O & \mathbf{v} & \rho \end{bmatrix} \right\|_\infty < 2^\ell \Big\}\end{aligned}$$

given parameters $(\mathbb{G}, P, n, Q, m, f)$ such that $f \in \mathbb{G}$ and $n, Q, m \in \mathbb{N}^*$, and $(\mathbf{g}, \mathbf{h}, \pi_{crs}) \in \mathbb{G}^{2n} \times \{0,1\}^*$. For fixed parameters n, Q and m, Section 9.5.1 shows that the protocol allows to prove the satisfiability of any polynomial in $\mathbb{Z}[X_1, \ldots, X_\nu]$ of total degree δ and with μ monomials if $\nu\lfloor \log \delta \rfloor + (\delta - 1)\mu \leq n$ and $1 + 2\nu\,(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu + m \leq Q$. The additional term m in the number of constraints compared to the previous section is to ensure the consistency between the committed variables $\mathbf{v}$ and the ones in the inner product.

Bünz et al. [BBB+18] gave a protocol for a similar relation in $\mathbb{Z}_p$ instead of $\mathbb{Z}$ to argue for the satisfiability of arithmetic circuits over $\mathbb{Z}_p$ (without the bounds related to integer polynomials as it was not their target) that is inspired by the one of Bootle et al. [BCC+16]. The general idea of our protocol for this relation is similar to the two previous ones, but there are key differences that arise from the fact that $\mathbb{Z}$ is not a field. These differences are highlighted in the overview below

Building Blocks. The protocol builds mainly on the protocol on Figure 9.2, and on three auxiliary protocols: a protocol Π_{crs} to prove that the CRS is well-formed (as in Section 9.4.3), a protocol Π' to aggregate arguments of opening to integer commitments (see Section 9.4.2) and a protocol $\tilde{\Pi}$ to argue knowledge of an integer vector that opens to commitments in different bases (Section 9.4.4), i.e., a base-switching argument. These arguments may be in the random-oracle model with an oracle $\mathcal{H}$.

Main Insights.

The main idea of the protocol is to reduce the verification of the Hadamard product and of the linear constraints to a single inner-product argument over the integers, and then invoke the protocol on Figure 9.2.

Starting with the product $\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O = 0_{\mathbb{Z}^n}$, the idea to verify all n equations at once is to consider each entry as the coefficient of a n-variate polynomial of total degree 1 and evaluate the polynomial at a vector $\mathbf{y} \leftarrow_\$ [\![0; P-1]\!]^n$ chosen by the verifier by computing $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y} \rangle$. If the polynomial is non-zero, then the evaluation will be zero will only negligible probability (this is the analog to the Schwartz-Zippel lemma in integral rings), i.e., $\langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y} \rangle = \langle \mathbf{a}_O, \mathbf{y} \rangle$ with high probability. The reason we choose $\mathbf{y} \leftarrow_\$ [\![0; P-1]\!]^n$ instead of choosing $y \leftarrow_\$ [\![0; P-1]\!]$ and setting $\mathbf{y} \leftarrow \begin{bmatrix} 1 & y & \cdots & y^{n-1} \end{bmatrix}$ (as did Bootle et al. and Bünz et al., and in which case the Schwartz-Zippel argument would still apply) is that it makes the integers in $\langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y} \rangle$ only mP times larger than those in $\mathbf{a}_L \circ \mathbf{a}_R$ instead of P^m. Similarly, the Q equations $\mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}}$ are reduced to a single equation by multiplying on the left by a random vector $\mathbf{z} \leftarrow_\$ [\![0; P-1]\!]^Q$. Naturally, the prover must commit to the inputs $\mathbf{a}_L, \mathbf{a}_R$ and the outputs $\mathbf{a}_O$ in the witness before receiving the values $\mathbf{y}$ and $\mathbf{z}$ for the Schwartz-Zippel lemma to apply since the coefficients of the polynomial cannot depend on the random point at which it is later evaluated. The inputs are committed in a group element C_I with some bases $(\mathbf{g}, \mathbf{h})$ and the outputs in a group element C_O with bases $\mathbf{g}$.

Now the goal is to verify both $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y} \rangle = 0$ and $\mathbf{z}\mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{z}\mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{z}\mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} - \mathbf{z}\mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{z}\mathbf{c}^{\mathrm{T}} = 0$ with a single inner-product. To do so, we introduce another variable X and construct two vectors of polynomials of small degree $l(X)$ and $r(X)$ in $\mathbb{Z}^n[X]$ of which the inner product is

denoted $t(X) \coloneqq \langle l(X), r(X)\rangle$. The coefficients of $l(X)$ and $r(X)$ depend on the two relations to be proved, and are designed so that some coefficients of $t(X)$ force these relations while being computable from public values, i.e., so that the verifier can prevent the prover from cheating. The explicit construction of $l(X)$ and $r(X)$ is given below. After the prover has committed to the coefficient of $t(X)$ *that the verifier cannot compute on his own*, this polynomial is be evaluated at a random point x chosen by the verifier. However, the prover cannot simply send $t(x)$ to the verifier as it contains information about the witness, and adding a random integer will not help as $\mathbb{Z}$ is infinite unlike $\mathbb{Z}_p$. This is the first main difference with the argument of Bünz et al. For this reason, the verifier cannot check that $t(x)$ is correctly computed with the committed coefficients of t, and the argument cannot be directly reduced to the full inner-product argument in Section 9.3.

Fortunately, the prover can instead argue that she knows the committed coefficients of t and the openings to the committed $\mathbf{v}$ with the aggregated argument Π', assuming that they are committed in the same base (e, f). After these arguments, we notice that the verifier is able to compute $e^{t(x)}$ at any x of her choice from the coefficients of t that she could compute on her own (and which force the relations of interest with $\mathbf{y}$ and $\mathbf{z}$) and the committed coefficients. The verifier can then choose x, send it to the prover, and then together proceed with an argument of knowledge of $l(x)$ and $r(x)$ of which the inner product is committed to with the base e using the protocol on Figure 9.2.

It now remains to define such polynomials $l(X)$ and $r(X)$ in $\mathbb{Z}^n[X]$. Consider first the relation with $\mathbf{z}$. The goal is to make use of $\mathbf{W}_V$, $\mathbf{V}$ and $\mathbf{c}$ which are public to force the equality with the rest that contains private parts by making sure that the coefficient of $t(X)$ will be computable from the public information. As the relation is an inner product itself, notice that $\mathbf{a}_L$ and $\mathbf{z}\mathbf{W}_L$ cannot be in the same polynomial, and similarly for $\mathbf{a}_R$ and $\mathbf{z}\mathbf{W}_R$ and $\mathbf{a}_O$ and $\mathbf{z}\mathbf{W}_O$. If the relation $\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O = 0_{\mathbb{Z}^n}$ were already enforced (with $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y}\rangle = 0$), one could put $\mathbf{a}_L$ and $\mathbf{z}\mathbf{W}_R$ at the same degree in $l(X)$, put $\mathbf{a}_R$ and $\mathbf{z}\mathbf{W}_L$ at the same degree in $r(X)$, and make sure that the sum of those degrees is equal to the degree at which $\mathbf{a}_0$ is in one of the polynomials (say $l(X)$), and leverage the equality $\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O = 0_{\mathbb{Z}^n}$ to eliminate parasite terms that are not publicly computable. With this reasoning, the minimal degree $\mathbf{a}_0$ can be is then 2, and the others at degree 1.

The relation $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y}\rangle = 0_{\mathbb{Z}^n}$ must still be proved, and unfortunately since one cannot invert integers modulo the unknown orders of the bases, this verification cannot be embedded in the same degrees as the relation with $\mathbf{z}$. It must then be shifted to higher degrees, and a different polynomial than that of Bünz et al. must be defined (this is the second main difference). The same reasoning as before leads to additional terms $\mathbf{a}_L X^3 - \mathbf{a}_O X^4$ in $l(X)$ and to additional terms $\mathbf{y}X^2 - \mathbf{y} \circ \mathbf{a}_R X^3$ in $r(X)$,

and the term of degree 6 of $t(X)$ is then 0.

The polynomials $l(X)$ and $r(X)$ are then respectively defined as $(\mathbf{a}_L + \mathbf{z}\mathbf{W}_R)X + \mathbf{a}_O X^2 + \mathbf{a}_L X^3 - \mathbf{a}_O X^4$ and $-\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L)\,X + \mathbf{y}X^2 + \mathbf{y} \circ \mathbf{a}_R X^3$. Note that for $t(X) \coloneqq \langle l(X), r(X)\rangle = \sum_{i=1}^{7} t_i X^i$ with

$$\begin{aligned} t_2 &= \langle \mathbf{a}_L, \mathbf{z}\mathbf{W}_L\rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R\rangle + \langle \mathbf{a}_O, \mathbf{z}\mathbf{W}_O\rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{z}\mathbf{W}_L\rangle \\ &\quad - \langle \mathbf{a}_O, \mathbf{1}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R\rangle \\ &= \langle \mathbf{a}_L, \mathbf{z}\mathbf{W}_L\rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R\rangle + \langle \mathbf{a}_O, \mathbf{z}\mathbf{W}_O\rangle + \underbrace{\langle \mathbf{z}\mathbf{W}_R, \mathbf{z}\mathbf{W}_L\rangle}_{\delta(\mathbf{z})} \\ &= \langle \mathbf{z}\mathbf{W}_V, \mathbf{v}\rangle + \delta(\mathbf{z}) \quad \text{and} \\ t_6 &= \langle \mathbf{a}_L, \mathbf{y} \circ \mathbf{a}_R\rangle - \langle \mathbf{a}_O, \mathbf{y}\rangle = \langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y}\rangle = 0. \end{aligned}$$

Since $\mathbf{c}$ is public and the verifier can compute $\delta(\mathbf{z})$ on his own, the verifier can compute e^{t_2} from the commitments $\mathbf{V}$.

A last hurdle arises from the fact the verifier must be guaranteed that the vector $\mathbf{a}_R$ in the term $\mathbf{y} \circ \mathbf{a}_R X^3$ of $r(X)$ is really the same as the input vector to which the prover committed to at the beginning of the protocol. In $\mathbb{Z}_p$, one can easily do so by simply interpreting a commitment of the form $\mathbf{h}^{\mathbf{a}_R}$ as a commitment to $\mathbf{y} \circ \mathbf{a}_R$ in the basis $\begin{bmatrix} h_1^{y_1^{-1}} & \cdots & h_n^{y_n^{-1}} \end{bmatrix}$. Nonetheless, the order of the group elements h_i are unknown in the present case (this is the third major difference with the proofs in groups of public order). That is why the prover must commit to $\mathbf{a}_L$ and $\mathbf{y} \circ \mathbf{a}_R$ in a new group element C_I' after receiving $\mathbf{y}$ and argue that $(\mathbf{a}_L, \mathbf{a}_R)$ opens to both C_I and C_I' respectively in the bases $(\mathbf{g}, \mathbf{h})$ and $(\mathbf{g}, \mathbf{h}')$ with $\mathbf{h}' = \begin{bmatrix} h_1^{y_1} & \cdots & h_n^{y_n} \end{bmatrix}$. To do so, the prover and the verifier run the base-switching argument $\tilde{\Pi}$.

Protocol Algorithms.

The protocol is denoted Π. The parameter-generation algorithm and the CRS generator are as in Section 9.3.1. The algorithms of the prover and the verifier are given on Figure 9.3. On that figure, $\mathbf{W}$ denotes the matrix $\begin{bmatrix} \mathbf{W}_L & \mathbf{W}_R & \mathbf{W}_O & \mathbf{W}_V \end{bmatrix}$. The values ℓ', $\tilde{\ell}$ and $\ell_{9.2}$ are given in Section 9.5.2.

Prover-Communication Complexity.

To estimate the communication complexity of the entire protocol from Figure 9.3, we estimate the complexity of each of its sub-protocol. To do so, one must assess the bit length of the integer witnesses given to each sub-protocol of the protocol on Figure 9.3, and this requires to give upper bounds on the polynomial evaluations in the protocol.

Heights of $l(X)$ and $r(X)$. Since $l(X) = (\mathbf{a}_L + \mathbf{z}\mathbf{W}_R)\,X + \mathbf{a}_O X^2 + \mathbf{a}_L X^3 - \mathbf{a}_O X^4 \in \mathbb{Z}^n[X]$, the height (i.e., the maximum of the absolute values of the

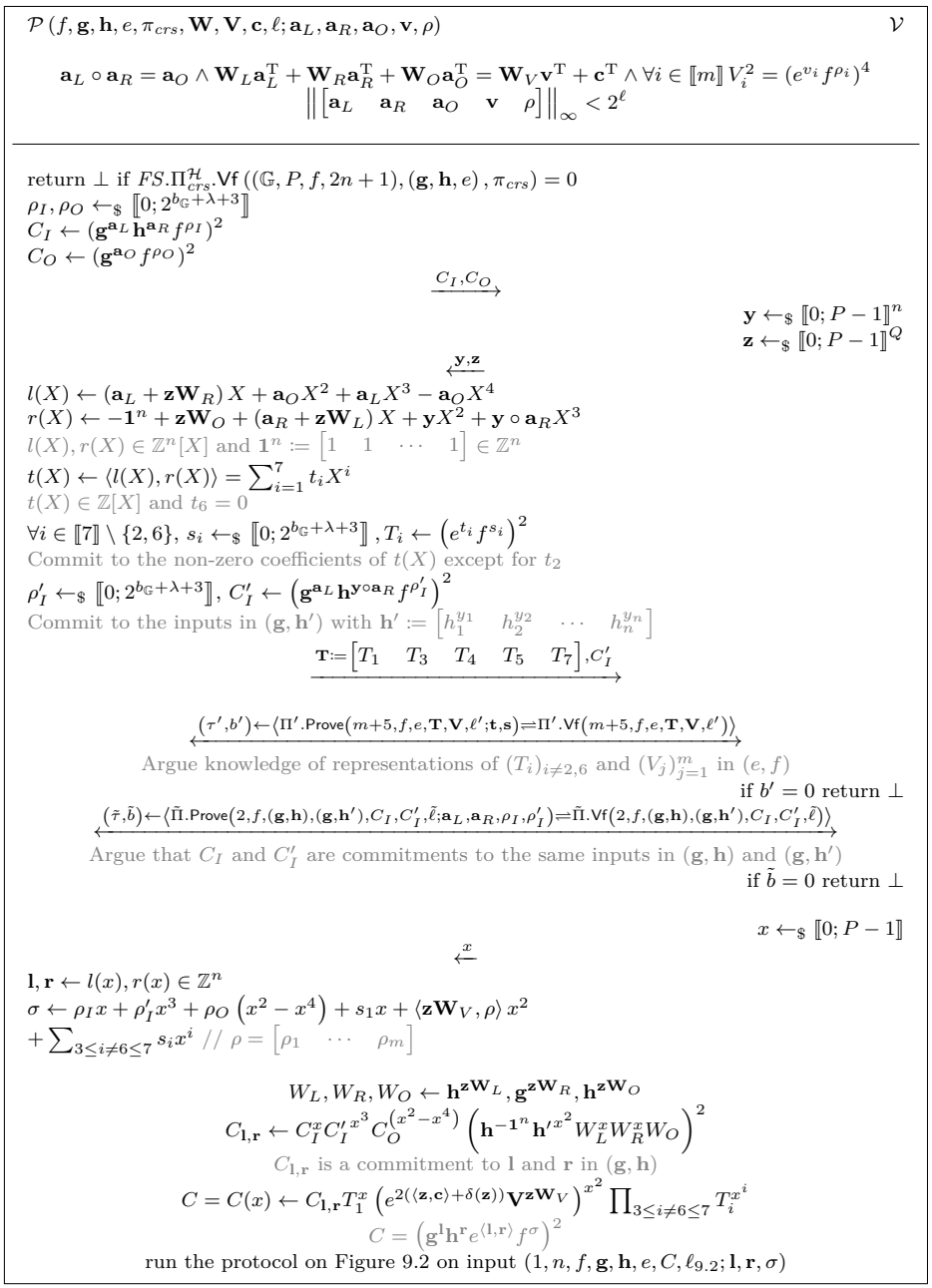

Figure 9.3: Succinct Argument of Diophantine-Equation Satisfiability.

coefficients) of the polynomials in $l(X)$ is at most $2^\ell - 1 + Q \|W\|_\infty (P-1)$ (recall that $\mathbf{W} = \begin{bmatrix}\mathbf{W}_L & \mathbf{W}_R & \mathbf{W}_O & \mathbf{W}_V\end{bmatrix}$). Concerning $r(X) = -\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L) X + \mathbf{y}X^2 + \mathbf{y} \circ \mathbf{a}_R X^3$, the height of the polynomials in $r(X)$ is at most

$$\max\left(2^\ell - 1 + Q \|W\|_\infty (P-1), (P-1)(2^\ell - 1)\right) \leq P\left(2^\ell - 1 + Q \|W\|_\infty\right).$$

Prover Complexity in Π'. The height of $t(X) = \langle l(X), r(X)\rangle$ is at most

$$7nP\left(2^\ell - 1 + Q \|W\|_\infty P\right)\left(2^\ell - 1 + Q \|W\|_\infty\right).$$

The bit length of the height of $t(X)$ is thus at most

$$\begin{aligned}
&3 + \lfloor \log n \rfloor + \lfloor \log P \rfloor + 2 + \max\left(\ell, \lfloor \log(Q \|W\|_\infty) \rfloor + 1\right) + 1 \\
&\quad + \max\left(\ell, \lfloor \log P \rfloor + \lfloor \log(Q \|W\|_\infty) \rfloor + 2\right) + 1 \\
&= 7 + \lfloor \log n \rfloor + \lfloor \log P \rfloor + \max\left(\ell, \lfloor \log(Q \|W\|_\infty) \rfloor + 1\right) \\
&\quad + \max\left(\ell, \lfloor \log P \rfloor + \lfloor \log(Q \|W\|_\infty) \rfloor + 2\right) \\
&\leq 10 + \lfloor \log n \rfloor + 2\left(\lfloor \log P \rfloor + \ell + \lfloor \log(Q \|W\|_\infty) \rfloor\right).
\end{aligned}$$

Therefore, the maximum bit length of the witness in this call is at most

$$\ell' \leftarrow \max\left(b_{\mathbb{G}} + \lambda + 4, 10 + \lfloor \log n \rfloor + 2\left(\lfloor \log P \rfloor + \ell + \lfloor \log(Q \|W\|_\infty) \rfloor\right)\right).$$

Π' is used to aggregate $m + 5$ arguments, so the bit-communication complexity of Π' is of order $O\left(\ell' + \log(m+5) + \log P + \log(n) b_{\mathbb{G}}\right)$, i.e.,

$$O\left(\ell + \log(n) b_{\mathbb{G}} + \log Q + \log(m+5) + \log \|W\|_\infty\right).$$

Prover Complexity in $\tilde{\Pi}$. The bit length of the witness in this call is at most $\tilde{\ell} \leftarrow \max\left(b_{\mathbb{G}} + \lambda + 4, \ell\right)$. As $\tilde{\Pi}$ is used to argue about two bases, the bit complexity of $\tilde{\Pi}$ is of order $O\left(\tilde{\ell} + \log P + \log(n) b_{\mathbb{G}}\right)$, i.e., $O\left(\ell + \log(n) b_{\mathbb{G}}\right)$.

Prover Complexity in the Protocol on Figure 9.2. The largest absolute value of the components in $\mathbf{l} = l(x)$ and $\mathbf{r} = r(x)$ is at most $P^5\left(2^\ell - 1 + Q \|W\|_\infty\right)$. Besides, the height of the polynomial

$$\sigma(X) \coloneqq \rho_I X + \rho'_I X^3 + \rho_O\left(X^2 - X^4\right) + s_1 X + \langle \mathbf{z}\mathbf{W}_V, \rho\rangle X^2 + \sum_{3 \leq i \neq 6 \leq 7} s_i x^i$$

is at most $2^{b_{\mathbb{G}}+\lambda+3} + mQ \|W\|_\infty P(2^\ell - 1)$ (given by the term of degree 2). The absolute value of randomness σ is thus at most $P^8\left(2^{b_{\mathbb{G}}+\lambda+3} + mQ \|W\|_\infty P\left(2^\ell - 1\right)\right)$. Therefore, the bit length of the integer witnesses (in absolute value) in the execution of the protocol on Figure 9.2 is at most

$$\ell_{9.2} \leftarrow 5 + \lfloor \log P \rfloor + \max\left(b_{\mathbb{G}} + \lambda + 4, \ell + \lfloor \log\left(mQ \|W\|_\infty P\right) \rfloor + 1\right).$$

In case $m = 0$, it is only $b_{\mathbb{G}} + \lambda + 4$. The bit complexity of the call to this protocol is then of order

$$O\left(\ell + \log(n) b_{\mathbb{G}} + \log Q + \log m + \log \|W\|_{\infty}\right).$$

(The term $\log m$ vanishes in case $m = 0$.)

Overall Prover-Communication Complexity. Based on the previous estimations, the prover sends $O\left(\ell + \log(n) b_{\mathbb{G}} + \log Q + \log m + \log \|W\|_{\infty}\right)$ bits during the protocol (the term $\log m$ disappears in case $m = 0$). Therefore, for a polynomial in $\mathbb{Z}[X_1, \ldots, X_\nu]$ of total degree δ, with μ monomials and with coefficients less than 2^H in absolute value, assuming that $\nu \lfloor \log \delta \rfloor + (\delta - 1)\mu \leq n$ and $1 + 2\nu\left(\lfloor \log \delta \rfloor - 1\right) + (\delta - 2)\mu + m \leq Q$, the communication complexity of the protocol is of order

$$O\left(\ell + \log\left(\delta \binom{\nu + \delta}{\delta}\right) b_{\mathbb{G}} + H\right) = O\left(\ell + \min(\nu, \delta) \log\left(\nu + \delta\right) b_{\mathbb{G}} + H\right).$$

The term $H = \lfloor \log \|W\|_{\infty} \rfloor + 1$ comes from the fact that the procedure gives linear constraints determined by the coefficients of the polynomial.

Verification Effiency.

Similarly to Section 9.3.1, the verifications of Π', $\tilde{\Pi}$ and the protocol on Figure 9.2 can each be done via single multi-exponentiations, with exponents of at most $O\left(\ell + b_{\mathbb{G}} + \log(n) \log(P) + \log Q + \log m + \log \|W\|_{\infty}\right)$ bits. For a polynomial in $\mathbb{Z}[X_1, \ldots, X_\nu]$ of total degree δ, with μ monomials and with coefficients less than 2^H in absolute value, that is $O\left(\ell + b_{\mathbb{G}} + \min(\nu, \delta) \log\left(\nu + \delta\right) \log P + H\right)$ bits.

9.5.3 Completeness and Security

This section formally states the properties achieved by the protocol.

Theorem 9.5.4 (Completeness)**.** Π *is complete if* Π_{crs}*,* Π' *and* $\tilde{\Pi}$ *are.*

Proof. The completeness of Π immediately follows from its definition and from the completeness of Π_{crs}, Π' and $\tilde{\Pi}$, and the completeness of the protocol on Figure 9.2. □

Theorem 9.5.5 (Honest-Verifier Zero-Knowledge Property)**.** *If* Π' *and* $\tilde{\Pi}$ *are respectively* $(T_{\Pi'}, T_{\Pi'.\mathsf{Sim}}, \varepsilon_{\Pi'})$ *and* $(T_{\tilde{\Pi}}, T_{\tilde{\Pi}.\mathsf{Sim}}, \varepsilon_{\tilde{\Pi}})$*-honest-verifier zero-knowledge, and if* $FS.\Pi^{\mathcal{H}}_{crs}$ *is is* $\left(T_{\Pi^{\mathcal{H}}_{crs}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}_{\Pi^{\mathcal{H}}_{crs}}\right)$*-sound, then* Π *is* $(T, O(b_{\mathbb{G}}) + T_{\Pi'.\mathsf{Sim}} + T_{\tilde{\Pi}.\mathsf{Sim}} + T_{9.2.\mathsf{Sim}}, \varepsilon^{\mathrm{zk}}_{\Pi^{\mathcal{H}}_{crs}} + \varepsilon_{\Pi'} + \varepsilon_{\tilde{\Pi}} + \varepsilon_{9.2})$*- honest-verifier zero-knowledge for* $T \leq \min(T_{\Pi^{\mathcal{H}}_{crs}}, T_{\Pi'}, T_{\tilde{\Pi}}, T_{9.2})$*, where* $(T_{9.2}, T_{9.2.\mathsf{Sim}}, \varepsilon_{9.2})$ *denote the bounds from Theorem 9.3.5.*

Proof. It suffices to define a simulator which, instead of computing the commitments $C_I, C_O, C'_I, T_1, T_3, T_4, T_5, T_7$ in the protocol, computes elements as f^α for $\alpha \leftarrow_\$ [\![0; 2^{b_\mathbb{G}+\lambda+3}]\!]$. These are then at a statistical distance of at most $2^{-\lambda}$ from the values computed in a real protocol execution, unless the CRS is ill-formed (which occurs with probability at most $\varepsilon^{\mathrm{zk}}_{\Pi^{\mathcal{H}}_{crs}}$). The protocol also runs the simulator of Π', $\tilde{\Pi}$ and the simulator of the protocol on Figure 9.2. □

Theorem 9.5.6 (Extractability)**.** *If $FS.\Pi_{crs}$ is $\left(T_{\tilde{\Pi}_{crs}.\mathsf{Sim}}, q_{\Pi_{crs}}, \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}}\right)$-statistically honest-verifier zero-knowledge, Π' is $(T_{\Pi',\mathcal{A}}, T_{\Pi',\mathsf{Prove}^*}, T_{\Pi'.\mathcal{E}}, q_{\Pi'}, \varepsilon^{\mathrm{ext}}_{\Pi'})$-extractable and $\tilde{\Pi}$ is $\left(T_{\tilde{\Pi}}, q_{\tilde{\Pi}}, \varepsilon^{\mathrm{snd}}_{\tilde{\Pi}}\right)$-sound, then under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, then Π is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{ext}}, \Sigma)$-extractable for $T_{\mathcal{A}}$ and T_{Prove^*} such that $T_{\mathcal{A}}+T_{\mathsf{Prove}^*} \leq \min\left(T_{\tilde{\Pi}.\mathsf{Sim}}, T_{\tilde{\Pi}}\right)$ and $T_{\mathcal{A}}$ and T_{Prove^*} satisfy the bounds for $T_{\Pi',\mathcal{A}}$ and $T_{\Pi',\mathsf{Prove}^*}$ and for $T_{9.2,\mathcal{A}}$ and $T_{9.2,\mathsf{Prove}^*}$, $T_{\mathcal{E}}$ explicited in the proof of the theorem, $q_{\mathcal{H}} \leq \min\left(q_{\Pi_{crs}}, q_{\tilde{\Pi}}, q_{\Pi'}\right)$ and $\varepsilon \leq \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}} + \varepsilon^{\mathrm{ext}}_{\Pi'} + \varepsilon^{\mathrm{snd}}_{\tilde{\Pi}} + \varepsilon_{9.2}$, where $T_{9.2,\mathcal{A}}$, $T_{9.2,\mathsf{Prove}^*}$ and $\varepsilon_{9.2}$ denote the bounds from Theorem 9.3.9.*

Proof. Suppose that for fixed vectors $\mathbf{y}$ and $\mathbf{z}$ and nine pairwise-distinct challenges $x_1, \ldots, x_9$, one can obtain representations $\left(\mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}}e^{\langle \mathbf{l},\mathbf{r}\rangle}f^\sigma\right)^4$ of $C^2(x_j)$ for $j \in [\![8]\!]$ ($\mathbf{l}$, $\mathbf{r}$ and σ depend on x_j). Denoting by $\mathbf{X}$ the Vandermonde matrix of $x_1, \ldots, x_8$, by $\mathrm{adj}(X)$ its adjugate matrix and by $\mathbf{e}_j$ (for $j \in [\![8]\!]$) the canonical row vector with 1 at position j and 0 elsewhere, the linear equations $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}_j^{\mathrm{T}}$ with indeterminate $\nu \in \mathbb{Z}^8$ have unique solutions if $\det \mathbf{X} \neq 0$, and these solutions are $\mathrm{adj}(\mathbf{X})\mathbf{e}_j^{\mathrm{T}}$. Therefore, one can solve the equation $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}_2^{\mathrm{T}}$ and compute via linear combinations integer vectors $\mathbf{a}_L$ and $\mathbf{a}_R$ and an integer ρ_I such that $C_I^{2\det\mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}e^{\langle \mathbf{a}_L,\mathbf{a}_R\rangle}f^{\rho_I}\right)^4$ since

$$C^2(x) = \left(C_I^x {C'_I}^{x^3} C_O^{x^2-x^4}\right)^2 \left(\mathbf{h}^{-\mathbf{1}^n+\mathbf{y}x^2}\mathbf{h}^{x\mathbf{z}\mathbf{W}_L}\mathbf{g}^{x\mathbf{z}\mathbf{W}_R}\mathbf{h}^{\mathbf{z}\mathbf{W}_O}\right)^4 \left(T_1^x\left(e^{2(\langle \mathbf{z},\mathbf{c}\rangle+\delta(\mathbf{z}))}\mathbf{V}^{\mathbf{z}\mathbf{W}_V}\right)^{x^2}\prod_{3\leq i\neq 6\leq 7} T_i^{x^i}\right)^2 \tag{9.10}$$

for all $x \in \{x_1, \ldots, x_9\}$. Likewise, by considering the equation $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}_4^{\mathrm{T}}$, one can compute integer vector $\mathbf{a}'_L$ and $\mathbf{a}'_R$ and an integer ρ'_I such that ${C'_I}^{2\det\mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}'_L}\mathbf{h}^{\mathbf{a}'_R}e^{\langle \mathbf{a}'_L,\mathbf{a}'_R\rangle}f^{\rho'_I}\right)^4$. Moreover, let $\mathbf{e}$ be the row vector with 1 at position 3, -1 at position 5 and 0 elsewhere. The equation $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}^{\mathrm{T}}$ has a unique solution if $\det(\mathbf{X}) \neq 0$, and it is $\mathrm{adj}(\mathbf{X})\mathbf{e}^{\mathrm{T}}$. It follows that one can compute integer vectors $\mathbf{a}_{O,L}$ and $\mathbf{a}_{O,R}$ and an integer ρ_O such that $C_O^{2\det\mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}_{O,L}}\mathbf{h}^{\mathbf{a}_{O,R}}e^{\langle \mathbf{a}_{O,L},\mathbf{a}_{O,R}\rangle}f^{\rho_O}\right)^4$.

Besides, assume to be given integers t_j and s_j for $j \in [\![7]\!] \setminus \{2,6\}$, and v_i and ρ_i for $i \in [\![m]\!]$ such that $T_j^2 = \left(e^{t_j} f^{s_j}\right)^4$ and $V_i^2 = \left(e^{v_i} f^{\rho_i}\right)^4$.

