# Transferable e-cash : an analysis in the algebraic group model

Balthazar Bauer

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

# Transferable e-cash: an analysis in the Algebraic Group Model


Soutenue par
**Balthazar BAUER**
Le 14 Décembre 2020


École doctorale n°386
**Sciences Mathématiques de Paris Centre**

Spécialité
**Informatique**

Composition du jury :

| | |
|---|---|
| Georg Fuchsbauer<br>TU Wien | *Directeur de thèse* |
| Damien Vergnaud<br>Université Pierre et Marie Curie | *Président du jury* |
| David Pointcheval<br>CNRS, École Normale Supérieure | *Directeur de thèse* |
| Olivier Blazy<br>Université de Limoges | *Rapporteur* |
| Eike Kiltz<br>Ruhr Universität Bochum | *Rapporteur* |
| Carla Ràfols<br>Universitat Pompeu Fabra | *Examinatrice* |
| Olivier Sanders<br>Orange Labs | *Examinateur* |

1794
ENS | PSL

# Transferable e-cash: an analysis in the Algebraic Group Model

Balthazar BAUER

14 Décembre 2020

# Abstract

Transferable e-cash is the most faithful digital analog of physical cash, as it allows users to transfer coins between them without interacting with the bank. Strong anonymity requirements and the need for mechanisms to trace illegal behavior (double-spending of coins) have made instantiating the concept notoriously hard. Baldimtsi et al. (PKC'15) have given a first instantiation, which however relied on a powerful cryptographic primitive that made the scheme non-practical. In this thesis we revisit the model for transferable e-cash, proposing simpler yet stronger security definitions and then give the first concrete instantiation of the primitive, basing it on bilinear groups, and analyze its concrete efficiency. Because to build our scheme, we are using non-standard assumption in a bilinear group context, we analyze the hardness of a broad class of assumptions in a relevant context: the algebraic group model.

# Acknowledgments

Il y a beaucoup de gens que je souhaiterais remercier à cette occasion, à commencer par mes encadrants de thèse Georg Fuchsbauer et David Pointcheval qui m'ont aidé aussi bien dans pour la partie scientifique de ma thèse que dans mes déboires administratifs.

Je remercie également les autres membres du jury, Damien Vergnaud, Olivier Sanders, Carla Ráfols, et tout particulièrement les rapporteurs Olivier Blazy et Eike Kiltz qui auront bien voulu prendre le temps de relire ce manuscrit.

Plus généralement, le cadre de travail à l'ENS était sans doute idéal pour effectuer cette thèse, entre autres grâce à la bonne humeur et à la gentillesse des nombreux membres du DI que j'ai eu l'occasion de côtoyer, comme Chloé, Romain, Azam, Brice, Antoine, Lenaïck, Pierrick, Louiza, Melissa, Michele, Michele, Thierry, Geoffroy, Anka, Pooya, Michel, Phong, Céline, Théo, Huy, Michael, Julia, Georges-Axel, Aurélien, Florian, Hoeteck, et Lelio, mais aussi grâce au personnel administratif et technique de l'ENS qui par leur travail invisible rendent beaucoup plus agréable le travail de recherche.

En prenant encore plus de distance avec le travail scientifique en lui même, il me semble important de remercier mes amis (hors du DI), qui au minimum sont responsables du fait que ces années de thèse resteront pour moi un souvenir agréable : Timothée, Charles, Paul, Bruno, Clément, Clément, Vincent, Vincent, Vincent, Hugo, Lucas, Ziyad, Ulysse, Bruno, Antonin, Antoine, Antoine, Cyril, Gaël, Lucas, Alexis, Tristan, Eoghen, Titouan, Jérôme, Léo, et Thomas, ainsi que ma famille, en particulier mes parents, mon frère, et ma sœur, qui ont toujours été là pour moi, et mes neveux qui sont une vraie source de bonne humeur.

Même si ils ne liront probablement jamais ces lignes, j'ai une pensée pour mes professeurs de mathématiques du secondaire (dont je vais peut-être mal écrire les noms) sans qui je ne serais pas en train de les écrire: Mesdames Januel, Daumarès, Giuly, Alt, et messieurs Hazan, Saint-Germain, et surtout monsieur Sablé qui en me parlant d'un certain livre de Neal Stephenson, une après-midi après le Bac, m'aura fait découvrir la cryptographie.

Et pour finir, je remercie ma compagne Juliette qui aura eu bien des désagréments à cause de cette thèse, ne serait-ce qu'en supportant mon stress, et ne s'en est jamais plainte, alors que franchement des fois il y aurait de quoi.

# Contents

# Chapter 1

# Introduction

## 1.1 Context

Trade and money are likely to have originated within the first societies with strong, centralized power. It was then a matter of taxing people to contribute to the administrations of a strong state. Since the methods of business transactions have been constantly evolving to allow for more efficient volumes, and ever-increasing transaction frequencies over ever-increasing distances. In particular the digitalization of these transactions has become widespread on whole swathes of the globe. It is notable, given the increase in the use of hardware and software from telecommunication, as with communication in general (a financial transaction being only a communicative exchange), digitalization is accompanied by security and confidentiality issues.

While the issue of confidentiality of transactions is not specific to digitized transactions, it is all the more serious in a technological context that requires that the security and confidentiality issues have taken place on a digital medium without guaranteeing to any the moment the digital data is erased.

**Anonymity and decentralization.** In the 1980s, David Chaum developed concepts such as blind signatures, that make the concept tangible of "confidential transactions". "Cypherpunks" manifests written in the 90s show a real interest in these problems from "anarcho-capitalist" movements. Various companies such as Digicash (1989–1998), Monero (2014–now), Z-Cash (2016–now) were founded, their vocation being to respond to this need for anonymity. It is noteworthy that the anarcho-capitalists (libertarians) have also had the other aim to develop parallel (and even competing) monetary systems to those of the states. Thus, following the logic of Bitcoin (2009–now) [Nak08], which aims decentralizing the issuance of money (and the validation of transactions), but in no way anonymous (David D. Friedman: "Bitcoin is, in a sense, the least anonymous money that has ever existed since every transaction is observable by anyone with a bitcoin account." [Fri]), contrary to the great misunderstanding by the media coverage, Z-Cash, and Monero thus realize both anonymity and the formal "decentralization" (see table). The interest in the confidentiality of transactions seems to echo with the willingness displayed by the states to control much more drastically financial transactions, in order to fight against tax evasion, black markets, and other illegal activities. Even though through public research budgets, the states also fund projects related to anonymous transactions. For example, this is the case of this thesis funded by the french national research agency (ANR). This paradox can be explained by the autonomy of state institutions, or by a lack of understanding of the technological challenges that decision-makers face.

| System | Withdrawing | Support | Transferable | Anonymous | Payment |
|---|---|---|---|---|---|
| Cash | Centralized | Physical | Yes | Yes [a] | Offline |
| Gold | Centralized [b] | Physical | Yes | Yes [c] | Offline |
| New Credit Card/Lydia | Centralized | Digital | No | No | Online |
| Moneo/Old Credit Card | Centralized | Digital | No | No | Offline |
| Monero/Z-Cash | Decentralized | Digital | No | Yes | Online |
| Digicash | Centralized | Digital | No | Yes | Offline |
| Transferable E-cash | Centralized | Digital | Yes | Yes | Offline |

[a] not designed to be
[b] with geological constraints
[c] not designed to be

**Online vs offline.** At a time when digital communications were scarcer, and more expensive, it was common to make commercial transactions offline, in particular by credit card. This method, transparent for the consumer, but much less so for the the merchant had become very widespread in the 1990s with credit card payments (for small amounts). It is in this context that David Chaum developed his company DigiCash, whose technology is based on its own schemes and theoretical models of anonymous e-cash. Today with the promotion of everything-online that accompanies the strong decrease in the cost of data transmission, interest for offline payments has fallen: credit card payments are increasing to the banking network, Digicash closed down. More anecdotally, in France, Moneo, the main mode of payment for university catering, which was offline payment method, has been superseded by its online counterpart Izly.

**Provable security.** Following the formalization of the notions of computation and algorithmic problems by Turing and Church (with for exemple the famous Turing machines [Tur37]), some mathematicians who were working on a computer system still in its infancy, have quite naturally begun classifying problems according to their "hardness", i.e., according to the computing time needed to solve them. This is how the field of research on algorithmic complexity was born (to which the famous "millennium problem" $\mathtt{P} = \mathtt{NP}$ belongs).

Realizing that cryptanalyzing an encrypted text or forge a digital signature is akin to an algorithmic problem, researchers like Diffie, Hellman [DH76], Rivest, Shamir and Adleman [RSA78] have therefore, reduced these problems to problems deemed difficult. (In complexity, reducing a problem $A$ to a problem $B$, consists of showing that $A$ is at least as difficult as $B$.) They have thus proved the security of these schemes, and then invented the paradigm of provable security.

Since then, many schemes have been developed, including satisfying new properties, thus diversifying the uses of cryptographic technologies, sometimes based on new cryptographic assumptions, i.e., assuming the hardness of new problems.

Faced with a profusion of cryptographic assumptions, researchers were struggling to analyze in detail the potential hardness of the considered problems, and therefore the security underlying the protocols which were based on the difficulty of these problems. Victor Shoup then invented a computational model in which it can be shown that a series of problems based on group theory are statistically difficult: The Generic Group Model (GGM) [Sho97]. More recently, an intermediate model has been developed by Fuchsbauer, Kiltz and Loss: the Algebraic Group Model (AGM) [FKL18].

**Overview of the thesis.** After a brief state of the art of the subjects dealt with in the thesis in this chapter, and a reminder of all necessary definitions and properties in Chapter 2, we will show in Chapter 3 why by using the AGM, one may consider as cryptographic assumptions a large family of various assumptions (in particular a non-standard assumption, such as one we will use later).

In Chapter 4, a new primitive is developed: Signatures on rerandomizable ciphertexts. Initially, it was supposed to be used to build transferable e-cash schemes. But we finally showed that it was not necessary, thereby improving the efficiency of our scheme.

Then in chapter 5, we will formally define the security properties of a transferable e-cash scheme in compare our models to previous ones, and finally, we will propose a scheme that addresses the goal of this thesis by formally proving it satisfies our scheme.

## 1.2 State of the art

### The Algebraic Group Model

A central paradigm for assessing the security of a cryptographic scheme or hardness assumption is to analyze it within an *idealized model of computation*. A line of work initiated by the seminal work of Nechaev [Nec94] introduced the *generic group model* (GGM) [Sho97, Mau05], in which all algorithms and adversaries are treated as generic algorithms, i.e., algorithms that do not exploit any particular structure of a group and hence can be run in *any* group. Because for many groups used in cryptography (in particular, groups defined over some elliptic curves), the best known algorithms are in fact generic, the GGM has for many years served as the canonical tool to establish confidence in new cryptographic hardness assumptions. Moreover, when cryptographic schemes have been too difficult to analyze in the standard model, they have also directly been proven secure in the GGM (for example LRSW signatures [LRSW99, CL04]).

Following the approach first used in [ABM15], a more recent work by Fuchsbauer, Kiltz, and Loss [FKL18] introduces the *algebraic group model*, in which all algorithms are assumed to be algebraic [PV05]. An *algebraic algorithm* generalizes the notion of a generic algorithm in that all of its output group elements must still be computed by generic operations; however, the algorithm can freely access the structure of the group and obtain more information than what would be possible by purely generic means. This places the AGM between the GGM and the standard model. In contrast to the GGM, one cannot give information-theoretic lower bounds in the AGM; instead, one analyzes the security of a scheme by giving security reductions from computational hardness assumptions.

Because of its generality and because it provides a powerful framework that simplifies the security analyses of complex systems, the AGM has readily been adopted, in particular in the context of SNARK systems [FKL18, MBKM19, Lip19, GWC19]. It has also recently been used to analyze blind (Schnorr) signatures [FPS20], which are notoriously difficult to prove secure in the standard or random oracle model. Another recent work by Agrikola, Hofheinz and Kastner [AHK20] furthermore shows that the AGM constitutes a plausible model, which is instantiable under falsifiable assumptions in the standard model.

Since its inception, many proofs in the AGM have followed a similar structure, which often consists of a series of tedious case distinctions. A natural question is whether it is possible to unify a large body of relevant hardness assumptions under a general 'Uber' assumption. This would avoid having to prove a reduction to a more well-studied hardness assumption for each of them in the AGM separately. In this work, we present a very rich framework of such Uber assumptions, which contain, as special cases, reductions between hardness assumptions in the AGM from prior work [FKL18, Los19]. We also show that there exists a natural hierarchy among Uber assumptions

of different strengths. Together, our results give an almost complete classification in the AGM of common hardness assumptions over (bilinear) groups of prime order.

### Signatures on randomizable ciphertexts

A standard approach for anonymous authentication is to combine signatures, which yield authentication, with zero-knowledge proofs, which allow to prove possession of a signature without revealing information about the latter and thus provide anonymity. This approach has been followed for (multi-show) anonymous credentials schemes, for which several showings of the same credential cannot be linked (in contrast to one-show credentials, e.g. [Bra00, BL13]).

The zero-knowledge proofs for these schemes are either instantiated using $\Sigma$-protocols [CL03, CL04] (and are thus interactive or in the random oracle model) or in the standard model [BCKL08] using Groth-Sahai proofs [GS08]. As this proof system only supports very specific types of statements in bilinear ("pairing-friendly") groups, signature schemes whose verification is of this type have been introduced: *structure-preserving signatures* [AFG+10] sign messages from a group $\mathbb{G}$ and are verified by checking equivalences of products of pairings of group elements from the verification key, the message and the signature.

**Equivalence-class signatures.** Hanser and Slamanig [HS14] extended this concept to *structure-preserving signatures on equivalence classes* (later improved in [FHS19]) for messages from $\mathbb{G}^2$, by adding a functionality called *signature adaptation*: given a signature on a message $\mathbf{m} \in \mathbb{G}^2$ and a scalar $r$, anyone can "adapt" the signature so it verifies for the message $r \cdot \mathbf{m}$. A signature thus authenticates the equivalence class of all multiples of the signed message.

Equivalence-class signatures (ECS) enable anonymous authentication that completely forgoes the layer of zero-knowledge proofs and thus yields considerable efficiency gains. Consider anonymous credentials. A credential is a signature on a message $\mathbf{m}$ (which typically contains a commitment to the user's attributes). In previous schemes, when authenticating, the user proves in zero knowledge that she knows a message $\mathbf{m}$ (and an opening of the contained commitment to the attributes she wants to show) as well as a signature on $\mathbf{m}$; several authentications with the same credential are thus unlinkable. Using ECS, this is possible *without* using any proof system [FHS19]: the user simply shows $r \cdot \mathbf{m}$ for a fresh random $r$ together with an adapted signature. Anonymity is implied by the following property of ECS: to someone that is given $\mathbf{m}$ and a signature on $\mathbf{m}$, the pair consisting of $\mathbf{m}' := r \cdot \mathbf{m}$ for a random $r$ and the signature adapted to $\mathbf{m}'$ is indistinguishable from a random element $\mathbf{m}''$ from $\mathbb{G}^2$ together with a *fresh* signature on $\mathbf{m}''$.