Lemma 9.3.7 shows that $2 \det X$ divides all the integers in the representations of $C_I^{2 \det \mathbf{X}}$, $C_I'^{2 \det \mathbf{X}}$ and $C_O^{2 \det \mathbf{X}}$ unless one can find non-trivial discrete-logarithm relations (the probability of this latter event is already accounted for in the bounds of Theorem 9.3.9). That is to say, one can obtain representations of C_I^2, $C_I'^2$ and C_O^2 (these integers obtained by dividing by $2 \det X$ are further denoted as before). Inserting these representations in Equation 9.10 for any $x \in \{x_1, \ldots, x_9\}$, it follows that

$$\begin{aligned}
\mathbf{l} &= (\mathbf{a}_L + \mathbf{z}\mathbf{W}_R)\, x + \mathbf{a}_{O,L} x^2 + \mathbf{a}'_L x^3 - \mathbf{a}_{O,L} x^4 \\
\mathbf{r} &= -\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L)\, x + \mathbf{y} x^2 + \mathbf{a}'_R x^3 + \mathbf{a}_{O,R} \left(x^2 - x^4\right)
\end{aligned}$$

and that

$$\begin{aligned}
\langle \mathbf{l}, \mathbf{r} \rangle = {} & \langle \mathbf{a}_L, \mathbf{a}_R \rangle\, x + \langle \mathbf{a}'_L, \mathbf{a}'_R \rangle\, x^3 + \langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle \left(x^2 - x^4\right) \\
& + s_1 x + \left(\langle \mathbf{z}\mathbf{W}_V, \mathbf{v} \rangle + \langle \mathbf{z}, \mathbf{c} \rangle + \delta(\mathbf{z})\right) x^2 + \sum_{3 \le i \ne 6 \le 7} t_i x^i,
\end{aligned}$$

unless one can obtain a non-trivial discrete-logarithm relation in $\langle f \rangle$. As the equation is satisfied for nine values of x although the polynomials are of degree at most 8 in x, then one can infer from the terms of degree 2, 6 and 8 that

$$\begin{aligned}
0 = {} & \langle \mathbf{a}_L, \mathbf{z}\mathbf{W}_L \rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R \rangle + \langle \mathbf{a}_{O,L}, \mathbf{z}\mathbf{W}_O \rangle - \langle \mathbf{a}_{O,L}, \mathbf{1}^n \rangle \\
& + \langle \mathbf{a}_L, \mathbf{a}_R \rangle - \langle \mathbf{z}\mathbf{W}_V, \mathbf{v} \rangle - \langle \mathbf{z}, \mathbf{c} \rangle - \langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle \\
0 = {} & \langle \mathbf{a}'_L, \mathbf{a}'_R \rangle - \langle \mathbf{a}_{O,L}, \mathbf{y} \rangle \quad \text{and} \\
0 = {} & \langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle .
\end{aligned}$$

Assuming an additional guarantee that $\mathbf{a}'_L = \mathbf{a}_L$ and $\mathbf{a}'_R = \mathbf{y} \circ \mathbf{a}_R$, these equations thus imply that

$$\begin{gathered}
\langle \mathbf{z}\mathbf{W}_L, \mathbf{a}_L \rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R \rangle + \langle \mathbf{z}\mathbf{W}_O, \mathbf{a}_{O,L} \rangle = \langle \mathbf{z}\mathbf{W}_V, \mathbf{v} \rangle + \langle \mathbf{z}, \mathbf{c} \rangle \quad \text{and} \\
\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_{O,L}, \mathbf{y} \rangle = 0.
\end{gathered}$$

If these equalities are verified for m vectors $\mathbf{y}_1, \ldots, \mathbf{y}_m \in \mathbb{Z}^m$ that are linearly independent over $\mathbb{Q}$ and for Q vectors $\mathbf{z}_1, \ldots, \mathbf{z}_Q \in \mathbb{Z}^Q$ that are linearly independent over $\mathbb{Q}$, then $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_{O,L}$ and $\mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_{O,L}^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}}$, with v_i committed in V_i with randomness ρ_i for all $i \in [\![m]\!]$. In other words, $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_{O,L}, \mathbf{v}, \rho$ is a valid witness.

The idea is now to gradually define an extractor for the entire protocol with a bottom-up approach starting from the extractor for the protocol on Figure 9.2 assuming vectors $\mathbf{y}, \mathbf{z}$ and an integer x to be fixed. The next extractor $\mathcal{E}_2$ builds on the previous extractor and can extract a witness given

fixed vectors $\mathbf{y}$ and $\mathbf{z}$ (sent as the first message from the verifier) as inputs. The following oracle $\mathcal{E}_1$ takes a fixed vector $\mathbf{y}$ and builds on $\mathcal{E}_2$. The last extractor $\mathcal{E}$ takes no fixed values except for the protocol inputs builds then builds on $\mathcal{E}_1$.

The running time of each of these extractor has a success probability which of course depends on the success probability of the prover with the fixed inputs, and for instance, what $\mathcal{E}_1$ does is simply to generate a vector $\mathbf{z}$ that leads to a success, and then using a heavy-row argument, one can show that with probability $1/2$ over the choice of $\mathbf{z}$, the success of the prover decreases by a factor at most $1/2$ if $\mathbf{z}$ is fixed to the chosen value, and then $\mathcal{E}_1$ can run $\mathcal{E}_2$ with its fixed inputs and $\mathbf{z}$, and extract a witness in reasonable expected time. The idea for the other steps is the same.

Consider now an algorithm $\mathcal{E}_2$ that runs the prover from the computation step right after it receives $\mathbf{y}$ and $\mathbf{z}$, and suppose that for some fixed vectors $\mathbf{y}$ and $\mathbf{z}$, the success probability of the prover is at least $\eta > 0$. Algorithm $\mathcal{E}_2$ runs $\Pi'.\mathcal{E}$ on the prover to extract representation of $(T_i)_{i\neq 2,6}$ and $(V_j)_{j=1}^m$. It continues by executed protocol $\tilde{\Pi}$ with the prover. As $\tilde{\Pi}$ is assumed to be sound, the extractor is guaranteed that with probability at least $1-\varepsilon_{\tilde{\Pi}}^{\mathrm{snd}}$, the inputs committed in C_I with bases $(\mathbf{g}, \mathbf{h})$ are the same as those committed in C_O with bases $(\mathbf{g}, \mathbf{h}')$ if the execution of $\tilde{\Pi}$ succeeds.

$\mathcal{E}_2$ generates $x \leftarrow_{\$} [\![0; P-1]\!]$, sends it to the prover and runs the protocol until the end. If the execution fails, $\mathcal{E}_2$ generates a fresh value x and proceeds as before. If the execution succeeds, then a heavy-row argument implies that with probability at least $1/2$ (over the choice of x), x is such that the prover succeeds in the rest of the protocol with probability at least $\eta/2$. Algorithm $\mathcal{E}_2$ then computes $C(x)$ as in the real protocol and runs the extractor of the protocol on Figure 9.2 (denote it $\Pi_{9.2}.\mathcal{E}$). If this latter does not return a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $\eta/2$ (Markov's inequality shows that this event occurs with probability at most $1/2$), algorithm $\mathcal{E}_2$ generates a fresh value x and proceeds as before. If it does, $\mathcal{E}_2$ returns the same value.

Denote by H the event in which x is such that the verifier of is convinced with probability at least $\eta/2$ and by T the event in which $\Pi_{9.2}.\mathcal{E}$ returns a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $\eta/2$. Algorithm $\mathcal{E}_2$ returns a value in the event $H \cap T$ and $\Pr[T \cap H] = \Pr[T|H]\Pr[H] \geq 1/4$, so $\mathcal{E}_2$ restarts from the generation of x an expected number of times at most 4. Furthermore, at each repetition, its running time is at most $T_{\mathsf{Prove}^*}/\eta + T_{\Pi_{9.2}.\mathcal{E}}(\eta/2)$, with $T_{\Pi_{9.2}.\mathcal{E}}(\eta/2)$ denoting the expected running time of $\Pi_{9.2}.\mathcal{E}$ with a prover that succeeds with probability at least $\eta/2$. Therefore, the expected running time given a prover that succeeds with probability at least η is at most $4\left(T_{\mathsf{Prove}^*}/\eta + 2T_{\Pi_{9.2}.\mathcal{E}}(\eta/2)\right)$.

$\mathcal{E}_2$ then repeats this process eight other times, with the restriction that

the new challenge x_j (for $j = 2, \ldots, 9$) is distinct from the ones previously chosen ones, i.e., x_j has a uniform distribution over $[\![0; P-1]\!] \backslash \{x_1, \ldots, x_{j-1}\}$. Note that in this case the prover succeeds with probability at least $\eta - (j-1)/P \geq \eta - 8/P =: \eta'$. Therefore, the total running time of $\mathcal{E}_2$ is at most $36\,(T_{\mathsf{Prove}^*}/\eta' + 2T_{\Pi_{9.2}.\mathcal{E}}(\eta'/2))$.

Now, for a fixed vector $\mathbf{y}_1 \in \mathbb{Z}^n$, suppose that a prover convinces the verifier with probability at least $\theta > 0$ conditioned on the vector $\mathbf{y}$ in the first message from the verifeir being $\mathbf{y}_1$; and consider an algorithm $\mathcal{E}_1$ which has black-box access to a prover Prove^* and proceeds as follows. $\mathcal{E}_1$ starts by generating a vector $\mathbf{z}_1$. If $\mathbf{z}_1 = 0_{\mathbb{Z}^Q}$, then it generates new vectors and otherwise playing the role of the verifier, runs with the prover the entire protocol with $(\mathbf{y}_1, \mathbf{z}_1)$ as first message from the verifier. If the execution is unsuccessful, then $\mathcal{E}_1$ rewinds Prove^* to the beginning of the protocol and generates a new vector $\mathbf{z}_1$.

Note that conditioned on the event in which $\mathbf{z}_1$ is non-zero, Prove^* convinces the verifier with probability at least $\theta - P^{-Q}$. Moreover, conditioned on the event in which the first message $(\mathbf{y}_1, \mathbf{z}_1)$ leads to at least one successful execution, a heavy-row argument implies that with probability at least $1/2$ (over the choice of $\mathbf{z}_1$), the first message $(\mathbf{y}_1, \mathbf{z}_1)$ is such that prover Prove^* convinces the verifier with probability at least $\left(\theta - P^{-Q}\right)/2$.

Algorithm $\mathcal{E}_1$ then runs $\mathcal{E}_2$ on Prove^* from the computation step right after the first message from the verifier is sent. If $\mathcal{E}_2$ returns a value in at most twice its expected running time with a prover that succeeds with probability at least $\eta \coloneqq \left(\theta - P^{-Q}\right)/2$, then $\mathcal{E}_1$ returns that value and otherwise rewinds Prove^* to the beginning of the protocol and generates fresh vectors $\mathbf{y}_1$ and $\mathbf{z}_1$.

Similarly to the analysis of $\mathcal{E}_2$, the expected running of $\mathcal{E}_1$ is at most

$$4\left(\eta^{-1}\left(1 - P^{-Q}\right)^{-1} T_{\mathsf{Prove}^*} + 2T_{\mathcal{E}_2}(\eta/2)\right).$$

The term $\left(1 - P^{-Q}\right)^{-1}$ simply comes from the expected time necessary to generate a non-zero vector $\mathbf{z}_1$.

Now, for $j = 2, \ldots, Q$, algorithm $\mathcal{E}_2$ generates $\mathbf{z}_j \leftarrow_{\$} [\![0; P-1]\!]^Q$. If $\mathbf{z}_1, \ldots, \mathbf{z}_j$ are linearly dependent over $\mathbb{Q}$, then $\mathcal{E}_2$ generates a new vector $\mathbf{z}_j$. Slinko [Sli01, Corollary 2] proved that this event occurs with probability at most P^{-Q+j-1}. Consequently, conditioned on the event in which $\mathbf{z}_1, \ldots, \mathbf{z}_j$ are linearly independent, Prove^* convinces the verifier with probability at least $\theta - P^{-Q+j-1} \geq \theta - P^{-1}$. Algorithm $\mathcal{E}_2$ then proceeds as in the case $j = 1$ with $\theta - P^{-1}$ instead of $\theta - P^{-Q}$.

The total running time of $\mathcal{E}_1$ is thus at most

$$4Q\left(\eta^{-1}\left(1 - P^{-1}\right)^{-1} T_{\mathsf{Prove}^*} + 2T_{\mathcal{E}_2}(\eta/2)\right)$$

with $\eta \coloneqq \left(\theta - P^{-1}\right)/2$. Note that if any two vectors $\mathbf{z}_i, \mathbf{z}_j$ for $i, j \in [\![Q]\!]$ lead to distinct witness returned by $\mathcal{E}_2$, then one obtains a non-trivial discrete-logarithm relation in $\langle f \rangle$.

The extractor $\mathcal{E}$ of the entire protocol can now be defined similarly to $\mathcal{E}_1$ to generate linearly independent vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{Z}^n$. Assuming that it has black-box access to a prover that succeeds with probability at least ε, its running time is at most

$$4n\left(\theta^{-1}\left(1 - P^{-1}\right)^{-1} T_{\mathsf{Prove}^*} + 2T_{\mathcal{E}_1}(\theta/2)\right)$$

with $\theta \coloneqq \left(\varepsilon - P^{-1}\right)/2$. □

9.6 Applications

In this section, we apply the Diophantine-satisfiability argument from Section 9.5 to several computational problems. The methodology is always the same: encode the problem as a polynomial, apply the degree-reduction procedure from Section 9.5.1 and then run the protocol from Section 9.5.2.

9.6.1 Arguing Knowledge of RSA signatures

It is often useful for a user to prove that an organization has supplied him with a signature or a certificate without revealing it. A natural approach is to commit to the value of the signature and to prove its knowledge without revealing any information on it. If the user can additionally prove it in such a way that the signature cannot be linked to its issuance and multiple proofs cannot be linked to each other, then one can use the primitive in privacy-preserving applications. Camenisch and Stadler [CS97] presented a proof of knowledge of committed $x \in \mathbb{Z}_N^*$ such that $x^e = y \bmod N$, where (N, e) is an RSA public key. Their proof is in a group $\mathbb{G}$ of known order N (obtained for instance by finding a prime number p such that N divides $(p-1)$ and considering $\mathbb{G}$ as the subgroup of $\mathbb{Z}_p^*$ of order N). The main issue with this approach is that it requires at least that the prover knows N at the time the commitment scheme is set up. It also requires a new instance of the commitment scheme for each RSA moduli.

As mentioned above, Damgård and Fujisaki [DF02] proposed an efficient proof of knowledge of the contents of commitments and proofs of multiplicative relations over committed values. They stated that this scheme allows to prove in zero-knowledge the knowledge of an RSA signature since the verification equation can also be written as $x^e - \lambda N = y$ for some integer $\lambda < N^{e-1}$. This gives a proof with communication complexity with $\Omega(eb_{\mathbb{G}})$ bits which makes it usable only for very small e. One can use the elegant technique from [CS97] to improve this to $\Omega(\log(e)b_{\mathbb{G}})$ using the square-and-multiply algorithm in the exponent (see [CS97, CGM16] for details). In the

following, we show that our succinct Diophantine satisfiability argument can be used to exponentially reduce this communication complexity.

Let N be some RSA integer and let e be a public RSA exponent with $\mathbf{e} = (e_0, e_1, \ldots, e_{T-1}) \in \{0,1\}^T$ as binary representation, i.e.,

$$e = \sum_{i=0}^{T-1} e_i 2^i.$$

with $e_{T-1} = 1$. Suppose that one wants to prove the knowledge of some $x \in \mathbb{Z}_N^*$ such that $x^e = y \bmod N$ for some public value y (typically the hash value of a message). Suppose that the prover commits to x using our integer commitment scheme and wants to prove that the opening of V is an e-th root of the public y modulo N.

First define integers $(y_1, \ldots, y_{T-1})$ such that

$$y_i = \begin{cases} x & \text{if } e_i = 1 \\ 1 & \text{if } e_i = 0 \end{cases}$$

for $i \in \{0, \ldots, T-1\}$. Set then $z_{T-1} = x$ and define by induction

$$\begin{aligned} w_i &= z_i^2 \bmod N \\ \lambda_i &= \left\lfloor z_i^2 / N \right\rfloor \qquad \triangleright \quad w_i = z_i^2 - \lambda_i N \qquad (9.11) \\ z_{i-1} &= y_i \cdot w_i \bmod N \\ \mu_i &= \lfloor (y_i \cdot w_i)/N \rfloor \qquad \triangleright \quad z_{i-1} = y_i \cdot w_i - \mu_i N \qquad (9.12) \end{aligned}$$

for i downward from $T-1$ to 0 such that if $x^e = y \bmod N$. Then, $z_{-1} = y$.

Reciprocally, if there exist five integer sequences $(w_0, \ldots, w_{T-1})$, $(y_0, \ldots, y_{T-1}, y_T)$, $(\lambda_0, \ldots, \lambda_{T-1})$, $(z_0, \ldots, z_{T-1})$, $(\mu_0, \ldots, \mu_{T-1})$ such that $w_0 y_0 - \mu_0 N = y$, Equations (9.11) and (9.12) hold for all $i \in \{0, \ldots, T-1\}$ and

$$\{y_i \colon e_i = 0, i \in \{0, \ldots, T-1\}\} = \{1\} \text{ and } \{y_i \colon e_i = 1, i \in \{0, \ldots, T\}\} = \{x\} \qquad (9.13)$$

where x, the unique element in the latter set, is committed in V, then $x^e = y \bmod N$.

One can now write down explicitly a Diophantine equation in such a way that the equation is satisfiable under the linear constraints (9.13) if and only if the value committed in V is an e-th root of the public y modulo N:

$$\sum_{i=0}^{T-1} \left(w_i - z_i^2 - \lambda_i N \right)^2 + \sum_{i=0}^{T-1} (z_i - y_i w_i - \mu_i N)^2 = 0$$

This Diophantine equation is directly suited for the reduction from Section 9.5.1 without introducing new variables. It indeed suffices to set

$$\begin{aligned}\mathbf{a}_L &\leftarrow \begin{bmatrix}\mathbf{z} & \begin{bmatrix}\mathbf{w} & 1\end{bmatrix} & \boldsymbol{\lambda} & \boldsymbol{\mu}\end{bmatrix}\\ \mathbf{a}_R &\leftarrow \begin{bmatrix}\mathbf{z} & \begin{bmatrix}\mathbf{y} & x\end{bmatrix} & \mathbf{0} & \mathbf{0}\end{bmatrix}\\ \mathbf{a}_O &\leftarrow \begin{bmatrix}\mathbf{w} & \begin{bmatrix}y & \mathbf{z}\end{bmatrix} & \mathbf{0} & \mathbf{0}\end{bmatrix} \in \mathbb{Z}^{4T+1}.\end{aligned}$$

Besides, there are $10T + 3$ linear equations that these vectors must satisfy ($2T$ for the terms in the sum of squares in the Diophantine equation, $4T$ for the zeros in $\mathbf{a}_R$ and $\mathbf{a}_0$, $2T$ for the consistency of $\mathbf{z}$, T for the consistency of $\mathbf{w}$, T for the consistency of $\mathbf{y}$ with V and 3 for the consistency of x and y).

Now, to estimate the bit complexity of the argument, note that each witness is upper-bounded by N and the maximum coefficient of the linear constraints is $O(N)$. According to the bounds in Section 9.5.2, the bit length of the argument is of order $O\left(\log(T)b_{\mathbb{G}} + \log N\right) = O\left(\log(\log(e))b_{\mathbb{G}} + \log N\right)$.

Finally, note that even if it is not necessary to commit to the value x, this technique can be used to argue knowledge of a modular e-th root modulo N with communication complexity $O\left(\log(\log(e))b_{\mathbb{G}}\right)$ bits (in a group with $b_{\mathbb{G}} = \Omega(\log N)$). In particular for $e < 2^{O(\lambda/\log(\lambda))}$ this improves asymptotically the communication of the well-known protocol due to Guillou-Quisquater [GQ88] which has communication complexity $O\left(\log(\lambda)/\log(e) \cdot \log(N)\right)$.

9.6.2 Argument of Knowledge of (EC)DSA Signatures

The Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) [JMV01] are standardized discrete-logarithm based efficient digital signature schemes. The underlying groups of public prime order are usually standardized and it may seem at first thought that one can use commitments in these groups (together with the classical zero-knowledge toolbox) in order to design a zero-knowledge argument of possession of an (EC)DSA signature. However, it turns out that this is not so easy since the verification equations of these signatures involve arithmetic modulo two different prime numbers. In this section, we show how one can use our Diophantine satisfiability argument and the approach we already used for RSA signatures to efficiently prove the knowledge of an (EC)DSA signature without revealing any further information.

DSA Signatures.

The DSA signature scheme consists of the following procedures (given two parameters λ_1 and λ_2).

$\mathsf{KG}(\lambda_1, \lambda_2) \rightarrow (vk, sk)$: Generate a prime q of bit-length λ_2 uniformly at random and a prime p of bit-length λ_1 uniformly at random such that q

divides $p-1$. Choose an integer $h \in \{2, \ldots, p-2\}$ uniformly at random and set $g = h^{(p-1)/q} \bmod p$ (in the rare case that $g = 1$, try again with a different h). Choose $x \in \mathbb{Z}_q$ uniformly at random, and compute the integer $y = g^x \bmod p$. The verification key is $vk \leftarrow (p, q, g, y, \mathcal{H})$ where $\mathcal{H}$ is a cryptographic hash function $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^{\lambda_2}$, while the signing key is $sk \leftarrow (p, q, g, x, \mathcal{H})$.

$\mathsf{Sign}(sk, m) \to \sigma$: Choose a secret $k \in \mathbb{Z}_q^*$ uniformly at random, then compute $g^k \bmod p$ and set $r = \left(g^k \bmod p\right) \bmod q$. Repeat the whole process until $r \neq 0$. Next, compute $s = k^{-1}(\mathcal{H}(m) + rx) \bmod q$ and repeat the signing process from the beginning until $s \neq 0$. The signature of the message m is the pair $\sigma \leftarrow (r, s)$.

$\mathsf{Vf}(vk, m, \sigma) \to b \in \{0,1\}$: To verify a signature pair $\sigma = (r, s)$ of a message m for vk, first check that $0 < r, s < q$. Then, compute

- $w = s^{-1} \bmod q$
- $u_1 = \mathcal{H}(m) \cdot w \bmod q$
- $u_2 = r \cdot w \bmod q$
- $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$

Accept the signature (i.e., return 1) if and only $r = v$.

If a user wants to prove the knowledge of such a signature $\sigma = (r, s)$ for a public message m and a public verification key is $vk = (p, q, g, y, \mathcal{H})$, she has to prove the knowledge of a 6-tuple of integers $(r, w, \alpha, \beta, \mu_1, \mu_2)$ such that

$$\begin{aligned} \alpha &= \mathcal{H}(m) \cdot w - \mu_1 \cdot q \\ \beta &= r \cdot w - \mu_2 \cdot q \\ \beta &\neq 0 \\ r &= \left(g^\alpha y^\beta \bmod p\right) \bmod q. \end{aligned}$$

In order to prove the last equality, introduce the value $\rho = \left(g^\alpha y^\beta \bmod p\right)$ and μ_3 such that

$$r = \rho - \mu_3 \cdot q.$$

Write $g_i = g^{2^i} \bmod p$ and $y_i = y^{2^i} \bmod p$ for $i \in \{0, \ldots, \lambda_2 - 1\}$ and

$$\alpha = \sum_{i=0}^{\lambda_2 - 1} \alpha_i 2^i \text{ with } \alpha_i \in \{0,1\} \text{ for } i \in \{0, \ldots, \lambda_2 - 1\}$$

$$\beta = \sum_{i=0}^{\lambda_2 - 1} \beta_i 2^i \text{ with } \alpha_i \in \{0,1\} \text{ for } i \in \{0, \ldots, \lambda_2 - 1\}.$$

The values α_i and β_i for $i \in \{0, \ldots, \lambda_2 - 1\}$ have to be kept secret by the user. Starting from $\rho_{-1} = 1$, recursively construct

$$\begin{aligned} \nu_i &= \rho_{i-1} \cdot (1 + \alpha_i \cdot (g_i - 1)) \bmod p \quad \triangleright \nu_i = \rho_{i-1} \cdot (1 + \alpha_i \cdot (g_i - 1)) + \delta_i p \\ \rho_i &= \nu_i \cdot (1 + \beta_i \cdot (y_i - 1)) \bmod p \qquad \triangleright \rho_i = \nu_i \cdot (1 + \beta_i \cdot (y_i - 1)) + \varepsilon_i p \end{aligned}$$

such that $\rho_j = g^{\sum_{i=0}^{j} \alpha_i 2^i} y^{\sum_{i=0}^{j} \beta_i 2^i} \bmod p$ for $j \in \{0, \ldots, \lambda_2 - 1\}$ and in particular $\rho_{\lambda_2} = \rho$. It thus leads to the Diophantine equation

$$\begin{aligned} &\sum_{i=0}^{\lambda_2 - 1} \left(\alpha_i^2 - \alpha_i\right)^2 + \left(\beta_i^2 - \beta_i\right)^2 + \left(\nu_i - \rho_{i-1} \cdot (1 + \alpha_i \cdot (g_i - 1)) - \delta_i p\right)^2 \\ &\quad + \sum_{i=0}^{\lambda_2 - 1} \left(\rho_i - \nu_i \cdot (1 + \beta_i \cdot (y_i - 1)) - \varepsilon_i p\right)^2 + \left(r - \rho_{\lambda_2} + \mu_3 \cdot q\right)^2 \\ &\quad + (\alpha - \mathcal{H}(m) \cdot w + \mu_1 \cdot q)^2 + (\beta - r \cdot w + \mu_2 \cdot q)^2 \\ &\quad + \left(\alpha - \sum_{i=0}^{\lambda_2 - 1} \alpha_i 2^i\right)^2 + \left(\beta - \sum_{i=0}^{\lambda_2 - 1} \beta_i 2^i\right)^2 = 0, \end{aligned}$$

with $(\alpha, \alpha_0, \ldots, \alpha_{\lambda_2})$, $(\beta, \beta_0, \ldots, \beta_{\lambda_2})$, $(\rho_0, \ldots, \rho_{\lambda_2})$, $(\nu_0, \ldots, \nu_{\lambda_2})$, $(\delta_0, \ldots, \delta_{\lambda_2})$, $(\epsilon_0, \ldots, \epsilon_{\lambda_2})$ and $(r, w, \mu_1, \mu_2, \mu_3)$ as unknowns.

Using the approach from Section 9.5.2, the bit length of the argument is of order $O(\log(\lambda_2) b_{\mathbb{G}} + \log(\lambda_1))$. Note also that this argument could be combined with proofs of non-algebraic statements [CGM16] to obtain proofs on committed messages.

ECDSA Signatures.

For ECDSA signatures, the underlying group is an elliptic curve $E(\mathbb{Z}_p)$ defined over $\mathbb{Z}_p$. The group law involves arithmetic operations modulo p, but the group order is usually some prime number q (of bit size equal to the bit size of p). The main difference in an ECDSA signature $(r, s) \in \mathbb{Z}_p^2$ is that the verification equation checks that r is the abscissa of some point on the elliptic curve obtained by a double scalar multiplication with exponents derived from (r, s) and the hash value of the signed message. It is therefore possible to follow the same strategy as for DSA by adding more variables in order to handle the more intricate group law on the elliptic curve. More precisely, one need only add a constant number of variables per bit in the exponents and using the approach from Section 9.5.2, the bit length of the argument is of order $O(\log(\log p) b_{\mathbb{G}})$.

9.6.3 Argument of Knowledge of List Permutation

Consider the classical problem of proving knowledge of openings $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ (together with the corresponding randomness) of two com-

mitment vectors $(V_1, \ldots, V_n)$ and $(V'_1, \ldots, V'_n)$, and of a permutation $\pi \in \mathfrak{S}_n$ (the symmetric group of order n) such that $y_i = x_{\pi(i)}$ for all $i \in [\![n]\!]$.

To provide a succinct argument of such a statement, one can write down a simple Diophantine equation that is satisfiable (under additional linear constraints) if and only if the statement indeed holds. Denote $V_i = e^{x_i} f^{\rho_i}$ and $V'_i = e^{y_i} f^{\eta_i}$ for $i \in [\![n]\!]$. Note that if there exists such a permutation π, one can consider the associated permutation binary matrix $U = (u_{i,j}) \in \mathbb{Z}^{n \times n}$ defined by $u_{i,j} = 1$ if $j = \pi(i)$ and $u_{i,j} = 0$ otherwise. Such a matrix has exactly one "1" per row and per column (and is null at all other indices). Therefore, $x_i = y_j$ for integers $i, j \in [\![n]\!]$ if $u_{i,j} = 1$. In particular, for all $i \in [\![n]\!]$

$$\sum_{j=1}^{n} \left(u_{i,j} \left(x_i - y_j\right)\right)^2 = 0. \tag{9.14}$$

Conversely, if there exists a permutation matrix $\mathbf{U} = (u_{i,j}) \in \mathbb{Z}^{n \times n}$ such that Equation (9.14) holds for all $i \in [\![n]\!]$, then there exist a permutation $\pi \in \mathfrak{S}_n$, such that $y_i = x_{\pi(i)}$ for all $i \in [\![n]\!]$. Consider then the following Diophantine equation

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(u_{i,j} \left(x_i - y_j\right)\right)^2 + \left(u_{i,j}^2 - u_{i,j}\right)^2 = 0$$

with the $2n$ linear constraints:

$$\sum_{i=1}^{n} u_{i,j} = 1 \text{ for all } j \in [\![n]\!] \quad \sum_{j=1}^{n} u_{i,j} = 1 \text{ for all } i \in [\![n]\!].$$

Note that these constraints could have been direclty embedded in the Diophantine equation.

Following the ideas in Section 9.5.1, by introducing variables $v_{i,j} \leftarrow u_{i,j} x_i$ for $i, j \in [\![n]\!]$, the previous equation is equivalent to the satisfiability of the equation

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(v_{i,j} - u_{i,j} y_j\right)^2 + \left(u_{i,j}^2 - u_{i,j}\right)^2 + \left(v_{i,j} - u_{i,j} x_i\right)^2 = 0.$$

This polynomial is now in a form which allows to immediately read a Hadamard product between variables and linear equations of which the satisfiability is equivalent to the satisfiability of the Diophantine equation. It suffices to set

$$\mathbf{a}_L \leftarrow \begin{bmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \mathbf{u} \end{bmatrix} \quad \mathbf{a}_R \leftarrow \begin{bmatrix} \mathbf{u} & \mathbf{u} & \mathbf{u} \end{bmatrix} \quad \mathbf{a}_O \leftarrow \begin{bmatrix} \mathbf{v} & \mathbf{v} & \mathbf{u} \end{bmatrix} \in \mathbb{Z}^{3n^2}.$$

where $\mathbf{u}$ is the vector of dimension n^2 obtained by concatenating the rows of $\mathbf{U}$, $\hat{\mathbf{x}}$ is the vector of dimension n^2 obtained by repeating n times each coordinate of $\mathbf{x}$ and $\hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} & \cdots & \mathbf{y} \end{bmatrix}$ is a vector of dimension n^2.

Besides, there are $7n^2 + 2n$ linear equations that these vectors must satisfy and the vectors $\mathbf{x}$ and $\mathbf{y}$ are committed individually (i.e., $2n$ commitments). To estimate the bit complexity of the argument, note that if have $\|\mathbf{x}\|_\infty < 2^\ell$ (and thus $\|\mathbf{y}\|_\infty < 2^\ell$), then, according to the analysis in Section 9.5.2, the bit length of the argument is of order $O\left(\ell + \log\left(3n^2\right) b_{\mathbb{G}}\right) = O\left(\ell + \log(n) b_{\mathbb{G}}\right)$.

9.6.4 3-SAT Satisfiability Argument

3-SAT is the prototypical NP-complete problem of deciding the satisfiability of a Boolean formula in conjunctive normal form, with each clause limited to at most three literals. Consider two integers $m, n \geq 1$ and a set of clause $C_1, \ldots, C_m$ where each C_i is the disjunction of exactly three literals from the set of Boolean variables $x_1, \ldots, x_n$ (a literal is a variable x_i or its negation $\neg x_i$). The problem is decide whether there exists an assignment of $(x_1, \ldots, x_n) \in \{0,1\}^n$ such that all clauses are satisfied. Write each clause C_i for $i \in [\![m]\!]$ as $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$, with $l_{i,1}$, $l_{i,2}$ and $l_{i,3}$ denoting the literals in C_i.

Such a Boolean 3-SAT formula can be readily turned into an equisatisfiable Diophantine equation. Indeed, for $i \in [\![m]\!]$ and $j \in [\![3]\!]$ and $k \in [\![n]\!]$, define

$$\varepsilon_{3(i-1)+j,k} = \begin{cases} 1 & \text{if } l_{i,j} = x_k \\ -1 & \text{if } l_{i,j} = \neg x_k \\ 0 & \text{otherwise} \end{cases}$$

so that setting $\mathbf{l} \coloneqq \begin{bmatrix} l_{1,1} & l_{1,2} & l_{1,3} & \cdots & l_{m,1} & l_{m,2} & l_{m,3} \end{bmatrix}$, $\mathbf{x} \coloneqq \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$ and $\mathbf{E} \coloneqq \begin{bmatrix} \varepsilon_{i,k} \end{bmatrix}_{1 \leq i \leq 3m, 1 \leq k \leq n}$, the equality $\mathbf{l}^{\mathrm{T}} = \mathbf{E} \cdot \mathbf{x}^{\mathrm{T}}$ holds. The Diophantine equation must ensure that (1) $x_i \in \{0,1\}$, i.e., $x_j\left(1 - x_j\right) = 0$ for $j \in [\![n]\!]$, and that (2) the clauses are all satisfied, i.e., $\left(1 - l_{i,1}\right)\left(1 - l_{i,2}\right)\left(1 - l_{i,3}\right) = 0$ for all $i \in [\![m]\!]$. The satisfiability of the 3-SAT instance is then equivalent to the satisfiability of the Diophantine equation

$$\sum_{j=1}^{n} \left(x_j - x_j^2\right)^2 + \sum_{i=1}^{m} \left(1 - l_{i,1} - l_{i,2} - l_{i,3} + l_{i,1}l_{i,2} + l_{i,1}l_{i,3} + l_{i,2}l_{i,3} - l_{i,1}l_{i,2}l_{i,3}\right)^2 = 0$$

Following the ideas in Section 9.5.1, by introducing variables $u_{i,1} \leftarrow l_{i,1}l_{i,2}$, $u_{i,2} \leftarrow l_{i,1}l_{i,3}$, $u_{i,3} \leftarrow l_{i,2}l_{i,3}$, $v_i \leftarrow u_{i,1}l_{i,3}$ for $i \in [\![m]\!]$, the previous equation

is equivalent to the satisfiability of the equation

$$\begin{aligned}&\sum_{j=1}^{n}\left(x_j - x_j^2\right)^2 + \sum_{i=1}^{m}(1 - l_{i,1} - l_{i,2} - l_{i,3} + u_{i,1} + u_{i,2} + u_{i,3} - v_i)^2 \\ &+\sum_{i=1}^{m}\Big((u_{i,1} - l_{i,1}l_{i,2})^2 + (u_{i,2} - l_{i,1}l_{i,3})^2 + (u_{i,3} - l_{i,2}l_{i,3})^2 + (v_i - u_{i,1}l_{i,3})\Big)^2 \\ &= 0.\end{aligned}$$

This polynomial is now in a form which allows to immediately read a Hadamard product between variables and linear equations of which the satisfiability is equivalent to the satisfiability of the Diophantine equation. It suffices to set

$$\begin{aligned}\mathbf{a}_L &\leftarrow \begin{bmatrix}\mathbf{x} & \mathbf{l}^{(1)} & \mathbf{l}^{(1)} & \mathbf{l}^{(2)} & \mathbf{l}^{(3)}\end{bmatrix} \\ \mathbf{a}_R &\leftarrow \begin{bmatrix}\mathbf{x} & \mathbf{l}^{(2)} & \mathbf{l}^{(3)} & \mathbf{l}^{(3)} & \mathbf{u}_1\end{bmatrix} \\ \mathbf{a}_O &\leftarrow \begin{bmatrix}\mathbf{x} & \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{v}\end{bmatrix} \in \mathbb{Z}^{n+4m}.\end{aligned}$$

Besides, there are $2n + 9m$ linear equations that these vectors must satisfy ($3m$ from matrix $\mathbf{E}$, m for the final Diophantine equation and $2n + 5m$ for the consistencies in $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_0$).

Now, to estimate the bit complexity of the argument, note that since $\|\mathbf{x}\|_\infty \leq 1$ and $\|\mathbf{E}\|_\infty \leq 1$, all witnesses and all variables are upper-bounded by 1, and according to the bounds in Section 9.5.2, the bit length of the argument is of order $O(\log(n + 4m)b_{\mathbb{G}}) = O(\log(n + m)b_{\mathbb{G}})$.

9.6.5 Integer-Linear-Programming Satisfiability Argument

A (decisional) Integer-Linear-Programming (ILP) problem is a feasibility linear program in which the variables are integers: given two positive integers m and n, a matrix $\mathbf{A} \in \mathbb{Z}^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{Z}^m$, the problem consists in deciding whether there exists a vector $\mathbf{x} \in \mathbb{N}^n$ such that $\mathbf{A}\mathbf{x}^{\mathrm{T}} \geq \mathbf{b}^{\mathrm{T}}$. This problem is a classical NP-complete problem and it models many real-life optimization problems, and the following shows how to succinctly argue knowledge of a solution to it using the techniques presented in Section 9.5.

In other to prove the positivity of an integer x, as previously done in the literature [Bou00, Lip03, CPP17], one can rely on Lagrange's four-square theorem which states that every natural integer can be represented as the sum of four integer squares. Actually, as remarked by Groth [Gro05], one can also rely on Legendre's three-square theorem which states that every natural number $x = 1 \bmod 4$ can be represented as the sum of three integer squares. Both theorems are effective and there exists efficient polynomial-time algorithms to find the decomposition as a sum of squares (see [Bou00, Lip03, Gro05, CPP17] and references therein for details).

To argue the satisfiability of the ILP problem, it is thus equivalent to argue knowledge of vectors $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}^n$ and vectors $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \in \mathbb{Z}^m$ such that

$$\begin{aligned} 4x_j + 1 - y_{1,j}^2 - y_{2,j}^2 - y_{3,j}^2 = 0 \quad \forall j \in [\![n]\!] \text{ and} \\ \sum_{j=1}^{n} 4a_{ij}x_j - 4b_i + 1 - z_{1,i}^2 - z_{2,i}^2 - z_{3,i}^2 = 0 \quad \forall i \in [\![m]\!]\,. \end{aligned}$$

The satisfiability of the ILP problem is then equivalent to the satisfiability of the Diophantine equation

$$\begin{aligned} &\sum_{j=1}^{n} \left(4x_j - y_{1,j}^2 - y_{2,j}^2 - y_{3,j}^2 + 1\right)^2 + \\ &\sum_{i=1}^{m} \left(\sum_{j=1}^{n} 4a_{ij}x_j - 4b_i - z_{1,i}^2 - z_{2,i}^2 - z_{3,i}^2 + 1\right)^2 = 0. \end{aligned}$$

Following the ideas in Section 9.5.1, by introducing variables $u_{1,j} \leftarrow y_{1,j}^2$, $u_{2,j} \leftarrow y_{2,j}^2$, $u_{3,j} \leftarrow y_{3,j}^2$, $v_{1,i} \leftarrow z_{1,i}^2$, $v_{2,i} \leftarrow z_{2,i}^2$ and $v_{3,i} \leftarrow z_{3,i}^2$, the previous equation equation is equivalent to the satisfiability of the equation

$$\begin{aligned} &\sum_{j=1}^{n} (4x_j - u_{1,j} - u_{2,j} - u_{3,j} + 1)^2 \\ &+ \sum_{i=1}^{m} \left(\sum_{j=1}^{n} 4a_{ij}x_j - 4b_i - v_{1,i} - v_{2,i} - v_{3,i} + 1\right)^2 \\ &+ \sum_{j=1}^{n} \left(u_{1,j} - y_{1,j}^2\right)^2 + \left(u_{2,j} - y_{2,j}^2\right)^2 + \left(u_{3,j} - y_{3,j}^2\right)^2 \\ &+ \sum_{i=1}^{m} \left(v_{1,i} - z_{1,i}^2\right)^2 + \left(v_{2,i} - z_{2,i}^2\right)^2 + \left(v_{3,i} - z_{3,i}^2\right)^2 = 0. \end{aligned}$$

This polynomial is now in a form which allows to immediately read a Hadamard product between variables and linear equations of which the satisfiability is equivalent to the satisfiability of the Diophantine equation. It suffices to set

$$\begin{aligned} \mathbf{a}_L &\leftarrow \begin{bmatrix} \mathbf{x} & \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \mathbf{z}_1 & \mathbf{z}_2 & \mathbf{z}_3 \end{bmatrix} \\ \mathbf{a}_R &\leftarrow \begin{bmatrix} \mathbf{0} & \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \mathbf{z}_1 & \mathbf{z}_2 & \mathbf{z}_3 \end{bmatrix} \\ \mathbf{a}_O &\leftarrow \begin{bmatrix} \mathbf{0} & \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \in \mathbb{Z}^{4n+3m}. \end{aligned}$$

Besides, there are $4(n+m)$ linear equations that these vectors must satisfy.

Now, to estimate the bit complexity of the argument, note that if $\|\mathbf{x}\|_\infty < 2^\ell$, then $\left|\sum_{j=1}^n 4a_{ij}x_j - 4b_i + 1\right| < 4\left(n\,\|\mathbf{A}\|_\infty\, 2^\ell + \|\mathbf{b}\|_\infty\right)+1$ for all $i \in [\![m]\!]$. Therefore, according to the bounds in Section 9.5.2, the bit length of the argument is of order $O\left(\ell + \log(4n+3m)b_{\mathbb{G}} + \log\|\mathbf{A}\|_\infty + \log\|\mathbf{b}\|_\infty\right)$.

Chapter 10

Public-Key Generation with Verifiable Randomness

This chapter addresses the problem of proving that a user algorithm selected and correctly used a truly random seed in the generation of her cryptographic key. A first approach was proposed in 2002 by Juels and Guajardo for the validation of RSA keys. Section 10.2 presents a new security model and general tools to efficiently prove that a private key was generated at random according to a prescribed process, without revealing any further information about the private key.

Section 10.3 gives a generic protocol for all key-generation algorithms based on probabilistic circuits, as well as for factoring-based cryptography, and proves their security. Section 10.4 instantiates the protocol for factoring-based cryptography. The instantiation relies on a new efficient zero-knowledge argument for the double discrete logarithm problem that achieves an exponential improvement in communication complexity compared to the state of the art, and is of independent interest. This argument relies on techniques similar to those presented in Chapter 9.

Contents

10.1 Preliminaries

This section introduces notions about imperfect randomness, randomness extraction, pseudo-random functions, and other tools used in this chapter.

10.1.1 Randomness Sources and Min-Entropy.

Imperfect randomness is modeled as arbitrary probability distributions with a certain amount of *entropy*. The *min-entropy* notion is used to measure the randomness in such an imperfect random source. A source is said to have k bits of min-entropy if its distribution has the property that each outcome occurs with probability at most 2^{-k}.

10.1.2 Randomness Extractors

For a discrete distribution X over a set S, its *min-entropy* is denoted $H_\infty(X) = \min_{x \leftarrow_\$ X}\{-\log \Pr[X = x]\}$. A distribution X is called a *κ-source* if $H_\infty(X) \geq \kappa$.

A *randomness extractor* is a function which, applied to the output of one or several discrete distributions (possibly together with an additional uniformly random seed), returns a value indistinguishable from a uniformly random one. It is well known that there does not exist good deterministic extractors for a single random source i.e., without the additional random seed. This chapter uses *two-source extractors* defined as follows.

Let $\mathcal{H} = \{H_x \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m\}_{x \in \{0,1\}^d}$ be a hash function family. $\mathcal{H}$ is said to be a *(κ, ε)-two-source extractor* if for any pair of discrete distributions I, J over $\{0,1\}^n$ such that $\max(H_\infty(I), H_\infty(J)) \geq \kappa$, for X uniformly random over $\{0,1\}^d$ and U is uniformly random over $\{0,1\}^m$,

the distributions of $(X, H_X(I,J))$ and (X,U) are ε-close. For some parameter sets, there exist deterministic two-source extractors (i.e., with $d=0$) (e.g., [Li16]). Using the random-oracle heuristic, two-source extractors with high security and efficient parameters can be efficiently constructed. Alternatively, one can also rely on the weaker notion of *universal computational extractor* [BHK13].

10.1.3 Universal Computational Extractors

Bellare, Hoang and Keelveedhi introduced Universal Computational Extractors [BHK13] (UCEs) as a standard-model security notion for keyed hash functions which relates them to random-oracles. Their definition features a two-stage adversary, i.e., a *source* $\mathcal{S}$ and a distinguisher $\mathcal{D}$. The source is given access to an oracle which either computes the hash function or is a random oracle, whereas the distinguisher is only given the hash-function key and some leakage information from the source. $\mathcal{D}$ is then suppose to guess whether the oracle computed a hash function or was a random oracle.

The formal definition of a UCE is actually given w.r.t. source classes instead of a single source. Formally, given a class of sources $\mathscr{S}$ and maps $d\colon \mathbb{N} \to \mathbb{N}$ and $N, M\colon \mathbb{N} \to 2^{\mathbb{N}}$, a family

$$\mathcal{H} = \left\{ H_x \in \bigcup_{n \in N(\lambda), m \in M(\lambda)} (\{0,1\}^m)^{\{0,1\}^n \times \{1\}^m} \right\}_{x \in \{0,1\}^{d(\lambda)}}$$

of (variable-input-length and variable-output-length) functions is (T, q, ε)-UCE secure w.r.t. $\mathscr{S}$ if for all $\lambda \in \mathbb{N}$, for every $T(\lambda)$-time adversary $(\mathcal{S}, \mathcal{D})$ such that $\mathcal{S} \in \mathscr{S}$ and such $\mathcal{S}$ makes at most q oracle queries,

$$\left| \Pr \left[b = b' \colon \begin{array}{l} x \leftarrow_{\$} \{0,1\}^{d(\lambda)}; Q \leftarrow \emptyset \\ b \leftarrow_{\$} \{0,1\} \\ L \leftarrow \mathcal{S}^{\mathcal{O}_b(x,Q,\cdot)}\left(1^\lambda\right) \\ b' \leftarrow \mathcal{D}\left(1^\lambda, x, L\right) \\ \text{return } (b, b') \end{array} \right] - 1/2 \right| \leq \varepsilon(\lambda),$$

with $\mathcal{O}_b(x, Q, \cdot)$ an oracle which, on input $(a, 1^m)$ such that $a \in \bigcup_{n \in N(\lambda)} \{0,1\}^n$ and $m \in M(\lambda)$, proceeds as follows:

- if there exists $h \in \{0,1\}^m$ such that $(a, 1^m, h) \in Q$, return h
- else
  - if $b = 1$, return $H_x(a, 1^m)$
  - else, generate $h \leftarrow_{\$} \{0,1\}^m$, add $(a, 1^m, h)$ to Q and return h.

As $\mathcal{S}$ could simply return one of its query and the response from the oracle in the leakage information, restrictions must imposed on the source for the security notion to be achievable. A classical requirement is then that $\mathscr{S}$ should be a class of *unpredictable* sources. A source $\mathcal{S}$ is unpredictable if it is computationally hard to determine its hash queries even given its leakage information L, in case it interacts with a random oracle. Formally, a source $\mathcal{S}$ is (T, q, ε)-simply unpredictable if for any $\lambda \in \mathbb{N}$, for every adversary or *simple*[1] *predictor* $\mathcal{P}$ running in time at most $T(\lambda)$ and making at most q oracle queries,

$$\Pr\left[Q \cap Q' \neq \emptyset \colon \begin{array}{l} x \leftarrow_{\$} \{0,1\}^{d(\lambda)}; Q \leftarrow \emptyset \\ L \leftarrow \mathcal{S}^{\mathcal{O}(x,Q,\cdot)}\left(1^{\lambda}\right) \\ Q' \leftarrow \mathcal{P}\left(1^{\lambda}, L\right) \\ \text{return } (Q, Q') \end{array}\right] \leq \varepsilon(\lambda),$$

with $\mathcal{O}(x, Q, \cdot)$ an oracle which, on input $(a, 1^m)$ such that $a \in \bigcup_{n \in N(\lambda)} \{0,1\}^n$ and $m \in M(\lambda)$, returns h if there exists $h \in \{0,1\}^m$ such that $(a, 1^m, h) \in Q$, and otherwise generates $h \leftarrow_{\$} \{0,1\}^m$, adds $(a, 1^m, h)$ to Q and returns h.

Pseudo-Random Functions.

A Pseudo Random Function (PRF) [GGM84] is an efficiently computable function of which the values are computationally indistinguishable from uniformly random values.

Formally, a function $\mathsf{PRF} \colon \mathcal{K}(\lambda) \times \mathcal{X}(\lambda) \to \mathcal{Y}(\lambda)$ is a (T, q, ε)-secure PRF with key space $\mathcal{K}$, input space $\mathcal{X}$ and range $\mathcal{Y}$ (all assumed to be finite) if the advantage

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{PRF}(K,\cdot)} \colon K \leftarrow_{\$} \mathcal{K}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{f(\cdot)} \colon f \leftarrow_{\$} \mathcal{Y}^{\mathcal{X}}\right]\right|$$

of every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$ is at most $\varepsilon(\lambda)$.

Dodis–Yampolskiy Pseudo-Random Function.

Let G be a group family generator. The Dodis–Yampolskiy pseudo-random function [DY05] in an ℓ-order group $(\mathbb{G}, \ell, g) \leftarrow_{\$} \mathsf{G}$ is the map $F \colon (K, x) \in \mathcal{K} \times \mathcal{X} \mapsto g^{1/(K+x)} \in \mathbb{G}^*$, with $\mathcal{K} = \mathbb{Z}_\ell^*$ and $\mathcal{X} \subset \mathbb{Z}_\ell^*$. They proved that it is $\left(T/\left(q\lambda^{O(1)}\right), q, \varepsilon q\right)$-secure under the (T, q, ε)-DDHI assumption (Definition 2.4.5) for G, where $q(\lambda) = O(\log \lambda)$ is an upper-bound on the bit-size of $\mathcal{X}$ for all λ [DY05, Section 4.2].

[1] The predictor is qualified as simple as it is not given access to oracle $\mathcal{O}$. Bellare, Hoang and Keelveedhi proved [BHK13, Lemma 4.3] that a source is unpredictable (i.e., with a predictor which is given oracle access) if and only if it is simply unpredictable.

10.1.4 Chernoff's Bound

The Chernoff bound gives bound on the tail distribution of sums of independent Bernoulli random variables. It will prove to be a useful tool from probabilistic methods for the analysis of the runtime of some algorithms in this chapter.

Theorem 10.1.5 ([MR95, Theorem 4.2]). *Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli random variables such that, for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$, with $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$, and any $0 < \delta \leq 1$,*

$$\Pr\left[X < (1-\delta)\mu\right] < \exp\left(-\mu\delta^2/2\right).$$

10.2 Model for Key Generation with Verifiable Randomness

This section formalizes key-generation protocols for arbitrary, predetermined key-generation algorithms. Such a protocol is executed between a *user* $\mathcal{U}$ and a *certification authority* $\mathcal{CA}$. At the end of the protocol, $\mathcal{U}$ obtains a pair of public–secret keys that $\mathcal{CA}$ certifies to be indistinguishable from keys generated by a fixed algorithm KeyGen, and to have been generated with proper randomness. These requirements are formally captured by a model for randomness verifiability given below. The security definition of the model ensures that a protocol satisfying its conditions fulfills the following properties:

1. $\mathcal{CA}$ can infer no more information about the secret key than it would from a public key generated by KeyGen if $\mathcal{U}$'s randomness source has high entropy
2. no external attacker can distinguish a public key generated via the protocol from a public key generation with KeyGen if the randomness source of either $\mathcal{U}$ or $\mathcal{CA}$ has high entropy
3. $\mathcal{U}$ cannot bias the generation of the keys if the randomness source of $\mathcal{CA}$ has high entropy. In particular, $\mathcal{U}$ cannot use the public key as a subliminal channel to convey information.

10.2.1 Syntax

An interactive asymmetric-key-generation protocol is a triple $\mathsf{IKG} = (\mathsf{Setup}, \mathsf{U}, \mathsf{CA})$ of algorithms such that $\mathsf{Setup}\left(1^\lambda\right) \to pp$ is a probabilistic algorithm which returns public parameters and

$$\langle \mathsf{U}(pp; r_{\mathcal{U}}) \rightleftharpoons \mathsf{CA}(pp; r_{\mathcal{CA}}) \rangle \to \langle (pk_{\mathcal{U}}, sk), pk_{\mathcal{CA}} \rangle$$

are interactive algorithms. At the end of the protocol, the user key-generation algorithm U returns a pair of public–secret keys, and the certificate-authority key-generation algorithm CA returns a public key.

Algorithm Setup may require some randomness, but the parameters it generates can be fixed once for all and used across multi sessions and by several users and authorities. Once parameters are fixed, high-entropy randomness is still needed to securely generate keys, and this is formalized in Section 10.2.3.

Definition 10.2.2 (Correctness)**.** *In the $\mathcal{O}$-oracle model, a key-generation protocol IKG is δ-correct w.r.t. a class $\mathscr{A}$ of algorithms if for all $\lambda \in \mathbb{N}$, for every $\mathcal{A} \in \mathscr{A}$,*

$$\Pr\left[pk_{\mathcal{U}} = pk_{\mathcal{CA}} \neq \bot : \begin{array}{l} pp \leftarrow_{\$} \mathsf{Setup}\left(1^{\lambda}\right) \\ (\mathcal{D}_{\mathcal{U}}, \mathcal{D}_{\mathcal{CA}}) \leftarrow_{\$} \mathcal{A}^{\mathcal{O}(\cdot)}\left(pp\right) \\ r_{\mathcal{U}} \leftarrow_{\$} \mathcal{D}_{\mathcal{U}}, r_{\mathcal{CA}} \leftarrow_{\$} \mathcal{D}_{\mathcal{CA}} \\ \langle (pk_{\mathcal{U}}, sk), pk_{\mathcal{CA}} \rangle \leftarrow \langle \mathsf{U}(pp; r_{\mathcal{U}}) \rightleftharpoons \mathsf{CA}(pp; r_{\mathcal{CA}}) \rangle \end{array}\right] \geq \delta.$$

Note that the last line of the probability event implicitly implies that U and CA *must terminate*.

The above definition is given in model in which $\mathcal{A}$ has oracle access to $\mathcal{O}$. This latter is used to "distinguish" different models: it may be a random oracle, but it could also simply be an oracle which returns a fixed value (i.e., the common-reference-string model) or no value at all (the standard model). The reason for this distinction is that if a component of the protocol (e.g. a randomized primality-testing algorithm) is not perfectly correct, then its correctness probability is only defined for perfect randomness although the parties only have access to imperfect randomness. However, in the random-oracle model for instance, this imperfect randomness chosen by the algorithm in the definition may depend on the random-oracle queries made by this latter.

10.2.3 Security

This section gives a game-based security model for key-generation protocols with verifiable randomness. It covers concurrent protocol executions with different instances of protocol algorithms. It is inspired by the BPR model for authenticated key exchange [BPR00] but with key differences.

Protocol Participants. A set of user identities U and a set of certificate-authority identities CA are assumed to be fixed. The union of the those sets form the overall identity space ID. For readability, it is implicitly assumed that during protocol executions, the messages exchanged are always prepended with the instance identifier of the receiving party. Note that

$\mathsf{Init}\left(1^\lambda, U, CA, I\right)$	$h \leftarrow_\$ \Omega$; $pp \leftarrow_\$ \mathsf{Setup}\left(1^\lambda\right)$ $ID \leftarrow U \cup CA$ for $i \in [\![I]\!]$ and $id \in ID$ do $st^i_{id} \leftarrow r^i_{id} \leftarrow \bot$ $used^i_{id} \leftarrow \mathsf{FALSE}$ $acc^i_{id} \leftarrow term^i_{id} \leftarrow flag^i_{id} \leftarrow \mathsf{FALSE}$ $sid^i_{id} \leftarrow pid^i_{id} \leftarrow \bot$ $sk^i_{id} \leftarrow pk^i_{id} \leftarrow \bot$ $Q_{\mathsf{Reveal}} \leftarrow Q_{\mathsf{Corrupt}} \leftarrow \emptyset$ return (pin, sin)
$\mathsf{Oracle}(m)$	return $h(m)$
$\mathsf{Dist}\left(id, i, \mathcal{D}^i_{id}\right)$	$r^i_{id} \leftarrow_\$ \mathcal{D}^i_{id}$ // r^i_{id} is simply generated and *not* returned to $\mathcal{A}_1$
$\mathsf{Exec}(\mathcal{U}, i, \mathcal{CA}, j)$	if $\left(\mathcal{U} \notin U$ or $\mathcal{CA} \notin CA$ or $used^i_{\mathcal{U}}$ or $used^j_{\mathcal{CA}}\right)$ return $\bot$ if $r^i_{\mathcal{U}} \neq \bot$ and $r^j_{\mathcal{CA}} \neq \bot$ return $\left\langle \mathsf{U}_i\left(pp, r^i_{\mathcal{U}}\right), \mathsf{CA}_j\left(pp, r^j_{\mathcal{CA}}\right)\right\rangle$ return $\bot$ // $\mathcal{A}_1$ must specify distributions beforehand
$\mathsf{Send}(id, i, m)$	if $r^i_{id} = \bot$ return $\bot$ // $\mathcal{A}_1$ must specify a distribution beforehand if $term^i_{id}$ return $\bot$ $used^i_{id} \leftarrow \mathsf{TRUE}$ $\left\langle m_{out}, acc, term, sid, pid, pk, sk, st^i_{id}\right\rangle \leftarrow \left\langle \mathsf{IKG}\left(id, st^i_{id}, m; r^i_{id}\right)\right\rangle$ if acc and $\neg acc^i_{id}$ $sid^i_{id} \leftarrow sid$; $pid^i_{id} \leftarrow pid$ $acc^i_{id} \leftarrow acc$ if $term$ and $\neg term^i_{id}$ // Set keys only after termination $pk^i_{id} \leftarrow pk$; $sk^i_{id} \leftarrow sk$ return $\left(m_{out}, sid, pid, pk, sk, acc, term^i_{id}\right)$
$\mathsf{Reveal}(id, i)$	$Q_{\mathsf{Reveal}} \leftarrow Q_{\mathsf{Reveal}} \cup \{(id, i)\}$ return $\left(pk^i_{id}, sk^i_{id}\right)$
$\mathsf{Corrupt}(id)$	$Q_{\mathsf{Corrupt}} \leftarrow Q_{\mathsf{Corrupt}} \cup \{id\}$ for $i \in [\![I]\!]$ $\left\{\text{if } \neg acc^i_{id} \text{ then } flag^i_{id} \leftarrow \mathsf{TRUE}\right\}$ return $\{st^i_{id}\}_{i \in [\![I]\!]}$

$\mathsf{Test}_b(id^*, i^*)$	if $\left(\exists(id_0, id_1, i, j)\colon pid^i_{id_0} = id_1$ and $pid^j_{id_1} = id_0$ and $acc^i_{id_0}$
	and $\neg term^j_{id_1}\right)$ return $\bot$
	// Once an instance accepts, its partner must eventually terminate
	if $\neg term^{i^*}_{id^*}$ return $\bot$
	if $flag^{i^*}_{id^*}$ return $\bot$
	// Reject if id^* was corrupt before (id^*, i^*) accepted
	if $(id^*, i^*) \in Q_{\mathsf{Reveal}}$ or $\left(\exists(id', j)\colon pid^{i^*}_{id^*} = id'$ and $pid^j_{id'} = id^*$
	and $(id', j) \in Q_{\mathsf{Reveal}}\right)$
	return $\bot$
	// Reject if the key of (id^*, i^*) or of its partner has been revealed
	if $pk^{i^*}_{id^*} \neq \bot$
	if $b = 0$
	$(pk, sk) \leftarrow_\$ \mathsf{KeyGen}\left(1^\lambda\right)$
	return pk
	return $pk^{i^*}_{id^*}$
	return $\bot$ // Reject if (id^*, i^*) does not have a key

Figure 10.1: Oracles for the Key-Generation Indistinguishability Experiment.

several instances of the same algorithm may concurrently run during the game.

Adversaries. The game features a two-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$. Adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ may agree on a common strategy before the beginning of the game. That is to say, the strategy may be part of their code, and it may dictate which queries to make (possibly depending on the oracle answers), the order of the queries and so forth. All but the challenge query can only be made by $\mathcal{A}_1$. The role of $\mathcal{A}_2$ is essentially only to guess whether a public key was generated with KeyGen or with the protocol, while $\mathcal{A}_1$ can make arbitrary queries according to the pre-established strategy.

However, $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot communicate after the beginning of the game. It reflects the fact that in practice, an implementer may distribute its key generator, but does not necessarily wiretap the execution of the key-generation protocol for a particular user. From a technical viewpoint, the reason is that in a key-generation protocol, a user has to prove to the authority that she correctly performed her computation. However, the randomness used in these proofs can be used as a subliminal channel to convey information about the secret key. For instance, an engineer could in practice implement a bogus key generator which only terminates the protocol if the first bits of the proof and secret key match. The proof then serves as subliminal channel to leak information about the secret key. Later on, when a user wants to generate a certified public key, if the engineer could wiretap the protocol execution, he could infer some information about her secret key through the proof of correct computation. It is the reason why communication between the adversaries cannot be allowed.

The restriction that $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot communicate after the beginning of the game means that the attacks in which the protocol executions are listened to are excluded, but as explained above, it seems to be a minimal requirement.

Game Overview. At the beginning of the game, the challenger first runs an initialization algorithm. After that, $\mathcal{A}_1$ can make several queries to the algorithm instances. It can in particular

* specify distributions from which randomness is drawn and given as an input to the instances,
* ask for the protocol to be executed between different instances of the protocol algorithms without its intervention, i.e., perform passive attacks,
* perform active attacks by sending messages to algorithm instances of its choice,

- later on reveal the keys that were generated by a particular instance,
- corrupt a party (user or certificate authority), and thereby gain access to the state of all its algorithm instances.

As for $\mathcal{A}_2$, it can reveal keys or make a test query that returns either (with probability 1/2 each) keys freshly generated by the key-generation algorithm or keys generated by instances of its choice via queries made by $\mathcal{A}_1$. Adversary $\mathcal{A}_2$ must eventually return a guess for the origin of the keys it was returned, and $(\mathcal{A}_1, \mathcal{A}_2)$ wins the game if the guess of $\mathcal{A}_2$ is correct.

Initialization & Game Variables. During the initialization phase, game variables are declared for every instance of the protocol algorithms. Assume that there are at most $I = I(\lambda)$ instances of any participant id. Each instance $i \in I$ of a participant id maintains a state st^i_{id}. A session identity sid^i_{id} and a partner identity pid^i_{id} allow to match instances together in protocol executions. It is assumed that for each sid^i_{id} there can be at most one partner instance, i.e., one pair (id', j) such that $pid^i_{id} = id'$ and $sid^i_{id} \coloneqq \left(id, i, id', j, {sid^i_{id}}'\right)$.

Public/secret-key variables (denoted pk^i_{id} and sk^i_{id}) hold the keys that were output, if any, by the ith instance of the algorithm of party id at that step of the computation. For certificate authorities, the secret keys are always set to $\perp$.

A variable $used^i_{id}$ indicates whether the adversary has performed an active attack on the ith algorithm instance of participant id.

Variables acc^i_{id} and $term^i_{id}$ respectively indicate whether the algorithm of the ith instance of participant id has accepted and terminated. As in the BPR model [BPR00], termination and acceptance are distinguished. When an instance terminates, it does not output any further message. However, it may accept at a certain point of the computation, and terminate later. In the present context, it may for instance occur when an instance expects no further random input from its partner instance, and the rest of its computation is purely deterministic. It may then only terminate after finishing its computation. This distinction is crucial for the security definition. It is important to exclude the trivial case in which, although every computation was honestly performed, a user discards the public key if it does not follow a certain pattern, thereby influencing the distribution of the output public key (i.e., perform rejection sampling), and possibly using it as a subliminal channel to convey information about the secret key.

Another variable $flag^i_{id}$ (new compared to the BPR model) indicates whether party id was corrupted before its ith instance had accepted. Recall that acceptance intuitively means that an instance expects no further random input from its partner instance. As long as $flag^i_{id}$ is set to FALSE, the

only information the adversary has about r^i_{id} is its distribution and therefore, if this distribution has high min-entropy, the adversary cannot bias the generation of the keys.

A variable r^i_{id} holds the random string to be used the ith instance of the algorithm of id.

The challenger maintains a set (initially empty) Q_{Reveal} of identity–instance pairs of which the keys were revealed. It also maintains a set (initially empty) Q_{Corrupt} of corrupt identities.

At the end of the initialization phase, the public parameters, the sets of participants and the user public keys are returned in a public input pin, and the rest is set in a secret input sin. That is, $pin \leftarrow (pp, U, CA, I, (pk_{id})_{id})$ and $sin \leftarrow \Big(pin, (sk_{id})_{id}, \big(st^i_{id}, sid^i_{id}, pid^i_{id}, acc^i_{id}, term^i_{id}, used^i_{id}\big)_{i,id}, Q_{\mathsf{Corrupt}}, Q_{\mathsf{Reveal}}\Big)$. The secret input sin is later made available to all oracles.

Oracles.

Throughout the game, adversary $\mathcal{A}_1$ is given access to the oracles summarized below and defined in Figure 10.1. It can query them *one at a time*.

* Oracle : gives access to a function h chosen uniformly at random from a probability space Ω. The adversary and the protocol may depend on h. The probability space Ω specifies the model in which the protocol is considered. If it is empty, then it is the standard model. If it is a space of random functions, then it is the random oracle model. As for the Common-Reference String (CRS) model, Ω is a space of constant functions.

* Dist : via this oracle, the adversary specifies the distribution $\mathcal{D}^i_{id}$ from which the randomness of the ith instance of id is drawn. These distributions are always assumed to be independent of oracle Oracle. However, the distributions specified by the adversary for different instances *can be correlated in any way*. Oracle Dist then generates a bit string r^i_{id} according to the input distribution and does *not* return it to the adversary. Whenever oracle Exec or Send is queried on (id, i), it uses randomness r^i_{id} for its computation.

  This new (compared to the BPR model) oracle is essential to express requirements on the minimal entropy used by the instances, and also to express reasonable winning conditions. It allows to properly capture properties like the fact that (1) the authority cannot infer any information about the secret key if the randomness of the user algorithm has high entropy, (2) that the output keys are indistinguishable from keys generated with the key-generation algorithm if the randomness used by the algorithm of either of the parties has high entropy, or (3) that a potentially malicious user algorithm cannot bias the distribution of

the output keys if the randomness of the authority algorithm has high entropy. That is first because the test query later made by $\mathcal{A}_2$ requires the min-entropy of the randomness of either the challenge instance or of its partner to be high. It is also due to the fact that the adversary cannot corrupt the challenge instance (thus learning its randomness) before the partner randomness necessary to generate the challenge key is committed, which is monitored by the flags. It for instance means that if the CA is the target of the test and the adversary plays the role of a user algorithm (in which case the partner randomness is considered to have nil entropy) and possibly deviates from the protocol, then the test CA must be given high-entropy randomness and the definition ensures that the resulting key is indistinguishable from keys generated with KeyGen.