Besides the first attribute-based anonymous credential scheme for which the complexity of showing is independent of the number of attributes [FHS19], ECS have also been used to build very efficient blind signatures with minimal interaction between the signer and the user that asks for the signature [FHS15, FHKS16], revocable anonymous credentials [DHS15], as well as efficient constructions [FGKO17, DS18] of both access-control encryption [DHO16] and dynamic group signatures [BSZ05].

The most efficient construction of ECS is the one from [FHS19], which was proven secure in the generic group model [Sho97]. A signature consist of 3 elements from a bilinear group, which the authors show to be optimal by relying on a result by Abe et al. [AGHO11]. Moreover, there is strong evidence that structure-preserving signatures of this size cannot be proved secure by a reduction to non-interactive assumptions [AGO11], meaning a proof in the generic group model is the best we can hope for. Less efficient constructions of EQS from standard assumptions have since then been given in the standard model by weakening the security guarantees [FG18] and in the common-reference string model [KSD19] (with signatures 6 times longer than [FHS19]).

*Signatures with flexible public key* [BHKS18] and *mercurials signatures* [CL19] are extensions of ECS that allow signatures to be adapted not only to multiples of the signed message, but also to

multiples of the verification key. This has been used to build delegatable anonymous credentials [BCC+09] in [CL19]. Delegatable credentials allow for hierarchical structures, in which users can delegate obtained credentials to users at lower levels.

**Shortcomings of ECS.** While schemes based on ECS offer (near-)optimal efficiency, a drawback is their weak form of anonymity. Consider a user who asks for a signature on $\mathbf{m} = (m_0G, m_1G)$ (where $G$ is the generator of the group $(\mathbb{G}, +)$). If the user later sees a randomization $(M'_0, M'_1)$ of this message, she can easily identify it as hers by checking whether $m_1M'_0 = m_0M'_1$. The notion of anonymity (which is called *class-hiding* in ECS) that can be achieved for these equivalence classes is thus akin to what has been called *selfless* anonymity [CG05] in the context of group signatures: in contrast to *full* anonymity [BMW03], signatures are only anonymous to those that do not know the secret values used to construct them (the signing key for group signatures; the values $m_0$ and $m_1$ in our example above).

This weakness can have concrete repercussions on the anonymity guarantees provided by schemes built from ECS, for example delegatable credentials. In previous instantiations [BCC+09, Fuc11] of the latter, the showing of a credential is anonymous to anyone, in particular to a user that has delegated said credential to the one showing it. However, in the construction from the ECS variant *mercurial signatures* [CL19], if Alice delegates a credential to Bob, she can identify Bob whenever he uses the credential to authenticate, which represents a serious infringement to Bob's privacy. In fact, anonymity towards the authority issuing (or delegating) credentials has been considered a fundamental property of anonymous credential schemes.

In [CL19], when Alice delegates a credential to Bob, she uses her secret key $(x_0, x_1) \in (\mathbb{Z}^*_{|\mathbb{G}|})^2$ to sign Bob's pseudonym under her own pseudonym $(P_0, P_1) = (rx_0G, rx_1G)$ for a random $r$, which becomes part of Bobs credential. When Bob shows it, he randomizes Alice's pseudonym to $(P'_0, P'_1) := (r'P_0, r'P_1)$ for a random $r'$, which Alice can recognize by checking whether $x_1P'_0 = x_0P'_1$.

**Signatures on randomizable ciphertexts.** To overcome this weakness in anonymity in ECS, we use a different type of equivalence class. Consider an ElGamal [ElG85] encryption $(C_0, C_1) = (rG, M + rP)$ of a message $M$ under an encryption key $P$. Such ciphertexts can be *randomized* by anyone, that is, without knowing the underlying message, a fresh encryption of the same message can be computed by choosing $r'$ and setting $(C'_0, C'_1) := (C_0 + r'G,\ C_1 + r'P) = ((r + r')G,\ M + (r + r')P)$. All possible encryptions of a message form an equivalence class, which, in contrast to multiples of pairs of group elements, satisfy a "full" anonymity notion: after randomization, the resulting ciphertext looks random *even to the one that created the original ciphertext* (see Proposition 4.7).

If such equivalence classes yield better anonymity guarantees, the question is whether we can have adaptable signatures on them, that is, signatures on ciphertexts that can be adapted to randomizations of the signed ciphertext. It turns out that this concept exists and even predates that of ECS and is called *signatures on randomizable ciphertexts* (SoRC) [BFPV11]. Since their introduction, SoRC have been extensively used in e-voting [CCFG16, CFL19, CGG19, HPP20] and other primitives, such as blind signatures and extensions thereof [BFPV13]. Blazy et al. [BFPV11] prove their instantiation of SoRC unforgeable under standard assumptions in bilinear groups. Its biggest drawback is that it only allows for efficiently signing messages that consist of a few bits.

### Transferable e-cash

**E-cash.** Starting with Chaum's seminal work [Cha83], since the 1980s cryptographers have sought to devise an electronic analogue to physical cash, which guarantees secure and private payments [Cha83, CFN90, Bra94, CHL05, CLM07, BCKL09]. Cryptographic e-cash defines three types of participants: a bank, users and merchants. Users withdraw electronic coins from the bank and

can then spend them to merchants, who later deposit them at the bank. The main two properties that e-cash systems are expected to satisfy are (1) *unforgeability*: no one, not even a collusion of adversarial users and merchants can spend more e-coins than they have withdrawn; and (2) *anonymity*: nobody (not even when the bank colludes with the merchant) can determine the spender's identity, that is, link a transaction to a particular withdrawal; spendings should moreover be unlinkable, meaning that no one can tell whether two coins were spent by the same user.

Arguably the most important difference between physical and electronic coins is that while both may be hard to forge, the latter are easy to duplicate. A central challenge in the design of e-cash systems are consequently countermeasures against *double-spending*, that is, multiple spending of the same coin. The first systems proposed were *online* e-cash schemes [Cha83], where the merchant must be connected to the bank when receiving a payment; so before accepting a coin the merchant can enquire with the bank whether it has already been spent. Necessitating constant connectivity is however a strong requirement and not desirable for constrained devices, or not even practicable for autonomous devices like vending machines. *Offline* e-cash [CFN90] eliminates this requirement, leading to a more versatile system, as payments can be made in isolation, without talking to the bank. Since in this setting merchants have no means of determining whether a coin has already been spent, double-spending cannot be prevented. Instead, elaborate mechanisms built into coins ensure that if a coin is deposited twice then the bank can determine the double-spender's identity—while as long as a coin is only spent once, the spender's anonymity is guaranteed.

In offline e-cash a coin withdrawn from the bank consists of the bank's signature $\sigma$ on a unique serial number $s$, which is unknown to the bank. When spending the coin with a merchant, a double-spending tag $t$ is computed, which encodes the identity of the spender in a non-retrievable way. The merchant then deposits $c = (s, \sigma, t)$ at the bank. If two coins $c, c'$ with the same serial number but with different double-spending tags $t, t'$ are deposited, these tags together reveal the identity of the user who double-spent.

Variants of e-cash proposed in the literature include *compact* e-cash [CHL05] and *divisible* e-cash [EO95, Oka95, CG07, CPST15], which aim at making withdrawal more efficient by withdrawing a large number of coins at once or by splitting coins later before spending. *Fair* e-cash [vSN92] adds a tracing mechanism that not only allows to persecute double-spenders, but using a special key an authority can trace any user or coin in the system at any time.

Research on e-cash had progressed slowly for quite some time, arguably due to the fact that banks were unlikely to support it and that credit cards and centralised payment services despite offering little privacy are broadly accepted for online payments. With the advent of Bitcoin [**?**], which proposed an electronic currency that simply bypassed the banks, there has been renewed interest in e-cash, and techniques from the literature on anonymous e-cash are being applied to decentralised cryptocurrencies as well [MGGR13, BCG+14].

**Transferable e-cash.** Classical e-cash requires merchants to deposit received coins at the bank, as these systems do not allow for coins to be transferred from one user to another. This is in stark contrast with physical cash, which naturally changes hands many times before eventually returning to a bank. A more truthful analogue of physical cash should therefore allow users to transfer coins to others, who should be able to further transfer the received coins, and so on; moreover, like spending in offline e-cash, these transfers should be possible offline, in particular, without being connected to the bank. This decreases the communication cost between the bank and the users, and also allows for faster realisations of coin transfers.

Transferable e-cash [vAE90, OO90, FY93] was proposed in the 1990s and a line of research analysed the desired security properties. Despite two decades of research, all proposed schemes are either flawed, only meet weak security and privacy requirements, or are of purely theoretical interest. Examples of the latter category are schemes whose coins grow exponentially in the number

of transfers [CG08], or schemes using involved techniques, such as [BCFK15], that make them far from being implementable on constrained mobile devices.

**Anonymity.** The first transferable e-cash schemes by Okamoto and Ohta [OO90, OO92] satisfy various properties such as divisibility and transferability but only provide very weak levels of anonymity. While withdrawals and spendings cannot be linked, it can be easily decided whether two payments were made by the same user, a notion later termed *weak anonymity* (WA). Chaum and Pedersen [CP93] extended a scheme by van Antwerpen [vAE90] by adding transferability of coins. Their scheme satisfies *strong anonymity* (SA), which, in addition to unlinkability of withdrawals and payments, guarantees that no one can tell whether two payments were made by the same user. However, a user in their scheme can recognise coins he observed in previous transactions. Strong anonymity is also satisfied by later schemes proposed in [Bla08, CGT08].

Chaum and Pedersen [CP93] also showed that transferred coins must necessarily grow in size and that an adversary with unbounded resources can always recognise a coin he owned when seeing it spent later. A formal treatment of anonymity notions for transferable e-cash was undertaken by Canard and Gouget [CG08], who introduce two strengthenings of weak and strong anonymity. *Full anonymity* (FA) requires that no adversary, even when colluding with the bank, can link a coin he previously observed to a coin he receives via a transfer or a spending ("observe-then-receive anonymity"). *Perfect anonymity* (PA) states that an adversary colluding with the bank cannot link a coin previously owned to a coin he receives. Note that these notions are not satisfied by decentralised cryptocurrencies like Bitcoin, where a coin can be traced throughout transfers.

This led to the following hierarchy of anonymity notions: WA $\Leftarrow$ SA $\Leftarrow$ FA $\Leftarrow$ PA. While it was known [CP93] that perfect anonymity cannot hold against unbounded adversaries, Canard and Gouget [CG08] showed that it cannot be achieved against bounded adversaries either. This led them to introduce two orthogonal relaxations of perfect anonymity, both incomparable to FA. PA1 (a.k.a. spend-then-observe): no adversary, controlling the bank, can link a previously owned coin to a coin he passively observes being transferred between two honest users; and PA2 (a.k.a. spend-then-receive): when the bank is honest then no adversary can link a coin previously owned to a coin he receives. (If the adversary colludes with the bank, this notion is not achievable, as it contradicts detection of double-spendings.) The authors then construct a transferable e-cash scheme that satisfies all achievable anonymity properties, but which is completely impractical as it relies on meta-proofs and thus Cook–Levin reductions.

**Implementable schemes.** The first practical scheme that satisfies (conditional versions of) FA, PA1 and PA2 was given by Fuchsbauer, Pointcheval and Vergnaud [FPV09] and was based on the efficient non-interactive zero-knowledge proofs due to Groth and Sahai [GS08] for which they introduced a compatible certification mechanism. The scheme has however two severe shortcomings: the users have to store the data of all transactions they were involved in, so they can prove innocence in case of fraud; and more seriously, when a double-spending is detected, all users up to the double-spender are revealed. Anonymity of a user therefore only holds as long as *every other* user in the system behaves honestly.

Based on the primitive of *commuting signatures* by Fuchsbauer [Fuc11], Blazy, Canard, Fuchsbauer, Gouget, Sibert and Traoré [BCF+11] proposed a scheme that overcomes the above drawbacks by assuming the existence of a trusted entity called the *judge*. This authority must be invoked every time a double-spending is detected, as it holds a key that can lift the anonymity of users. It can however also use it to trace all coins and users in the system at any time, which conflicts with one of the main goals of e-cash: that users remain anonymous as long as they do not double-spend.

The first transferable e-cash scheme that satisfies all anonymity properties suggested in the literature (FA, PA1, PA2) was given last year by Baldimtsi, Chase, Fuchsbauer and Kohlweiss

[BCFK15]. It does not assume any trusted parties nor does it rely on Cook–Levin reductions or heuristics like the random-oracle model [BR93] and is proven secure under standard assumptions from elliptic-curve cryptography. Although coins only grow linearly in the number of transfers and the scheme can be implemented in principle, it is hardly practical. The work first revisits (game-based) anonymity definitions and introduces a new notion that had not been captured before, which is a strengthening of PA2 (a.k.a. spend-then-receive), the strongest anonymity notion defending against a malicious bank. While it is impossible to prevent an adversary that colludes with the bank from linking a coin he previously owned to one he receives [CG08], the new notion demands that he should not learn anything about the users that possessed the coin in between.

For transferable e-cash, the owner of a coin should be able to pass a coin, containing the bank's signature, to another user in a way that maintains the validity of the coin, carries all necessary information to detect double-spending, and all this while preserving anonymity. This can be abstracted as computing from a signature a fresh variant (unlinkable to the original one to ensure anonymity) that includes further information (such as double-spending information for the new owner). Generalising homomorphic signatures [BFKW09] and commuting signatures [Fuc11], Chase et al. [CKLM14] proposed *malleable* signatures, where anyone can transform a signature on a message $m$ into a signature on $m'$, as long as $T(m) = m'$ for some allowed transformation $T$. They show that malleable signatures yield delegatable credentials, and the transferable e-cash scheme in [BCFK15] builds on this construction, while adding traceability of double-spenders. A coin is first signed by the bank and when transferring the coin, a user "mauls" this signature using a valid transformation that guarantees that the coin is owned by the spender and the new coin/signature encodes the right information on the receiver. Serial number and double-spending tag are encrypted under the public key of the bank, which can therefore be used to decrypt to check for double-spending on deposit. Anonymity is ensured by re-randomising these ciphertexts before every transfer.

While malleable signatures are a useful abstraction of the concepts required to transfer coins in an anonymous manner, they dissimulate the considerable complexity underlying their actual instantiations.