* Exec : returns the transcript of an honest (i.e., without the interference of the adversary) protocol execution between the ith instance of U and the jth instance of CA. The protocol is executed with the random strings generated for these instances by oracle Dist on the input of adversarial distributions. The notations U_i and CA_j mean that algorithms U and CA are executed using the state of the ith instance of U and the jth instance of CA respectively. It is implicitly assumed that the states $acc^i_{\mathcal{U}}$, $term^i_{\mathcal{U}}$, $acc^j_{\mathcal{CA}}$ and $term^j_{\mathcal{CA}}$ are set to TRUE after an honest protocol execution. Moreover, if the termination variable of either party is set to TRUE, the protocol is not executed and $\perp$ is returned. In essence, by querying oracle Exec, adversary $\mathcal{A}_1$ performs a passive eavesdropping attack.

* Send : adversary $\mathcal{A}_1$ can perform active attacks via this oracle. $\mathcal{A}_1$ can send any message to an instance of its choice, e.g., the ith instance of a user algorithm, which runs the honest protocol algorithm of the corresponding party on the input of the message chosen by the adversary.

  To prompt the ith instance of id to initiate a protocol execution with the jth instance of id', adversary $\mathcal{A}_1$ can make a Send query on $(id, i, (id', j))$.

  IKG$(id, *)$ denotes the IKG algorithm of party id, i.e., either U or CA. The algorithm is executed using the randomness generated by oracle Dist for that instance. (Note that the input random string may be used only at certain steps of the computation.) The oracle then returns the output of the instance to the adversary. It also specifies if this instance accepted and/or terminated, and returns the session identifier and the identity of its partner in the protocol execution, as well as the public and secret keys returned by this instance, if any. Note that if the instance is that of a certificate-authority algorithm, the secret key is

always set to $\perp$.

* Reveal : on input (id, i), returns the keys held by the ith instance of the algorithm of id. The couple (id, i) is added to the set Q_{Reveal} of revealed keys.

* $\mathsf{Corrupt}$: on input id, returns the states of all the instances of the algorithm of id. The identity id is added to the set Q_{Corrupt} of corrupt identities. Besides, for any instance i of id, if it has not yet accepted, $flag_{id}^{i}$ is set to TRUE.

Remark 10.2.4. *The first main difference with the BPR model is the new oracle* Dist. *It allows to capture an adversary running several instances of the protocol with correlated randomness. In the new model, it is also important to express winning conditions that exclude the trivial (and unavoidable) rejection-sampling attack. Another difference is that the variable* $flag_{id}^{i}$ *is set to* TRUE *if* $\mathcal{A}_1$ *corrupts* id *before its* i*th instance has accepted. It is to say that for instance, if an adversary (e.g., a malicious user algorithm) knows the randomness of the other party (by corrupting the CA) before it has "committed" to its randomness, then that party can influence the resulting key and break property (3).*

As for adversary $\mathcal{A}_2$, it is given access to oracles Oracle, Reveal and to oracle

* Test_b : on input (id^*, i^*), it returns the public key $pk_{id^*}^{i^*}$ generated via IKG (with an Exec query or Send queries) if $b = 0$ or a fresh public key generated via KeyGen if $b = 1$.

  An important restriction on this query is that the following condition must be satisfied: for any instance i of the algorithm of a party id_0, once it has accepted, i.e., once $acc_{id_0}^{i}$ is set to TRUE, the partner instance algorithm, say the jth instance of id_1, must eventually terminate, i.e., $term_{id_1}^{j}$ must have been set to TRUE as well by the time of query Test. It prevents $\mathcal{A}_1$ from biasing the distribution of the keys by prematurely aborting the protocol although it was followed, if the resulting key does not follow a certain pattern, and which would allow $\mathcal{A}_2$ to guess b with a non-negligible advantage.

  The other restrictions are simply that i^*-th instance of id^* must have terminated, that id^* was not corrupt before (id^*, i^*) had accepted[2], that neither the key of the i^*th instance of id^* nor of its partner instance has been revealed, and that the i^*th instance of id^* must already hold a key.

  Note that $\mathcal{A}_2$ can query Test only once. A definition with multiple queries would be asymptotically equivalent via a standard hybrid argument.

Adversary $\mathcal{A}_2$ must eventually return a bit b' as a guess for the origin (i.e., either IKG or KeyGen) of the key returned by oracle Test_b.

To achieve any form of indistinguishability from a key-generation algorithm, it is clear that either the distribution $\mathcal{D}^{i^*}_{id^*}$ or the distributions $\mathcal{D}^{j}_{id'}$ for the partner instance (j, id') of (i^*, id^*) must have high entropy. Indeed, if distributions with low entropy were allowed, $\mathcal{A}_1$ and $\mathcal{A}_2$ could agree on these identities, instances and distributions beforehand. Adversary $\mathcal{A}_2$ could then simply return 1 if and only if the challenge key is the most likely key w.r.t. $\mathcal{D}^{i^*}_{id^*}$ and $\mathcal{D}^{j}_{id'}$, and thereby win the game with a non-negligible advantage.

A parameter κ for the maximal min-entropy of $\mathcal{D}^{i^*}_{id^*}$ and $\mathcal{D}^{j}_{id'}$ specified by $\mathcal{A}_1$ is therefore introduced. If the adversary modified any message from the partner (j, id') of (id^*, i^*) before (id^*, i^*) accepts, then $\mathcal{D}^{j}_{id'}$ is set to be the Dirac mass at the zero bit-string by convention (and it thus has no entropy). The underlying idea is that as long as at least one of the two parties has a randomness source with high entropy, the key returned at the end of the protocol should be indistinguishable from a key generated by the KeyGen algorithm, which implies properties (1), (2) and (3). The security of a key-generation protocol is then defined for adversaries that specify challenge distributions with min-entropy at least κ.

Definition 10.2.5 (Indistinguishability)**.** *An interactive key-generation protocol* IKG *is* $(T, q_{\mathsf{Oracle}}, q_{\mathsf{Dist}}, q_{\mathsf{Exec}}, q_{\mathsf{Send}}, q_{\mathsf{Reveal}}, q_{\mathsf{Corrupt}}, \kappa, \varepsilon)$*-indistinguishable from a key-generation algorithm* KeyGen *(running on uniform randomness) if for all* $\lambda \in \mathbb{N}$*, for every adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *that runs in time at most* $T(\lambda)$ *and makes at most* $q_{\mathcal{O}}$ *queries to* $\mathcal{O} \in \{\mathsf{Oracle}, \mathsf{Dist}, \mathsf{Exec}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{Corrupt}\}$*, and such that* $\max\left(H_\infty\left(\mathcal{D}^{i^*}_{id^*}\right), H_\infty\left(\mathcal{D}^{j}_{id'}\right)\right) \geq \kappa$ *for query* Test*, the advantage (function of* λ*)*

$$\left|\Pr\left[b = b' : \begin{array}{l} (pin, sin) \leftarrow \mathsf{Init}\left(1^\lambda, U, CA, I\right) \\ \mathcal{O}_1 \leftarrow \{\mathsf{Oracle}, \mathsf{Dist}, \mathsf{Exec}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{Corrupt}\} \\ \mathcal{A}_1^{\mathcal{O}_1(sin,\cdot)}(pin) \\ b \leftarrow_{\$} \{0,1\} \\ \mathcal{O}_2 \leftarrow \{\mathsf{Oracle}, \mathsf{Reveal}, \mathsf{Test}_b\} \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2(sin,\cdot)}(pin) \\ \textit{return } (b, b') \end{array}\right] - 1/2\right|$$

[2] To understand why it is necessary for id^* not to be corrupt before (id^*, i^*) accepts even though $\mathcal{A}_1$ and $\mathcal{A}_2$ do not communicate, suppose that this condition were not imposed and consider the following strategy which allows $(\mathcal{A}_1, \mathcal{A}_2)$ to trivially win: $\mathcal{A}_1$ and $\mathcal{A}_2$ agree on (id^*, i^*) and on a distribution $\mathcal{D}^{i^*}_{id^*}$. Adversary $\mathcal{A}_1$ prompts (id^*, i^*) to initiate a protocol execution by making a Send query. It then corrupts id^* and obtains $st^{i^*}_{id^*}$, from which it can read $r^{i^*}_{id^*}$. Adversary $\mathcal{A}_1$ could then play the role of its partner and adapt the messages it sends to make sure that the resulting public key follows a certain pattern known to $\mathcal{A}_2$. This latter would then be able to win the game with a non-negligible advantage.

of $(\mathcal{A}_1, \mathcal{A}_2)$ *is at most* $\varepsilon(\lambda)$.

From a practical perspective, this definition (which implies requirement 3 as it enforces indistinguishability from keys generated by IKG) means that keys generated via a protocol satisfying the definition above are not subject to randomness vulnerabilities such as the ROCA vulnerabilities [NSS+17] mentioned in introduction (in which only specific primes were selected by user key generation algorithms) and those [LHA+12, HDWH12] in which several public keys are generated with correlated randomness sources.

10.3 Generic Constructions

This section presents a protocol that covers a wide class of key-generation algorithms, namely those that can be represented as probabilistic circuits, and another protocol specific to the generation of RSA keys. The first protocol is of theoretical interest and shows that randomness verifiability can be achieved for wide class of key-generation algorithms, whereas the second protocol is a solution that can actually be used in practice.

10.3.1 Key-Generation Protocol with Verifiable Randomness for Probabilistic Circuits

This section gives a key-generation protocol with verifiable randomness for a large class of key-generation algorithms. The focus is here on the class of key-generation algorithms that can be modeled as *probabilistic circuits*.

The advantage of probabilistic circuits compared to general Turing Machines for this purpose is that the running time of a probabilistic circuit is independent of the random inputs. In a key-generation protocol with verifiable randomness, the user has to prove to the authority that she correctly performed her computation. Having a constant running time then ensures that no one can infer any information about the secret key from the statement proved by the user or the proof itself. It prevents malicious user algorithms from using the proofs as subliminal channels to pass information about the secret key.

To understand why it is important for the running time to be constant, consider the artificial random number generator described on Algorithm 10.3.5. To generate a k-bit string $t = (t_0, \ldots, t_{k-1})$, it essentially consists in flipping a random coin s several times for each bit t_i and to set this bit to the parity of the number of flipped coins to obtain the first "Head". It produces a k-bit string uniformly distributed within expected time complexity $O(k)$ and it could be used as a secret-key generation algorithm (and the public key would then be a deterministic function of the generated secret key).

Algorithm 10.3.5 RNG with Non-Constant Running Time.

Require: Integer k.
Ensure: Uniformly random k bit-string x.

```
for i = 0 to k − 1 do
    c ← 0
    while TRUE do
        c ← (c + 1 mod 2)
        s ←$ {0, 1}
        if s = 0 then
            t_i ← c
            break
        end if
    end while
end for
t ← t_0 ‖ ··· ‖ t_{k−1}
return t
```

For a user to prove that she correctly generated the random bit string t, she would have to commit to the t_i values and compute a proof on the successive s values. However, each t_i is simply the parity of the number of trials before $s = 0$. Therefore, from the number of s values for which the user has to perform a proof, the authority can infer t_i. For example, if the user generated two s values for t_1, the authority knows that $t_1 = 0$.

In other words, the statement of the proof itself reveals some information about the secret key to the certification authority; and the issue is here that the running time changes from one random run of the algorithm to the other. Restricting to probabilistic circuits eliminates this issue.

The restriction to circuits comes at a cost though. It for instance excludes the class of algorithms for which there is no known circuit that can represent them. It is for instance the case of algorithms that must efficiently generate primes during the process. Indeed, there is no known circuit that can efficiently generate prime numbers. On this ground, the generic protocol for probabilistic circuits of Section 10.3.1 does not apply to the RSA-key generation for instance[3]. See rather Section 10.3.4 for the specific case of RSA-key generation with verifiable randomness for arbitrary properties that the keys must satisfy.

[3]One can construct families of probabilistic "circuits" which output an RSA key but *only* with overwhelming probability (and not probability 1) by relying on the prime number theorem and Chernoff's bound. However, to obtain a b-bit RSA public key with probability $1 - 2^{\lambda}$, such constructions would have $O\left(\lambda^2 b^4\right)$ gate complexity and $O\left(\lambda^2 b^2\right)$ randomness complexity (based on Miller-Rabin's primality test) or $O\left(\lambda b^7\right)$ gate complexity and $O\left(\lambda b^2\right)$ randomness complexity (based on Agrawal-Kayal-Saxena primality test [AKS04]). Applying our generic construction to such circuits family would result in schemes with prohibitive efficiency.

Before describing our protocol, we first formally define probabilistic circuits.

Probabilistic Circuits.

A probabilistic circuit is essentially a deterministic circuit augmented with uniformly random gates. The random gates produce independent and uniform random bits that are sent along their output wires.

We equivalently define a probabilistic circuit as a uniform random variable over a finite collection of deterministic boolean circuits. These boolean circuits are restricted to have the same amount n of input variables, and r fixed inputs. The number r of fixed inputs depends on the security parameter 1^λ. Denote such a circuit as $\Gamma_{b_1 \cdots b_r}(x_1, \ldots, x_n)$, with $x_1, \ldots, x_n$ the input variables and $b_1, \ldots, b_r$ the fixed inputs. To each element in $\{0,1\}^r$ corresponds a circuit in the collection with the bit string as fixed inputs, so that there are 2^r circuits in the collection. However, these circuits are not required to form a uniform family (i.e., they are not required to be output by a single Turing machine); the circuit families here considered can be non-uniform.

A probabilistic circuit Γ is then defined as a uniform random variable over the set (of circuits) $\{\Gamma_b\}_{b\in\{0,1\}^r}$. Namely, for input variables $x_1, \ldots, x_n$, the evaluation $\Gamma(x_1, \ldots, x_n)$ is a uniform random variable over the set (of values) $\{\Gamma_b(x_1, \ldots, x_n)\}_{b\in\{0,1\}^r}$. If $\omega \in \{0,1\}^r$ denotes the random input to the probabilistic circuit Γ, the evaluation $\Gamma(x_1, \ldots, x_n; \omega)$ is then $\Gamma_\omega(x_1, \ldots, x_n)$.

The advantage of this second definition is that randomness is invoked only once instead of invoking it for each of the r random gates. To generate keys, PRFs are often used to provide random bit strings from small secret seeds. As the goal is to build a key-generation protocol which allows the CA to certify that the keys are generated with high-entropy randomness, the user will have to prove that she correctly performed the pseudo-random evaluations. Invoking randomness only once then allows to invoke the PRF only once in the protocol.

Generic Protocol.

We now give a two-party protocol in the CRS model to generate, with verifiable randomness, keys computed by probabilistic circuits. Requiring that keys are generated with verifiable randomness here means that the random inputs to the circuits must be uniformly generated in a verifiable manner. The deterministic inputs to the circuits can simply be considered as public parameters.

Building Blocks. The protocol involves

- a function family $\mathcal{H} = \{H_{hk}\}_{hk \in \{0,1\}^{d(\lambda)}}$ which is a universal computational extractor w.r.t. unpredictable sources (Section 10.1.3)
- a two-source extractor Ext (Section 10.1.2) with key space $\{0,1\}^{\delta(\lambda)}$
- an extractable commitment scheme $\mathscr{C} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{ComVf}, \mathsf{TSetup}, \mathsf{ExtCom})$ for the user algorithm to commit to her random string *before receiving any input from the CA*, thereby preventing it from biasing the distribution of the keys. The parameters returned by Setup are implicit inputs to the other algorithms of $\mathscr{C}$
- a non-interactive, extractable, zero-knowledge proof system $\Pi = (\mathsf{Setup}, , \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{TSetup}^{zk}, \mathsf{Sim}, \mathsf{TSetup}^{ext}, \mathsf{Ext})$ for relation

$$\mathcal{R}_\Pi \coloneqq \{((x_i)_i, k, \mathcal{C}, r_{\mathcal{CA}}, pk; r'_{\mathcal{U}}, d, sk) : \mathsf{ComVf}(\mathcal{C}, r'_{\mathcal{U}}, d) = 1 \\ \wedge (pk, sk) = \Gamma(x_1, \ldots, x_n; \mathsf{Ext}_k(r'_{\mathcal{U}}, r_{\mathcal{CA}}))\},$$

- a pseudo-random function PRF to generate the randomness for $\Pi.\mathsf{Prove}$.

Parameters. Given a circuit Γ with deterministic inputs $x_1, \ldots, x_n$, to generate public parameters for the protocol on the input of a security parameter 1^λ, run $pp_{\mathscr{C}} \leftarrow \mathscr{C}.\mathsf{Setup}\left(1^\lambda\right)$ ($pp_{\mathscr{C}}$ is a tacit input to the algorithms of $\mathscr{C}$), $pp \leftarrow \Pi.\mathsf{Setup}\left(1^\lambda\right)$, $crs \leftarrow \Pi.\mathsf{CRSGen}(pp)$, and generate $hk \leftarrow_\$ \{0,1\}^{d(\lambda)}$ and $k \leftarrow_\$ \{0,1\}^{\delta(\lambda)}$. Return $pp \leftarrow (crs, pp_{\mathscr{C}}, hk, k, x_1, \ldots, x_n)$.

Formal Description. Consider the interactive protocol IKG_Γ on Figure 10.2 between a user $\mathcal{U}$ and a certification authority $\mathcal{CA}$. Each algorithm maintains acceptance and termination variables acc_{id} and $term_{id}$, for $id \in \{\mathcal{U}, \mathcal{CA}\}$, initially set to FALSE. On the input of pp and of their respective random strings $r_{\mathcal{U}}$ and $r_{\mathcal{CA}}$, the party algorithms proceed as follows:

1. U separates the domain of H_{hk} in two and applies it to its randomness. It commits to the first output with the second output as randomness, and sends the resulting commitment $\mathcal{C}$ to $\mathcal{CA}$
2. CA, upon receiving the commitment from $\mathcal{U}$, sets $acc_{\mathcal{CA}} \leftarrow \mathsf{TRUE}$ and sends its random string $r_{\mathcal{CA}}$ to $\mathcal{U}$
3. U, upon receiving $r_{\mathcal{CA}}$ from $\mathcal{CA}$, sets $acc_{\mathcal{U}} \leftarrow \mathsf{TRUE}$. Next, it extracts a seed s with Ext from the joint randomness. It evaluates Γ on $x_1, \ldots, x_n$ and s, and obtains a key pair (pk, sk). It generates another seed s' with H_{hk}. Algorithm U then evaluates PRF mode on s' to generate the randomness necessary to compute $\Pi.\mathsf{Prove}$ since U has no other random string than $r_{\mathcal{U}}$ available, i.e., it computes

$r_\Pi \leftarrow \mathsf{PRF}(s', 0)$. Algorithm U then proves that it followed the protocol and correctly evaluated Γ at $x_1, \ldots, x_n$, i.e., it computes a proof $\pi \leftarrow \Pi.\mathsf{Prove}\left(crs, ((x_i)_i, k, \mathcal{C}, r_{\mathcal{CA}}, pk), (r'_{\mathcal{U}}, d, sk); r_\Pi\right)$. After that, it erases all variables but pk, sk, π, sends pk and π to $\mathcal{CA}$, returns (pk, sk) and sets $term_{\mathcal{U}} \leftarrow \mathsf{TRUE}$

4. CA, upon receiving (pk, π) from $\mathcal{U}$, verifies the proof. If the proof is valid, it returns pk, otherwise it returns $\perp$. It then sets $term_{\mathcal{CA}} \leftarrow \mathsf{TRUE}$.

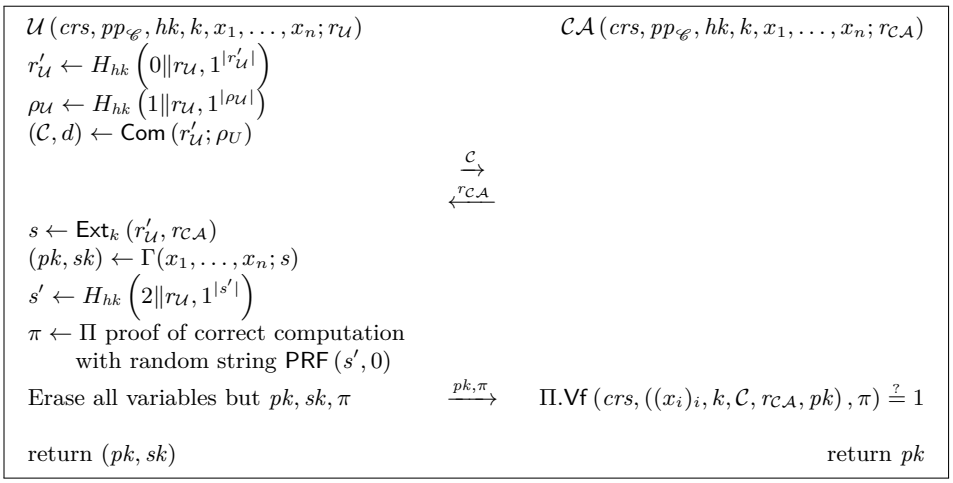

Figure 10.2: Key-Generation Protocol with Verifiable Randomness for Probabilistic Circuits.

Theorem 10.3.2 (Correctness)**.** *Protocol* IKG_Γ *is* 1*-correct w.r.t. all algorithms if* $\mathscr{C}$ *is correct and if* Π *is complete.*

Proof. The theorem immediately follows from the correctness of $\mathscr{C}$ and the completeness of Π. □

Theorem 10.3.3 (Indistinguishability)**.** *Suppose that* $\mathcal{H}$ *is* $(T^{\text{uce}}_{\mathcal{H}}, 3, \varepsilon^{\text{uce}}_{\mathcal{H}})$*-UCE secure w.r.t. simply unpredictable sources of min-entropy at least* $\kappa_{\mathcal{H}}$*. Suppose also that* Ext *is a* $(\kappa_{\mathsf{Ext}}, \varepsilon_{\mathsf{Ext}})$*-extractor for* $\kappa_{\mathsf{Ext}} \leq \min(|r'_{\mathcal{U}}|, |r_{\mathcal{CA}}|)$*. If* $\mathscr{C}$ *is* $\left(T^{\text{hide}}_{\mathscr{C}}, \varepsilon^{\text{hide}}_{\mathscr{C}}\right)$*-hiding, and satisfies* $\left(T^{\text{setup}-\text{ind}}_{\mathscr{C}}, \varepsilon^{\text{setup}-\text{ind}}_{\mathscr{C}}\right)$*-setup indistinguishability and* $\left(T^{\text{biding}}_{\mathscr{C},\mathsf{ExtCom}}, T^{\text{biding}}_{\mathscr{C},\mathcal{A}}, 0, \varepsilon^{\text{biding}}_{\mathscr{C}}\right)$*-binding extractability, if* Π *is* $\left(T^{\text{ext}}_{\mathsf{Ext}}, T^{\text{ext}}_{\mathcal{A},\Pi}, \varepsilon^{\text{ext}}\right)$*-extractable and* $\left(T^{\text{zk}}_{\Pi}, \varepsilon^{\text{zk}}\right)$*-composable zero-knowledge, and if* PRF *is* $(T_{\mathsf{PRF}}, 1, \varepsilon_{\mathsf{PRF}})$*-secure, then protocol* IKG_Γ *is* $(T, q_{\mathcal{O}}, \kappa, \varepsilon)$*-indistinguishable from* Γ *in the CRS model, for*

$$T = \min\left(T^{\text{uce}}_{\mathcal{H}},\ T^{\text{hide}}_{\mathscr{C}}, T^{\text{setup}-\text{ind}}_{\mathscr{C}}, T^{\text{biding}}_{\mathscr{C},\mathsf{ExtCom}}, T^{\text{biding}}_{\mathscr{C},\mathcal{A}}, T^{\text{ext}}_{\Pi,\mathsf{Ext}}, T^{\text{ext}}_{\Pi,\mathcal{A}}, T^{\text{zk}}_{\Pi}, T_{\mathsf{PRF}}\right),$$

$\mathcal{O} \in \{\mathsf{Dist}, \mathsf{Exec}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{Corrupt}\}$, $\kappa = \max(\kappa_{\mathcal{H}}, \kappa_{\mathsf{Ext}})$ *and* $\varepsilon \coloneqq 5\varepsilon_{\mathcal{H}}^{\mathrm{uce}} + 5\varepsilon_{\mathsf{Ext}} + \varepsilon_{\mathsf{PRF}} + \varepsilon_{\Pi}^{\mathrm{zk}} + \varepsilon_{\Pi}^{\mathrm{ext}} + \varepsilon_{\mathscr{C}}^{\mathrm{setup-ind}} + \varepsilon_{\mathscr{C}}^{\mathrm{bind-ext}} + \varepsilon_{\mathscr{C}}^{\mathrm{hide}}$.

Proof. First note that if $(\mathcal{A}_1, \mathcal{A}_2)$ wins the indistinguishability game with a probability at least ε, then there exists an adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ which wins with probability at least $\varepsilon/|ID||I|$ a variant of the game in which the adversary is required to specify the pair (id^*, i^*) of its Test query before being given access to the game oracles, i.e., a *selective* variant of the indistinguishability game. Indeed, $(\mathcal{A}'_1, \mathcal{A}'_2)$ can simply run $(\mathcal{A}_1, \mathcal{A}_2)$ as a subroutine and guess the pair (id^*, i^*) at the beginning of the game. If $(\mathcal{A}_1, \mathcal{A}_2)$ later makes its Test query on a different pair, $(\mathcal{A}'_1, \mathcal{A}'_2)$ simply aborts and sends to the challenger a bit chosen uniformly at random.

A message is subsequently said to be *oracle-generated* if it was computed by the challenger as a response to a Send query and it was not altered. Otherwise, the message is said to be *adversarially generated*.

Further distinguish two cases

1. $id^* \in CA$, or $id^* \in U$ and the random string $r_{\mathcal{CA}}$ the i^*th instance of id^* receives is oracle-generated.
2. $id^* \in U$ and the random string $r_{\mathcal{CA}}$ the i^*th instance of id^* receives is adversarially generated.

In the first case, as the commitment scheme satisfies binding extractability and as the proof system satisfies simulation extractability, the randomness used to generate the keys is really drawn from $\mathcal{D}^{i^*}_{id^*}$ and $\mathcal{D}^{j}_{id'}$. Moreover, since either $\mathcal{D}^{i^*}_{id^*}$ or $\mathcal{D}^{j}_{id'}$ has min-entropy at least κ, family $\mathcal{H}$ and extractor Ext guarantee that the public key generated during the protocol execution is indistinguishable from a one from the key-generation algorithm.

To prove it, consider the following sequence of games.

Game 0. This is the real selective game.

Game 1. In this game, to generate the common-reference string, the challenger runs $(crs, \tau_{ext}) \leftarrow \Pi.\mathsf{TSetup}^{ext}\left(1^{\lambda}\right)$ instead of $\Pi.\mathsf{setup}$ and $\Pi.\mathsf{CRSGen}$. This game is perfectly indistinguishable from the previous one since Π is extractable.

Game 2. To answer an Exec query on $(id^*, i^*, *, *)$ or on $(*, *, id^*, i^*)$, the challenger generates a transcript of an honest protocol execution. However, instead of setting $pk^{i^*}_{id^*} \leftarrow pk$, it generates a uniformly random string σ, runs $(pk', sk') \leftarrow \Gamma(x_1, \ldots, x_n; \sigma)$ and sets $pk^{i^*}_{id^*} \leftarrow pk'$.

Recall that $\mathcal{A}'_1$ and $\mathcal{A}'_2$ share no state, and that in the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, key $pk^{i^*}_{id^*}$ is not in Q_{Reveal} (nor is the key of the partner instance of (id^*, i^*)). Denoting by (id, j')

the partner instance of (id^*, i^*), since $\max\left(H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right), H_\infty\left(\mathcal{D}_{id'}^{j}\right)\right) \geq \kappa$, adversary $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}$.

Indeed, first suppose that $id^* \in U$. If $H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right) \geq \kappa \geq \kappa_{\mathcal{H}}$, then $r_{\mathcal{U}}'$ is $\varepsilon_{\mathcal{H}}^{\text{uce}}$-computationally indistinguishable from a uniformly random value. Since $|r_{\mathcal{U}}'| \geq \kappa_{\mathsf{Ext}}$, $\max\left(H_\infty\left(r_{\mathcal{U}}'\right), H_\infty\left(r_{id'}^{j}\right)\right) \geq \kappa_{\mathsf{Ext}}$ and s is then ε^{ext}-computationally indistinguishable from a uniformly random value. On the other hand, if $H_\infty\left(\mathcal{D}_{id'}^{j}\right) \geq \kappa \geq \kappa_{\mathsf{Ext}}$ then $\max\left(H_\infty\left(r_{\mathcal{U}}'\right), H_\infty\left(r_{\mathcal{CA}}\right)\right) \geq \kappa_{\mathsf{Ext}}$ and s is thus ε^{ext}-computationally indistinguishable from a uniformly random source. In either sub-case, $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}$.

In case $id^* \in CA$, a symmetric argumentation implies that $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}$.

Game 3. If $id^* \in U$, the challenger answers a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ as follows. It computes (pk, π) honestly, but instead of setting $pk_{id^*}^{i^*} \leftarrow pk$, it generates a uniformly random string σ, runs $(pk', sk') \leftarrow \Gamma(x_1, \ldots, x_n; \sigma)$ and sets $pk_{id^*}^{i^*} \leftarrow pk'$.

Recall that in case 1), since $r_{\mathcal{CA}}$ is always oracle-generated. Therefore, the same indistinguishability between the last two games still apply and $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}$.

Game 4. If $id^* \in CA$, the challenger answers a Send query on $(id^*, i^*, (pk, \pi))$ as follows. If (pk, π) is adversarially generated and valid, the challenger runs $(r_{\mathcal{U}}', d, sk) \leftarrow \Pi.\mathsf{Ext}\left(crs, \tau_{ext}, ((x_i)_i, k, \mathcal{C}, r_{\mathcal{CA}}, pk), \pi\right)$ and checks whether $((x_i)_i,\ k, \mathcal{C}, r_{\mathcal{CA}}, pk; r_{\mathcal{U}}', d, sk)$ is in $\mathcal{R}_\Pi$. If not, it aborts.

This game can be distinguished from the one only if the extractability of Π is contradicted. Therefore, $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the one with an advantage of at most $\varepsilon_\Pi^{\text{ext}}$.

Game 5. The challenger now runs $(pp_{\mathsf{Com}}, \tau_{\mathsf{Com}}) \leftarrow \mathscr{C}.\mathsf{TSetup}\left(1^\lambda\right)$ instead of running $\mathscr{C}.\mathsf{setup}$. The setup indistinguishability of $\mathscr{C}$ implies that adversary $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathscr{C}}^{\text{setup}-\text{ind}}$.

Game 6. If $id^* \in CA$, the challenger of this game answers a Send query on $(id^*, i^*, (pk, \pi))$ as follows. If (pk, π) is adversarially generated and valid, the challenger runs $(r_{\mathcal{U}}', \rho_{\mathcal{U}}, sk) \leftarrow \Pi.\mathsf{Ext}\left(crs, \tau_{ext}, ((x_i)_i, k, \mathcal{C}, r_{\mathcal{CA}}, pk), \pi\right)$. If $((x_i)_i,\ k, \mathcal{C}, r_{\mathcal{CA}}, pk; r_{\mathcal{U}}', d, sk)$ is in $\mathcal{R}_\Pi$, it extracts $r_{\mathcal{U}}'' \leftarrow \mathsf{ExtCom}(\tau_{\mathsf{Com}}, \mathcal{C})$ and if $r_{\mathcal{U}}' \neq r_{\mathcal{U}}''$, the challenger aborts.

This game can be distinguished from the previous one only if the binding extractability of $\mathscr{C}$ is contradicted. It follows that adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous with an advantage of at most $\varepsilon_{\mathscr{C}}^{\text{bind}-\text{ext}}$.

Game 7. If $id^* \in CA$, to answer a Send query on $(id^*, i^*, (pk, \pi))$, the challenger proceeds as the challenger of the previous game, and if it does not aborts, it generates a uniformly random string σ, computes $(pk', sk') \leftarrow \Gamma(x_1, \ldots, x_n; \sigma)$ and sets $pk_{id^*}^{i^*} \leftarrow pk'$.

Note that if $\mathcal{C}$ is adversarially generated, then $H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right) \geq \kappa$ as the distribution of the partner instance is set to a Dirac mass since (id^*, i^*) has not yet accepted, by definition of oracle Test. If $\mathcal{C}$ is oracle-generated, then $\max\left(H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right), H_\infty\left(\mathcal{D}_{id'}^{j}\right)\right) \geq \kappa$. The same arguments for the computational indistinguishability of Game 2 and Game 1 imply that $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}$.

In the last game, the condition that if an instance accepts then its partner must eventually terminate implies that $pk_{id^*}^{i^*}$ is computed by generating a uniformly random string σ and evaluating Γ at $(x_1, \ldots, x_n; \sigma)$. The advantage of the adversary in the last game is thus nil. IT follows that in case 1), the advantage of $(\mathcal{A}_1, \mathcal{A}_2)$ is at most

$$|ID|\,|I|\left(3\left(\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}\right) + \varepsilon_{\Pi}^{\text{ext}} + \varepsilon_{\mathscr{C}}^{\text{setup}-\text{ind}} + \varepsilon_{\mathscr{C}}^{\text{bind}-\text{ext}}\right).$$

In the second case, $id^* \in U$ and since $r_{\mathcal{CA}}$ is adversarially generated, $\mathcal{D}_{id^*}^{i^*}$ has min-entropy at least κ. The security of $\mathcal{H}$ and of Ext ensure that the commitment scheme is hiding and that the proof is zero-knowledge, which ensures that the protocol execution leaks no information about seed s. In addition to that, the high min-entropy of $\mathcal{D}_{id^*}^{i^*}$ also guarantees that s is indistinguishable from a uniformly random string. As a consequence, the resulting key is indistinguishable from one generated by the key-generation algorithm.

To prove it, consider the following sequence of games.

Game 0. This is the real selective game.

Game 1. In this game, to generate the common-reference string, the challenger runs $(crs, \tau_{zk}) \leftarrow \Pi.\mathsf{TSetup}^{zk}\left(1^\lambda\right)$ instead of $\Pi.\mathsf{Setup}$ and $\Pi.\mathsf{CRSGen}$. Adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most ε_{Π}^{zk}.

Game 2. This game is defined as in the previous case, and $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathcal{H}}^{\text{uce}} + \varepsilon_{\mathsf{Ext}}$.

Game 3. To answer a prompting Send query on $(id^*, i^*, (*, *))$, the challenger now generates $r'_{\mathcal{U}}$, ρ_U and s' uniformly at random. Since $r_{\mathcal{CA}}$ is adversarially generated in case 2) and that $\mathcal{U}$ accepts only after receiving it, the only information $(\mathcal{A}'_1, \mathcal{A}'_2)$ has about $r_{\mathcal{U}}$ is that it has a distribution $\mathcal{D}^{i^*}_{id^*}$ such that $H_\infty\left(\mathcal{D}^{i^*}_{id^*}\right) \geq \kappa \geq \kappa_{\mathcal{H}}$. In the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, instance (id^*, i^*) is not corrupt before it accepts, so the only information $\mathcal{A}'_1$ has about $r^{i^*}_{id^*}$ is that its distribution is $\mathcal{D}^{i^*}_{id^*}$. Moreover, if id^* is later corrupt before the end of the protocol execution, (id^*, i^*) will have already erased $r'_{\mathcal{U}}$ and $\rho_{\mathcal{U}}$ and s'.

Consequently, the UCE security of $\mathcal{H}$ can be reduced to distinguishing this game from the previous one, and $(\mathcal{A}'_1, \mathcal{A}'_2)$ can thus distinguish this game from the previous one with an advantage of at most $\varepsilon^{\text{uce}}_{\mathcal{H}}$.

Game 4. In this game, the challenger answers a Send query on $(id^*, i^*, r_{\mathcal{CA}})$, with $r_{\mathcal{CA}}$ adversarially generated, by generating uniformly random values instead of evaluating PRF at $(s', 0)$. If id^* is later corrupt before the end of the protocol execution, (id^*, i^*) will have already erased s' and r_Π. Adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathsf{PRF}}$.

Game 5. To answer a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ such that $r_{\mathcal{CA}}$ is adversarially generated, the challenger simulates a proof $\pi \leftarrow \Pi.\mathsf{Sim}\,(crs, \tau_{zk}, ((x_i)_i, k, \mathcal{C}, \ r_{\mathcal{CA}}, pk))$. By the composable zero-knowledge property of Π, this game is perfectly indistinguishable from the previous one.

Game 6. To answer a prompting Send query on $(id^*, i^*, (*, *))$, the challenger runs $(\mathcal{C}, d) \leftarrow \mathsf{Com}\left(0^{|r'_{\mathcal{U}}|}; \rho_{\mathcal{U}}\right)$ and sends $\mathcal{C}$. In the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, adversary $\mathcal{A}'_1$ does not corrupt id^* before (id^*, i^*) accepts, so not before $\rho_{\mathcal{U}}$ is erased. As $\mathscr{C}$ is $\varepsilon^{\text{hide}}_{\mathscr{C}}$-hiding, $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{\text{hide}}_{\mathscr{C}}$.

Game 7. To answer a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ such that $r_{\mathcal{CA}}$ is adversarially generated, the challenger generates s uniformly at random. As $|r'_{\mathcal{U}}| \geq \kappa_{\mathsf{Ext}}$, s is ε^{ext}-computationally indistinguishable from a uniformly random value. $(\mathcal{A}'_1, \mathcal{A}'_2)$ can then distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathsf{Ext}}$.

In the last game, the condition that if an instance accepts then its partner must eventually terminate then implies that $pk^{i^*}_{id^*}$ is computed by evaluating Γ at $(x_1, \ldots, x_n; \sigma)$, where σ is a uniformly random string. The advantage of $(\mathcal{A}'_1, \mathcal{A}'_2)$ in that game is then nil. As a result, the advantage of $(\mathcal{A}_1, \mathcal{A}_2)$

in the second case is at most

$$|ID|\,|I|\left(\varepsilon_{\Pi}^{zk}+2\varepsilon_{\mathcal{H}}^{\text{uce}}+2\varepsilon_{\mathsf{Ext}}+\varepsilon_{\mathsf{PRF}}+\varepsilon_{\mathscr{C}}^{\text{hide}}\right).$$

□

Discrete-Logarithm Keys.

This section instantiates the generic protocol of Figure 10.2 in the case of discrete-logarithm keys. For simplicity, we present a scheme based on Pedersen commitments and classical Schnorr-like zero-knowledge proofs.

Let G be a group family generator, λ be an integer and $(\mathbb{G},\ell,g)\leftarrow\mathsf{G}(\lambda)$. Let g_1,g_2 denote two generators of $\mathbb{G}$ used for the Pedersen commitment scheme. The key-generation protocol for discrete-logarithm keys is then the following.

1. U applies the random oracle $\mathcal{H}$ twice to its randomness $r_{\mathcal{U}}$ to compute $r'_{\mathcal{U}}\leftarrow\mathcal{H}(0\|r_{\mathcal{U}})$ and $\rho_U\leftarrow\mathcal{H}(1\|r_{\mathcal{U}})$, commits to $r'_{\mathcal{U}}$ (with randomness ρ_U) using Pedersen's commitment, computes a proof $\pi_{\mathcal{C}}$ of knowledge of an opening to the commitment with a Schnorr-type proof $\Pi_{\mathcal{C}}$ in the random-oracle model (with an oracle $\mathcal{H}_{\mathcal{C}}$), and sends the resulting commitment to $\mathcal{CA}$.

2. CA sets $acc_{\mathcal{CA}}\leftarrow\mathsf{TRUE}$, verifies the proof, and it is correct, sends its randomness $r_{\mathcal{CA}}$ to $\mathcal{U}$; otherwise it returns $\perp$ and sets $term_{\mathcal{CA}}\leftarrow\mathsf{TRUE}$

3. U extracts a seed from the joint randomness by computing

   $$s=r'_{\mathcal{U}}+\mathcal{H}(r_{\mathcal{CA}})\bmod\ell.$$

   It then computes a non-interactive zero-knowledge proof that it correctly performed its computation, i.e., it computes, in the random oracle model (with a different oracle $\mathcal{H}_{\Pi}$), a proof

   $$\pi\leftarrow\Pi.\mathsf{Prove}\left\{r'_{\mathcal{U}}\colon\mathcal{C}=g_1^{r'_{\mathcal{U}}}g_2^{\rho_{\mathcal{U}}}\wedge pk=g_1^{r'_{\mathcal{U}}}\cdot g_1^{\mathcal{H}(r_{\mathcal{CA}})}\right\}$$

   which is a standard Schnorr-like proof of representation. Since the only randomness available to U is $r_{\mathcal{U}}$, it is also used to generate the randomness necessary to compute the zero-knowledge proof. For this reason, U derives another seed $s'\leftarrow\mathcal{H}(0\|s)$ and then uses $\mathcal{H}$ with a counter on s'.

   After computing π, U erases $r_{\mathcal{U}}$, s and all temporary variables used to compute π. Algorithm U then sends $pk\leftarrow(g_1,g_1^s)$ and π to $\mathcal{CA}$, returns $(pk_{\mathcal{U}}\leftarrow pk, sk_{\mathcal{U}}\leftarrow s)$ and sets $term_{\mathcal{U}}\leftarrow acc_{\mathcal{U}}\leftarrow\mathsf{TRUE}$

4. CA verifies π. If this verification did not succeed, it returns $\perp$, otherwise it returns $pk_{\mathcal{CA}}\leftarrow pk$. It sets $term_{\mathcal{CA}}\leftarrow\mathsf{TRUE}$.

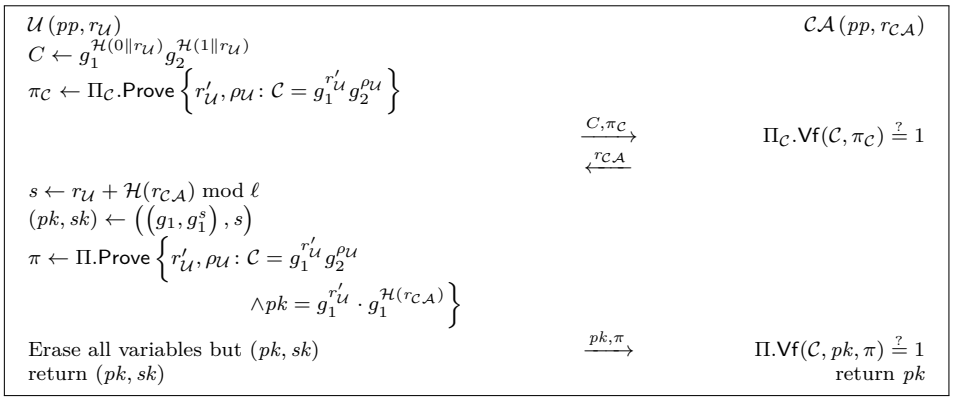

Figure 10.3: Discrete-Logarithm Key-Generation Protocol with Verifiable Randomness.

This protocol is 1-correct as Schnorr proofs are. In the random-oracle model and under the discrete-logarithm assumption over G, it also satisfies the indistinguishability security notion of Section 10.2.3 w.r.t. randomness sources with $\Omega(\lambda)$ min-entropy which are independent of the random-oracle.

10.3.4 RSA-Key Generation Protocol with Verifiable Randomness

This section gives a two-party protocol for RSA-key generation with verifiable randomness between a user $\mathcal{U}$ and a certification authority $\mathcal{CA}$. The resulting keys can be used in any RSA cryptosystem. The protocol attests that the resulting keys were generated with high-entropy randomness and that they satisfy (fixed) arbitrary properties. These properties are captured by a relation

$$\mathcal{R}_W \coloneqq \{(N, e \in \mathbb{Z}; p, q \in W \subseteq \mathbb{P}) : p \neq q \wedge N = pq \wedge \gcd(e, \varphi(N)) = 1\}$$

to which the keys generated should belong, where W is a set that defines the predicates p and q must satisfy, e.g., $p = q = 3 \mod 4$ or p and q are safe primes. Its relative language is denoted $\mathcal{R}_W$. Efficient proof systems for such properties exist [vP88, CM99, AP18], though none of them aims at proving that the keys were generated with proper randomness.

In comparison, the protocol by Juels and Guajardo [JG02] only guarantees the first two properties, and does *not* ensure that the user algorithm cannot bias the distribution of the keys. Without the third property, an interactive key-generation protocol is only beneficial if the user does not have high-entropy randomness locally whereas the CA does, otherwise it is only a burden for the user. On the other hand, the third property additionally guarantees the end user that if the CA has high-entropy randomness, her keys are not faulty.

As for the attestation scheme of Benhamouda et al. [BFGN17], it allows to prove that the RSA primes were generated with an arbitrary generator; and the protocols of Camenisch and Michels [CM99], of Auerbach and Poettering [AP18], and of Goldberg et al. [GRSB19], only allow to prove that RSA primes satisfy certain properties, not that they were generated with high entropy. In a sense, our goal is complementary to that of proving that RSA moduli satisfy certain properties without proving that the keys were generated with high-entropy randomness.

RSA Key-Generation Algorithm. The NIST standard [NIS13] for the RSA [RSA78] key-generation algorithm, further denoted $\mathsf{KeyGen}_{\mathrm{RSA}}$, is the following:

- choose at random two distinct large primes p and q
- compute $N \leftarrow pq$ and $\varphi(N) \leftarrow (p-1)(q-1)$
- choose an integer $2^{16} < e < 2^{256}$ such that $\gcd(e, \varphi(N)) = 1$ (e may be chosen deterministically or at random); compute $d \leftarrow e^{-1} \mod \varphi(N)$
- Return $pk \leftarrow (N, e)$ and $sk \leftarrow (N, d)$.

Alternatively, the secret key sk can be set to (p, q, e) instead of (N, d) as one can compute (N, d) from (p, q, e). It is this variant that is hereafter considered. To formally capture the requirement on p and q to be large, a parameter $b = b(\lambda)$ that specifies the bit-length of p and q is introduced.

Interpretation. There is some ambiguity as to how p and q are generated. The interpretation (which follows how the algorithm would implemented in practice) of $\mathsf{KeyGen}_{\mathrm{RSA}}$ in the rest of the chapter is first that there exists a PPT primality-test algorithm $\mathsf{PrimeTest}_W(\lambda, b, e, p) \to \zeta \in \{0, 1\}$ (parameter λ is further omitted from its syntax) which tests whether an integer p is in W, b-bit long and such that $\gcd(e, (p-1)) = 1$. Algorithm $\mathsf{KeyGen}_{\mathrm{RSA}}$ then generates, uniformly at random, integers in $[\![2^{b-1}, 2^b - 1]\!]$ until it finds an integer p such that $\mathsf{PrimeTest}_W(b, e, p) = 1$, and continues until it finds a second one $q \neq p$ such that $\mathsf{PrimeTest}_W(b, e, q) = 1$. If no such two integers are found in a specified number of iterations $T_{\mathrm{RSA}}(\lambda)$, the algorithm aborts and returns an invalid pair, e.g., $(0, 0)$. The random variable with values in $\{0, 1, 2\}$ that counts the number of distinct primes found in at most $T_{\mathrm{RSA}}(\lambda)$ iterations is further denoted ctr_{RSA}.

Protocol.

We now describe our protocol, further denoted $\mathsf{IKG}_{\mathrm{RSA}}$, to generate RSA keys with verifiable randomness. The protocol is given in the random-oracle model to allow for practical efficiency.

Building Blocks. The protocol builds on

- the same primality-test algorithm $\mathsf{PrimeTest}_W$ as the one run by algorithm $\mathsf{KeyGen}_{\mathrm{RSA}}$. It is said to be δ-*correct* if with probability at most $1-\delta$, $\mathsf{PrimeTest}_W(b,e,p)=0$ for $p \in W \cap [\![2^{b-1}, 2^b-1]\!]$ such that $\gcd(e,(p-1))=1$, or $\mathsf{PrimeTest}_W(b,e,p)=1$ for $p \notin W \cap [\![2^{b-1}, 2^b-1]\!]$ or such that $\gcd(e,(p-1))>1$ (i.e., it is an upper-bound on the probability that it returns a false negative or a false positive)

- a random oracle of which the domain is separated to obtain pairwise independent random oracles $\mathcal{H}$, $\mathcal{H}_{\mathcal{C}}$, $\mathcal{H}_{\Pi}$ and $\mathcal{H}_{\Pi_W}$

- a commitment scheme $\mathscr{C} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{ComVf})$ (App. 9.2) for the user algorithm to commit to its random string *before* receiving any input from the CA. The parameters returned by Setup are tacit inputs to $\mathscr{C}$ other algorithms.

- a pseudo-random function PRF with range (non-empty) $R_{\mathsf{PRF}} \subseteq \mathbb{N}$ for $\mathcal{U}$ to generate the RSA primes from the seed extracted with $\mathcal{H}$

- an extractable, non-interactive zero-knowledge argument system $\Pi_{\mathcal{C}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{Sim}, \mathsf{Ext})$ for the relation

$$\{(C; r'_{\mathcal{U}}, d) : \mathsf{ComVf}(C, r'_{\mathcal{U}}, d) = 1\}$$

  with random oracle $\mathcal{H}_{\mathcal{C}}$, i.e., for the user to prove knowledge of an opening to her committed randomness

- an extractable NIZK argument system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{Sim}, \mathsf{Ext})$ with random oracle $\mathcal{H}_{\Pi}$ for the relation

$$\left\{(C, r_{\mathcal{CA}}, N, (a_\gamma)_{\gamma \neq i,j}; r'_{\mathcal{U}}, d, a_i, a_j) : \begin{array}{r} \mathsf{ComVf}\,(\mathcal{C}, r'_{\mathcal{U}}, d) = 1, \\ s = r'_{\mathcal{U}} \oplus \mathcal{H}(r_{\mathcal{CA}}), \forall \gamma \in [\![j]\!]\ a_\gamma = \mathsf{PRF}(s,\gamma), \\ 2^{b(\lambda)-1} \leq a_i, a_j \leq 2^{b(\lambda)} - 1, N = a_i a_j \text{ in } \mathbb{N} \end{array}\right\},$$

  i.e., for the user to prove the RSA primes are really the *first two primes* generated with the seed derived from the committed randomness and the randomness of the CA. This relation is further denoted $\mathcal{R}_{\Pi}$

- a NIZK argument system $\Pi_W = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{Sim})$ with random oracle $\mathcal{H}_{\Pi_W}$ for relation $\mathcal{R}_W$

- another pseudo-random function PRF' with variable output length (encoding in unary as last input) for $\mathcal{U}$ to generate the randomness necessary[4] to compute $\Pi_{\mathcal{C}}.\mathsf{Prove}$, $\mathsf{PrimeTest}_W$, $\Pi.\mathsf{Prove}$ and $\Pi_W.\mathsf{Prove}$, as

the only available randomness to the parties are their input random bit strings.

Throughout the protocol, e is assumed (without loss of generality) to be a fixed[5], hard-coded value in U. For the sake of simplicity, e is further assumed to be prime, e.g., $e = 65537$ (it is a value commonly used in practice).

Parameters. Given a security parameter 1^λ and a function $T\colon \mathbb{N} \to \mathbb{N}_{>1}$ that gives an upper bound on the number of iterations in Algorithm 10.3.6 (and thus the running time of U), to generate parameters for $\mathsf{IKG}_{\mathrm{RSA}}$, run $pp_{\mathscr{C}} \leftarrow \mathscr{C}.\mathsf{Setup}\left(1^\lambda\right)$, $pp_{\Pi_{\mathcal{C}}} \leftarrow \Pi_{\mathcal{C}}.\mathsf{Setup}\left(1^\lambda\right)$, $pp_{\Pi} \leftarrow \Pi.\mathsf{Setup}\left(1^\lambda\right)$ and $pp_{\Pi_W} \leftarrow \Pi_W.\mathsf{Setup}\left(1^\lambda\right)$. Set and return $pp \leftarrow \Big(b(\lambda), T(\lambda), pp_{\mathscr{C}}, pp_{\Pi_{\mathcal{C}}}, pp_{\Pi}, pp_{\Pi_W}\Big)$.

1. U applies the random oracle $\mathcal{H}$ twice to its randomness $r_{\mathcal{U}}$ to compute $r'_{\mathcal{U}} \leftarrow \mathcal{H}(0\|r_{\mathcal{U}})$ and $\rho_U \leftarrow \mathcal{H}(1\|r_{\mathcal{U}})$, commits to $r'_{\mathcal{U}}$ with ρ_U as random string. Next, a seed $s' \leftarrow \mathcal{H}(2\|r_{\mathcal{U}})$ from which it derives the randomness necessary to compute $\Pi_{\mathcal{C}}.\mathsf{Prove}$, and computes of proof of knowledge of an opening to the commitment. U sends the commitment and the proof to $\mathcal{CA}$

2. CA, upon receiving the commitment and the proof from $\mathcal{U}$, sets $acc_{\mathcal{CA}} \leftarrow \mathsf{TRUE}$. It verifies the proof and if it holds, sends its randomness to $\mathcal{U}$, and otherwise returns $\perp$ and sets $term_{\mathcal{CA}} \leftarrow \mathsf{TRUE}$

3. U, upon receiving $r_{\mathcal{CA}}$ from $\mathcal{CA}$, sets $acc_{\mathcal{U}} \leftarrow \mathsf{TRUE}$. It extracts a seed s with $\mathcal{H}$ from the joint randomness. It continues by generating by running $\Big((a_\gamma)_{\gamma=1}^{j}, i\Big) \leftarrow$ Algorithm 10.3.6.

   (a) if $i = \perp$ (i.e., Algorithm 10.3.6 did not find 2 primes such that $\mathsf{PrimeTest}_W(b, e, a_j; \mathsf{PRF}(s, j)) = 1$ in T iterations; this case is not depicted on Figure 10.4), U sends $\Big(r_{\mathcal{U}}, (a_\gamma)_{\gamma=1}^{j}\Big)$ to $\mathcal{CA}$, returns $(0, 0)$ and sets $term_{\mathcal{U}} \leftarrow \mathsf{TRUE}$

   (b) if $i \neq \perp$, U computes a proof π that it correctly performed its computation with Π, and a proof π_W that the RSA public key is in $\mathcal{L}_W$ with Π_W. After computing the proofs, U erases all variables

[4] Juels and Guajardo's protocol [JG02] allows for probabilistic primality-test algorithms and makes uses of proof systems, but does not specify the origin of the randomness necessary for their computation or the zero-knowledge property of the proof systems.

[5] Alternatively, in the protocol on Figure 10.4, after N is computed, U could continue to generate pseudo-random values until it finds one that is coprime with $\varphi(N)$ and then sets it as e. Algorithm U would then also have to reveal the values that did not satisfy this property and prove that they did not, and also to prove that the chosen e and $\varphi(N)$ are coprime. Assuming e to be fixed in advance avoids this complication.

Algorithm 10.3.6

Require: $\mathsf{PrimeTest}_W$, integers T, b, e, pseudo-random function PRF, seed s.

Ensure: Pseudo-random numbers a_γ and integer i.

```
ctr, i, j ← 0
while ctr < 2 and j < T do
    j ← j + 1; a_j ← PRF(s, j)
    if PrimeTest_W (b, e, a_j; PRF(s, j)) then
        if ctr = 0 then
            i ← j
        end if
        ctr ← ctr + 1
    end if
end while
if ctr < 2 then
    return ((a_γ)_{γ=1}^{j}, ⊥)
else
    return ((a_γ)_{γ=1}^{j}, i)
end if
```

but N, e, p, q, i, π, π_W and $(a_\gamma)_{\gamma\neq i,j}$. It sends these latter to $\mathcal{CA}$, except p and q, returns $(pk_\mathcal{U} \leftarrow (N,e), sk \leftarrow (p,q,e))$, and sets $term_\mathcal{U} \leftarrow \mathsf{TRUE}$

4a. CA, upon receiving $\left(r_\mathcal{U}, (a_\gamma)_{\gamma=1}^{j}\right)$ from $\mathcal{U}$, computes $r'_\mathcal{U}$, $\rho_\mathcal{U}$ and s as U, computes $(\mathcal{C}', d') \leftarrow \mathsf{Com}\,(r'_\mathcal{U}, \rho_\mathcal{U})$, and verifies that $\mathcal{C}' = \mathcal{C}$ and that $\mathsf{PRF}(s,\gamma) = a_\gamma$ for all $\gamma \in [\![j]\!]$. If all verifications succeed, CA returns 0, otherwise it returns $\perp$. It sets $term_\mathcal{CA} \leftarrow \mathsf{TRUE}$

4b. CA, upon receiving $(N, e, \pi, \pi_W, i, (a_\gamma)_{\gamma\neq i,j})$ from $\mathcal{U}$, generates a seed s'' with $\mathcal{H}$ from its randomness, and uses it to generate the randomness necessary to compute $\mathsf{PrimeTest}_W$. The resulting random string is denoted r'_W. It verifies that for all $\gamma \in [\![j-1]\!] \setminus \{i\}$, $\mathsf{PrimeTest}_W(b, e, a_\gamma; r'_W) = 0$, and that π and π_W are valid. If one of the verifications did not succeed, CA returns $\perp$, otherwise it returns $pk_\mathcal{CA} \leftarrow (N,e)$. It sets $term_\mathcal{CA} \leftarrow \mathsf{TRUE}$.

Theorem 10.3.5 (Correctness)**.** *Let j be the number of iterations of Algorithm 10.3.6, and suppose that $\mathsf{PrimeTest}_W$ is δ-correct and that PRF' is $(T_{\mathsf{PRF}'}, j+3, \varepsilon_{\mathsf{PRF}'})$-secure for $T_{\mathsf{PRF}'} = \Omega(T \cdot T_{\mathsf{PrimeTest}})$. If $\mathscr{C}$ is correct and if $\Pi_\mathcal{C}$, Π and Π_W are complete, protocol $\mathsf{IKG}_{\mathrm{RSA}}$ is*

$$\max\left(1 - j(1-\delta) - 2\varepsilon_{\mathsf{PRF}'} - q_\mathcal{H}\left(2^{-\kappa_\mathcal{U}} + 2^{-\kappa_\mathcal{CA}}\right), 0\right)$$

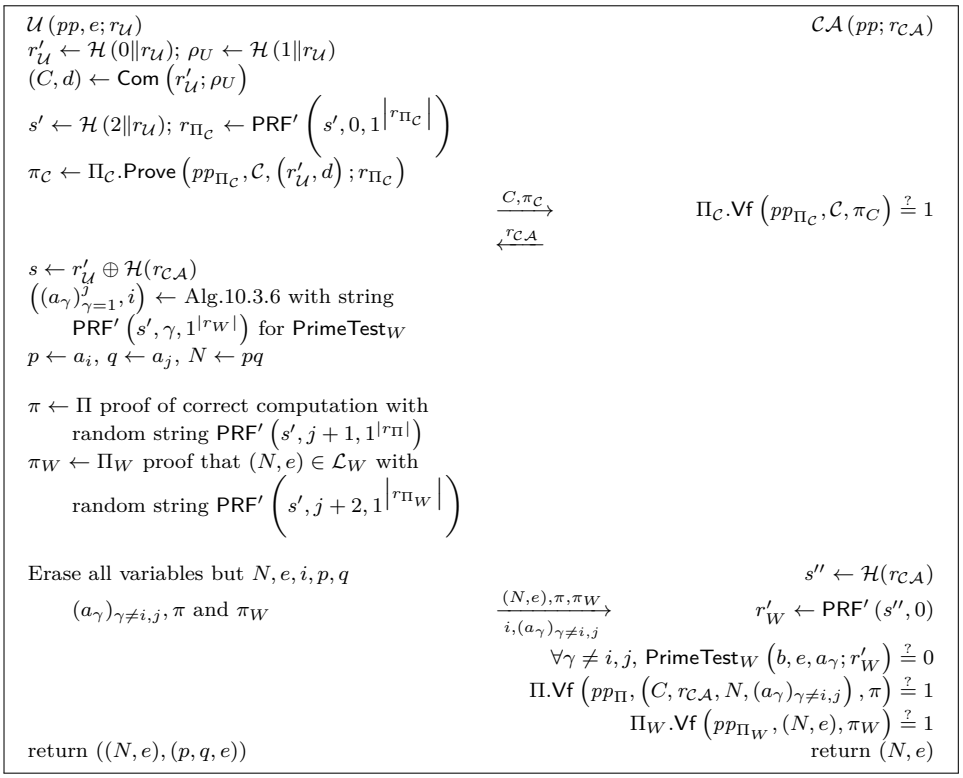

Figure 10.4: RSA-Key Generation Protocol with Verifiable Randomness for an Arbitrary Relation $\mathcal{R}_W$.

-correct in the random-oracle model w.r.t. to the class of algorithms that run in time at most $T_{\mathsf{PRF}'}$, make at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$ and return distributions $\mathcal{D}_{\mathcal{U}}$ and $\mathcal{D}_{\mathcal{CA}}$ independent of the random oracle and with min-entropy at least $2^{-\kappa_{\mathcal{U}}}$ and $2^{-\kappa_{\mathcal{CA}}}$, respectively.

Proof. Assuming $\mathscr{C}$ to be correct and $\Pi_{\mathcal{C}}$, Π and Π_W to be complete, a protocol in which uniformly random values are generated instead of s' and s'' is $\max(1 - j(1-\delta), 0)$-correct as an honest protocol execution fails only if $\mathsf{PrimeTest}$ does.

Since $\mathcal{D}_{\mathcal{U}}$ and $\mathcal{D}_{\mathcal{CA}}$ are assumed to be independent of the random oracle and to have min-entropy at least with min-entropy at least $2^{-\kappa_{\mathcal{U}}}$ and $2^{-\kappa_{\mathcal{CA}}}$, respectively, a protocol in which s and s' are uniformly random is $q\left(2^{-\kappa_{\mathcal{U}}} + 2^{-\kappa_{\mathcal{CA}}}\right)$-statistically indistinguishable from the previous one.

Consider now the following algorithm which interacts with the PRF challenger for PRF'. Given user and certification-authority distributions, the algorithm runs the key-generation protocol as a subroutine with random strings drawn from the specified distributions, and queries the challenger when PRF' is to be evaluated on s' (which is exactly $j+3$ times). The algorithm returns 1 if the protocol execution fails and 0 otherwise. The

security of PRF' implies that a protocol execution with PRF' evaluations on s' is $\max\left(1-j(1-\delta)-\varepsilon_{\mathsf{PRF}'}-q\left(2^{-\kappa_{\mathcal{U}}}+2^{-\kappa_{\mathcal{CA}}}\right),0\right)$-correct.

Likewise, consider another algorithm which runs the key-generation protocol as a subroutine with random strings drawn from the specified distributions, and queries the challenger when PRF' is to be evaluated on s'' (which is exactly once). The algorithm returns 1 if the protocol execution fails and 0 otherwise. The security of PRF' implies that a protocol execution with PRF' evaluations on s' and on s'' is

$$\max\left(1-j(1-\delta)-2\varepsilon_{\mathsf{PRF}'}-q\left(2^{-\kappa_{\mathcal{U}}}+2^{-\kappa_{\mathcal{CA}}}\right),0\right)$$

-correct. □

The following theorem shows that the output of $\mathsf{IKG}_{\mathrm{RSA}}$ is indistinguishable from that of $\mathsf{KeyGen}_{\mathrm{RSA}}$ in the random oracle if $\mathscr{C}$ is hiding and binding, if $\Pi_{\mathcal{C}}$ and Π are zero-knowledge and extractable, if Π_W is zero-knowledge and sound, if PRF and PRF' are secure PRFs, if the probability that Algorithm 10.3.6 fails although $\mathsf{IKG}_{\mathrm{RSA}}$ does not is small, and if the adversary makes few random-oracle queries compared to the min-entropy of the test distributions.