**Other works.** For the sake of completeness, let us also mention the following works on transferable e-cash. Zhang et al. [ZLG12] proposed a transferable variant of *conditional* e-cash [SCS07], where the validity of a coin depends on the outcome of an event. The scheme extends the security definitions and the scheme from [BCF+11], but the additional security requirements are not formalised and the proofs are not convincing. Sarkar [Sar13] constructed a protocol, which was later shown to be neither anonymous nor to prevent double-spending by Barguil and Barreto [BB15]. Tiwari and Gupta [TG14] proposed "biometric-based" transferable e-cash and claimed security under hardness of the discrete logarithm problem, but gave no formal security analysis or security proofs.

Very recently, Tewari and Hughes [TH16] proposed "fully anonymous transferable ecash", a title that is doubly misleading. As their transfer protocol provides no means for tracing double-spenders, they require every transfer to be included in a blockchain. The scheme does thus not meet a central requirement of transferable e-cash, namely offline transferability; it is also not anonymous (let alone "fully"), as every coin has a non-hidden unique identifier, which lets anyone trace coins across transfers.

**Instantiation.** Our main contribution is an instantiation of transferable e-cash, which we prove satisfies our security model, and which is much more efficient than the only previous realization [BCFK15]. To do so, we depart from the use of malleable signatures, which due to their generality and strong security guarantees in the spirit of simulation-sound extractability result in very inefficient schemes.

Instead, we give a direct instantiation based on Groth-Sahai proofs [GS08], which are randomizable, structure-preserving signatures [AFG+10], which are compatible with GS proofs, and rerandomizable encryption satisfying RCCA-security [CKN03] (a the corresponding variant of CCA security). While we use signature schemes from the literature [AGHO11, Fuc11], we construct a new RCCA-secure encryption scheme based on [LPQ17] that is tailored to our scheme. Finally, we reuse the (efficient) tags introduced in [BCFK15] to trace double-spenders.

Due to the existence of an omnipotent "judge", no such tags were required in [BCF+11]. Surprisingly, although we do not assume any active trusted parties, we achieve a comparable efficiency, which we do by realizing that the full potential of these tags had not been leveraged in [BCFK15]: they had only been used to encode a user's identity; but, as we show, they can at the same time be used to commit the user. This means that, contrary to all previous instantiations, we can omit the inclusion of signatures by the users in the coins, which makes them lighter. For an informal, yet more detailed overview of our scheme see Sect. 5.3.

## 1.3 Our results

### The Algebraic Group Model

#### An Uber-Assumption Framework for the AGM

The main challenge in analyzing Boyen's framework in the AGM setting is that we can no longer prove lower bounds as in the GGM. The next best thing would be to reduce the Uber assumption to a well-established assumption such as the discrete logarithm (DLog) assumption. Due to the general nature of the Uber assumption, this turns out to be impossible; in particular, our negative result (see below) establishes that *algebraic reductions in the AGM* can only reduce DLog to Uber assumptions that are defined by *linear polynomials*.

Indeed, as for Boyen's [Boy08] proofs in the GGM, the degrees of the involved polynomials are expected to appear in our reductions. In our first theorem in Sect. 3.1 we show that in the AGM any Uber assumption is implied by a parametrized variant of the discrete logarithm problem: in the $q$-DLog problem the adversary, on top of the instance $g^z$, is also given $g^{z^2}, \ldots, g^{z^q}$ and must compute $z$. We prove that if the maximum total degree of the challenge polynomials in $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{T})$ of an Uber assumption is at most $q$, then it is implied by the hardness of the $q$-DLog problem. This establishes that under $q$-DLog, anything that is not trivially computable from a given instance (represented by $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{T})$) is infeasible to compute. We prove this by generalizing a technique first used by Fuchsbauer et al. [FKL18] to prove soundness of Groth's SNARK [Gro16] under the $q$-DLog assumption in the AGM.

**Proof idea.** To convey our main idea, consider a simple instance of the Uber assumption parametrized by polynomials $R_1, \ldots, R_r, F_1$ and let $\overrightarrow{S} = \overrightarrow{T} = \emptyset$. That is, the the adversary is given group elements $\mathbf{U}_1 = g_1^{R_1(\overrightarrow{x})}, \ldots, \mathbf{U}_r = g_1^{R_r(\overrightarrow{x})}$ for a random $\overrightarrow{x}$ and must compute $\mathbf{U}' = g_1^{F_1(\overrightarrow{x})}$. For this problem to be non-trivial, $F_1$ must be linearly independent of $R_1, \ldots, R_r$, that is, for all $\overrightarrow{a} \in \mathbb{Z}_p^r$ we have $R'(\overrightarrow{X}) \neq \sum_i a_i R_i(\overrightarrow{X})$.

Since the adversary is assumed to be algebraic (see Def. 2.4), it computes its output $\mathbf{U}'$ from its inputs $\mathbf{U}_1, \ldots, \mathbf{U}_r$ by generic group operations, that is, for some vector $\overrightarrow{\mu}$ we have $\mathbf{U}' = \prod_i \mathbf{U}_i^{\mu_i}$. In the AGM, the adversary is assumed to output this vector $\overrightarrow{\mu}$. Taking the logarithm of the previous equation yields

$$R'(\overrightarrow{x}) = \sum_{i=1}^{r} \mu_i R_i(\overrightarrow{x}). \tag{1.1}$$

Since $R'$ is independent from $\overrightarrow{R}$, the polynomial $P(\overrightarrow{X}) := R'(\overrightarrow{X}) - \sum_i \mu_i R_i(\overrightarrow{X})$ is non-zero. On the other hand, (1.1) yields $P(\overrightarrow{x}) = 0$ for a successful adversary.

The adversary has thus (implicitly) found a non-zero polynomial $P$, which has the secret $\overrightarrow{x}$ among its roots. Now, in order to use this to solve a $q$-DLog instance $(g_1, g_1^z, \ldots, g_1^{z^q})$, we embed a randomized version of $z$ into every coordinate of $\overrightarrow{x}$. In particular, for random vectors $\overrightarrow{y}$ and $\overrightarrow{v}$, we implicitly let $x_i := y_i z + v_i \bmod p$. By leveraging linearity, the reduction can compute the group elements $\mathbf{U}_i = g_1^{R_i(\overrightarrow{x})}$, etc, from its DLog instance.

If $P(\overrightarrow{X})$ is non-zero then $Q(Z) := P(y_1 Z + v_1, \ldots, y_m Z + v_m)$ is non-zero with overwhelming probability: the values $v_i$ guarantee that the values $y_i$ are perfectly hidden from the adversary and, as we show (Lemma 2.2), the leading coefficient of $Q$ is a non-zero polynomial evaluated at $y_1, \ldots, y_m$, values that are independent of the adversary's behavior. Schwartz-Zippel thus bounds the probability that the leading coefficient of $Q$ is zero, and thus, that $Q \equiv 0$. Since $Q(z) = P(\overrightarrow{x}) = 0$, we can factor the univariate polynomial $Q$ and find the DLog solution $z$, which is among its roots.

**Extensions.** We next extend our approach to a flexible (i.e., adaptive) version of the static Uber assumption, where the adversary can adaptively choose the polynomials (Sect. 3.2) as well as a generalization from polynomials to rational fractions (Sect. 3.3). We combine the flexible framework with the rational fraction framework in Sect. 3.4. After these generalizations, our framework covers assumptions such as strong Diffie-Hellman [BB08], where the adversary must compute a rational fraction of its own choice in the exponent.

In a next step (Sect. 3.5), we extend our framework to also cover gap-type assumptions such as Gap Diffie-Hellman (GDH) [OP01], which was recently proven equivalent to the DLog assumption in the AGM by Loss [Los19]. GDH states that the CDH assumption remains true even when the DDH assumption no longer holds. Informally, the idea of the proof given in [Los19] (first presented in [FKL18] for a restricted version of GDH) is to argue that the DDH oracle given to an algebraic adversary is useless, unless the adversary succeeds in breaking CDH during an oracle query. The reduction simulates the DDH oracle by always returning false. We generalize this to a broader class of assumptions, using a different simulation strategy, which avoids a security loss.

We also present (Sect. 3.6) an extension of our (adaptive) framework that allows to capture assumptions as strong as the LRSW assumption [LRSW99], which forms the basis of the Camenisch-Lysyanskaya signature scheme [CL04]. The LRSW assumption falls outside (even the adaptive version of) Boyen's Uber framework, since the adversary need not output the polynomial it is computing in the exponent.

The LRSW and GDH assumptions were previously studied in the AGM in the works of [FKL18, Los19], who gave very technical proofs spanning multiple pages of case distinctions. By comparison, our Uber Framework offers a more general and much simpler proof for both of these assumptions. Finally, we are able to prove all these results using *tight reductions*. This, in particular, improves upon the non-tight reduction of DLog to LRSW in [FKL18].

### Classifying Assumptions in our Framework

Finally, we prove two separation results that show the following:

**Separating $(q+1)$-DLog from $q$-DLog.** This shows that with respect to currently known (i.e., algebraic) reduction techniques, the Uber assumption, for increasing degrees of the polynomials, defines a natural hierarchy of assumptions in the AGM. More concretely, the $q$-lowest class within the established hierarchy consists of all assumptions that are covered by a specific instantiation of the Uber assumption which can be reduced from the $q$-DLog problem. Our separation result (Theorem 3.15) shows that there is no algebraic reduction from the $q$-DLog problem to the $(q+1)$-

DLog problem in the AGM. This implies that assumptions within different classes are separated with respect to algebraic reductions. Interestingly, we are even able to show our separation for reductions that can rewind and choose the random coins of the solver for the $(q+1)$-DLog problem freely.

**Separating OMDL from $q$-DLog.** Our second result (Theorem 3.17) shows a separation result between the one-more-DLog problem (OMDL) (where the adversary has to solve $q$ DLog instances and is given an oracle that computes discrete logarithms, which it can access $q-1$ times) and the $q$-DLog problem (for any $q$) in the AGM. Our result strengthens a previous result by Bresson, Monnerat, and Vergnaud [BMV08], who showed a separation between the discrete logarithm problem (i.e, where $q=1$) and the 2-one-more-DLog problem with respect to *black-box reductions*. By comparison, our result holds even in the AGM, where reductions are inherently non-black-box, as the AGM implicitly assumes an efficient extractor algorithm that extracts algebraic coefficients from the algebraic adversary. As the extractor is non-black-box (since it depends on the algebraic adversary), neither is any reduction that non-trivially leverages the AGM.

Our result clearly establishes the limits of our framework, as it excludes the OMDL family of assumptions. Unlike our first separation, this one comes with the caveat that it only applies to reductions that are "black-box in the AGM", meaning that they simply obtain the algebraic coefficients via the extractor, but cannot rewind the adversary or choose its random coins.

### Related Work

A long line of research has considered frameworks to capture general classes of assumptions. We give an overview of the most closely related works. The first Uber assumptions were introduced by Boyen et al. [BBG05, Boy08]. Others later gave alternative concepts to classify assumptions within cyclic groups. The works of Chase et al. [CM14, CMM16] study assumptions in bilinear groups of *composite* order, which are not considered in the original Uber framework. They show that several $q$-type assumptions are implied by (static) "subgroup-hiding" assumptions. This gives evidence that this type of assumption, which is specific to composite-order groups, is particularly strong.

More recently, Ghadafi and Groth [GG17] studied a broader class of assumptions in which the adversary must compute a group element from $\mathbb{G}_T$. Like our work, their framework applies to prime-order groups and extends to the case where the exponents can be described by rational fractions, and they also separate classes of assumptions from each other. However, their framework only deals with non-interactive assumptions, which do not cover the adaptive type of assumptions we study in our flexible variants (in fact, the authors mention extending their work to interactive assumptions as an open problem [GG17]). Their work does not cover assumptions such as GDH or LRSW, which we view as particularly interesting (and challenging) to classify. Indeed, our framework appears to be the first in this line of work that offers a classification comprising this type of assumptions.

A key difference is that Ghadafi and Groth's results are in the standard model whereas we work in the AGM. While this yields stronger results for reductions, their separations are weaker (in addition to separating less broad types of Uber assumptions), as they are with respect to generic reductions, whereas ours hold against algebraic reductions that can assume that the *adversary is algebraic*. Furthermore, their work considers *black-box reductions* that treat the underlying solver as an (imperfect) oracle, while we show the non-existence of reductions in the AGM, which are, by definition, non-black-box (see above). A final difference to the work of [GG17] lies in the tightness of all our reductions, whereas non of theirs are tight.

At CT-RSA'19 Mizuide, Takayasu, and Takagi [MTT19] studied static (i.e., non-flexible) variants and generalizations of the Diffie-Hellman problem in prime-order groups (also with extensions to the bilinear setting) by extending proofs from [FKL18] in the obvious manner. Most of their results

are special cases of our Uber assumption framework. Concretely, when restricting the degrees of all input polynomials to 1 in our static Uber Assumption, our non-triviality condition implies all corresponding theorems in their paper (except the ones relating to Matrix assumptions, which are outside the scope of this work). By our separation of $q$-DLog for different $q$, our results for higher degrees do not follow from theirs by currently known techniques. Finally, they do not cover the flexible (adaptive) variants nor oracle-enhanced- and hidden-polynomial-type assumptions (such as GDH and LRSW).

A further distinction that sets our work apart from these prior works is our formulation of the aforementioned 'hidden-type' assumptions, where we allow the adversary to solve the problem with respect to a group generator *of its own choice* instead of the one provided by the game. A recent work [BMZ19] shows that even in the GGM, allowing randomly chosen generators results in unexpected complications when proving lower bounds. Similarly, giving the adversary this additional freedom makes proving (and formalizing) our results more challenging. We also give this freedom to the reductions that we study (and prove impossible) in our separation results.

### Signatures on randomizable ciphertexts

**Our contribution.** Our aim was to construct a scheme of signatures on randomizable ciphertexts with a large message space and short signatures. But first we strengthen the notion of signature unforgeability. In SoRC, signatures are produced (and verified) on pairs of encryption keys and ciphertexts $(ek, c)$. In the original unforgeability notion [BFPV11] the adversary is given a signature verification key and a set of encryption keys $ek_1, \ldots, ek_n$ and can then make queries $(i, c)$ to get a signature for $(ek_i, c)$. Its goal is to return $(i^*, c^*)$ and a signature for $(ek_{i^*}, c^*)$, so that $c^*$ encrypts a message of which no encryption has been submitted to the signing oracle. Signatures thus authenticate plaintexts irrespective of the encryption key.