Theorem 10.3.6 (Indistinguishability)**.** *Suppose that* $\mathscr{C}$ *is* $\left(T_{\mathscr{C}}^{\mathrm{hide}},\varepsilon_{\mathscr{C}}^{\mathrm{hide}}\right)$*-hiding and* $\left(T_{\mathscr{C}}^{\mathrm{bind}},\varepsilon_{\mathscr{C}}^{\mathrm{bind}}\right)$*-binding, that* PRF *is* $(T_{\mathsf{PRF}},j,\varepsilon_{\mathsf{PRF}})$*-secure, that* PRF' *is* $(T_{\mathsf{PRF}'},\ j+3,\varepsilon_{\mathsf{PRF}'})$*-secure, that* $\Pi_{\mathcal{C}}$ *is* $\left(T_{\Pi_{\mathcal{C}}}^{\mathrm{ext}},q_{\mathcal{H}_{\mathcal{C}}},\varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{ext}}\right)$*-extractable and* $\left(T_{\Pi_{\mathcal{C}}}^{\mathrm{zk}},q_{\mathcal{H}_{\mathcal{C}}},1,\varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{zk}}\right)$*-zero-knowledge, that* Π *is* $(T_{\Pi}^{\mathrm{ext}},q_{\mathcal{H}_{\Pi}},\varepsilon_{\Pi}^{\mathrm{ext}})$*-extractable and* $\left(T_{\Pi}^{\mathrm{zk}},q_{\mathcal{H}_{\Pi}},1,\varepsilon_{\Pi}^{\mathrm{zk}}\right)$*-zero-knowledge, and that* Π_W *is* $\left(T_{\Pi_W}^{\mathrm{sound}},q_{\mathcal{H}_W},\varepsilon_{\Pi_W}^{\mathrm{sound}}\right)$*-extractable and* $\left(T_{\Pi_W}^{\mathrm{zk}},q_{\mathcal{H}_W},1,\varepsilon_{\Pi_W}^{\mathrm{zk}}\right)$*-zero-knowledge. Protocol* $\mathsf{IKG}_{\mathrm{RSA}}$ *is* $(T,$ $q_{\mathcal{O}},\kappa,\varepsilon)$*-indistinguishable from* $\mathsf{KeyGen}_{\mathrm{RSA}}$ *in the random-oracle model, for*

$$T=\min\left(T_{\mathscr{C}}^{\mathrm{hide}},T_{\mathscr{C}}^{\mathrm{bind}},\ T_{\mathsf{PRF}},T_{\mathsf{PRF}'},T_{\Pi_{\mathcal{C}}}^{\mathrm{ext}},T_{\Pi_{\mathcal{C}}}^{\mathrm{zk}},T_{\Pi}^{\mathrm{ext}},\ T_{\Pi}^{\mathrm{zk}},T_{\Pi_W}^{\mathrm{sound}},T_{\Pi_W}^{\mathrm{zk}}\right),$$

$\mathcal{O}\in\{\mathsf{Oracle},\mathsf{Dist},\mathsf{Exec},\mathsf{Send},\mathsf{Reveal},\mathsf{Corrupt}\}$*,* $q_{\mathsf{Oracle}}\geq q_{\mathcal{H}}+q_{\mathcal{H}_{\mathcal{C}}}+q_{\mathcal{H}_{\Pi}}+q_{\mathcal{H}_W}$*, and*

$$\begin{aligned}\varepsilon\coloneqq|ID|\,|I|\Big(&2^{-\kappa}q_{\mathcal{H}}+5\left(2^{-\kappa}q_{\mathcal{H}}+\varepsilon_{\mathsf{PRF}}+|R_{\mathsf{PRF}}|^{-1}+\Pr[ctr<2,ctr_{\mathrm{RSA}}=2]\right)\\&+\varepsilon_{\mathsf{PRF}'}+\varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{ext}}+\varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{zk}}+\varepsilon_{\Pi}^{\mathrm{ext}}+\varepsilon_{\Pi}^{\mathrm{zk}}+\varepsilon_{\Pi_W}^{\mathrm{sound}}+\varepsilon_{\Pi_W}^{\mathrm{zk}}+\varepsilon_{\mathscr{C}}^{\mathrm{bind}}+\varepsilon_{\mathscr{C}}^{\mathrm{hide}}\Big).\end{aligned}$$

Proof. First note that if $(\mathcal{A}_1,\mathcal{A}_2)$ wins the indistinguishability game with a probability at least ε, then there exists an adversary $(\mathcal{A}_1',\mathcal{A}_2')$ which wins with probability at least $\varepsilon/|ID||I|$ a variant of the game in which the adversary is required to specify the pair (id^*,i^*) of its Test query before being given access to the game oracles, i.e., a *selective* variant of the indistinguishability game. Indeed, $(\mathcal{A}_1',\mathcal{A}_2')$ can simply run $(\mathcal{A}_1,\mathcal{A}_2)$ as a subroutine and

guess the pair (id^*, i^*) at the beginning of the game. If $(\mathcal{A}_1, \mathcal{A}_2)$ later makes its Test query on a different pair, $(\mathcal{A}'_1, \mathcal{A}'_2)$ simply aborts and sends to the challenger a bit chosen uniformly at random.

A message is subsequently said to be *oracle-generated* if it was computed by the challenger as a response to a Send query and it was not altered. Otherwise, the message is said to be *adversarially generated*.

Further distinguish two cases

1. $id^* \in CA$, or $id^* \in U$ and the random string $r_{\mathcal{CA}}$ the i^*th instance of id^* receives is oracle-generated.
2. $id^* \in U$ and the random string $r_{\mathcal{CA}}$ the i^*th instance of id^* receives is adversarially generated.

In the first case, the situation is similar to the first case of the proof of Theorem 10.3.3, except that the arguments are in the random-oracle model, and that $\mathscr{C}$ is not assumed to be extractable, so the extraction is rather done via $\Pi_{\mathcal{C}}$. Consider then the following sequence of games.

Game 0. This is the real selective game.

Game 1. To answer an Exec query on $(id^*, i^*, *, *)$ or on $(*, *, id^*, i^*)$, the challenger generate an honest transcript of the protocol. However, instead of setting $pk^{i^*}_{id^*} \leftarrow pk$, it generates a uniformly random string σ, runs $(pk', sk') \leftarrow \mathsf{KeyGen}_{\mathrm{RSA}}(b, e, W; \sigma)$ and sets $pk^{i^*}_{id^*} \leftarrow pk$.

Recall that $\mathcal{A}'_1$ and $\mathcal{A}'_2$ share no state, and that in the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, key $pk^{i^*}_{id^*}$ is not in Q_{Reveal} (nor is the key of the partner of (id^*, i^*). Denoting by (id, j') the partner instance of (id^*, i^*), since $\max\left(H_\infty\left(\mathcal{D}^{i^*}_{id^*}\right), H_\infty\left(\mathcal{D}^{j}_{id'}\right)\right) \geq \kappa$, adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}} + \varepsilon_{\mathsf{PRF}} + |R_{\mathsf{PRF}}|^{-1} + \Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$.

Indeed, in Algorithm 10.3.6, if $ctr < 2$ after T iterations (with ctr depending on $\mathcal{D}^{i^*}_{id^*}$ and $\mathcal{D}^{j}_{id'}$), then the key pair generated by $\mathsf{IKG}_{\mathrm{RSA}}$ is $(0, 0)$, i.e., invalid RSA keys, although the key pair generated by $\mathsf{KeyGen}_{\mathrm{RSA}}$ is valid. Moreover, since $s = \mathcal{H}(0 \| r_{\mathcal{U}}) \oplus \mathcal{H}(r_{\mathcal{CA}})$, it is $2^{-\kappa} q_{\mathcal{H}}$-statistically indistinguishable from a uniformly random value if $H_\infty\left(\mathcal{D}^{i^*}_{id^*}\right) \geq \kappa$ or $H_\infty\left(\mathcal{D}^{j}_{id'}\right) \geq \kappa$, for $\mathcal{D}^{i^*}_{id^*}$ and $\mathcal{D}^{j}_{id'}$ are independent of $\mathcal{H}$.

For a uniformly random seed s, the security of PRF implies that its $j+3$ evaluations on s are $\varepsilon_{\mathsf{PRF}}$-computationally indistinguishable from uniformly random values.

Lastly, if $p = q$, then the key generated is not a valid RSA key, but it only occurs with probability at most $|R_{\mathsf{PRF}}|^{-1}$.

Therefore, $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}} + \varepsilon_{\mathsf{PRF}} + |R_{\mathsf{PRF}}|^{-1} + \Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$.

Game 2. If $id^* \in U$, the challenger answers a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ as follows. It then computes (pk, π) honestly, but instead of setting $pk_{id^*}^{i^*} \leftarrow pk$, it generates a uniformly random string σ, runs $(pk', sk') \leftarrow \mathsf{KeyGen}_{\mathrm{RSA}}(b, e, W; \sigma)$ and sets $pk_{id^*}^{i^*} \leftarrow pk$.

Recall that in case 1), since $r_{\mathcal{CA}}$ is always oracle-generated. Therefore, the same indistinguishability between the last two games still apply and $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}} + \varepsilon_{\mathsf{PRF}} + |R_{\mathsf{PRF}}|^{-1} + \Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$.

Game 3. If $id^* \in CA$, the challenger answers a Send query on $(id^*, i^*, (N, e, \pi, \pi_W, i, (a_\gamma)_{\gamma \neq i,j}))$ as follows. If $(N, e, \pi, \pi_W, i, (a_\gamma)_{\gamma \neq i,j})$ is adversarially generated, for all $\gamma \neq i, j$, $\mathsf{PrimeTest}_W(b, e, a_\gamma; r'_W) = 0$, and proofs π and π_W are valid, then the challenger runs $\Pi.(\mathsf{Ext}_0, \mathsf{Ext}_1)$ on $\mathcal{A}'_1$. If extraction fails, the challenger aborts; otherwise it obtains a tuple $(r'_{\mathcal{U}}, d, a_i, a_j)$, and checks whether $(C, r_{\mathcal{CA}}, N, (a_\gamma)_{\gamma \neq i,j}; r'_{\mathcal{U}}, d, a_i, a_j)$ is in $\mathcal{R}_\Pi$. If not, it aborts. If so, $N = a_i a_j$ in $\mathbb{N}$, and the challenger checks whether $(N, e; a_i, a_j)$ of $(N, e; a_j, a_i)$ is in $\mathcal{R}_{\Pi_W}$. If not, it aborts. Indeed, Since $W \subseteq \mathbb{P}$, if a_i and a_j are not prime, then the soundness of Π_W is contradicted. If a_i and a_j are prime, then unless the soundness of Π_W is contradicted, the only possible witness for (N, e) is (a_i, a_j) or (a_j, a_i) by the fundamental theorem of arithmetic. Therefore, if neither $(N, e; a_i, a_j)$ nor $(N, e; a_j, a_i)$ is in $\mathcal{R}_{\Pi_W}$, the soundness of $\Pi_{\mathcal{W}}$ is contradicted.

This game can be distinguished from the one only if the extractability of Π or the soundness of Π_W is contradicted. Therefore, $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the one with an advantage of at most $\varepsilon_{\Pi}^{\mathrm{ext}} + \varepsilon_{\Pi_W}^{\mathrm{sound}}$.

Game 4. If $id^* \in CA$, the challenger answers a Send query on $(id^*, i^*, (N, e, \pi, \pi_W, i, (a_\gamma)_{\gamma \neq i,j}))$ the challenger proceeds as the one of the previous game, but if $(C, r_{\mathcal{CA}}, N, (a_\gamma)_{\gamma \neq i,j}; r'_{\mathcal{U}}, d, a_i, a_j)$ is in the relation, it runs $\Pi_{\mathcal{C}}.(\mathsf{Ext}_0, \mathsf{Ext}_1)$. If extraction fails, the challenger aborts; otherwise it obtains a tuple $(r''_{\mathcal{U}}, d')$. If $r'_{\mathcal{U}} \neq r''_{\mathcal{U}}$, the challenger aborts.

This game can be distinguished from the previous one only if the extractability of $\Pi_{\mathcal{C}}$ or the binding property of $\mathscr{C}$ is contradicted. It follows that adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous with an advantage of at most $\varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{ext}} + \varepsilon_{\mathscr{C}}^{\mathrm{bind}}$.

Game 5. If $id^* \in CA$, to answer a Send query on $(id^*, i^*, (N, e, \pi, \pi_W, i, (a_\gamma)_{\gamma \neq i,j}))$, the challenger proceeds as the one of the previous game,

and if it does not aborts, it generates a uniformly random string σ, runs $(pk', sk') \leftarrow \mathsf{KeyGen}_{\mathrm{RSA}}(b, e, W; \sigma)$ and sets $pk_{id^*}^{i^*} \leftarrow pk$.

Note that if $\mathcal{C}$ is adversarially generated, then $H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right) \geq \kappa$ as the distribution of the partner instance is set to a Dirac mass since (id^*, i^*) has not yet accepted, by definition of oracle Test. If $\mathcal{C}$ is oracle-generated, then $\max\left(H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right), H_\infty\left(\mathcal{D}_{id'}^{j}\right)\right) \geq \kappa$. The same arguments for the computational indistinguishability of Game 2 and Game 1 imply that $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}} + \varepsilon_{\mathsf{PRF}} + |R_{\mathsf{PRF}}|^{-1} + \Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$.

In the last game, the condition that if an instance accepts then its partner must eventually terminate implies that $pk_{id^*}^{i^*}$ is computed by generating a uniformly random string σ and running $\mathsf{KeyGen}_{\mathrm{RSA}}$ on input $(b, e, W; \sigma)$. The advantage of the adversary in the last game is thus nil. IT follows that in case 1), the advantage of $(\mathcal{A}_1, \mathcal{A}_2)$ is at most

$$|ID|\,|I|\left(3\left(2^{-\kappa} q_{\mathcal{H}} + \varepsilon_{\mathsf{PRF}} + |R_{\mathsf{PRF}}|^{-1} + \Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]\right) + \varepsilon_{\Pi}^{\mathrm{ext}} + \varepsilon_{\Pi_W}^{\mathrm{sound}} + \varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{ext}} + \varepsilon_{\mathscr{C}}^{\mathrm{bind}}\right).$$

In the second case, the situation is similar to the second case of the proof of Theorem 10.3.3, except that the arguments are in the random-oracle model, and that Π_W is only assumed to be sound, not extractable. Consider then the following sequence of games.

Game 0. This is the real selective game.

Game 1. This game is defined as in the previous case, and $(\mathcal{A}_1', \mathcal{A}_2')$ can distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}} + \varepsilon_{\mathsf{PRF}} + |R_{\mathsf{PRF}}|^{-1} + \Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$.

Game 2. To answer a prompting Send query on $(id^*, i^*, (*, *))$, the challenger now generates $r_{\mathcal{U}}'$, ρ_U and s' uniformly at random. Since $r_{\mathcal{CA}}$ is adversarially generated in case 2) and that $\mathcal{U}$ accepts only after receiving it, the only information $(\mathcal{A}_1', \mathcal{A}_2')$ has about $r_{\mathcal{U}}$ is that it has a distribution $\mathcal{D}_{id^*}^{i^*}$ such that $H_\infty\left(\mathcal{D}_{id^*}^{i^*}\right) \geq \kappa$. In the event in which the Test query of $\mathcal{A}_2'$ is not replied to with $\perp$, instance (id^*, i^*) is not corrupt before it accepts, so the only information $\mathcal{A}_1'$ has about $r_{id^*}^{i^*}$ is that its distribution is $\mathcal{D}_{id^*}^{i^*}$. Moreover, if id^* is later corrupt before the end of the protocol execution, (id^*, i^*) will have already erased $r_{\mathcal{U}}'$ and $\rho_{\mathcal{U}}$ and s'.

Consequently, $(\mathcal{A}_1', \mathcal{A}_2')$ can thus distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}}$.

Game 3. In this game, the challenger answers a Send query on $(id^*, i^*, r_{\mathcal{CA}})$, with $r_{\mathcal{CA}}$ adversarially generated, if $ctr < 2$, the challenger aborts. $(\mathcal{A}'_1, \mathcal{A}'_2)$ can thus distinguish this game from the previous one with an advantage of at most $\Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$.

Game 4. In this game, the challenger answers a Send query on $(id^*, i^*, r_{\mathcal{CA}})$, with $r_{\mathcal{CA}}$ adversarially generated, by generating uniformly random values instead of evaluating PRF' at $(s', 0)$. If id^* is later corrupt before the end of the protocol execution, (id^*, i^*) will have already erased s' and r_Π. Adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathsf{PRF}'}$.

Game 5. Denoting by $Q_{\mathcal{H}_\Pi}$ the of queries to $\mathcal{H}$ and their responses, and the challenger now runs $\Pi.\mathsf{Sim}(0, Q_{\mathcal{H}_\Pi}, \cdot)$ to answer random-oracle queries. To answer a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ such that $r_{\mathcal{CA}}$ is adversarially generated, the challenger simulates a proof $\pi \leftarrow \Pi.\mathsf{Sim}\,(1, Q_{\mathcal{H}_\Pi}, (C, r_{\mathcal{CA}}, N, (a_\gamma)_{\gamma \neq i,j}; r'_{\mathcal{U}}, d, a_i, a_j))$. Likewise, the challenger simulates a proof $\pi_{\mathcal{W}}$.

In the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, adversary $\mathcal{A}'_1$ does not corrupt id^* before (id^*, i^*) accepts, so not before r_Π and r_{Π_W} are erased. By the zero-knowledge property of Π and $\Pi_{\mathcal{W}}$, adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{\mathrm{zk}}_\Pi + \varepsilon^{\mathrm{zk}}_{\Pi_W}$.

Game 6. To answer a prompting Send query on $(id^*, i^*, (*, *))$, the challenger simulates a proof $\Pi_{\mathcal{C}}$. In the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, adversary $\mathcal{A}'_1$ does not corrupt id^* before (id^*, i^*) accepts, so not before $r_{\Pi_{\mathcal{C}}}$ is erased.

By the zero-knowledge property of $\Pi_{\mathcal{C}}$ adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{\mathrm{zk}}_{\Pi_C}$.

Game 7. To answer a prompting Send query on $(id^*, i^*, (*, *))$, the challenger runs $(\mathcal{C}, d) \leftarrow \mathsf{Com}\left(0^{|r'_{\mathcal{U}}|}; \rho_{\mathcal{U}}\right)$ and sends $\mathcal{C}$. In the event in which the Test query of $\mathcal{A}'_2$ is not replied to with $\perp$, adversary $\mathcal{A}'_1$ does not corrupt id^* before (id^*, i^*) accepts, so not before $\rho_{\mathcal{U}}$ is erased. As $\mathscr{C}$ is $\varepsilon^{\mathrm{hide}}_{\mathscr{C}}$-hiding, $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{\mathrm{hide}}_{\mathscr{C}}$.

Game 8. To answer a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ such that $r_{\mathcal{CA}}$ is adversarially generated, the challenger generates s uniformly at random. As $H_\infty\left(\mathcal{D}^{i^*}_{id^*}\right) \geq \kappa$, seed s is $2^{-\kappa} q_{\mathcal{H}}$-statistically indistinguishable from a uniformly random value. $(\mathcal{A}'_1, \mathcal{A}'_2)$ can then distinguish this game from the previous one with an advantage of at most $2^{-\kappa} q_{\mathcal{H}}$.

Game 9. To answer a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ such that $r_{\mathcal{CA}}$ is adversarially generated, the challenger generates uniformly random numbers instead of evaluating PRF on s. Algorithm $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $\varepsilon_{\mathsf{PRF}}$.

Game 10. To answer a Send query on $(id^*, i^*, r_{\mathcal{CA}})$ such that $r_{\mathcal{CA}}$ is adversarially generated, the challenger aborts if $p = q$. Algorithm $(\mathcal{A}'_1, \mathcal{A}'_2)$ can distinguish this game from the previous one with an advantage of at most $|R_{\mathsf{PRF}}|^{-1}$.

In the last game, the condition that if an instance accepts then its partner must eventually terminate then implies that $pk_{id^*}^{i^*}$ is computed by running $\mathsf{KeyGen}_{\mathrm{RSA}}(b, e, W; \sigma)$, where σ is a uniformly random string. The advantage of $(\mathcal{A}'_1, \mathcal{A}'_2)$ in that game is then nil. As a result, the advantage of $(\mathcal{A}_1, \mathcal{A}_2)$ in the second case is at most

$$|ID|\,|I|\left(3q_{\mathcal{H}}2^{-\kappa} + 2\varepsilon_{\mathsf{PRF}} + \varepsilon_{\mathsf{PRF}'} + \varepsilon_{\Pi}^{zk} + \varepsilon_{\Pi_W}^{\mathrm{zk}} + \varepsilon_{\Pi_{\mathcal{C}}}^{\mathrm{zk}} + \varepsilon_{\mathscr{C}}^{\mathrm{hide}} + 2\,|R_{\mathsf{PRF}}|^{-1} + 2\Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]\right).$$

□

Theorem 10.3.7 (Running Time of $\mathsf{IKG}_{\mathrm{RSA}}$)**.** *Let $n(b, W, e, R_{\mathsf{PRF}})$ denote the number of primes p in $[\![2^{b-1}, 2^b - 1]\!] \cap W \cap R_{\mathsf{PRF}}$ such that $\gcd(e, p-1) = 1$, and let β denote $n(b, W, e, R_{\mathsf{PRF}})\,|R_{\mathsf{PRF}}|^{-1}$. If $n(b, W, e, R_{\mathsf{PRF}}) \geq 1$, then for numbers chosen uniformly at random in R_{PRF}, the number of trials necessary to obtain two primes that satisfy the above conditions is $2/\beta$ in expectation. Moreover, for any $\delta > 0$, setting $J \leftarrow \lceil 2(1+\delta)/\beta \rceil$, if PRF is $(T_{\mathsf{PRF}}, J, \varepsilon_{\mathsf{PRF}})$-secure and $\mathcal{D}_{\mathcal{U}}$ and $\mathcal{D}_{\mathcal{CA}}$ are independent of the random oracle, have respective min-entropy at least $2^{-\kappa_{\mathcal{U}}}$ and $2^{-\kappa_{\mathcal{CA}}}$, and are returned by an algorithm that runs in time at most T_{PRF} and makes at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$, then, in Algorithm 10.3.6, $\Pr[j > J] \leq \exp\left(-(2\delta/\beta J)^2\,\beta J/2\right) + q_{\mathcal{H}}\left(2^{-\kappa_{\mathcal{U}}} + 2^{\kappa_{\mathcal{CA}}}\right) + \varepsilon_{\mathsf{PRF}}$.*

Proof. Let $(X_\mu)_{\mu \geq 1}$ be a family of independent random variables with the same binomial distribution of parameter β. In essence, X_μ indicates whether the μth trial, if it were done with a uniformly random function, would result in a prime number. For an integer $\nu \geq 1$, let $T_\nu := \min\{\mu \geq 1 \colon X_1 + \cdots + X_\mu = \nu\}$. It is a stopping time which indicates the number of trials before getting ν primes numbers. The variable $T_\nu - \nu$, which indicates the number of non-primes before ν primes are found, has a negative binomial distribution with parameters ν and $1-\beta$. Its expectation is then $\nu(1-\beta)/\beta$. Therefore, by $\mathbb{E}[T_\nu] = \nu/\beta$. In the case of RSA keys, $\nu = 2$.

Moreover, the Chernoff bound (Section 10.1.4) implies that the probability that the number of trials is higher that this expectation decreases exponentially fast. Formally, consider a real number $\delta > 0$, and set $J \leftarrow$

$\lceil 2(1+\delta)/\beta \rceil$. Note that $T_2 > J$ if and only if $X_1 + \cdots + X_J < 2$, which is equivalent to $X_1 + \cdots + X_J - J\beta < 2 - J\beta \leq -2\delta$. Note also that $0 < 2\delta/\beta J \leq 1$ for all $\delta > 0$. The Chernoff bound implies that $\Pr(T_2 > J) \leq P(X_1 + \cdots + X_J - J\beta < -2\delta) \leq \exp(-(2\delta/\beta J)^2 \beta J/2)$.

In Algorithm 10.3.6, integers are not generated uniformly at random, but rather with PRF. However, for any $\delta > 0$, setting $J \leftarrow \lceil 2(1+\delta)/\beta \rceil$, if PRF is $(T_{\mathsf{PRF}}, J, \varepsilon_{\mathsf{PRF}})$-secure and $\mathcal{D}_{\mathcal{U}}$ and $\mathcal{D}_{\mathcal{CA}}$ are independent of the random oracle, have respective min-entropy at least $2^{-\kappa_{\mathcal{U}}}$ and $2^{-\kappa_{\mathcal{CA}}}$, and are returned by an algorithm that runs in time at most T_{PRF} and makes at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$, then, the inequality of the theorem statement holds.

To show it, first notice that an execution of protocol $\mathsf{IKG}_{\mathrm{RSA}}$ is $q_{\mathcal{H}}(2^{\kappa_{\mathcal{U}}} + 2^{\kappa_{\mathcal{CA}}})$-statistically indistinguishable from one in which s is generated uniformly at random. Consider now an adversary for the PRF game with the DY PRF which makes oracle queries until it either obtains two primes (possibly at the Jth query) or reaches J query with at most one prime returned by the oracle. The adversary returns 1 in the first case and 0 in the second. The number of trials with PRF is then at most the number of trials with a uniformly random function plus the advantage of this adversary in the PRF game, and the theorem follows. □

Corollary 10.3.8. *In Algorithm 10.3.6, if $T > 2/\beta$, setting $\delta \leftarrow \beta T/2 - 1$ and $J \leftarrow \lceil 2(1+\delta)/\beta \rceil$,* $\Pr[ctr < 2] \leq \exp\left(-(2\delta/\beta J)^2 \beta J/2\right) + q_{\mathcal{H}}(2^{-\kappa_{\mathcal{U}}} + 2^{\kappa_{\mathcal{CA}}}) + \varepsilon_{\mathsf{PRF}}$.

Theorem 10.3.9 (Running Time of $\mathsf{KeyGen}_{\mathrm{RSA}}$)**.** *Let $n(b, W, e)$ denote the number of primes p in $[\![2^{b-1}, 2^b - 1]\!] \cap W$ such that $\gcd(e, p-1) = 1$, and let β_{RSA} denote $n(b, W, e)2^{-b+1}$. If $n(b, W, e) \geq 1$, then the expected running time of $\mathsf{KeyGen}_{\mathrm{RSA}}$ is $2/\beta_{\mathrm{RSA}}$. Moreover, for $T_{\mathrm{RSA}} > 2/\beta_{\mathrm{RSA}}$, setting $\delta_{\mathrm{RSA}} \leftarrow \beta_{\mathrm{RSA}} T_{\mathrm{RSA}}/2 - 1$ and $J_{\mathrm{RSA}} \leftarrow \lceil 2(1+\delta_{\mathrm{RSA}})/\beta_{\mathrm{RSA}} \rceil$,*

$$\Pr[ctr_{\mathrm{RSA}} = 2] \geq 1 - \exp\left(-\left(\frac{2\delta_{\mathrm{RSA}}}{\beta_{\mathrm{RSA}} J_{\mathrm{RSA}}}\right)^2 \frac{\beta_{\mathrm{RSA}} J_{\mathrm{RSA}}}{2}\right).$$

Proof. It follows by the exact same analysis as in the first part of the proof of Theorem 10.3.7. □

As ctr and ctr_{RSA} are independent random variables (the randomness of by $\mathsf{KeyGen}_{\mathrm{RSA}}$ in query Test is independent of the distributions given by $\mathcal{A}_1'$), $\Pr[ctr < 2, ctr_{\mathrm{RSA}} = 2]$ is upper-bounded by the product of the upper-bounds of Corollary 10.3.8 and Theorem 10.3.9, and this upper bound mainly (but not only) depends on T, T_{RSA} and R_{PRF}.

10.4 Instantiation of the RSA-Key Generation Protocol

This section instantiates the protocol of Section 10.3.4 for RSA key-generation with verifiable randomness. To do so, it provides efficient instantiations for each of the building blocks.

Recently, several important advancements have been made on the efficiency of the commit-and-prove paradigm on committed values which combine algebraic and non-algebraic statements [CGM16, BBB+18, BHH+19]. These improvements for *cross-domains* statements allow to prove efficiently for instance that some committed value corresponds to a pre-image of some value of a given hash function such as SHA-256 or that some value is the output of some non-algebraic PRF (i.e. HMAC-SHA-256 or AES) using some committed key. To generate an RSA modulus of 3072 bits (for 128-bit security) using the generic protocol from Section 10.3.4, the PRF must return 1536-bit integers and the use of non-algebraic PRF with the technique from [CGM16, BBB+18, BHH+19] would result in prohibitive schemes.

On this account, the instantiation in this section is based on an algebraic PRF, namely the Dodis–Yampolskiy PRF, and uses techniques due to Bünz et al. [BBB+18] for range proofs and arithmetic-circuit satisfiability to obtain short proofs of correct computation (i.e., Π in Section 10.3.4). These techniques are akin to those presented in Chapter 9 for Diophantine satisfiability, though less intricate as they in groups with public prime orders.

In the process, it presents the first logarithmic-size (in the bit-length of the group order) argument of knowledge of double discrete logarithms, and argument of equality of a discrete logarithm in a group and a double discrete logarithm in another related group. The underlying ideas are similar to those given in Section 9.5.1 to derive a Hadamard product and linear constraints from a Diophantine equation. In contrast, the protocol of Camenisch and Stadler [CS97] for the first relation, and the protocol of Chase et al. [CGM16] for the second are linear in the security parameter.

Parameters. We consider two related group-family generators G_1 and G_2. Given a security parameter λ, to generate an RSA modulus which is the product of two $b(\lambda)$-bit prime numbers, let ℓ be the smallest prime of binary length equal to $b(\lambda)$ such that $2\ell+1$ is also a prime number (i.e., ℓ is a Sophie Germain prime number, or equivalently $2\ell+1$ is a $b(\lambda)+1$-bit *safe prime*). G_2 returns, on input λ, the group $\mathbb{G}_2$ of quadratic residues modulo $2\ell+1$ (which is of prime order ℓ). The group-family generator G_1 returns on input λ some group $\mathbb{G}_1$ of prime order Λ such that ℓ divides $\Lambda-1$ and $\Lambda > (2\ell+1)^2$. In practice[6], $\mathbb{G}_1$ can be taken as a prime order subgroup Λ of $\mathbb{Z}_r^*$ for some prime number r such that Λ divides $r-1$.

The restriction to quadratic residues is necessary for assumptions like

the DDH and the q-DDHI assumptions to hold over $\mathbb{G}_2$. However, it introduces a bias by design (not from the user algorithm) in the RSA keys generated: p and q are necessarily quadratic residues modulo $2\ell + 1$. The reason is that the values returned by the DY PRF are not actually integers but $\mathbb{G}_2$ elements. Nonetheless, it is already the case for 1/4 of all RSA moduli since the factors p and q returned by $\mathsf{keygen}_{\mathrm{RSA}}$.

Commitment Scheme. Scheme $\mathscr{C}$ is the Pedersen commitment scheme [Ped92] in $\mathbb{G}_2$ for the user to commit her randomness.

used to commit to the secret RSA primes p and q, and the same Pedersen scheme.

Pseudo-Random Functions. PRF is the Dodis–Yampolskiy (DY) PRF (see Section 10.1.3) in the group $\mathbb{G}_2 = QR_{2\ell+1}$ of quadratic residues modulo $2\ell + 1$. It is used to generate the secret RSA primes p and q. Since $2\ell + 1$ is $b(\lambda) + 1$ bits long, p and q are $b(\lambda)$ bits long with probability close to 1/2. The reason $2\ell + 1$ is chosen to be one bit larger than p and q is to ensure that *all* primes of $b(\lambda)$ bits can be returned by the PRF so as not to introduce a bias. As for PRF', it can be any efficient pseudo-random function, e.g., HMAC [BCK96].

Argument for R_W. The argument system Π_W depends on the properties that the prime factors of N must satisfy, e.g., they must be congruent to 3 modulo 4 or be safe primes. To prove that $p = q = 3 \mod 4$, one can prove that N is of the form $p^r q^s$ with $p = q = 3 \mod 4$ using the protocol of van de Graaf and Peralta [vP88], and run in parallel the protocol of Boyar et al. [BFL90] to prove that N is square-free. To prove that p and q are safe primes, there exist proof systems in the literature such as Camenisch and Michel's [CM99]. Besides, Goldberg et al. [GRSB19] recently built a protocol to prove that $\gcd(e, \phi(N)) = 1$.

Argument of Correct Computation. The last component is an extractable zero-knowledge argument system Π in the random-oracle model for the user algorithm to prove that it correctly performed its computation, i.e., an argument system for $\mathcal{R}_\Pi$. Section 10.4.1 presents a perfectly honest-verifier zero-knowledge interactive protocol for $\mathcal{R}_\Pi$ that also satisfies extractability.

[6] To generate RSA moduli which are products of two 1536-bit primes, the instantiation with the Dodis–Yampolskiy PRF uses $\ell = 2^{1535} + 554415$ which is a Sophie Germain prime, $\Lambda = (4\ell + 18)\ell + 1$ and $r = 1572 \cdot \Lambda + 1$.

10.4.1 Zero-Knowledge Argument with the Dodis–Yampolskiy PRF

This section gives a zero-knowledge argument Π in the case of the DY PRF in $\mathbb{G}_2 = QR_{2\ell+1}$. Formally, let $2\ell+1$ be a $b(\lambda)+1$-bit (i.e., $b(\lambda)+1 = \lfloor \log(2\ell+1) \rfloor + 1$) safe prime (i.e., ℓ is a Sophie Germain prime) and let Λ be a prime integer such that ℓ divides $\Lambda - 1$ and $\Lambda > (2\ell+1)^2$. Consider $\mathbb{G}_1 = \langle G_1 \rangle$ a group of prime order Λ (in which p and q will be committed) and $\mathbb{G}_2 = \langle G_2 \rangle = QR_{2\ell+1}$ the group of quadratic residues modulo $2\ell+1$, which is a cyclic group of order ℓ. Recall that the DY PRF is defined as the map $(K, x) \mapsto G_2^{1/(K+x)}$.

Proof Strategy.