In more detail, once a query $(1, \mathsf{Enc}(ek_1, m))$ was made, a signature for $(ek_2, \mathsf{Enc}(ek_2, m))$ is *not* considered a forgery. In contrast, in our new definition (Def. 4.6), this is considered a forgery, since we view a signature as (obliviously) authenticating a message *for a particular encryption key*. That is, if from a signature on an encryption of a message for one key one can forge a signature on the same message for another key, this is considered a break of the scheme. A further difference is that, while in [BFPV11] encryption keys are generated by the challenger, we let the adversary choose (in any, possibly malicious, way) the encryption keys (in addition to the ciphertexts) on which it wishes to see a signature, as well as the key for its forgery.

We then construct a scheme which signs ElGamal ciphertexts and whose signatures consist of 4 elements of an (asymmetric) bilinear group (3 elements from $\mathbb{G}_1$ and 1 from $\mathbb{G}_2$). Our scheme (given in Fig. 4.3) is inspired by the original equivalence-class signature scheme [FHS19], whose equivalence classes only provide "selfless" anonymity. We show that signatures adapted to a randomization of a ciphertext are equivalently distributed to fresh signatures on the new ciphertext (Proposition 4.8). We then prove that our scheme satisfies our strengthened unforgeability notion in the generic group model (Theorem 4.9).

**Comparison with Blazy et al.** Apart from the stronger unforgeability notion we achieve, the main improvement of our scheme over [BFPV11] concerns its efficiency. The Blazy et al. scheme builds on (a new variant of) Waters signatures [Wat05] and Groth-Sahai proofs [GS08], which allows them to prove unforgeability from standard assumptions. However, encrypting and signing a $k$-bit message yields a ciphertext/signature pair consisting of $12 + 12k$ group elements of an asymmetric bilinear group. In our scheme, a message is a group element (as for ElGamal encryption), which lets us encode 128-bit messages (or messages of unbounded length by hashing into the group). A ciphertext/signature pair consists of 6 group elements. We also propose a generalization to messages of $n$ group elements for which a ciphertext/signature pair consists of $n + 5$ group elements.

The price we pay for this length reduction by a factor of over 250 (for 128-bit messages or longer) is an unforgeability proof in the generic group model. But, as we argue next, this is to be expected. Since we sign group elements and verification consists in checking pairing-product equations, our scheme is *structure-preserving* [AFG+10]. Signatures for such schemes must at least contain 3 group elements [AGHO11] and schemes with such short signatures cannot be proved from non-interactive (let alone standard) assumptions [AGO11]. Our 4-element signatures, which provide additional functionalities, and its unforgeability proof are therefore close to being optimal. We also note that a security reduction to computational hardness assumptions for schemes satisfying our unforgeability notion seems challenging, as the challenger cannot efficiently decide whether the adversary has won (in contrast to the weaker notion [BFPV11]).

### Transferable e-cash

**Our contribution.** Our contributions are two-fold:

**Security model.** We first revisit the formal model for transferable e-cash, departing from [BCFK15], whose model had itself already been a refined version of earlier ones. We start with giving a definition of correctness, which was lacking in previous works. Moreover, we discovered attacks against users following the protocol, against which previous models did not protect:

- When a user receives a coin (that is, the protocol accepts the received coin), then in previous models there is no guarantee that this coin will be accepted by other users when transferred. An adversary could thus send a mal-formed coin to a user, which the latter accepts but can then not spend.

- There are no guarantees for user against a malicious bank which at coin deposit refuses to credit the user's account (e.g., by claiming that the coin was invalid or had been double-spent). In our model, when the bank refuses a coin, it is forced to accuse a user of double-spending and exhibiting a proof for this.

We moreover simplify the anonymity definitions, which in earlier version had been cluttered with numerous oracles the adversary has access to, and for which the intuitive notion that they were formalizing was hard to grasp. While our definitions are simpler, they are stronger in that they imply previous definitions (except for the previous notion of "spend-then-receive (StR) anonymity", whose version we argue is not relevant in practice).

We also show that the proof of StR anonymity of the previous scheme [BCFK15] is flawed.

# Chapter 2

# Preliminaries

In this chapter, we introduce the notations and basic assumptions and primitives employed throughout this manuscript. We start by recalling some standard mathematical and computational notions, then we briefly introduce provable security. We also recall some well-known number-theoretic assumptions, to introduce the cryptographic primitives used throughout this work.

## 2.1 Notations

**Sets, integers, moduli, and associated rings and groups.** We denote real numbers by $\mathbb{R}$, integers by $\mathbb{Z}$ and non-negative integers by $\mathbb{N}$. If $a, b$ are two integers such that $a < b$, we denote the (closed) integer interval from $a$ to $b$ by $[\![a; b]\!]$.

If $q$ is a positive integer, we denote by $\mathbb{Z}_q$ the ring of integers modulo $q$.
In all of our constructions, the order of $\mathbb{Z}_q$ will be public. Therefore, elements of $\mathbb{Z}_q$ are represented as integers of the set $[\![0; q-1]\!]$. For an integer $x \in \mathbb{Z}$, $x \mod q$ is the reminder of the Euclidean division of $x$ by $q$. It can be seen both as an integer in $[\![0; q-1]\!]$ and as an element of $\mathbb{Z}_q$. Vectors are denoted by letters with arrows, like $\overrightarrow{v}$. We indicated a vector $\overrightarrow{v}$'s entry by $v_i$. We use $\overrightarrow{v}^T$ to denote the transpose of a vector $\overrightarrow{v}$.

**Random variables.** For a random variable $X$ over a probability space $(\Omega, \Sigma, \Pr)$, and a possible outcome $x$, we write $\Pr[X = x]$ to indicate the measure of the preimage of $\{x\} \in \Sigma$ under $\Pr$. We denote the action of sampling uniformly at random $x$ from $X$ with $x \leftarrow X$. Let us call *range* of a random variable $X : \Omega \to E$ the set of elements $x \in E$ for which the probability that $X$ has outcome $x$ is strictly positive. In this work, we refer only to random variables whose range is finite.

A random variable $X$ defined on a finite probability space $(\Omega, \Pr)$ is said to have the *uniform distribution* if $\Pr[X = x] = 1/|\operatorname{Im}(X)|$ where $\operatorname{Im}(X)$ denotes the image of $X$. Given a non-empty finite set $S$, we let $x \xleftarrow{\$} S$ denote the operation of sampling an element $x$ from $S$ uniformly at random.

**Asymptotics.** Given a function $f : \mathbb{N} \to \mathbb{R}$, the set $O(f)$ describes all functions that can be asymptotically upper-bounded by $f$, that is, all $g$ such that $\exists c, \lambda_0 \in \mathbb{N},\ 0 \leq g(\lambda) \leq cf(\lambda)$ for all $\lambda \geq \lambda_0$. With a slight abuse of notation, we write $g = O(f)$ to denote that $g \in O(f)$. The set $\tilde{O}(f)$ describes all functions that can be upper-bounded by $f$ ignoring logarithmic factors, that is, all $g$ such that $\exists c, k, \lambda_0 \in \mathbb{N},\ 0 \leq g(\lambda) \leq cf(\lambda)\log^k(\lambda)$ for all $\lambda \geq \lambda_0$. We use the notation $o(f)$ to identify all functions that can be asymptotically upper-bounded by $f$, where the upper-bound is strict. That is, all $g$ such that $\forall c\ \exists \lambda_0 \in \mathbb{N},\ 0 \leq g(\lambda) < cf(\lambda)$ for all $\lambda \geq \lambda_0$.

A function $\mu\colon \mathbb{N} \to [0, 1]$ is negligible (denoted $\mu = \mathsf{negl}(\lambda)$) if $\mu(\lambda) = o(\lambda^{-c})$ for any fixed constant $c$. That is, if for all $c \in \mathbb{N}$ there exists $\lambda_c \in \mathbb{N}$ such that $\mu(\lambda) < \lambda^{-c}$ for all $\lambda \geq \lambda_c$. A

function $\nu$ is *overwhelming* if $1 - \nu = \mathsf{negl}(\lambda)$. We let $\mathsf{poly}(\lambda)$ denote the set of polynomials in $\lambda$ (more precisely, functions upper-bounded by a polynomial in $\lambda$) with integer coefficients.

**Languages, machines, function families and complexity classes.** Languages are denoted in calligraphic, e.g. $\mathcal{L}$. We focus on languages whose alphabet is the set of bits $\{0,1\}$.

Algorithms are formalized as Turing Machines and denoted in serif, e.g. $\mathsf{M}$. We let $\mathsf{M.rl}(\lambda)$ be a *length function* (i.e., a function $\mathbb{N} \to \mathbb{N}$ polynomially bounded) in $\lambda$ defining the length of the randomness for a probabilistic interactive Turing Machine $\mathsf{M}$. By $y \coloneqq \mathsf{M}(x_1, \ldots; r)$ we denote the operation of running algorithm $\mathsf{M}$ on inputs $x_1, \ldots$ and coins $r \in \{0,1\}^{\mathsf{M.rl}(\lambda)}$ and letting $y$ denote the output. We see $\mathsf{M}$ both as a Turing Machine as well as a random variable. By $y \leftarrow \mathsf{M}(x_1, \ldots)$, we denote $y \coloneqq \mathsf{M}(x_1, \ldots; r)$ for random $r \in \mathsf{M.rl}(\lambda)$, and $[\mathsf{M}(x_1, \ldots)]$ the range of $\mathsf{M}$ on inputs $x_1, \ldots$. Unless otherwise specified, all the algorithms defined throughout this work are assumed to be probabilistic Turing machines that run in time $\mathsf{poly}(\lambda)$ – i.e., in $\mathsf{PPT}$. An adversary is denoted by $\mathcal{A}$; when it is interacting with an oracle $\mathcal{O}$, we write $\mathcal{A}^{\mathcal{O}}$. For two $\mathsf{PPT}$ machines $\mathsf{A}, \mathsf{B}$, with $(\mathsf{A} \| \mathsf{B})(x)$ we denote the execution of $\mathsf{A}$ followed by the execution of $\mathsf{B}$ on the same input $x$ and with the same random coins. The output of the two machines is concatenated and separated with a semicolon, e.g., $(out_{\mathsf{A}}; out_{\mathsf{B}}) \leftarrow (\mathsf{A} \| \mathsf{B})(x)$.

**Polynomials and rational fractions.** We denote polynomials by uppercase letters $P, Q$ and specify them by a list of their coefficients. If $m$ is an integer, we denote by $\mathbb{Z}_p[X_1, \ldots, X_m]$ the set of $m$-variate polynomials with coefficients in $\mathbb{Z}_p$ and by $\mathbb{Z}_p(X_1, \ldots, X_m)$ the set of rational fractions in $m$ variables with coefficients in $\mathbb{Z}_p$. We define the *total degree* of a polynomial $P(X_1, \ldots, X_m) = \sum_{\vec{i} \in \mathbb{N}^m} \lambda_{i_1,\ldots,i_m} \prod_{j=1}^{m} X_j^{i_j} \in \mathbb{Z}_p[X_1, \ldots, X_m]$ as $\max_{\vec{i} \in \mathbb{N}^m : \lambda_{i_1,\ldots,i_m} \not\equiv_p 0} \{\sum_{j=1}^{m} i_j\}$.

For the degree of rational fractions we will use the "French" definition [AW98]: for $(P, Q) \in \mathbb{Z}_p[X_1, \ldots, X_m] \times (\mathbb{Z}_p[X_1, \ldots, X_m] \setminus \{0\})$ we define

$$\deg \frac{P}{Q} := \deg P - \deg Q.$$

This definition has the following properties: The degree does not depend on the choice of the representative; it generalizes the definition for polynomials; and the following holds: $\deg(F_1 \cdot F_2) = \deg F_1 + \deg F_2$, and $\deg(F_1 + F_2) \leq \max\{\deg F_1, \deg F_2\}$.

**Schwartz-Zippel lemma.** We will use the following version of the Schwartz-Zippel lemma [DL77]:

**Lemma 2.1.** *Let $P \in \mathbb{Z}_p[X_1, \ldots, X_m]$ be a non-zero polynomial of total degree $d$. Let $r_1, \ldots, r_m$ be selected at random independently and uniformly from $\mathbb{Z}_p^*$. Then*

$$\Pr\left[P(r_1, \ldots, r_m) \equiv_p 0\right] \leq \frac{d}{p-1}.$$

We state the following technical lemma, which we will use in our reductions.

**Lemma 2.2.** *Let $P$ be a non-zero multivariate polynomial in $\mathbb{Z}_p[X_1, \ldots, X_m]$ of total degree $d$. Define $Q(Z) \in (\mathbb{Z}_p[Y_1, \ldots, Y_m, V_1, \ldots, V_m])[Z]$ as $Q(Z) := P(Y_1 Z + V_1, \ldots, Y_m Z + V_m)$. Then the coefficient of maximal degree of $Q$ is a polynomial in $\mathbb{Z}_p[Y_1, \ldots, Y_m]$ of degree $d$.*

*Proof.* $P$ is of the form $P(\overrightarrow{X}) = \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{i_1,\ldots,i_m}\prod_{j=1}^m X_j^{i_j}$ for coefficients $\lambda_{i_1,\ldots,i_m}$ and thus

$$\begin{aligned}
Q(Z) &= \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{i_1,\ldots,i_m}\prod_{j=1}^m (Y_j Z + V_j)^{i_j}\\
&= \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{\overrightarrow{i}}\prod_{j=1}^m \left(\textstyle\sum_{k=0}^{i_j}\binom{i_j}{k} Y_j^k\, Z^k\, V_j^{i_j-k}\right)\\
&= \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{\overrightarrow{i}}\sum_{k_1=0}^{i_1}\cdots\sum_{k_m=0}^{i_m}\prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j}\, Z^{k_j}\, V_j^{i_j-k_j}\\
&= \sum_{\ell=0}^{d}\ \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{\overrightarrow{i}}\sum_{\overrightarrow{k}\in\mathbb{N}^m\,:\,\Sigma_j k_j=\ell}\prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j}\, Z^{k_j}\, V_j^{i_j-k_j} = \textstyle\sum_{\ell=0}^d \lambda'_\ell\, Z^\ell\\
&\qquad\qquad \text{with}\quad \lambda'_\ell := \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{\overrightarrow{i}}\sum_{\overrightarrow{k}\in\mathbb{N}^m\,:\,\Sigma_j k_j=\ell}\prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j}\, V_j^{i_j-k_j}.
\end{aligned}$$

By assumption $P \not\equiv 0$. Thus for some $i_1,\ldots,i_m \ge 0$ with $\sum_j i_j = d$ we have $\lambda_{i_1,\ldots,i_m}\neq 0$, while $\lambda_{i_1,\ldots,i_m} = 0$ when $\sum_j i_j > d$. By the latter we have

$$\lambda'_d \;=\; \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j\le d} \lambda_{\overrightarrow{i}}\sum_{\overrightarrow{k}\,:\,\Sigma_j k_j=d}\prod_{j=1}^m \binom{i_j}{k_j} Y_j^{k_j}\, V_j^{i_j-k_j} \;=\; \sum_{\overrightarrow{i}\in\mathbb{N}^m\,:\,\Sigma_j i_j=d} \lambda_{\overrightarrow{i}}\prod_{j=1}^m Y_j^{i_j},$$

where the last step follows since $k_j \le i_j$ for all $j$ and $\sum_j i_j \le d$, $\sum_j k_j = d$ implies that $k_j = i_j$ for all $j$. Since $P$ is a polynomial of total degree $d$, for some $\overrightarrow{i}\in\mathbb{N}^m$ we have $\sum_j i_j = d$ and $\lambda_{\overrightarrow{i}}\neq 0$. We conclude that $\lambda'_d$ is a polynomial in $(Y_1,\ldots,Y_m)$ of total degree $d$. □

**Bilinear Groups.** We next state the definition of a *bilinear group*.

**Definition 2.3** (Bilinear group)**.** *A* bilinear group (description) *is a tuple* $\mathcal{G} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,\phi,\psi,p)$ *such that*

- $\mathbb{G}_i$ *is a cyclic group of prime order* $p$*, for* $i\in\{1,2,T\}$*;*
- $e$ *is a non-degenerate bilinear map* $e\colon \mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$*, that is, for all* $a,b\in\mathbb{Z}_p$ *and all generators* $g_1$ *of* $\mathbb{G}_1$ *and* $g_2$ *of* $\mathbb{G}_2$ *we have that* $g_T := e(g_1,g_2)$ *generates* $\mathbb{G}_T$ *and* $e(g_1^a,g_2^b) = e(g_1,g_2)^{ab} = g_T^{ab}$*;*
- $\phi$ *is an isomorphism* $\phi\colon\mathbb{G}_1\to\mathbb{G}_2$*, and* $\psi$ *is an isomorphism* $\psi\colon\mathbb{G}_2\to\mathbb{G}_1$*.*

*All group operations and the bilinear map* $e$ *must be efficiently computable.* $\mathcal{G}$ is of Type 1 *if the maps* $\phi$ *and* $\psi$ *are efficiently computable;* $\mathcal{G}$ is of Type 2 *if there is no efficiently computable map* $\phi$*; and* $\mathcal{G}$ is of Type 3 *if there are no efficiently computable maps* $\phi$ *and* $\psi$*. We require that there exist an efficient algorithm* $\mathsf{GenSamp}$ *that returns generators* $g_1$ *of* $\mathbb{G}_1$ *and* $g_2$ *of* $\mathbb{G}_2$*, so that* $g_2$ *is uniformly random, and (for Types 1 and 2)* $g_1 = \psi(g_2)$ *or (Type 3)* $g_1$ *is also uniformly random. By* $\mathsf{GenSamp}_i$ *we denote a restricted version that only returns* $g_i$*.*

We assume the existence of a probabilistic polynomial-time (p.p.t.) algorithm $\mathsf{BGGen}$ that takes as input an integer $\lambda$ in unary and outputs a description of an (asymmetric) bilinear group $(p,\mathbb{G},G,\hat{\mathbb{G}},\hat{G},\mathbb{G}_T,e)$ consisting of groups $(\mathbb{G},\cdot)$ and $(\hat{\mathbb{G}},\cdot)$, generated by $G$ and $\hat{G}$, resp., and $(\mathbb{G}_T,\cdot)$, all of cardinality a prime number $p\in\{2^\lambda,\ldots,2^{\lambda+1}\}$, and a bilinear map $e\colon\mathbb{G}\times\hat{\mathbb{G}}\to\mathbb{G}_T$, such that $e(G,\hat{G})$ generates $\mathbb{G}_T$, called *pairing*.

**(Algebraic) Security games.** We use a variant of (code-based) *security games* [BR04]. In game $\mathbf{G}_{\mathcal{G}}$ (defined relative to $\mathcal{G}$), an adversary $\mathsf{A}$ interacts with a challenger that answers oracle queries issued by $\mathsf{A}$. The game has a main procedure and (possibly zero) oracle procedures which describe how oracle queries are answered. We denote the output of a game $\mathbf{G}_{\mathcal{G}}$ between a challenger and an adversary $\mathsf{A}$ by $\mathbf{G}^{\mathsf{A}}_{\mathcal{G}}$. $\mathsf{A}$ is said to *win* if $\mathbf{G}^{\mathsf{A}}_{\mathcal{G}} = 1$. We define the *advantage* of $\mathsf{A}$ in $\mathbf{G}_{\mathcal{G}}$ as $\mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}} := \Pr\left[\mathbf{G}^{\mathsf{A}}_{\mathcal{G}} = 1\right]$ and the running time of $\mathbf{G}^{\mathsf{A}}_{\mathcal{G}}$ as $\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}}$. In this work, we are primarily concerned with *algebraic security games* $\mathbf{G}_{\mathcal{G}}$, in which we syntactically distinguish between elements of groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ (written in bold, uppercase letters, e.g., $\mathbf{Z}$) and all other elements, which must not depend on any group elements.

We next define algebraic algorithms. Intuitively, the only way for an algebraic algorithm to output a new group element $\mathbf{Z}$ is to derive it via group operations from known group elements.

**Definition 2.4** (Algebraic algorithm for bilinear groups)**.** *An algorithm $\mathsf{A}_{\mathsf{alg}}$ executed in an algebraic game $\mathbf{G}_{\mathcal{G}}$ is called* algebraic *if for all group elements $\mathbf{Z} \in \mathbb{G}$ (where $\mathbb{G} \in \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$) that $\mathsf{A}_{\mathsf{alg}}$ outputs, it additionally provides a representation in terms of received group elements in $\mathbb{G}$ and those from groups from which there is an efficient mapping to $\mathbb{G}$; in particular: if $\mathbf{U}_0, \ldots, \mathbf{U}_\ell \in \mathbb{G}_1$, $\mathbf{V}_0, \ldots, \mathbf{V}_m \in \mathbb{G}_2$ and $\mathbf{W}_0, \ldots, \mathbf{W}_t \in \mathbb{G}_T$ are the group elements received so far then $\mathsf{A}_{\mathsf{alg}}$ provides vectors $\vec{\mu}, \vec{\nu}, \vec{\zeta}, \vec{\eta}, \vec{\delta}$ and matrices $A = (\alpha_{i,j}), B = (\beta_{i,j}), \Gamma = (\gamma_{i,j})$ such that*

- $\mathbf{Z} \in \mathbb{G}_1$ *(Type 1 and 2):* $\mathbf{Z} = \prod_i \mathbf{U}_i^{\mu_i} \cdot \prod_i \psi(\mathbf{V}_i)^{\nu_i}$
  *(Type 3):* $\mathbf{Z} = \prod_i \mathbf{U}_i^{\mu_i}$
- $\mathbf{Z} \in \mathbb{G}_2$ *(Type 1):* $\mathbf{Z} = \prod_i \phi(\mathbf{U}_i)^{\zeta_i} \cdot \prod_i \mathbf{V}_i^{\eta_i}$
  *(Type 2 and 3):* $\mathbf{Z} = \prod_i \mathbf{V}_i^{\eta_i}$
- $\mathbf{Z} \in \mathbb{G}_T$*:* $\mathbf{Z} = \prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_i \prod_j e(\mathbf{U}_i, \phi(\mathbf{U}_j))^{\beta_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{\gamma_{i,j}} \cdot \prod_i \mathbf{W}_i^{\delta_i}$, *where $\beta_{i,j} = 0$ for Type 2 and $\beta_{i,j} = \gamma_{i,j} = 0$ for Type 3.*

We remark that oracle access to an algorithm $\mathsf{B}$ in the AGM includes any (usually non-black-box) access to $\mathsf{B}$ that is needed to extract the algebraic coefficients. Thus, our notion of black-box access in the AGM mainly rules out techniques such as rewinding $\mathsf{B}$ or running it on non-uniform random coins.

## 2.2 Security Model

**Generic group model.** *Generic algorithms* $\mathsf{A}_{\mathsf{gen}}$ are only allowed to use generic properties of a group. Informally, an algorithm is generic if it works regardless of what group it is run in. This is usually modeled by giving an algorithm indirect access to group elements via abstract handles. It is straight-forward to translate all of our algebraic games into games that are syntactically compatible with generic algorithms accessing group elements only via abstract handles. We measure the running times of generic algorithms as queries to an oracle that implements the abstract group operation, i.e., every query accounts for one step of the algorithm. We highlight this difference by denoting the running time of a generic algorithm with the letter $o$ rather than $t$. We say that winning algebraic game $\mathbf{G}_{\mathcal{G}}$ is $(\varepsilon, o)$*-hard in the generic group model* if for every generic algorithm $\mathsf{A}_{\mathsf{gen}}$ it holds that

$$\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq o \implies \mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq \varepsilon.$$

As all of our reductions run the adversary only once and without rewinding, the overhead in the running time of our reductions is *additive* only. We make the reasonable assumption that,

compared to the running time of the adversary, this is typically small, and therefore ignore the losses in the running times for this work in order to keep notational overhead low.

We assume that a generic algorithm $\mathsf{A}_{\mathsf{gen}}$ provides the representation of $\mathbf{Z}$ relative to all previously received group elements, for all group elements $\mathbf{Z}$ that it outputs. This assumption is w.l.o.g. since a generic algorithm can only obtain new group elements by querying two known group elements to the generic group oracle; hence a reduction can always extract a valid representation of a group element output by a generic algorithm. This way, every generic algorithm is also an algebraic algorithm.

Furthermore, if $\mathsf{B}_{\mathsf{gen}}$ is a generic oracle algorithm and $\mathsf{A}_{\mathsf{alg}}$ is an algebraic algorithm, then $\mathsf{B}_{\mathsf{alg}} := \mathsf{B}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ is also an algebraic algorithm. We refer to [Mau05] for more on generic algorithms.

**Security Reductions.** All our security reductions are (bilinear) generic algorithms, which allows us to compose all of our reductions with hardness bounds in the (bilinear) generic group model (see next paragraph). Let $\mathbf{G}_{\mathcal{G}}, \mathbf{H}_{\mathcal{G}}$ be security games. We say that algorithm $\mathsf{R}_{\mathsf{gen}}$ is a *generic* $(\Delta_\varepsilon^{(\cdot)}, \Delta_\varepsilon^{(+)}, \Delta_o^{(\cdot)}, \Delta_o^{(+)})$*-reduction from* $\mathbf{H}_{\mathcal{G}}$ *to* $\mathbf{G}_{\mathcal{G}}$ if $\mathsf{R}_{\mathsf{gen}}$ is generic and if for every algebraic algorithm $\mathsf{A}_{\mathsf{alg}}$, algorithm $\mathsf{B}_{\mathsf{alg}}$ defined as $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ satisfies

$$\begin{aligned}\mathbf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} &\geq \frac{1}{\Delta_\varepsilon^{(\cdot)}} \cdot \left(\mathbf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \Delta_\varepsilon^{(+)}\right),\\ \mathbf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} &\leq \Delta_o^{(\cdot)} \cdot \left(\mathbf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} + \Delta_o^{(+)}\right).\end{aligned}$$

Furthermore, for simplicity of notation, we will make the convention of referring to $(1, \Delta_\varepsilon, 1, \Delta_o)$-reductions as $(\Delta_\varepsilon, \Delta_o)$-reductions.

**Composing information-theoretic lower bounds with reductions in the AGM.** The following lemma from [Los19] explains how statements in the AGM compose with bounds from the GGM.

**Lemma 2.5.** *Let* $\mathbf{G}_{\mathcal{G}}$ *and* $\mathbf{H}_{\mathcal{G}}$ *be algebraic security games and let* $\mathsf{R}_{\mathsf{gen}}$ *be a generic* $(\Delta_\varepsilon^{(\cdot)}, \Delta_\varepsilon^{(+)}, \Delta_o^{(\cdot)}, \Delta_o^{(+)})$ *-reduction from* $\mathbf{H}_{\mathcal{G}}$ *to* $\mathbf{G}_{\mathcal{G}}$*. If* $\mathbf{H}_{\mathcal{G}}$ *is* $(\varepsilon, o)$*-secure in the GGM, then* $\mathbf{G}_{\mathcal{G}}$ *is* $(\varepsilon', o')$*-secure in the GGM where*

$$\varepsilon' = \varepsilon \cdot \Delta_\varepsilon^{(\cdot)} + \Delta_\varepsilon^{(+)}, \quad o' = o/\Delta_o^{(\cdot)} - \Delta_o^{(+)}.$$

## 2.3 Computational assumptions

**Definition 2.6** (SXDH)**.** *The* Symmetric External Diffie-Hellman Assumption *states that given* $(g^r, g^s, g^t)$ *for random* $r, s \in \mathbb{Z}_p$*, it is hard to decide whether* $t = rs$ *or* $t$ *is random; moreover, given* $(\hat{g}^{r'}, \hat{g}^{s'}, \hat{g}^{t'})$ *for random* $r', s' \in \mathbb{Z}_p$*, it is hard to decide whether* $t' = r's'$ *or* $t'$ *is random.*

The two following assumptions have been introduced in [AFG+10]. We call it the Asymetric Double Hidden Strong Diffie Hellman (ADHSDH) assumption and the Asymetric Weak Flexible Computational Diffie Hellman (AWFCDH) assumption.

**Definition 2.7** ($q$-ADHSDH)**.** *Given* $\lambda \in \mathbb{Z}_p$ *and* $(g, f, k, x = g^\lambda, \hat{g}, \hat{y} = \hat{g}^\lambda) \leftarrow \mathbb{G}^4 \times \hat{\mathbb{G}}^2$*,* $(a_i = (kg^{\beta_i})^{\frac{1}{\lambda+\alpha_i}}, b_i = f^{\alpha_i}, v_i = g^{\beta_i}, \hat{d}_i = \hat{g}^{\alpha_i}, \hat{w}_i = \hat{g}^{\beta_i})_{i=1}^q$*, for* $c_i, v_i \xleftarrow{\$} \mathbb{Z}_p$*, it is hard to output a new tuple* $(a, b, v, \hat{d}, \hat{w}) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^2$ *of this form, i.e., a tuple that satisfies*

$$e(a, \hat{y}\hat{d}) = e(kv, \hat{g}) \ \wedge \ e(b, \hat{g}) = e(f, \hat{d}) \ \wedge \ e(v, \hat{g}) = e(g, \hat{w}).$$

| $q$-**dlog** $^{\mathsf{A}}_{\mathcal{G}_i}$ | $(q_1, q_2)$-**dlog** $^{\mathsf{A}}_{\mathcal{G}}$ |
|---|---|
| 01 $g \xleftarrow{\$} \mathsf{GenSamp}_i$ | 01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ |
| 02 $z \xleftarrow{\$} \mathbb{Z}_p^*$ | 02 $z \xleftarrow{\$} \mathbb{Z}_p^*$ |
| 03 $z^* \xleftarrow{\$} \mathsf{A}(g, g^z, g^{z^2}, \ldots, g^{z^q})$ | 03 $z^* \xleftarrow{\$} \mathsf{A}(g_1, g_1^z, g_1^{z^2}, \ldots, g_1^{z^{q_1}}, g_2, g_2^z, \ldots, g_2^{z^{q_2}})$ |
| 04 Return $(z^* = z)$ | 04 Return $(z^* = z)$ |

Figure 2.1: $q$-discrete logarithm game $q$-**dlog** $_{\mathcal{G}_i}$ (left) and $(q_1, q_2)$-discrete logarithm game $(q_1, q_2)$-**dlog** $_{\mathcal{G}}$ (right) relative to group $\mathcal{G}_i, i \in \{1, 2\}$ and $\mathcal{G}$, resp., and adversary $\mathsf{A}$.