To prove knowledge of a witness for the membership of $(C, r_{\mathcal{CA}}, N, (a_\gamma)_{\gamma \neq i,j})$ to the language relative to $\mathcal{R}_\Pi$, the user algorithm commits to $p = a_i$ and $q = a_j$ in $\mathbb{G}_1$ with the Pedersen commitment scheme and respective randomness r_p and r_q. The commitments are denoted P and Q.

The user algorithm then proves knowledge of a witness for $\mathcal{R}_0 \cap \mathcal{R}_1$, with

$$\begin{aligned}\mathcal{R}_0 \coloneqq \{&(\mathcal{C}, r_{\mathcal{CA}}, N, P, Q, (a_\gamma)_{\gamma \neq i,j}; r'_{\mathcal{U}}, \rho_u, a_i, a_j, r_p, r_q) : \\ &\mathsf{ComVf}(\mathcal{C}, r'_{\mathcal{U}}, \rho_u) = 1,\ s = r'_{\mathcal{U}} + h(r_{\mathcal{CA}}) \bmod \ell \\ &\forall \gamma \in [\![j]\!], a_\gamma = \mathsf{PRF}(s, \gamma), \mathsf{ComVf}(P, a_i, r_p) = \mathsf{ComVf}(Q, a_j, r_q) = 1\}\end{aligned}$$

and

$$\begin{aligned}\mathcal{R}_1 \coloneqq \{&(\mathcal{C}, r_{\mathcal{CA}}, N, P, Q, (a_\gamma)_{\gamma \neq i,j}; r'_{\mathcal{U}}, \rho_u, a_i, a_j, r_p, r_q) : \mathsf{ComVf}(P, a_i, r_p) = 1 \\ &\mathsf{ComVf}(Q, a_j, r_q) = 1, 2^{b(\lambda)-1} \leq a_i, a_j \leq 2^{b(\lambda)} - 1, N = a_i a_j \text{ in } \mathbb{N}\Big\}.\end{aligned}$$

To prove knowledge of a witness for relation $\mathcal{R}$, it then suffices to prove in parallel knowledge of a witness for $\mathcal{R}_0$ and of a witness for $\mathcal{R}_1$ on the same public inputs. Note that the binding property of the Pedersen commitment scheme in $\mathbb{G}_1$ (relying on the DLOG assumption) guarantees that the a_i and a_j values used in both proofs are the same (up to a relabeling).

Relation $\mathcal{R}_0$. We start by giving two preliminary protocols:

- a logarithmic-size zero-knowledge argument of knowledge of a double-discrete logarithm (Section 10.4.2) using Bulletproof techniques [BBB+18]. The resulting proofs are of size logarithmic in the bit-length of the group order. In comparison, the protocol of Camenisch and Stadler [CS97] has proofs of size linear in the security parameter
- a logarithmic-size argument of equality of a discrete logarithm in a group and a double discrete logarithm in another related group (Section 10.4.3). In contrast, the protocol of Chase et al. [CGM16, Section 4.3] for this relation uses the techniques of Camenisch and Stadler and therefore has proofs of size linear in the security parameter.

We then combine the latter proof with the proof in Section 10.4.4 to obtain a proof for relation $\mathcal{R}_0$.

Relation $\mathcal{R}_1$. The aggregated logarithmic range proof of Bünz et al. [BBB$^+$18, Section4.2] is sufficient to prove that the values committed in P and Q modulo Λ are in $[\![2^{b-1}, 2^b - 1]\!]$ (which is equivalent to proving that the values committed in $PG_1^{-2^{b-1}}$ and $QG_1^{-2^{b-1}}$ are in $\left\{0, \ldots, 2^{b-1} - 1\right\}$). With the hypotheses on the parameters Λ and ℓ, the verifier is convinced that the equation $N = a_i a_j$ holds in $\mathbb{N}$. Indeed, the equation $N = a_i a_j \mod \Lambda$ implies that there exists $m \in \mathbb{Z}$ such that $N = a_i a_j + m\Lambda$. Integer m cannot be strictly positive as otherwise N would be strictly greater than Λ. Besides, m cannot be strictly negative since $\Lambda > (2\ell + 1)^2 > a_i a_j$; it is therefore nil and the equation $N = a_i a_j$ holds in $\mathbb{N}$.

10.4.2 Logarithmic-Size Argument of Double Discrete Logarithm

This section gives a zero-knowledge argument with logarithmic communication size for proving knowledge of a double discrete logarithm. It uses as a sub-argument the logarithmic-size inner-product argument for arithmetic-circuit satisfiability of Bünz et al. [BBB$^+$18, Section 5.2]. This latter argument share similarities with the one presented in Section 9.5.2.

As mentioned in Section 9.5.1, following the ideas of Bootle et al. [BCC$^+$16], Bünz et al. convert any arithmetic circuit with n multiplications gates into a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$ and $Q \leq 2n$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for $q \in [\![Q]\!]$, with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q} \in \mathbb{Z}_p^n$ and $c_q \in \mathbb{Z}_p$. The vectors $\mathbf{a}_L$, $\mathbf{a}_R$ respectively denote the vectors of left and right inputs to the multiplications gates, and $\mathbf{a}_O$ the vector of outputs. The linear constraints ensure the consistency between the outputs and the inputs of two consecutive depth levels of the circuit.

Bünz et al. actually give an argument for a more general relation which includes Pedersen commitments of which the openings are included in the linear consistency constraints. Concretely, given a group $\mathbb{G}$ of prime order p and positive integers n, m and Q, Bünz et al. give a zero-knowledge argument for the relation

$$\Big\{\Big(g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^n, \mathbf{V} \in \mathbb{G}^m, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}, \mathbf{W}_V \in \mathbb{Z}_p^{Q \times m}, \mathbf{c} \in \mathbb{Z}_p^Q;$$
$$\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n, \mathbf{v}, \gamma \in \mathbb{Z}_p^m\Big) : V_j = g^{v_j} h^{\gamma_j} \forall j \in [\![m]\!]$$
$$\wedge \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}}\Big\}.$$

The soundness of their argument relies on the discrete-logarithm assumption over the generator of $\mathbb{G}$. Concerning the proof size, the prover sends $2\lceil \log_2 n \rceil + 8$ group elements and 5 $\mathbb{Z}_p$ elements. Note that this setting is similar to the one in Section 9.5.2, but over $\mathbb{Z}_p$ instead of $\mathbb{Z}$.

The main difficulty in the case of a proof of a double discrete logarithm relation is to re-write the problem in a way that is suitable to apply the proof for arithmetic circuits. The goal is to give a zero-knowledge argument for:

$$\mathcal{R}_{\text{2DLOG}} \coloneqq \left\{ (G_1, H_1, G_2, Y; x \in \mathbb{Z}_\ell, r \in \mathbb{Z}_\Lambda) \colon Y = G_1^{G_2^x} H_1^r \right\}.$$

First, let $n(\lambda)+1 \coloneqq b(\lambda)$ be the bit-length of ℓ. Given the bit representation $(x_i)_{i=0}^n$ of x, $G_2^x = G_2^{\sum_{i=0}^n x_i 2^i} = \prod_i \left(G_2^{2^i}\right)^{x_i}$. An important observation is that for $x_i \in \{0,1\}$, $\left(G_2^{2^i}\right)^{x_i} = x_i G_2^{2^i} + (1 - x_i) = x_i \left(G_2^{2^i} - 1\right) + 1$. The addition here is over $\mathbb{Z}_\Lambda$, although the notation is purely formal since $x_i \in \{0,1\}$. It thus follows that an argument for $\mathcal{R}_{\text{2DLOG}}$ is equivalent to an argument for:

$$\left\{ (G_1, H_1, G_2, Y; (x_i)_{i=0}^n \in \{0,1\}^n, r \in \mathbb{Z}_\Lambda) \colon Y = G_1^{\prod_i \left(x_i \left(G_2^{2^i} - 1\right) + 1\right)} H_1^r \right\},$$

which is also equivalent to an argument for:

$$\left\{ (G_1, H_1, G_2, Y; (a_i)_{i=0}^n, r \in \mathbb{Z}_\Lambda) \colon Y = G_1^{\prod_i a_i} H_1^r \wedge a_i \in \left\{1, G_2^{2^i}\right\} \right\}.$$

To this end, consider the following array

$$\begin{array}{ccccc} a_0 & a_1 & a_2 & \cdots & a_n \\ 1 & a_0 & a_0 a_1 & \cdots & a_0 a_1 \cdots a_{n-1} \\ a_0 & a_0 a_1 & a_0 a_1 a_2 & \cdots & a_0 \cdots a_n. \end{array}$$

Notice that its third row is the product of the first two. In other words, if $\mathbf{a} \leftarrow (a_0, a_1, \ldots, a_n) \in \mathbb{Z}_\Lambda^{n+1}$ and $\mathbf{b} \leftarrow (b_0 = a_0, b_1 = a_0 a_1, \ldots, b_{n-1} = a_0 a_1 \cdots a_{n-1}) \in \mathbb{Z}_\Lambda^n$, then $\mathbf{a} \circ (1 \quad \mathbf{b}) = (\mathbf{b} \quad y)$ for $y \coloneqq G_2^x$.

Moreover, for $\mathbf{a}_L \coloneqq \left[\mathbf{a} \quad \mathbf{a} - \mathbf{1}^{n+1}\right]^{\mathrm{T}}$, $\mathbf{a}_R \coloneqq \left[1 \quad \mathbf{b} \quad \mathbf{a} - \mathbf{G_2^{2^{n+1}}}\right]^{\mathrm{T}}$ and $\mathbf{a}_O \coloneqq \left[\mathbf{b} \quad y \quad \mathbf{0}^{n+1}\right]^{\mathrm{T}} \in \mathbb{Z}_\Lambda^{2(n+1)}$, where $\mathbf{G_2^{2^{n+1}}}$ denotes the vector

$$\left(G_2, G_2^2, G_2^{2^2}, \ldots, G_2^{2^n}\right),$$

one has $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$. If one can prove knowledge of scalars $y, r \in \mathbb{Z}_\Lambda$ and of vectors $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ such that $Y = G_1^y H_1^r$ and $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, and such that the vectors are of the form above, then one can prove knowledge of $(a_i)_{i=0}^n \in \prod_i \left\{1, G_2^{2^i}\right\}$ and $(b_i)_{i=0}^{n-1}$ such that $y = a_n b_{n-1} = a_n a_{n-1} b_{n-2} =$

$\cdots = a_n a_{n-1} \cdots a_1 b_0 = a_n \cdots a_0$ and $Y = G_1^y H_1^r$. That is to say, one can prove knowledge of a double discrete logarithm.

To prove such a relation, one can use the argument of Bünz et al. [BBB+18] for arithmetic circuits with the right linear constraints to ensure that the vectors are of the appropriate form. To express these constraints, consider matrices

$$\mathbf{W}_L \coloneqq \begin{bmatrix} \mathbf{0}_{(n+2)\times 2(n+1)} \\ \mathbf{I}_{n+1} \quad -\mathbf{I}_{n+1} \\ \mathbf{I}_{n+1} \quad \mathbf{0}_{(n+1)\times(n+1)} \\ \mathbf{0}_{(n+1)\times 2(n+1)} \end{bmatrix},$$

$$\mathbf{W}_R \coloneqq \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & \\ 0 & 0 & 1 & 0 & \cdots & 0 & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \\ 0 & 0 & \cdots & 0 & 1 & 0 & \mathbf{0}_{(n+2)\times(n+1)} \\ 0 & 0 & \cdots & 0 & 0 & 1 & \\ 0 & 0 & \cdots & 0 & 0 & 0 & \\ 1 & 0 & \cdots & 0 & 0 & 0 & \\ & & & \mathbf{0}_{(n+1)\times 2(n+1)} & & & \\ & & & \mathbf{0}_{(n+1)\times(n+1)} & \mathbf{-I}_{n+1} & & \\ & & & \mathbf{0}_{(n+1)\times 2(n+1)} & & & \end{bmatrix},$$

$$\mathbf{W}_O \coloneqq \begin{bmatrix} & & & & 0 & \\ & & -\mathbf{I}_n & & \vdots & \\ & & & & 0 & \mathbf{0}_{(n+2)\times(n+1)} \\ 0 & \cdots & 0 & & 1 & \\ 0 & \cdots & 0 & & 0 & \\ & & & \mathbf{0}_{2(n+1)\times 2(n+1)} & & \\ & & & \mathbf{0}_{(n+1)\times(n+1)} & \mathbf{I}_{n+1} & \end{bmatrix},$$

$$\mathbf{W}_V \coloneqq \begin{bmatrix} \mathbf{0}_{n\times 2} \\ 0 \quad 1 \\ 1 \quad 0 \\ \mathbf{0}_{3(n+1)\times 2} \end{bmatrix},$$

and vectors $\mathbf{v} \coloneqq \begin{bmatrix} 1 \\ y \end{bmatrix}$, $\mathbf{c}^T \coloneqq \begin{bmatrix} \mathbf{0}_{1\times(n+2)} & \mathbf{1}^{n+1} & \mathbf{G_2^{2^{n+1}}} & \mathbf{0}_{1\times(n+1)} \end{bmatrix}$.

Three vectors $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O \in \mathbb{Z}_\Lambda^{2(n+1)}$ satisfy the equation $\mathbf{W}_L\mathbf{a}_L + \mathbf{W}_R\mathbf{a}_R + \mathbf{W}_O\mathbf{a}_O = \mathbf{W}_V\mathbf{v} + \mathbf{c}$ if and only if there exists $\mathbf{a} \in \mathbb{Z}_\Lambda^{n+1}$ and $\mathbf{b} \in \mathbb{Z}_\Lambda^n$ such that $\mathbf{a}_L^T \coloneqq \begin{bmatrix} \mathbf{a} & \mathbf{a} - \mathbf{1}^{n+1} \end{bmatrix}$, $\mathbf{a}_R^T \coloneqq \begin{bmatrix} 1 & \mathbf{b} & \mathbf{a} - \mathbf{G_2^{2^{n+1}}} \end{bmatrix}$ and $\mathbf{a}_O^T \coloneqq \begin{bmatrix} \mathbf{b} & y & \mathbf{0}^{n+1} \end{bmatrix} \in \mathbb{Z}_\Lambda^{2(n+1)}$.

Indeed,

* the first n rows of the equation guarantee that for $i = 0, \ldots, n-1$, $\mathbf{a}_{O,i} = \mathbf{a}_{R,i+1}$,

* the $n+1$th line ensures that $\mathbf{a}_{O,n+1} = y$,
* the $n+2$th line imposes $\mathbf{a}_{R,1} = 1$ (G_1 is here used a commitment to 1),
* the next $n+1$ lines are satisfied if and only if $\mathbf{a}_{L,[:n+1]} = \mathbf{a}_{L,[n+1:]} + \mathbf{1}^{n+1}$,
* the next $n+1$ lines guarantee that $\mathbf{a}_{R,[:n+1]} = \mathbf{a}_{L,[n+1:]} + \mathbf{G_2^{2^{n+1}}}$,
* the last $n+1$ lines ensure that $\mathbf{a}_{O,[2(n+1):]} = \mathbf{0}^{n+1}$.

If vectors $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ additionally satisfy $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, then $a_i = \mathbf{a}_{L,i} \in \left\{1, G_2^{2^i}\right\}$ for $i = 0, \ldots, n$ and $\mathbf{a} \circ (1 \quad \mathbf{b}) = (\mathbf{b} \quad y)$.

Note that the procedure to derive such vectors $\mathbf{a}_L, \mathbf{a}_R$ and $\mathbf{a}_O$ and the linear constraints is the same as the one in Section 9.5.1.

The argument of Bünz et al. is therefore sufficient to prove in zero-knowledge knowledge of a double discrete logarithm. The soundness of the proof relies on the discrete-logarithm assumption over $\mathbb{G}_1$.

Regarding the proof size, the prover sends $(2\lceil \log_2 2(n+1) \rceil + 8)$ $\mathbb{G}_1$ elements and 5 $\mathbb{Z}_\Lambda$ elements. Notice that the argument of Bünz et al. requires $4(n+1)$ elements of $\mathbb{G}_1^*$ in addition to G_1 and H_1. To guarantee its soundness, no discrete logarithm relation between these elements, G_1 and H_1 must be known to the prover. They can then be choosen uniformly at random during set-up.

10.4.3 Logarithmic-Size Argument of Discrete-Logarithm Equality in two Groups

Building on the argument of double discrete logarithm of Section 10.4.2, this section gives a zero-knowledge argument for the relation

$$\mathcal{R}_{\text{DLOG}-2} \coloneqq \Big\{(G_1, H_1, G_2, H_2, Y, X; x \in \mathbb{Z}_\ell, r_1, r_2 \in \mathbb{Z}_\Lambda) \colon Y = G_1^{G_2^x} H_1^{r_1}, \\ X = G_2^x H_2^{r_2}\Big\}.$$

As in Section 10.4.2, write G_2^x as $\prod_i \overbrace{\left(x_i(G_2^{2^i} - 1) + 1\right)}^{a_i}$ for $x_i \in \{0,1\}$, and $H_2^{r_2}$ as $\prod_i \overbrace{\left(r_{2,i}(H_2^{2^i} - 1) + 1\right)}^{c_i}$ for $r_{2,i} \in \{0,1\}$. Note that $G_1^X = G_1^{\prod_i a_i \prod_i c_i}$.

An argument for $\mathcal{R}$ is then equivalent to an argument for

$$\Big\{(G_1, H_1, G_2, H_2, Y, X; (a_i)_{i=0}^n, (c_i)_{i=0}^n, r \in \mathbb{Z}_\Lambda) \colon Y = G_1^{\prod_i a_i} H_1^r, \\ G_1^X = G_1^{\prod_i a_i \prod c_i} \wedge a_i, c_i \in \left\{1, G_2^{2^i}\right\}\Big\}.$$

To give an argument for this latter relation, consider the following array (written over several lines)

$$\begin{array}{cccc} a_0 & a_1 & \cdots & a_n & c_0 \\ 1 & b_0 = a_0 & \cdots & b_{n-1} = a_0 \cdots a_{n-1} & b_{n-1}a_n \\ a_0 & b_1 = a_0 a_1 & \cdots & G_2^x = a_0 \cdots a_n & a_0 \cdots a_n c_0 \end{array}$$

$$\begin{array}{cccc} \cdots & c_1 & \cdots & c_n \\ \cdots & d_0 = a_0 \cdots a_n c_0 & \cdots & d_{n-1} = a_0 \cdots a_n c_0 \cdots c_{n-1} \\ \cdots & a_0 \cdots a_n c_0 c_1 & \cdots & X = \prod_i a_i \prod_i c_i. \end{array}$$

Its third row is the product of the first two. It follows that for

$$\begin{aligned} \mathbf{a}_L^T &\coloneqq \left[\mathbf{a} \quad \mathbf{a} - \mathbf{1}^{n+1} \quad \mathbf{c} \quad \mathbf{c} - \mathbf{1}^{n+1}\right], \\ \mathbf{a}_R^T &\coloneqq \left[1 \quad \mathbf{b} \quad \mathbf{a} - \mathbf{G_2^{2^{n+1}}} \quad \prod_i a_i \quad \mathbf{d} \quad \mathbf{c} - \mathbf{H_2^{2^{n+1}}}\right], \\ \mathbf{a}_O^T &\coloneqq \left[\mathbf{b} \quad \prod_i a_i \quad \mathbf{0}^{n+1} \quad \mathbf{d} \quad X \quad \mathbf{0}^{n+1}\right] \in \mathbb{Z}_\Lambda^{4(n+1)}, \end{aligned}$$

the equality $\mathbf{a}_L \circ a_R = a_O$ holds.

As in Section 10.4.2, it suffices to find linear constraints, i.e., matrices $\mathbf{M}_L$, $\mathbf{M}_R$, $\mathbf{M}_O$ and $\mathbf{M}_V$, and a vector $\mathbf{c}$, to enforce that three vectors $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ that satisfy the equation $\mathbf{M}_L\mathbf{a}_L + \mathbf{M}_R\mathbf{a}_R + \mathbf{M}_O\mathbf{a}_O = \mathbf{M}_V\mathbf{v} + \mathbf{c}$, where $\mathbf{v} = \begin{bmatrix} 1 \\ X \end{bmatrix}$ (G_1 and G_1^X are respectively used as commitments to 1 and X), are of the form above.

To express such constraints, consider the matrices $\mathbf{W}_L$, $\mathbf{W}_R$ and $\mathbf{W}_O$ of Section 10.4.2, and let $\mathbf{W}'_L$, $\mathbf{W}'_R$ and $\mathbf{W}'_O$ be their respective sub-matrices obtained by removing the $n+1$th line (the one that enforced that $\mathbf{a}_{O,n+1} = y$ therein) and $\mathbf{W}''_L$, $\mathbf{W}''_R$ and $\mathbf{W}''_O$ their sub-matrices obtained by removing the $n+2$th line (the one which imposed that $\mathbf{a}_{R,1} = 1$). Define also $\mathbf{W}'_V$ as the sub-matrix of $\mathbf{W}_V$ obtained by removing its $n+1$th line and its second column and $\mathbf{W}''_V$ as the sub-matrix of $\mathbf{W}'_V$ obtained by removing its $n+2$th line and its first column.

Consider then the following matrices

$$\begin{aligned} \mathbf{M}_L &\coloneqq \begin{bmatrix} \mathbf{W}'_L & \\ & \mathbf{W}''_L \end{bmatrix}, \ \mathbf{M}_R \coloneqq \begin{bmatrix} \mathbf{W}'_R & \\ & \mathbf{W}''_R \end{bmatrix} \\ \mathbf{M}_O &\coloneqq \begin{bmatrix} \mathbf{W}'_O & \\ & \mathbf{W}''_O \end{bmatrix}, \ \mathbf{M}_V \coloneqq \begin{bmatrix} \mathbf{W}'_V & \\ & \mathbf{W}''_V \end{bmatrix}. \end{aligned}$$

By definitions of $\mathbf{M}_L$, $\mathbf{M}_R$, $\mathbf{M}_O$, $\mathbf{M}_V$, $\mathbf{v}$ and $\mathbf{c}$, three vectors $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O \in \mathbb{Z}_\Lambda^{(n+1)(n+8)}$ satisfy the equations $\mathbf{M}_L\mathbf{a}_L + \mathbf{M}_R\mathbf{a}_R + \mathbf{M}_O\mathbf{a}_O = \mathbf{M}_V\mathbf{v} + \mathbf{d}$ and $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$ if and only if they are of the form above.

The argument of Bünz et al. is therefore sufficient to prove the relation $\mathcal{R}_{\text{DLOG}-2}$. The prover sends $2\lceil \log_2 4(n+1)\rceil + 8$ $\mathbb{G}_1$ elements and $5\mathbb{Z}_\Lambda$ elements.

10.4.4 An Intermediate Protocol in $\mathbb{G}_2$

This section gives an perfect honest verifier zero-knowledge protocol for relation

$$\begin{aligned}\mathcal{R}'_0 \coloneqq \{(G_2, H_2, X_p, X_q, U, K_u, (K_\gamma)_{\gamma\neq i,j}, (a_\gamma)_\gamma; x_p, x_q, r_p, r_q, u, \rho_u) :\\ \forall \pi \in \{p,q\}, X_\pi = G_2^{x_\pi} H_2^{r_\pi}, U = G_2^u H_2^{\rho_u},\\ \forall \gamma, a_\gamma = G_2^{x_\gamma}, x_\pi(u+K_\pi) = x_\gamma(u+K_\gamma) = 1 \mod \ell\}.\end{aligned}$$

Note that for $\pi \in \{p,q\}$, $\left(UG_2^{K_\pi}\right)^{x_\pi} H_2^{-x_\pi\rho_u} = G_2$, and that $\forall\gamma$, $a_\gamma^u = G_2 a_\gamma^{-K_\gamma}$, i.e., the discrete logarithms of $G_2 a_\gamma^{-K_\gamma}$ in base a_γ for all γ are the same.

The protocol is given on Figure 10.5). As the proof system is public-coin, it can be made non-interactive in the random-oracle model via the Fiat–Shamir heuristic by computing c as $h(G_2, H_2, (X_\pi), U, (K_\pi), (K_\gamma)_\gamma, (a_\gamma)_\gamma, (Y_\pi), V, (H_\pi), (A_\gamma))$ for a random oracle h with $\mathbb{Z}_\ell$ as range. The proof then consists of $(c, (z_\pi, t_\pi)_{\pi\in\{p,q\}}, w, \tau_u, (\tau_\pi)_\pi)$, i.e., 9 $\mathbb{Z}_\ell$ elements.

The protocol is complete, perfectly honest-verifier zero-knowledge, and satisfies extractability under the discrete-logarithm assumption over $\mathbb{G}_2$. The protocol completeness and its zero-knowledge property are straightforward. To prove that the protocol satisfies the extractability property, note that from two distinct accepting transcripts

$$(((Y_\pi), V, (H_\pi), (A_\gamma)_\gamma), c, ((z_\pi), (t_\pi), w, \tau_u, (\tau_\pi)))$$

and

$$(((Y_\pi), V, (H_\pi), (A_\gamma)_\gamma), c', ((z'_\pi), (t_\pi)', \ w', \tau'_u, (\tau'_\pi))),$$

one has

$$\begin{aligned} G_2^{(z_\pi - z'_\pi)/(c'-c)} H_2^{(t_\pi - t'_\pi)/(c'-c)} &= X_\pi,\\ G_2^{(w-w')/(c'-c)} H_2^{(\tau_u - \tau'_u)/(c'-c)} &= U,\\ \left(UG_2^{K_u}\right)^{(z_\pi - z'_\pi)/(c'-c)} H_2^{(\tau_\pi - \tau'_\pi)/(c'-c)} &= G_2,\\ a_\gamma^{(w-w')/(c'-c)} &= G_2 a_\gamma^{-K_\gamma}. \end{aligned}$$

Replacing U in the third line with the expression in the second, one has $G_2^{(w-w')+K_u(z-z'))/(c'-c)} H_2^{(\tau_u-\tau'_u)(z-z')/(c'-c)^2+(\tau_\pi-\tau'_\pi)/(c'-c)} = G_2$. Under the discrete-logarithm assumption over $\mathbb{G}_2$, $(\tau_u - \tau'_u)(z_\pi - z'_\pi)/(c'-c)^2 + (\tau_\pi -$

$\tau'_\pi)/(c'-c) = 0 \mod \ell$ and $(w-w')/(c'-c) + K_u(z_\pi - z'_\pi)/(c'-c) = 1 \mod \ell$. Moreover, $a_\gamma^{(w-w')/(c'-c)} = G_2 a_\gamma^{-K_\gamma}$ for all $\gamma \neq i, j$.

It follows that setting $x_\pi \leftarrow (z_\pi - z'_\pi)/(c'-c)$, $r \leftarrow (t_\pi - t'_\pi)/(c'-c)$, $u \leftarrow (w-w')/(c'-c)$ and $\rho_u \leftarrow (\tau_u - \tau'_u)/(c'-c)$, the tuple $((x_\pi), (r_\pi), u, \rho_u)$ is a valid witness for relation $\mathcal{R}'_0$.

To extract a witness from a prover with a fixed random string, it suffices to repeat the following procedure:

1. run the protocol once, rewind the protocol to the computation step right after the prover sent its first message and run it a second time with a fresh verifier randomness
2. if the verifier accepts both executions, and the challenges of these executions are different, extract a witness as above; otherwise restart.

The expected running time of this procedure is at most $\left(\psi^2 - \ell^{-1}\right)^{-1}$, with ψ the probability that the verifier accepts a protocol execution.

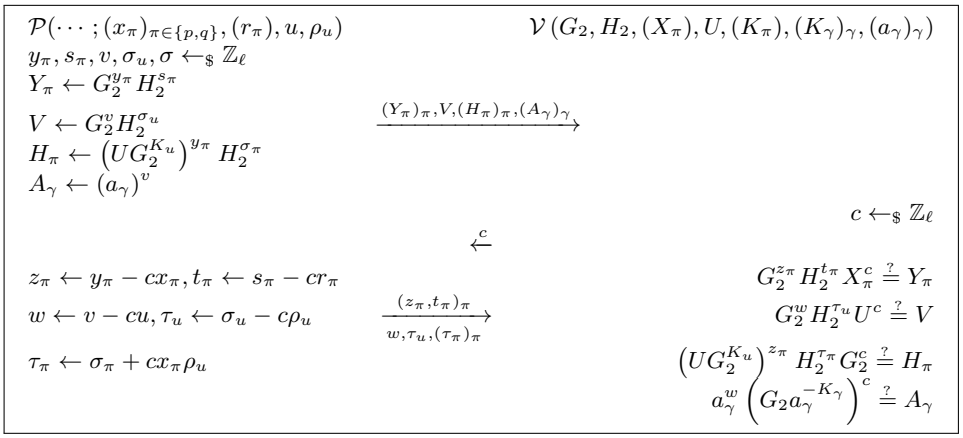

Figure 10.5: Honest-Verifier Zero-Knowledge Protocol for Relation $\mathcal{R}_1$.

10.4.5 Protocol for $\mathcal{R}_0$

To prove knowledge of a witness for $\mathcal{R}_0$, the prover starts setting by setting $K_p \coloneqq h(r_{\mathcal{CA}}) + i$, $K_q \coloneqq h(r_{\mathcal{CA}}) + j$, and $u \coloneqq r'_{\mathcal{U}}$. It then

- computes two commitments $X_p = G_2^{x_p} H_2^{r_p}$ and $X_q = G_2^{x_q} H_2^{r_q}$, for $x_p = (u + K_p)^{-1} \mod \ell$ and $x_q = (u + K_q)^{-1} \mod \ell$
- computes a proof $\pi_{\mathrm{DLOG}-2,p}$ that the double discrete-logarithm of P is the discrete logarithm of X_p, and similarly a proof $\pi_{\mathrm{DLOG}-2,q}$ for Q and X_q
- computes a proof π' for relation $\mathcal{R}'_0$ with X_p and X_q.

The final proof π_0 for $\mathcal{R}_0$ consists of $(X_p, X_q, \pi_{\mathrm{DLOG}-2,p}, \pi_{\mathrm{DLOG}-2,q}, \pi'_0)$.

Security.

It is important to note that the security of the generated key is weakened compared to an RSA-key of the same size since the CA can recover seed s (and thus the prime factors) by solving a discrete logarithm problem in $\mathbb{G}_2$. For 3072-bit RSA moduli, this protocol therefore only provides 96 bits of security (with respect to the CA) instead of the expected 128-bit security level. To avoid this issue, one can increase the bit size of the prime numbers to 3072 bits (but at the cost of generating RSA moduli of twice this size). Another possibility is to use other groups for $\mathbb{G}_2$ with (alleged) harder discrete logarithm problem, e.g., the group of points of an elliptic curve over $\mathbb{F}_p$ or an algebraic torus defined over $\mathbb{F}_{p^2}$ (with compact representation of group elements in $\mathbb{F}_p$) for a 1536-bit prime p. This may however introduce a new bias for the generated primes and require to adapt the zero-knowledge proofs.

Efficiency.

The asymptotic complexity of the communication size depends on the number of trials to obtain two primes in W since the prover has to send $(a_\gamma)_{\gamma \neq i,j}$. However, even though the communication is asymptotically linear in the number of trials, the overhead incurred by the proof of correct computation should in practice be small.

	$\mathbb{Z}_\ell$	$\mathbb{Z}_N$	$\mathbb{Z}_\Lambda$	$\mathbb{G}_1$	$\mathbb{G}_2$	Total (kB)
$\mathcal{R}_0$	9	0	10	$4\lceil\log_2 4b\rceil + 16$	2	346
$\mathcal{R}_1$	0	0	5	$2\lceil\log_2 2(b-1)\rceil + 4$	0	142

Figure 10.6: Size of the Arguments (for a 96-bit security level and 3072-bit RSA moduli).

Total Proof Size.

As discussed in Section 10.4.3, proofs $\pi_{\text{DLOG}-2,p}$ and $\pi_{\text{DLOG}-2,q}$ both consists of $2\lceil\log_2 4(n+1)\rceil + 8$ $\mathbb{G}_1$ elements and $5\mathbb{Z}_\Lambda$ elements.

Proof π' consists of 9 $\mathbb{Z}_\ell$ elements (see Section 10.4.4). Proof π_0 for $\mathcal{R}_0$ therefore consists of 2 $\mathbb{G}_2$ elements, $4\lceil\log_2 4(n+1)\rceil + 16$ $\mathbb{G}_1$ elements, 10 $\mathbb{Z}_\Lambda$ elements and 9 $\mathbb{Z}_\ell$ elements. As for the proof for $\mathcal{R}_1$, the aggregated proof that 2 values committed in $\mathbb{G}_1$ are in $[\![0, 2^{b-1} - 1]\!]$ consists of $2\lceil\log_2 2(b-1)\rceil + 4$ $\mathbb{G}_1$ elements (recall that $n + 1 = b$) and 5 $\mathbb{Z}_\Lambda$ elements.