**Definition 2.8** (AWFCDH)**.** *Given random generators $g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}}$, and $a = g^\alpha$ for $\alpha \leftarrow \mathbb{Z}_p$, it is hard to output $(g^\nu, g^{\nu\alpha}, \hat{g}^\nu, \hat{g}^{\nu\alpha})$, i.e., a tuple $(r, m, \hat{s}, \hat{n})$ that satisfies*

$$e(a, \hat{s}) = e(m, \hat{g}) \ \wedge \ e(m, \hat{g}) = e(g, \hat{n}) \ \wedge \ e(r, \hat{g}) = e(g, \hat{s}).$$

**The $q$-discrete logarithm assumption and variants.** For this work, we consider two generalizations of the DLog assumption, which are parametrized (i.e., "$q$-type") variants of the DLog assumption. We describe them via the algebraic security games $q$-**dlog** $_{\mathcal{G}_i}$ and $(q_1, q_2)$-**dlog** $_{\mathcal{G}}$ in Fig. 2.1.

The following lemma, which follows similarly to the generic security of $q$-SDH [BB08], was proved (asymptotically) by Lipmaa [Lip12]. For completeness, we give a concrete proof in Section 2.5.

**Lemma 2.9.** *Let $o, q_1, q_2 \in \mathbb{N}$, let $q := \max\{q_1, q_2\}$. Then $q$-DLog and $(q_1, q_2)$-DLog are $(\frac{1+(o+q+1)^2 q}{p-1}, o)$-secure in the bilinear generic group model.*

We remark that all though our composition results are stated in the bilinear GGM, it is straight forward to translate them to the standard GGM if the associated hardness assumption is stated over a pairing-free group. This is true, because in those cases, our reductions will also be pairing-free and hence are standard generic algorithms themselves.

## 2.4 Primitives used

### Bilinear groups

The building blocks of our scheme will be defined over a (Type-3, i.e., asymmetric) bilinear group, which is a tuple $Gr = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, \hat{g})$, where $\mathbb{G}, \hat{\mathbb{G}}$ and $\mathbb{G}_T$ are groups of prime order $p$; $\langle g \rangle = \mathbb{G}$, $\langle \hat{g} \rangle = \hat{\mathbb{G}}$, and $e\colon \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is a bilinear map (i.e., for all $a, b \in \mathbb{Z}_p$: $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab}$) so that $e(g, \hat{g})$ generates $\mathbb{G}_T$. We assume that the groups are discrete-log-hard and other computational problems (DDH, CDH, SXDH, etc. defined in Section 2.3) are infeasible as well. We assume that there exists an algorithm $\mathsf{BGGen}$ that, on input the security parameter $\lambda$ in unary, outputs the description of a bilinear group with $p \geq 2^{\lambda-1}$.

### Randomizable proofs of knowledge and signatures

#### Commit-and-prove proof systems

As coins must be unforgeable, at their core lie digital signatures. To achieve anonymity, these must be hidden, which can be achieved via non-interactive zero-knowledge (NIZK) proofs of knowledge; if these proofs are *re-randomizable*, then they can not even be recognized by a past owner. We will

use Groth-Sahai NIZK proofs [GS08], which are randomizable [BCC+09] and include commitments to the witnesses.

We let $\mathcal{V}$ be set of values that can be committed, $\mathcal{C}$ be the set of commitments, $\mathcal{R}$ the randomness space and $\mathcal{E}$ the set of equations (containing equality) whose satisfiability can be proved. We assume that $\mathcal{V}$ and $\mathcal{R}$ are groups. We will use an extractable commitment scheme, which consists of the following algorithms:

$\mathsf{C.Setup}(Gr)$ takes as input a description of a bilinear group and returns a commitment key $ck$, which implicitly defines the sets $\mathcal{V}, \mathcal{C}, \mathcal{R}$ and $\mathcal{E}$.

$\mathsf{C.ExSetup}(Gr)$ returns an extraction key $xk$ in addition to a commitment key $ck$.

$\mathsf{C.SmSetup}(Gr)$ returns a commitment key $ck$ and a simulation trapdoor $td$.

$\mathsf{C.Cm}(ck, v, \rho)$, on input a key $ck$, a value $v \in \mathcal{V}$ and randomness $\rho \in \mathcal{R}$, returns a commitment in $\mathcal{C}$.

$\mathsf{C.ZCm}(ck, \rho)$, used when simulating proofs, is defined as $\mathsf{C.Cm}(ck, 0_{\mathcal{V}}, \rho)$.

$\mathsf{C.RdCm}(ck, c, \rho)$ randomizes a commitment $c$ to a fresh $c'$ using randomness $\rho$.

$\mathsf{C.Extr}(xk, c)$, on input extraction key $xk$ and a commitment $c$, outputs a value in $\mathcal{V}$. (This is the only algorithm that might not be polynomial-time.)

We extend $\mathsf{C.Cm}$ to vectors in $\mathcal{V}^n$: for $M = (v_1, \ldots, v_n)$ and $\rho = (\rho_1, \ldots, \rho_n)$ then we define $\mathsf{C.Cm}(ck, M, \rho) := (\mathsf{C.Cm}(ck, v_1, \rho_1), \ldots, \mathsf{C.Cm}(ck, v_n, \rho_n))$ and likewise $\mathsf{C.Extr}(xk, (c_1, \ldots, c_n)) := (\mathsf{C.Extr}(xk, c_1), \ldots, \mathsf{C.Extr}(xk, c_n))$.

We now define a NIZK proof system that proves that committed values satisfy a given equation from $\mathcal{E}$. Given a proof for commitments, the proof can be adapted to a randomization (via $\mathsf{C.RdCm}$) of the commitments using $\mathsf{C.AdptPrf}$.

$\mathsf{C.Prv}(ck, E, (v_1, \rho_1), \ldots, (v_n, \rho_n))$, on input a commitment key $ck$, a set of equations $E \subseteq \mathcal{E}$ and vectors $(v_1, \ldots, v_n)$ of values and $(\rho_1, \ldots, \rho_n)$ of randomness, outputs a proof $\pi$.

$\mathsf{C.Verify}(ck, E, c_1, \ldots, c_n, \pi)$, on input a commitment key $ck$, a set of equations in $\mathcal{E}$, a commitment vector $(c_1, \ldots, c_n)$, and a proof $\pi$, outputs a bit $b$.

$\mathsf{C.AdptPrf}(ck, E, c_1, \rho_1, \ldots, c_n, \rho_n, \pi)$, on input a set of equations, commitments $(c_1, \ldots, c_n)$, randomness $(\rho_1, \ldots, \rho_n)$ and a proof $\pi$, outputs a proof $\pi'$.

$\mathsf{C.SmPrv}(td, E, \rho_1, \ldots, \rho_n)$, on input the simulation trapdoor, an equation $E$ with $n$ variables and randomness, outputs a proof $\pi$.

#### $\mathcal{M}$-structure-preserving signatures

To prove knowledge of signatures, we require a scheme that is compatible with Groth-Sahai proofs [AFG+10].

$\mathsf{S.Setup}(Gr)$, on input the bilinear group description, outputs signature parameters $par_S$, defining a message space $\mathcal{M}$. We require $\mathcal{M} \subseteq \mathcal{V}^n$ for some $n$.

$\mathsf{S.KeyGen}(par_S)$, on input the parameters $par_S$, outputs the signing and the verification key $(sk, vk)$. We require that $vk$ is composed of values in $\mathcal{V}$.

$\mathsf{S.Sign}(sk, M)$, on input a signing key $sk$ and a message $M \in \mathcal{M}$, outputs a signature $\Sigma$. We require that $\Sigma$ is composed of values in $\mathcal{V}$.

$\mathsf{S.Verify}(vk, M, \Sigma)$, on input a verification key $vk$, a message $M$ and a signature $\Sigma$, outputs a bit $b$. We require that $\mathsf{S.Verify}$ proceeds by evaluating equations from $\mathcal{E}$ (which we denote by $E_{\mathsf{S.Verify}(\cdot,\cdot,\cdot)}$).

### $\mathcal{M}$-commuting signatures

As in a previous instantiation of transferable e-cash [BCF+11], we will use commuting signatures [Fuc11], which let the signer, given a commitment to a message, produce a commitment to a signature on that message, together with a proof, via the following functionality:

$\mathsf{SigCm}(ck, sk, c)$, on input a signing key $sk$ and a commitment $c$ of a message $M \in \mathcal{M}$, outputs a committed signature $c_\Sigma$ and a proof $\pi$ that the signature in $c_\Sigma$ is valid on the value in $c$, i.e., the committed values satisfy $\mathsf{S.Verify}(vk, \cdot, \cdot)$.

$\mathsf{SmSigCm}(xk, vk, c, \Sigma)$, on input the an extraction key $xk$, a verification key $vk$, a commitment $c$ and a signature $\Sigma$, outputs a committed signature $c_\Sigma$ and a proof $\pi$ of validity for $c_\Sigma$ and $c$.

**Correctness and soundness properties.** We require the following properties of commitments, proofs and signatures, when the setup algorithms are run on any output $Gr \leftarrow \mathsf{BGGen}(1^\lambda)$ for any $\lambda \in \mathbb{N}$:

*Perfectly binding commitments:* $\mathsf{C.Setup}$ and the first output of $\mathsf{C.ExSetup}$ are distributed equivalently. Let $(ck, xk) \leftarrow \mathsf{C.ExSetup}$; then for every $c \in \mathcal{C}$ there exists exactly one $v \in \mathcal{V}$ such that $c = \mathsf{C.Cm}(ck, v, \rho)$ for some $\rho \in \mathcal{R}$. Moreover, $\mathsf{C.Extr}(xk, c)$ extracts that value $v$.

*$\mathcal{V}'$-extractability:* We require that committed values from a subset $\mathcal{V}' \subseteq \mathcal{V}$ can be efficiently extracted. Let $(ck, xk) \leftarrow \mathsf{C.ExSetup}$; then $\mathsf{C.Extr}(xk, \cdot)$ is efficient on all values $\mathsf{C.Cm}(ck, v, \rho)$ for $v \in \mathcal{V}'$ and $\rho \in \mathcal{R}$.

*Proof completeness:* Let $ck \leftarrow \mathsf{C.Setup}$; then for all $(v_1, \ldots, v_n) \in \mathcal{V}^n$ satisfying $E \subseteq \mathcal{E}$, and $(\rho_1, \ldots, \rho_n) \in \mathcal{R}^n$ and $\pi \leftarrow \mathsf{C.Prv}(ck, E, (v_1, \rho_1), \ldots, (v_n, \rho_n))$ we have

$$\mathsf{C.Verify}(ck, E, \mathsf{C.Cm}(ck, v_1, \rho_1), \ldots, \mathsf{C.Cm}(ck, v_n, \rho_n), \pi) = 1.$$

*Proof soundness:* Let $(ck, xk) \leftarrow \mathsf{C.ExSetup}$, $E \subseteq \mathcal{E}$, and $(c_1, \ldots, c_n) \in \mathcal{C}^n$.

If $\mathsf{C.Verify}(ck, E, c_1, \ldots, c_n, \pi) = 1$ for some $\pi$, then letting $v_i := \mathsf{C.Extr}(xk, c_i)$, we have that $(v_1, \ldots, v_n)$ satisfy $E$.

*Randomizability:* Let $ck \leftarrow \mathsf{C.Setup}$, and $E \in \mathcal{E}$; then for all $(v_1, \ldots, v_n) \in \mathcal{V}^n$ that satisfy $E$ and $\rho_1, \rho'_1, \ldots, \rho_n, \rho'_n \in \mathcal{R}$ the following two are distributed equivalently:

$$\begin{aligned}
&(\mathsf{C.RdCm}(\mathsf{C.Cm}(ck, v_1, \rho_1), \rho'_1), \ldots, \mathsf{C.RdCm}(\mathsf{C.Cm}(ck, v_n, \rho_n), \rho'_n),\\
&\qquad \mathsf{C.AdptPrf}(ck, \mathsf{C.Prv}(ck, E, (v_1, \rho_1), \ldots, (v_n, \rho_n))), \rho'_1, \ldots, \rho'_n) \text{ and}\\
&(\mathsf{C.Cm}(ck, v_1, \rho_1 + \rho'_1), \ldots, \mathsf{C.Cm}(ck, v_n, \rho_n + \rho'_n),\\
&\qquad \mathsf{C.Prv}(ck, E, (v_1, \rho_1 + \rho'_1), \ldots, (v_n, \rho_n + \rho'_n))).
\end{aligned}$$

*Signature correctness:* Let $(sk, vk) \leftarrow \mathsf{S.KeyGen}(\mathsf{S.Setup})$ and $M \in \mathcal{M}$;

then we have $\mathsf{S.Verify}(vk, M, \mathsf{S.Sign}(sk, M)) = 1$.

*Correctness of signing committed messages:* Let $(ck, xk) \leftarrow \mathsf{C.ExSetup}$ and $(sk, vk) \leftarrow \mathsf{S.KeyGen}$, and let $M \in \mathcal{M}$; if $\rho, \rho' \xleftarrow{\$} \mathcal{R}$, then the following three are distributed equivalently:

$$(\mathsf{C.Cm}(ck, \mathsf{S.Sign}(sk, M), \rho'),\ \mathsf{C.Prv}(ck, E_{\mathsf{S.Verify}(vk,\cdot,\cdot)}, (M, \rho), (\Sigma, \rho'))) \quad \text{and}$$
$$\mathsf{SigCm}(ck, sk, \mathsf{C.Cm}(ck, M, \rho)) \quad \text{and}$$
$$\mathsf{SmSigCm}(xk, vk, \mathsf{C.Cm}(ck, M, \rho), \mathsf{S.Sign}(sk, M)).$$

The first equality also holds for $ck \leftarrow \mathsf{C.Setup}$, since it is distributed like $ck$ output by $\mathsf{C.Setup}$.

**Security properties**

*Mode indistinguishability:* Let $Gr \leftarrow \mathsf{BGGen}(1^\lambda)$; then the outputs of $\mathsf{C.Setup}(Gr)$ and the first output of $\mathsf{C.SmSetup}(Gr)$ are computationally indistinguishable.

*Perfect zero-knowledge in hiding mode:* Let $(ck, td) \leftarrow \mathsf{C.SmSetup}(Gr)$, $E \subseteq \mathcal{E}$ and $v_1, \ldots, v_n \in \mathcal{V}$ such that $E(v_1, \ldots, v_n) = 1$. For $\rho_1, \ldots, \rho_n \xleftarrow{\$} \mathcal{R}$ the following are equivalently distributed:

$$(\mathsf{C.Cm}(ck, v_1, \rho_1), \ldots, \mathsf{C.Cm}(ck, v_n, \rho_n), \mathsf{C.Prv}(ck, E, (v_1, \rho_1), \ldots, (v_n, \rho_n)))$$
$$\text{and} \quad (\mathsf{C.ZCm}(ck, \rho_1), \ldots, \mathsf{C.ZCm}(ck, \rho_n), \mathsf{C.SmPrv}(td, E, \rho_1, \ldots, \rho_n)).$$

*Signature unforgeability (under chosen message attack):* No PPT adversary that is given $vk$ output by $\mathsf{S.KeyGen}$ and an oracle for adaptive signing queries on messages $M_1, M_2, \ldots$ of its choice can output a pair $(M, \Sigma)$, such that $\mathsf{S.Verify}(vk, M, \Sigma) = 1$ and $M \notin \{M_1, M_2, \ldots\}$.

### Rerandomizable encryption schemes

In order to trace double-spenders, some information must be retrievable from the coin by the bank, so we encrypt it. Since coins must change appearance in order to achieve coin transparency (Def. 5.4), we must use rerandomizable encryption. In our e-cash scheme we will prove consistency of encrypted messages with values used elsewhere, and to produce such a proof, knowledge of part of the randomness is required; we therefore make this an explicit input of some algorithms, which thus are still probabilistic.

A rerandomizable encryption scheme consists of 4 poly-time algorithms:

$\mathsf{E.KeyGen}(Gr)$, on input the group description, outputs an encryption key $ek$ and a corresponding decryption key $dk$.

$\mathsf{E.Enc}(ek, M, \nu)$ is probabilistic and on input an encryption key $ek$, a message $M$ and (partial) randomness $\nu$ outputs a ciphertext.

$\mathsf{E.ReRand}(ek, C, \nu')$, on input an encryption key, a ciphertext and some randomness, outputs a new ciphertext. If no randomness is explicitly given to $\mathsf{E.Enc}$ or $\mathsf{E.ReRand}$ then it is assumed to be picked uniformly.

$\mathsf{E.Dec}(dk, C)$, on input a decryption key and a ciphertext, outputs either a message or $\perp$ indicating an error.

In order to prove statements about encrypted messages, we add two functionalities: $\mathsf{E.Verify}$ lets one verify that a ciphertext encrypts a given message $M$, for which it is also given partial randomness $\nu$. This will allow us to prove that a commitment $c_M$ and a ciphertext $C$ contain the same message. For this, we require that the equations defining $\mathsf{E.Verify}$ are in the set $\mathcal{E}$ supported by $\mathsf{C.Prv}$.

This lets us define an equality proof $\tilde{\pi} = (\pi, c_\nu)$, where $c_\nu$ is a commitment of the randomness $\nu$, and $\pi$ a proof that the values $c_M$ and $c_\nu$ verify the equations of $\mathsf{E.Verify}(ek, \cdot, \cdot, C)$. In order to support rerandomization of ciphertexts, we define a functionality $\mathsf{E.AdptPrf}$, which adapts a proof $(\pi, c_\nu)$ to a rerandomization.

$\mathsf{E.Verify}(ek, M, \nu, C)$, on input an encryption key, a message, randomness and a ciphertext, outputs a bit.

$\mathsf{E.AdptPrf}(ck, ek, c_M, C, \tilde{\pi} = (\pi, c_\nu), \nu')$, a probabilistic algorithm which, on input a commitment key, an encryption key, a commitment, a ciphertext, an equality proof (i.e., a proof and a commitment) and randomness, outputs a new equality proof.

**Correctness properties**

We require the scheme to satisfy the following correctness properties for all key pairs $(ek, dk) \leftarrow \mathsf{E.KeyGen}(Gr)$:

- For all $M \in \mathcal{M}$ and randomness $\nu$ we have: $\mathsf{E.Enc}(ek, M, \nu) = C$ if and only if $\mathsf{E.Verify}(ek, M, \nu, C) = 1$.
- For all $M \in \mathcal{M}$ and $\nu$: $\mathsf{E.Verify}(ek, M, \nu, C) = 1$ implies $\mathsf{E.Dec}(dk, C) = M$. (These two notions imply the standard correctness notion.)
- For all $M \in \mathcal{M}$ and randomness $\nu, \nu'$, if $C \leftarrow \mathsf{E.Enc}(ek, M, \nu)$ then the following are equally distributed: $\mathsf{E.ReRand}(ek, C, \nu')$ and $\mathsf{E.Enc}(ek, M, \nu + \nu')$.
- For every $ck \leftarrow \mathsf{C.Setup}$, every $(ek, dk) \leftarrow \mathsf{E.KeyGen}$, $M \in \mathcal{M}$ and randomness $\nu, \nu', \rho_M, \rho_\nu$, if we let

$$\begin{aligned} c_M &\leftarrow \mathsf{C.Cm}(ck, M, \rho_M) & C &\leftarrow \mathsf{E.Enc}(ek, M, \nu) \\ c_\nu &\leftarrow \mathsf{C.Cm}(ck, \nu, \rho_\nu) & \pi &\leftarrow \mathsf{C.Prv}(ck, \mathsf{E.Verify}(ek, \cdot, \cdot, C), (M, \rho_M), (\nu, \rho_\nu)) \end{aligned}$$

  then the following are equivalently distributed:

$$\begin{aligned} &\mathsf{E.AdptPrf}(ck, ek, c_M, \mathsf{ReRand}(pk, C, \nu'), (\pi, c_\nu), \nu') \qquad \text{and} \\ &(\mathsf{C.Prv}(ck, \mathsf{E.Verify}(ek, \cdot, \cdot, \mathsf{ReRand}(ek, C, \nu')), (M, \rho_M), (\nu + \nu', \rho_\nu)), c_\nu). \end{aligned}$$

**Security properties**

We require two properties from rerandomizable encryption: the first one is the standard (strongest possible) variant of CCA security; the second one is a new notion, which is easier to achieve.

**Replayable-CCA (RCCA) security.** We use the definition from Canetti et al. [CKN03], formalized in Fig. 2.2.

**Indistinguishability of adversarially chosen and randomized ciphertexts (IACR).** An adversary that is given a public key, chooses two ciphertexts and is then given the randomization of one of them cannot, except with a negligible advantage, distinguish which one it was given. The game is formalized in Fig. 2.2.

**Definition 2.10.** *For* $\mathtt{x} \in \{\mathtt{RCCA}, \mathtt{IACR}\}$*, a rerandomizable encryption scheme is* x-secure *if* $\Pr[\mathbf{Expt}^{\mathtt{x}}_{\mathcal{A},1}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\mathtt{x}}_{\mathcal{A},0}(\lambda) = 1]$ *is negligible in* $\lambda$ *for any PPT* $\mathcal{A}$*.*

$\mathbf{Expt}^{\mathtt{RCCA}}_{\mathcal{A},b}(\lambda)$:
$(ek, dk) \leftarrow \mathsf{E.KeyGen}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Dec}(dk,\cdot)}(ek)$
$C \leftarrow \mathsf{E.Enc}(ek, m_b)$
$b' \leftarrow \mathcal{A}^{\mathtt{GDec}(\cdot)}(C)$
Return $b'$

$\mathtt{GDec}(C)$:
$m \leftarrow \mathsf{E.Dec}(dk, C)$
If $m \notin \{m_0, m_1\}$
Return $m$
Else return `replay`

$\mathbf{Expt}^{\mathtt{IACR}}_{\mathcal{A},b}(\lambda)$:
$(ek, dk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(C_0, C_1) \leftarrow \mathcal{A}(ek)$
$C \leftarrow \mathsf{E.ReRand}(ek, C_b)$
$b' \leftarrow \mathcal{A}(ek, C)$
Return $b'$

Figure 2.2: Security games for rerandomizable encryption schemes

### Double-spending tag schemes

We follow the general approach from [BCFK15], in which the bank represents a coin in terms of its *serial number* $sn = sn_0 \| \ldots \| sn_k$, which grows with every transfer. In addition, a coin contains a tag $tag = tag_1 \| \ldots \| tag_k$, which enables tracing of double-spenders. The part $sn_i$ is chosen by a user when she receives the coin, while the tag $tag_i$ is computed by the sender as a function of $sn_{i-1}$ and $sn_i$ and her secret key.

Baldimtsi et al. [BCFK15] show how to construct such tags in a way so they perfectly hide the user's identity, except when a user computes two tags with the same $sn_{i-1}$ but different values $sn_i$, in which case her identity can be computed from the two tags. Note that this precisely corresponds to double-spending the coin that ends in $sn_{i-1}$ to two users that choose different values for $sn_i$ when receiving it.

We use the construction of [BCFK15], which we first formally define and then show that its full strength had not been leveraged yet: in particular, we realize that the double-spending tag can also be used as method for users to *authenticate* the coin transfer. In earlier works [BCF+11, BCFK15], at each transfer the spender computed a signature that was included in a coin, and that committed the user to the spending (and made her accountable in case of double spending). Our construction *does not require any user signatures* and thus gains in efficiency.

Furthermore, in [BCFK15] (there were no tags in [BCF+11]), the malleable signatures took care of ensuring well-formedness of the tags, while we give an explicit construction. To be compatible with Groth-Sahai proofs, we define structure-preserving proofs of well-formedness for serial numbers and tags.

**Syntax.** An $\mathcal{M}$-double-spending tag scheme is composed of the following 10 polynomial-time algorithms:

$\mathsf{T.Setup}(Gr)$, on input a group description, outputs the parameters $par_{\mathsf{T}}$ (which is an implicit input to all of the following).

$\mathsf{T.KeyGen}(par_{\mathsf{T}})$, on input the parameters, outputs a tag key pair $(sk, pk)$.

$\mathsf{T.SGen}(sk, n)$, the serial-number generation function, on input a secret key and a nonce $n \in \mathcal{N}$ (the nonce space), outputs a serial-number component $sn$ and a proof $sn\text{-}pf$ of well-formedness.

$\mathsf{T.SGen}_{\mathrm{init}}(sk, n)$ is a variant of $\mathsf{T.SGen}$ that also outputs a message $M \in \mathcal{M}$. ($\mathsf{SGen}_{\mathrm{init}}$ is used for the first component of the serial number, which is signed by the bank using a signature scheme that requires messages to be in $\mathcal{M}$.)

$\mathsf{T.SVfy}(pk, sn, sn\text{-}pf)$, on input a public key, a serial number and a proof verifies that $sn$ is consistent with $pk$ by outputting a bit $b$.

$\mathsf{T.SVfy}_{\mathrm{init}}(pk, sn, M)$, on input a public key, a serial number and a message in $\mathcal{M}$, checks their consistency by outputting a bit $b$.

$\mathsf{T.SVfy}_{\mathrm{all}}$, depending on the type of the input, runs $\mathsf{T.SVfy}_{\mathrm{init}}$ or $\mathsf{T.SVfy}$.

$\mathsf{T.TGen}(sk, n, sn)$, the double-spending tag function, takes as input a secret key, a nonce $n \in \mathcal{N}$ and a serial number, and outputs a double-spending tag $tag \in \mathcal{T}$ (the set of the double-spending tags) and a tag proof $t\text{-}pf$.

$\mathsf{T.TVfy}(pk, sn, sn', tag, t\text{-}pf)$, on input a public key, two serial numbers, a double-spending tag, and a proof, checks consistency of tag w.r.t. the key and the serial numbers by outputting a bit $b$.

$\mathsf{T.Detect}(sn, sn', tag, tag', \mathcal{L})$, double-spending-detection, takes as input two serial numbers $sn$ and $sn'$, two tags $tag, tag' \in \mathcal{T}$ and a list of public keys $\mathcal{L}$ and outputs a public key $pk$ (of the accused user) and a proof $\Pi$.

$\mathsf{T.VfyGuilt}(pk, \Pi)$, the incrimination-proof verification function, takes as input a public key and a proof and outputs a bit $b$.

**Correctness properties**

For any double-spending tag scheme $\mathsf{T} = (\mathsf{T.Setup}, \mathsf{T.KeyGen}, \mathsf{T.SGen}, \mathsf{T.SGen}_{\text{init}}, \mathsf{T.SVfy}, \mathsf{T.SVfy}_{\text{init}}, \mathsf{T.TGen}, \mathsf{T.TVfy}, \mathsf{T.Detect}, \mathsf{T.VfyGuilt})$ we require the following properties:

*SN-identifiability:* For all public tag keys $pk_1$ and $pk_2$, all serial numbers $sn$ and all $X_1$ and $X_2$, which can be messages in $\mathcal{M}$ or SN proofs, if

$$\mathsf{T.SVfy}_{\text{all}}(pk_1, sn, X_1) = \mathsf{T.SVfy}_{\text{all}}(pk_2, sn, X_2) = 1$$

then $pk_1 = pk_2$.

*2-show extractability:* Let $pk_0$, $pk_1$ and $pk_2$ be public tag keys, $sn_0$, $sn_1$ and $sn_2$ be serial numbers, $tag_1$ and $tag_2$ be tags, $sn\text{-}pf_1$ and $sn\text{-}pf_2$ be SN proofs, $t\text{-}pf_1$ and $t\text{-}pf_2$ be tag proofs and $X_0$ be either an SN proof or a message in $\mathcal{M}$. Let $\mathcal{L}$ be a set of tag public keys with $pk_0 \in \mathcal{L}$. If

$$\begin{aligned} \mathsf{T.SVfy}(pk_1, sn_1, sn\text{-}pf_1) = \mathsf{T.SVfy}(pk_2, sn_2, sn\text{-}pf_2) &= 1 \\ \mathsf{T.TVfy}(pk_1, sn_0, sn_1, tag_1, t\text{-}pf_1) &= 1 \\ \mathsf{T.TVfy}(pk_2, sn_0, sn_2, tag_2, t\text{-}pf_2) &= 1 \\ \mathsf{T.SVfy}_{\text{all}}(pk_0, sn_0, X_0) &= 1 \end{aligned}$$

and $sn_1 \neq sn_2$ then $\mathsf{T.Detect}(sn_1, sn_2, tag_1, tag_2, \mathcal{L})$ efficiently extracts $(pk_0, \Pi)$ and we have $\mathsf{T.VfyGuilt}(pk_0, \Pi) = 1$.