Running Time.

An important question about the protocol is the number of necessary PRF trials to obtain two primes that satisfy the conditions required for the factors

of N (captured by $W \subseteq \mathbb{P}$). We estimate the number j of necessary trials in the case $W = \mathbb{P} \cap [\![2^{b-1}, 2^b - 1]\!]$, i.e., when $\mathcal{U}$ simply has to prove that p and q are prime of $b(\lambda)$ bits. The following analysis shows (using a number-theoretic heuristic) that the number of trials exceeds $17b(\lambda) = O(\log \lambda)$ (so the DY PRF remains secure), and that the probability that it is larger than that decreases exponentially fast.

For an integer x, denote by $\pi(x)$ the number of primes smaller or equal to x. For $x \geq 30$, Chebyshev's theorem [Dus98, p. 9] ensures that $0.92x/\ln(x) < \pi(x) < 1.11x/\ln(x)$. Therefore, if $b > \log_2 30 + 1$, $\pi\left(2^b\right) > 0.92 \cdot 2^b / b\ln(2)$ and $\pi\left(2^{b-1}\right) < 1.11 \cdot 2^b/2(b-1)\ln(2)$.

For two integers q and a and an integer x, denote by $\pi(x; q, a)$ the number of primes congruent to a modulo q and smaller or equal to x. For all $x > q$, then the Brun–Titchmarsh theorem [MV73] implies that $\pi(x; q, a) \leq \frac{2x}{\varphi(q)\ln(x/q)}$.

The number of primes in $[\![2^{b-1}, 2^b - 1]\!]$ which are not congruent to 1 modulo e where e is a prime number greater than 17 is then at least

$$0.92 \cdot \frac{2^b}{b\ln(2)} - \frac{2 \cdot 2^b}{(e-1)(b\ln(2) - \ln(e))} - 1.11 \cdot \frac{2^b}{2(b-1)\ln(2)} \geq 0.12 \frac{2^b}{(b - \ln(e)/\ln(2))}$$

for $b \geq 6\ln(e)/\ln(2)$.

To apply Theorem 10.3.7, it remains to estimate the number of b-bit primes p in $QR_{2\ell+1}$ and such that $\gcd(e, p-1) = 1$.

Assuming that around half of the primes in $[\![2^{b-1}, 2^b - 1]\!]$ that are not congruent to 1 modulo are quadratic residues modulo $2\ell + 1$, the factor β in Theorem 10.3.7 is at least $0.12/(b - \ln(e)/\ln(2)) \geq 0.12/b$ (the range of the PRF is has size ℓ).

This heuristic is supported by the fact that half of the integers in $[\![2\ell]\!]$ are quadratic residues, and that primes chosen uniformly at random often have properties similar to those of general integers chosen uniformly at random.

Overall Communication Size.

In the last flow of the protocol, the prover then sends an integer N, two commitments in $\mathbb{G}_1$, $17b(\lambda) - 2$ integers in $[\![0, 2\ell]\!]$ with high probability, i.e., the $(a_\gamma)_{\gamma \neq i,j}$ values which are integers returned by the PRF and not in W, and the proof of correct computation of which the size is summarized in Table 10.4.5.

Chapter 11

Conclusion and Future Work

Chapter 9 presented a succinct argument for Diophantine satisfiability based on generic assumptions over hidden-order group generators. More precisely, it gave a procedure to turn any Diophantine equation into a Hadamard product and set of linear constraints, an then an argument for such relations.

Although these techniques were given in generic hidden-order groups, it would be worth investigating whether they also hold under weaker assumptions in specific groups, e.g., the RSA assumption instead of the strong RSA assumption in RSA groups.

Besides, to instantiate the construction in ideal-class groups, the standard method consists in selecting a discriminant uniformly at random from a large set of values. Various heuristic arguments indicate that the discriminant should then contain large and random prime factors, and thus be difficult to factorize; in which case root extraction appears to be computationally hard. However, choosing such large value inexorably impacts the efficiency of the group operations. To be able to choose smaller values, variants of the strong-root assumption [CCL+20] have been introduced. They exclude powers of 2 as square roots can be efficiently computed in ideal-class groups of quadratic number fields if the discriminant is easy to factorize. For improved practical efficiency, it would be worthwhile to determine whether the commitment scheme and the argument for Diophantine satisfiability give in Chapter 9 are still secure under such variants of the strong-root assumption.

Chapter 10 formalized the problem of verifiable randomness in public-key generation. It presented a game-based security model that accounts for concurrent sessions and guarantees that the only attack possible is the unavoidable "halting attack". It then gave a protocol for key-generation algorithms that can be represented as probabilistic circuits as well as a protocol for factoring-based keys.

Despite the wide range of scenarios captured by the game-based model, it does not guarantee that a protocol proved secure in it remains composably

secure. Providing a universal-composability definition seems natural in this setting, but the main hurdle in doing so comes from the fact that the sampler cannot communicate at all with the distinguisher since it would otherwise allow for covert channels as explained in Section 10.2.3. As a consequence, a universal-composability definition would need functionalities with local adversaries, which would be an interesting research direction.

Moreover, designing practical protocols for modern post-quantum cryptosystems based on the learning-with-error problem [Reg05] and variants therefor would be another important contribution.

List of Figures

List of Tables

Bibliography

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 214–234. Springer, Heidelberg, December 2013.

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 332–352. Springer, Heidelberg, March / April 2015.

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, Heidelberg, August 2000.

[ACNP16] Michel Abdalla, Mario Cornejo, Anca Nitulescu, and David Pointcheval. Robust password-protected secret sharing. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016: 21st European Symposium on Research in Computer Security, Part II*, volume 9879 of *Lecture Notes in Computer Science*, pages 61–79. Springer, Heidelberg, September 2016.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, Heidelberg, August 2009.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. Math. (2)*, 2004(2):781–793, 2004.

[AP18] Benedikt Auerbach and Bertram Poettering. Hashing solutions instead of generating problems: On the interactive certification of RSA moduli. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 403–430. Springer, Heidelberg, March 2018.

[Art10] Michael Artin. *Algebra*. Pearson, 2010.

[Art17] Article 29 Data Protection Working Party. Opinion 03/2017 on processing personal data in the context of cooperative intelligent transport systems (C-ITS) - wp252. `http://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=610171`, 2017.

[ASM10] Man Ho Au, Willy Susilo, and Yi Mu. Proof-of-knowledge of representation of committed value and its applications. In Ron Steinfeld and Philip Hawkes, editors, *ACISP 10: 15th Australasian Conference on Information Security and Privacy*, volume 6168 of *Lecture Notes in Computer Science*, pages 352–369. Springer, Heidelberg, July 2010.

[Ava05] Roberto Maria Avanzi. The complexity of certain multi-exponentiation techniques in cryptography. *Journal of Cryptology*, 18(4):357–373, September 2005.

[Bab85] László Babai. Trading group theory for randomness. In *17th Annual ACM Symposium on Theory of Computing*, pages 421–429. ACM Press, May 1985.

[Bar68] Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comput.*, 22(103):565–578, 1968.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, Heidelberg, May 2004.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE*

Symposium on Security and Privacy, pages 315–334. IEEE Computer Society Press, May 2018.

[BBC+13] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475. Springer, Heidelberg, August 2013.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, Heidelberg, August 2004.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 339–369. Springer, Heidelberg, December 2016.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004: 11th Conference on Computer and Communications Security*, pages 132–145. ACM Press, October 2004.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer, Heidelberg, August 2009.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume

9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, Heidelberg, May 2016.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 449–458. ACM Press, October 2008.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, Heidelberg, August 1996.

[BCN+10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, Heidelberg, September 2010.

[BCV16] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 3–18. Springer, Heidelberg, February / March 2016.

[BD19] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, Oct 2019.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464. Springer, Heidelberg, December 2018.

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619. Springer, Heidelberg, August 2006.

[BFGN17] Fabrice Benhamouda, Houda Ferradi, Rémi Géraud, and David Naccache. Non-interactive provably secure attestations for arbitrary RSA prime generation algorithms. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017: 22nd European Symposium on Research in Computer Security, Part I*, volume 10492 of *Lecture Notes in Computer Science*, pages 206–223. Springer, Heidelberg, September 2017.

[BFL90] Joan Boyar, Katalin Friedl, and Carsten Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 155–172. Springer, Heidelberg, April 1990.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 777–804. Springer, Heidelberg, December 2016.

[BG13] Stephanie Bayer and Jens Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 646–663. Springer, Heidelberg, May 2013.

[BGK15] Razvan Barbulescu, Pierrick Gaudry, and Thorsten Kleinjung. The tower number field sieve. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 31–55. Springer, Heidelberg, November / December 2015.

[BHH+19] Michael Backes, Lucjan Hanzlik, Amir Herzberg, Aniket Kate, and Ivan Pryvalov. Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 11442 of *Lecture Notes in Computer Science*, pages 286–313. Springer, Heidelberg, April 2019.

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415. Springer, Heidelberg, August 2013.

[BJL16] Johannes Blömer, Jakob Juhnke, and Nils Löken. Short group signatures with distributed traceability. In Ilias S. Kotsireas, Siegfried M. Rump, and Chee K. Yap, editors, *Mathematical Aspects of Computer and Information Sciences*, pages 166–180, Cham, 2016. Springer International Publishing.

[BJSL11] Ali Bagherzandi, Stanislaw Jarecki, Nitesh Saxena, and Yanbin Lu. Password-protected secret sharing. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011: 18th Conference on Computer and Communications Security*, pages 433–444. ACM Press, October 2011.

[BL09] Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing. Cryptology ePrint Archive, Report 2009/095, 2009. `http://eprint.iacr.org/2009/095`.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, Heidelberg, December 2001.

[BLS11] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. On the correct use of the negation map in the pollard rho method. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 128–146. Springer, Heidelberg, March 2011.

[BMP+14] Norbert Bißmeyer, Sebastian Mauthofer, Jonathan Petit, Mirko Lange, Martin Moser, Daniel Estor, Michel Sall, Michael Feiri, Rim Moalla, Marcello Lagana, and Frank Kargl. V2X security architecture v2. PRESERVE Project, Deliverable D1.3, 2014.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general

assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, Heidelberg, May 2003.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 390–399. ACM Press, October / November 2006.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 501–510. ACM Press, October 2008.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, Heidelberg, January 2003.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer, Heidelberg, May 2000.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, Heidelberg, September 2008.

[BP97] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, Heidelberg, May 1997.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks.

In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, Heidelberg, May 2000.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19. Springer, Heidelberg, August 2014.

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111. Springer, Heidelberg, March 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[BR95] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th Annual ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, May / June 1995.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004: 11th Conference on Computer and Communications Security*, pages 168–177. ACM Press, October 2004.

[BS17] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2017.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, Heidelberg, February 2005.

[BTV20] Olivier Blazy, Patrick Towa, and Damien Vergnaud. Public-key generation with verifiable randomness. *IACR Cryptol. ePrint Arch.*, 2020:294, 2020.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCD+17] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *2017 IEEE Symposium on Security and Privacy*, pages 901–920. IEEE Computer Society Press, May 2017.

[CCL+20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 266–296. Springer, Heidelberg, May 2020.

[CDL+20a] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. Short threshold dynamic group signatures. *IACR Cryptol. ePrint Arch.*, 2020:16, 2020.

[CDL+20b] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. Zone encryption with anonymous authentication for V2V communication. *IACR Cryptol. ePrint Arch.*, 2020:43, 2020.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, Heidelberg, May 2001.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, Heidelberg, August 1994.

[CG05] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In Carlo Blundo and

Stelvio Cimato, editors, *SCN 04: 4th International Conference on Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 120–133. Springer, Heidelberg, September 2005.

[CG07] Sébastien Canard and Aline Gouget. Divisible e-cash systems can be truly anonymous. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 482–497. Springer, Heidelberg, May 2007.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57. Springer, Heidelberg, February 2004.

[CGM16] Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 499–530. Springer, Heidelberg, August 2016.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, Heidelberg, May 2005.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 201–210. ACM Press, October / November 2006.

[CJW05] Jong Youl Choi, Markus Jakobsson, and Susanne Wetzel. Balancing auditability and privacy in vehicular networks. In *Q2SWinet'05*, pages 79–87, 2005.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300. Springer, Heidelberg, April 2012.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, Heidelberg, August 2003.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, Heidelberg, August 2002.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, Heidelberg, September 2003.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, Heidelberg, August 2004.

[CLLN14] Jan Camenisch, Anja Lehmann, Anna Lysyanskaya, and Gregory Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 256–275. Springer, Heidelberg, August 2014.

[CLNS16] Jan Camenisch, Anja Lehmann, Gregory Neven, and Kai Samelin. Virtual smart cards: How to sign with a password

and a server. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 353–371. Springer, Heidelberg, August / September 2016.

[CM99] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer, Heidelberg, May 1999.

[CMBF13] Henry Corrigan-Gibbs, Wendy Mu, Dan Boneh, and Bryan Ford. Ensuring high-quality randomness in cryptographic key generation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 685–696. ACM Press, November 2013.

[CMG+16] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the juniper dual EC incident. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 468–479. ACM Press, October 2016.

[CNR12] Jan Camenisch, Gregory Neven, and Markus Rückert. Fully anonymous attribute tokens from lattices. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12: 8th International Conference on Security in Communication Networks*, volume 7485 of *Lecture Notes in Computer Science*, pages 57–75. Springer, Heidelberg, September 2012.

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 321–350. Springer, Heidelberg, April / May 2017.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In

Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, Heidelberg, August 1997.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, Heidelberg, August 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, Heidelberg, April / May 2002.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, Heidelberg, April 1991.

[Des96] Yvo Desmedt. Simmons' protocol is not free of subliminal channels. In *Ninth IEEE Computer Security Foundations Workshop, March 10 - 12, 1996, Dromquinna Manor, Kenmare, County Kerry, Ireland*, pages 170–175, 1996.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, Heidelberg, August 1990.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer, Heidelberg, December 2002.

[dFGSV17] José María de Fuentes, Lorena González-Manzano, Jetzabel Serna-Olvera, and Fatbardh Veseli. Assessment of attribute-based credentials for privacy-preserving road traffic services in smart cities. *Personal and Ubiquitous Computing*, 21(5):869–891, 2017.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHO16] Ivan Damgård, Helene Haagh, and Claudio Orlandi. Access control encryption: Enforcing information flow with cryptography. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 547–576. Springer, Heidelberg, October / November 2016.

[dLS19] Rafaél del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for FHE and ring-LWE ciphertexts. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 11442 of *Lecture Notes in Computer Science*, pages 344–373. Springer, Heidelberg, April 2019.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18: 13th ACM Symposium on Information, Computer and Communications Security*, pages 551–565. ACM Press, April 2018.

[Dus98] Pierre Dusart. *Autour de la fonction qui compte le nombre de nombres premiers*. PhD thesis, Université de Limoges, 1998.

[Dwo04] Morris Dworkin. Request for review of key wrap algorithms. Cryptology ePrint Archive, Report 2004/340, 2004. `http://eprint.iacr.org/2004/340`.

[Dwo07] Morris J. Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. `https://www.nist.gov/`, 2007.

[Dwo16] Morris J. Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication. `https://www.nist.gov/`, 2016.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume

3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, Heidelberg, January 2005.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147. Springer, Heidelberg, August 2013.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[ETS14] Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. Technical Report ETSI EN 302 637-2 V1.3.1, ETSI, 2014.

[ETS18] Intelligent transport systems (ITS); security; ITS communications security architecture and security management. Technical Report ETSI TS 102 940 v1.3.1, ETSI, 2018.

[ETS19] Intelligent transport systems (ITS); security; trust and privacy management. Technical Report ETSI TS 102 941 V1.3.1, ETSI, 2019.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437. IEEE Computer Society Press, October 1987.

[FFHR19] Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 159–190. Springer, Heidelberg, December 2019.

[FKK17] Cody Freitag, Jonathan Katz, and Nathan Klein. Symmetric-key broadcast encryption: The multi-sender case. In Shlomi Dolev and Sachin Lodha, editors, *Cyber Security Cryptography and Machine Learning - First International Conference, CSCML 2017*, volume 10332 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2017.

[FM19] Georgios Fotiadis and Chloe Martindale. Optimal TNFS-secure pairings on elliptic curves with composite embedding degree. Cryptology ePrint Archive, Report 2019/555, 2019. https://eprint.iacr.org/2019/555.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, Heidelberg, August 1997.

[FO18] Georg Fuchsbauer and Michele Orrù. Non-interactive zaps of knowledge. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*, volume 10892 of *Lecture Notes in Computer Science*, pages 44–62. Springer, Heidelberg, July 2018.

[FRF+07] Julien Freudiger, Maxim Raya, Márk Félegyházi, Panos Papadimitratos, and Jean-Pierre Hubaux. Mix-zones for location privacy in vehicular networks. *ACM Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS)*, 2007.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, Heidelberg, August 1987.

[fS15] International Organization for Standardization, 2015.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

[GGI19] Rosario Gennaro, Steven Goldfeder, and Bertrand Ithurburn. Fully distributed group signatures, 2019.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479. IEEE Computer Society Press, October 1984.

[Gha15] Essam Ghadafi. Efficient distributed tag-based encryption and its application to group signatures with efficient

distributed traceability. In Diego F. Aranha and Alfred Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014: 3rd International Conference on Cryptology and Information Security in Latin America*, volume 8895 of *Lecture Notes in Computer Science*, pages 327–347. Springer, Heidelberg, September 2015.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310. Springer, Heidelberg, May 1999.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer, Heidelberg, April 2015.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, Heidelberg, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM Press, May 1982.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GM02] Lutz Gollan and Christophe Meinel. Digital signatures for automobiles. In *Systemics, Cybernetics and Informatics (SCI) 2002*, pages 1–5, 2002.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304. ACM Press, May 1985.

[GMR98] Rosario Gennaro, Daniele Micciancio, and Tal Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In Li Gong and Michael K. Reiter, editors, *ACM CCS 98: 5th Conference on Computer and Communications Security*, pages 67–72. ACM Press, November 1998.

[GMT19] Aurore Guillevic, Simon Masson, and Emmanuel Thomé. Cocks–pinch curves of embedding degrees five to eight and optimal ate pairing computation. 2019.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, Heidelberg, May / June 2006.

[GPC+] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03. ACM.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer, Heidelberg, May 1988.

[Gre11] Matthew Green. Secure blind decryption. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 265–282. Springer, Heidelberg, March 2011.

[Gro05] Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 467–482. Springer, Heidelberg, June 2005.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology –*

ASIACRYPT 2006, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, Heidelberg, December 2006.

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 192–208. Springer, Heidelberg, August 2009.

[Gro14] Trusted Computing Group. Trusted platform module library specification,family "2.0", 2014.

[GRSB19] Sharon Goldberg, Leonid Reyzin, Omar Sagga, and Foteini Baldimtsi. Efficient noninteractive certification of RSA moduli and beyond. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 700–727. Springer, Heidelberg, December 2019.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, Heidelberg, April 2008.

[GS19] Aurore Guillevic and Shashank Singh. On the alpha value of polynomials in the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2019/885, 2019. `https://eprint.iacr.org/2019/885`.

[Gui19] Aurore Guillevic. A short-list of STNFS-secure pairing-friendly curves at the 128-bit security level. Cryptology ePrint Archive, Report 2019/1371, 2019. `https://eprint.iacr.org/2019/1371`.

[HCL04] Jean-Pierre Hubaux, Srdjan Capkun, and Jun Luo. The security and privacy of smart vehicles. *IEEE Security & Privacy*, 2(3):49–55, 2004.

[HDWH12] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In Tadayoshi Kohno, editor, *USENIX Security 2012: 21st USENIX Security Symposium*, pages 205–220. USENIX Association, August 2012.

[IPV10] Malika Izabachène, David Pointcheval, and Damien Vergnaud. Mediated traceable anonymous encryption. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *Progress in Cryptology - LATINCRYPT 2010: 1st International Conference on Cryptology and Information Security in Latin America*, volume 6212 of *Lecture Notes in Computer Science*, pages 40–60. Springer, Heidelberg, August 2010.

[JG02] Ari Juels and Jorge Guajardo. RSA key generation with verifiable randomness. In David Naccache and Pascal Paillier, editors, *PKC 2002: 5th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 357–374. Springer, Heidelberg, February 2002.

[JKKX17] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. TOPPSS: Cost-minimal password-protected secret sharing based on threshold OPRF. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 39–58. Springer, Heidelberg, July 2017.

[JMV01] Don Johnson, Alfred Menezes, and Scott A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1):36–63, 2001.

[Jou14] Antoine Joux. A new index calculus algorithm with complexity $L(1/4+o(1))$ in small characteristic. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013: 20th Annual International Workshop on Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 355–379. Springer, Heidelberg, August 2014.

[JR17] Charanjit S. Jutla and Arnab Roy. Improved structure preserving signatures under standard bilinear assumptions. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 183–209. Springer, Heidelberg, March 2017.

[KB16] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lec-*

ture Notes in Computer Science, pages 543–571. Springer, Heidelberg, August 2016.

[KBT06] Pandurang Kamat, Arati Baliga, and Wade Trappe. An identity-based security framework for VANETs. In *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*, VANET '06, pages 94–95, New York, NY, USA, 2006. ACM.

[KJ17] Taechan Kim and Jinhyuck Jeong. Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 388–408. Springer, Heidelberg, March 2017.

[Kra22] M. Kraitchik. *Théorie des nombres: Congruences et divisibilités avec une préface de M. d'Ocagne*. Théorie des nombres. Gauthier-Villars, 1922.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, Heidelberg, August 2010.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer, Heidelberg, March 2011.

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 101–128. Springer, Heidelberg, April 2015.

[KW19] Thorsten Kleinjung and Benjamin Wesolowski. Discrete logarithms in quasi-polynomial time in finite fields of fixed characteristic. Cryptology ePrint Archive, Report 2019/751, 2019. `https://eprint.iacr.org/2019/751`.

[LHA+12] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In Reihaneh Safavi-Naini and Ran

Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 626–642. Springer, Heidelberg, August 2012.

[Li16] Xin Li. Improved two-source extractors, and affine extractors for polylogarithmic entropy. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 168–177. IEEE Computer Society Press, October 2016.

[Lip03] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415. Springer, Heidelberg, November / December 2003.

[LPQ17] Benoît Libert, Thomas Peters, and Chen Qian. Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 247–276. Springer, Heidelberg, March 2017.

[LY10] Benoît Libert and Moti Yung. Dynamic fully forward-secure group signatures. In Dengguo Feng, David A. Basin, and Peng Liu, editors, *ASIACCS 10: 5th ACM Symposium on Information, Computer and Communications Security*, pages 70–81. ACM Press, April 2010.

[Man05] Mark Manulis. Democratic group signatures on example of joint ventures. Cryptology ePrint Archive, Report 2005/446, 2005. `http://eprint.iacr.org/2005/446`.

[Mat70] Yu. V. Matiyasevich. Enumerable sets are diophantine. *Sov. Math., Dokl.*, 11:354–358, 1970.

[Mil04] Victor Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17:235–261, 09 2004.

[Mon87] Peter L Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001:*

8th Conference on Computer and Communications Security, pages 245–254. ACM Press, November 2001.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 2019.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge International Series on Parallel Computation. Cambridge University Press, 1995.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, Heidelberg, February 2004.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 657–686. Springer, Heidelberg, April 2015.

[MSJ06] Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. *Journal of Cryptology*, 19(1):27–66, January 2006.

[MSS16] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of nfs advances on the security of pairing-based cryptography. Cryptology ePrint Archive, Report 2016/1102, 2016. `https://eprint.iacr.org/2016/1102`.

[MV73] H. L. Montgomery and R. C. Vaughan. The large sieve. *Mathematika*, 20(2):119–134, 1973.

[NBCN17a] Gregory Neven, Gianmarco Baldini, Jan Camenisch, and Ricardo Neisse. Privacy-preserving attribute-based credentials in cooperative intelligent transport systems. In *2017 IEEE Vehicular Networking Conference, VNC 2017*, pages 131–138. IEEE, 2017.

[NBCN17b] Gregory Neven, Gianmarco Baldini, Jan Camenisch, and Ricardo Neisse. Privacy-preserving attribute-based credentials in cooperative intelligent transport systems. In *2017 IEEE Vehicular Networking Conference, VNC 2017*, pages 131–138. IEEE, 2017.

[NIS13] NIST. National Institute of Standards and Technology – Digital signature standard (dss). `https://csrc.nist.gov/publications/detail/fips/186/4/final`, 2013.

[NSS+17] Matús Nemec, Marek Sýs, Petr Svenda, Dusan Klinec, and Vashek Matyas. The return of coppersmith's attack: Practical factorization of widely used RSA moduli. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1631–1648. ACM Press, October / November 2017.

[oTNHTSA17] US Department of Transportation; National Highway Traffic Safety Administration. Notice of proposed rulemaking for federal motor vehicle safety standards; V2V communications. *Federal Register*, 82(8), 2017.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Heidelberg, August 1992.

[Pol78] John M Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[PR07] Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 517–534. Springer, Heidelberg, August 2007.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 111–126. Springer, Heidelberg, February / March 2016.

[PS18] David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor,

Topics in Cryptology – CT-RSA 2018, volume 10808 of *Lecture Notes in Computer Science*, pages 319–338. Springer, Heidelberg, April 2018.

[PSFK15] Jonathan Petit, Florian Schaub, Michael Feiri, and Frank Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE Communications Surveys and Tutorials*, 17(1):228–255, 2015.

[RAF11] Marshall Riley, Kemal Akkaya, and Kenny Fong. Group-based hybrid authentication scheme for cooperative collision warnings in VANETs. *Security and Communication Networks*, 4(12):1469–1482, 2011.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[RLS+17] Leonid Reyzin, Anna Lysyanskaya, Vitaly Shmatikov, Adam D. Smith, and Center for Democracy & Technology. Comments on NHTSA notice of proposed rule for FMVSS no. 150, V2V communications. `https://cdt.org/files/2017/04/FMVSS150CommentsOnPrivacy-as-submitted.pdf`, 2017.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, Heidelberg, May / June 2006.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[SABB16] Katrin Sjöberg, Peter Andres, Teodor Buburuzan, and Achim Brakemeier. C-ITS deployment in europe - current status and outlook. *CoRR*, abs/1609.03876, 2016.

[SABD18] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. *CoRR*, abs/1802.07344, 2018.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, Heidelberg, August 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[SF17] Ankit Singh and Hervais Simo Fhom. Restricted usage of anonymous credentials in vehicular ad hoc networks for misbehavior detection. *Int. J. Inf. Sec.*, 16(2):195–211, 2017.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. of Symp. Math. Soc., 1971*, volume 20, pages 41–440, 1971.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society Press, November 1994.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, Heidelberg, May 1997.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[Sko50] Thoralf Skolem. *Diophantische Gleichungen.* Ergebnisse der Mathematik und ihrer Grenzgebiete. New York, Chelsea Pub. Co., 1950.

[Sli01] Arkadii Slinko. A generalization of Komlos theorem on random matrices. *New Zealand J. Math.*, 30(1):81–86, 2001.

[SMK09] F. Schaub, Z. Ma, and F. Kargl. Privacy requirements in vehicular communication systems. In *2009 International Conference on Computational Science and Engineering*, volume 3, pages 139–145, Aug 2009.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, Heidelberg, May 1996.

[TBNM16] Hamza Toulni, Mohcine Boudhane, Benayad Nsiri, and Mounia Miyara. An adaptive key exchange procedure for VANET. *International Journal of Advanced Computer Science and Applications*, 7(4), 2016.

[TCMDS11] Carmela Troncoso, Enrique Costa-Montenegro, Claudia Diaz, and Stefan Schiffner. On the difficulty of achieving anonymity for vehicle-2-x communication. *Computer Networks*, 55(14), 2011.

[Tes01] Edlyn Teske. On random walks for pollard's rho method. *Mathematics of computation*, 70(234):809–825, 2001.

[Tur36] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

[TV20] Patrick Towa and Damien Vergnaud. Succinct diophantine-satisfiability arguments. *IACR Cryptol. ePrint Arch.*, 2020:682, 2020.

[VHG19] E. Verheul, C. Hicks, and F. D. Garcia. Ifal: Issue first activate later certificates for v2x. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, 2019.

[vP88] Jeroen van de Graaf and René Peralta. A simple and secure way to show the validity of your public key. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 128–134. Springer, Heidelberg, August 1988.

[WWKH13] William Whyte, André Weimerskirch, Virendra Kumar, and Thorsten Hehn. A security credential management system for V2V communications. In *2013 IEEE Vehicular Networking Conference*, pages 1–8. IEEE, 2013.

[YY97] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, Heidelberg, May 1997.

[Zen06] Ke Zeng. Pseudonymous PKI for ubiquitous computing. In *EuroPKI 2006*, volume 4043 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2006.

[ZLM+08] Dong Zheng, Xiangxue Li, Changshe Ma, Kefei Chen, and Jianhua Li. Democratic group signatures with threshold traceability. Cryptology ePrint Archive, Report 2008/112, 2008. `http://eprint.iacr.org/2008/112`.

RÉSUMÉ

Ce manuscrit propose des nouveaux protocoles cryptographiques qui sont respectueux de la vie privée des utilisateurs et qui ont des applications dans la vie réelle.
Dans une première partie, l'accent est mis sur les signatures de groupe, une primitive cryptographique qui permet aux membres d'un groupe d'utilisateurs de signer anonymement au nom du groupe, et sur la confidentialité des messages. Pour éviter de faire confiance à des autorités uniques, les signatures de groupe sont ici définies avec plusieurs autorités et permettent l'émission à seuil de titres de créance ainsi que l'ouverture à seuil. Ces signatures de groupe sont alors utilisées comme mécanisme d'authentification pour la communication entre véhicules, et, combinées au chiffrement par zone, une nouvelle primitive permettant aux véhicules de chiffrer éfficacement leur communication, elles assurent de fortes garanties de sécurité bien définies pour les systèmes de transport coopératifs et intelligents. Par la suite, le chiffrement à clef publique est étudié dans un contexte plus général dans lequel les utilisateurs n'ont pas accès à un support de stockage sécurisé pour leurs clefs secrètes, mais peuvent tirer parti de mots de passe et de l'interaction avec des serveurs pour obtenir des garanties de sécurité comparables tout en préservant leurs vies privées.
Dans une deuxième partie, nous étudions des primitives cryptographiques à portée générale qui ont des applications à la protection de la vie privée. Dans un premier temps, nous étudions les arguments à divulgation nulle, un type de schémas cryptographiques qui permettent à un prouveur avec une puissance de calcul limitée de convaincre un vérifieur d'une assertion sans révéler aucune information supplémentaire. Plus précisement, nous étudions des arguments de satisfiabilité d'équations diophantiennes qui ont une complexité de communication et une complexité de tour logarithmiques, ainsi que leurs applications à la cryptographie qui vise à protéger la vie privée. Ensuite, nous considérons la question de prouver que l'algorithme d'un utilisateur a correctement choisi et utilisé une graine réellement aléatoire pour générer les clefs de l'utilisateur, un problème d'une importance capitale pour la sécurité de tout système cryptographique à clef publique.

MOTS CLÉS

Vie Privée, Signatures de Groupe, Cryptographie à seuil, V2X, Chiffrement par Zone, Chiffrement à Clef Publique, Cryptographie à Base de Mots de Passe, Preuves à Divulgation Nulle, Équations Diophantiennes, Certification d'Aléa

ABSTRACT

This manuscript proposes new cryptographic protocols that are respectful of users' privacy and which have real-world applications.
In a first part, the focus is on group signatures, a primitive which allows members of a user group to anonymously sign on behalf of the group, and on message confidentiality. To remove the trust on single authorities, group signatures are here defined in a setting with multiple authorities and support both threshold issuance and threshold opening. These group signatures are then used as authentication mechanism for vehicle-to-vehicle communication, and combined with zone encryption, a new primitive whereby vehicles can efficiently encrypt their communication, they provide strong, well-defined privacy guarantees for cooperative intelligent transport systems. Thereafter, public-key encryption is studied in a more general context in which users do not have access to secure storage to protect their secret keys, but can leverage passwords and interaction with servers to obtain comparable security guarantees without renouncing their privacy.
In a second part, the topic of study are general-purpose cryptographic primitives which have privacy-preserving applications. First come zero-knowledge arguments, a type of cryptographic schemes which enable a computationally bounded prover to convince a verifier of a statement without disclosing any information beyond that. More specifically, we study arguments to prove the satisfiability of Diophantine equations which have logarithmic communication and round complexity, as well as their applications to privacy-preserving cryptography. Then, we tackle the problem of proving that a user algorithm selected and correctly used a truly random seed in the generation of her cryptographic key, a problem of fundamental importance to the security of any public-key cryptographic scheme.

KEYWORDS

Privacy, Group Signatures, Threshold Cryptography, V2X, Zone Encryption, Public-Key Encryption, Password-Based Cryptography, Zero-Knowledge Proofs, Diophantine Equations, Randomness Certification