*Verifiability:* For every $n, n' \in \mathcal{N}$, and after computing

- $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
- $(sk, pk) \leftarrow \mathsf{T.KeyGen}(Gr)$
- $(sk', pk') \leftarrow \mathsf{T.KeyGen}(Gr)$
- $(sn, X) \leftarrow \mathsf{T.SGen}(sk, n)$ or $(sn, X) \leftarrow \mathsf{T.SGen}_{\text{init}}(sk, n)$
- $(sn', sn\text{-}pf') \leftarrow \mathsf{T.SGen}(sk', n')$
- $(tag, t\text{-}pf) \leftarrow \mathsf{T.TGen}(sk, n, sn')$

we have $\mathsf{T.TVfy}(pk, sn, sn', tag, t\text{-}pf) = \mathsf{T.SVfy}(pk, sn, X) = 1$.

*Bootability:* It is impossible to find an SN message $M$ and 2 different serial numbers $sn_1$ and $sn_2$ and two tag keys (distinct or not) $pk_{\mathsf{T}}^{(1)}, pk_{\mathsf{T}}^{(2)}$ such that:

$$\mathsf{T.SVfy}_{\text{init}}(pk_{\mathsf{T}}^{(1)}, sn_1, M) = \mathsf{T.SVfy}_{\text{init}}(pk_{\mathsf{T}}^{(2)}, sn_2, M) = 1.$$

$\mathbf{Expt}^{\texttt{tag-anon}}_{\mathcal{A},b}(\lambda)$:
$Gr \leftarrow \mathsf{BGGen}(1^\lambda)$
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
$k := 0$
$(sk_0, sk_1) \leftarrow \mathcal{A}(par_{\mathsf{T}})$
$b^* \leftarrow \mathcal{A}^{O_1(sk_b), O_2(sk_b,\cdot,\cdot)}(par_{\mathsf{T}}, sk_0, sk_1)$
Return $(b = b^*)$

$O_1(sk)$:
$n \xleftarrow{\$} \mathcal{N}$; $T[k] := n$; $k := k + 1$
$(sn, sn\text{-}pf) \leftarrow \mathsf{T.SGen}(sk, n)$
Return $sn$.

$O_2(sk, sn', i)$:
If $T[i] = \perp$, abort the oracle call
$n := T[i]$; $T[i] := \perp$
$(tag, t\text{-}pf) \leftarrow \mathsf{T.TGen}(sk, n, sn')$
Return $tag$

Figure 2.3: Game for *tag anonymity* (with oracles also used in *exculpability*) for double-spending tag schemes

$\mathcal{N}$*-injectivity:* For all $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$ and any secret key $sk$, the function $\mathsf{T.SGen}(sk, \cdot)$ is injective.

**Security properties**

*Exculpability:* This notion requires soundness of proofs, that is, one cannot produce a proof of a false statement. Let $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}$ and $(sk, pk) \leftarrow \mathsf{T.KeyGen}(par_{\mathsf{T}})$. Then we require that is computationally hard for an adversary given $pk$ and black-box access to oracles $O_1(sk)$ and $O_2(sk, \cdot, \cdot)$ defined on the right of Fig. 2.3 to return a proof $\Pi$ with $\mathsf{T.VfyGuilt}(pk, \Pi) = 1$.

*Tag anonymity:* Finally, our anonymity notions for transferable e-cash should hold even against a malicious bank, which gets to see the serial numbers and double-spending tags for deposited coins, and the *secret keys* of the users. Thus, we require that as long as the nonce $n$ is random and only used once, the serial numbers and tags reveal nothing about the user-specific values, such as $sk$ and $pk$, that were used to generate them. The game is given in Fig. 2.3.

**Definition 2.11** (Tag anonymity)**.** *A double-spending tag scheme is* anonymous *if* $\Pr[\mathbf{Expt}^{\texttt{tag-anon}}_{\mathcal{A},1}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\texttt{tag-anon}}_{\mathcal{A},0}(\lambda) = 1]$ *is negligible in* $\lambda$ *for any PPT* $\mathcal{A}$.

## 2.5 Proof of Lemma 2.9

We prove the statement for $(q_1, q_2)$-DLog; the proof for $q$-DLog follows analogously. We give a proof in Maurer's version of the GGM [Mau05], in which an adversary $\mathsf{A}_{\mathsf{gen}}$ can access elements from the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ only via abstract handles. These are maintained by the challenger in lists $L_1$, $L_2$, and $L_T$, which correspond to the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively.

We define a slightly different model (the ideal one) in which one the challenger considers polynomials instead of group elements. For $(q_1, q_2)$-DLog, it means that the lists $L_1$ and $L_2$ initially contain the handles to the elements $1, Z, Z^2, \ldots, Z^{q_1}$ and $1, Z, Z^2, \ldots, Z^{q_2} \in \mathbb{Z}_p[Z]$, respectively, that correspond to the $(q_1, q_2)$-DLog challenge given to $\mathsf{A}_{\mathsf{gen}}$. The challenger also samples $z \xleftarrow{\$} \mathbb{Z}_p^*$, the solution, and we will argue that this value remains information-theoretically hidden from $\mathsf{A}_{\mathsf{gen}}$.

The adversary is granted access to oracles of two types. Oracles $\mathsf{O}_i(\cdot)$, for $i \in \{1, 2, T\}$, take as input two handles $h_1, h_2 \in L_i$ for polynomials $P_1(Z), P_2(Z) \in \mathbb{Z}_p[Z]$, respectively, and output a handle $h$ for the polynomial $P_1(Z) + P_2(Z)$; $L_i$ is accordingly updated with the handle $h$. Oracle $\mathsf{O}_e(\cdot)$, on input two handles $h_1 \in L_1$ and $h_2 \in L_2$ for $P_1(Z)$ and $P_2(Z)$, outputs the handle $h_T$ to the element $P_1(Z) \cdot P_2(Z)$ and updates $L_T$ accordingly.

In this game the adversary does not have access any information on $z$ ($z$ could be chosen at the end of the game), and thus has probability $\frac{1}{p-1}$.

Following the standard argument in the generic group model proofs, we consider a so-called *collision event* $\mathcal{E}$ which occurs if there exist two distinct handles $h_1, h_2 \in L_1 \cup L_2 \cup L_T$ that point to polynomials $P_1(Z)$ and $P_2(Z)$, respectively, such that $P_1(Z) \neq P_2(Z)$, yet $P_1(z) = P_2(z)$. Now have to determine how many this ideal game differs from the real one (in the generic group mode). We can upper bound the statistical disctance between these games by $\Pr(\mathcal{E})$.

Thus, to lower-bound the number of oracle interactions that are needed until $\mathsf{A}_{\mathsf{gen}}$ finds $z$, it suffices to lower-bound the time until $\mathcal{E}$ happens.

**Analysis of $\Pr(\mathcal{E})$.** Before $\mathcal{E}$ occurs, $z$ is a uniformly random value and thus the probability that two computed elements are equal after $t$ steps of computation (i.e., oracle calls) can be upper-bounded by the Schwartz-Zippel Lemma (Lemma 2.1). In particular, let $P_1(Z)$ and $P_2(Z)$ be distinct polynomials of degree at most $2q$. Then Schwartz-Zippel upper-bounds the probability that $P_1(z) \equiv_p P_2(z)$ for a uniform element $z \in \mathbb{Z}_p^*$ by $2q/(p-1)$. As initially the set $L_1 \cup L_2 \cup L_T$ is of size $q+1$ (where $q := \max\{q_1, q_2\}$) and an oracle call by $\mathsf{A}_{\mathsf{gen}}$ adds at most one polynomial of degree at most $q_1 + q_2 < 2q$ to one of the lists, there are at most $\binom{t+q+1}{2}$ such equations after $t$ steps of computation. Thus, the probability of a collision occurring is at most $(t+q+1)^2 \cdot 2q/(2(p-1)) = (t+q+1)^2 \cdot q/(p-1)$.

# Chapter 3

# Classification of computational assumptions in the algebraic group model

*This work was published in the proceedings of the 2020 CRYPTO Conference. It was completed with co-authors Georg Fuchsbauer, and Julian Loss.*

---

**Overview.** In this chapter, we give a taxonomy of computational assumptions in the algebraic group model (AGM). We first analyze Boyen's Uber assumption family for bilinear groups and then extend it in several ways to cover assumptions as diverse as Gap Diffie-Hellman and LRSW. We show that in the AGM every member of these families is implied by the $q$-discrete logarithm (DL) assumption, for some $q$ that depends on the degrees of the polynomials defining the Uber assumption.

Using the meta-reduction technique, we then separate $(q+1)$-DL from $q$-DL, which yields a classification of all members of the extended Uber-assumption families. We finally show that there are strong assumptions, such as one-more DL, that provably fall outside our classification, by proving that they cannot be reduced from $q$-DL even in the AGM.

## 3.1 The Uber-Assumption Family

Boyen [Boy08] extended the Uber-assumption framework he initially introduced with Boneh and Goh [BBG05]. We start with defining notions of independence for polynomials and rational fractions (of which polynomials are a special case):

**Definition 3.1.** *Let* $\overrightarrow{R} \in \mathbb{Z}_p(X_1,\ldots,X_m)^r$ *and* $W \in \mathbb{Z}_p(X_1,\ldots,X_m)$. *We say that* $W$ *is* linearly dependent *on* $\overrightarrow{R}$ *if there exist coefficients* $(a_i)_{i=1}^r \in \mathbb{Z}_p^r$ *such that*

$$W = \sum_{i=1}^{r} a_i R_i.$$

*We say that* $W$ *is* (linearly) independent *from* $\overrightarrow{R}$ *if it is not linearly dependent on* $\overrightarrow{R}$.

**Definition 3.2.** *Let $\overrightarrow{R}$, $\overrightarrow{S}$, $\overrightarrow{F}$ and $W$ be vectors of rational fractions from $\mathbb{Z}_p(X_1,\ldots,X_m)$ of length $r, s, f$ and 1, respectively. We say that $W$ is* ("bilinearly") dependent *on $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$ if there exist coefficients $\{a_{i,j}\}$, $\{b_{i,j}\}$, $\{c_{i,j}\}$ and $\{d_k\}$ in $\mathbb{Z}_p$ such that*

$$W = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{i,j}R_iS_j + \sum_{i=1}^{r}\sum_{j=1}^{r} b_{i,j}R_iR_j + \sum_{i=1}^{s}\sum_{j=1}^{s} c_{i,j}S_iS_j + \sum_{k=1}^{f} d_kF_k.$$

*We call the dependency of* Type 2 *if $b_{i,j} = 0$ for all $i, j$ and of* Type 3 *if $b_{i,j} = c_{i,j} = 0$ for all $i, j$. Else, it is of* Type 1. *We say that $W$ is* (Type-$\tau$) independent *from $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$ if it is not (Type-$\tau$) dependent on $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$. (Thus, $W$ can be Type-3 independent but Type-2 dependent.)*

Consider the Uber-assumption game in Fig. 3.1, which is parametrized by vectors of polynomials $\overrightarrow{R}$, $\overrightarrow{S}$ and $\overrightarrow{F}$ and polynomials $R'$, $S'$ and $F'$. For a random vector $\overrightarrow{x}$, the adversary receives the evaluation of the (vectors of) polynomials in the exponents of the generators $g_1$, $g_2$ and $g_T$; its goal is to find the evaluation of the polynomials $R'$, $S'$ and $F'$ at $\overrightarrow{x}$ in the exponents. Note that we do not explicitly give the generators to the adversary. This is without loss of generality because we can always set $R_1 = S_1 = F_1 \equiv 1$.

The game can be efficiently solved if one of the following conditions hold (where we distinguish the different types of bilinear groups and interpret all polynomials over $\mathbb{Z}_p$):

(Type 1) If $R'$ is dependent on $\overrightarrow{R}$ and $\overrightarrow{S}$, and $S'$ is dependent on $\overrightarrow{R}$ and $\overrightarrow{S}$, and $F'$ is Type-1 dependent (Def. 3.2) on $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$.

(Type 2) If $R'$ is dependent on $\overrightarrow{R}$ and $\overrightarrow{S}$, $S'$ is dependent on $\overrightarrow{S}$, and $F'$ is Type-2 dependent (Def. 3.2) on $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$.

(Type 3) If $R'$ is dependent on $\overrightarrow{R}$, $S'$ is dependent on $\overrightarrow{S}$, and $F'$ is Type-3 dependent (Def. 3.2) on $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$.

For example, in Type-2 groups, if $R' = \sum_i a'_iR_i + \sum_i b'_iS_i$ and $S' = 1$, and $F' = \sum_i\sum_j a_{i,j}R_iS_i + \sum_i\sum_j c_{i,j}S_iS_j$, then from a challenge $(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$, one can easily compute a solution $\mathbf{U}' := \prod_i \mathbf{U}_i^{a'_i} \cdot \psi(\prod_i \mathbf{V}_i^{b'_i})$, $\mathbf{V}' := g_2$, $\mathbf{W}' := \prod_i\prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{a_{i,j}} \cdot \prod_i\prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{b_{i,j}}$.

In our main theorem, we show that whenever the game in Fig. 3.1 cannot be trivially won, then for groups of Type $\tau \in \{1, 2\}$, it can be reduced from $q$-**dlog**$_{\mathcal{G}_2}$, and for Type-3 groups, it can be reduced from $(q_1, q_2)$-**dlog**$_{\mathcal{G}}$ (for appropriate values of $q, q_1, q_2$). To state the theorem for all types of groups, we first define the following non-triviality condition (which again we state for the more general case of rational fractions):

**Definition 3.3** (Non-triviality)**.** *Let $\overrightarrow{R} \in \mathbb{Z}_p(X_1,\ldots,X_m)^r$, $\overrightarrow{S} \in \mathbb{Z}_p(X_1,\ldots,X_m)^s$, $\overrightarrow{F} \in \mathbb{Z}_p(X_1,\ldots,X_m)^f$, $R', S', F' \in \mathbb{Z}_p(X_1,\ldots,X_m)$. We say that the tuple $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ is* non-trivial for groups of type $\tau$, *for $\tau \in \{1, 2, 3\}$, if the following holds:*

- **either** *$R'$ is linearly independent from $\overrightarrow{R}$ and $\overrightarrow{S}$ in case $\tau \in \{1, 2\}$,*
  *$R'$ is linearly independent from $\overrightarrow{R}$ in case $\tau = 3$;* *($\tau$.1)*
- **or** *$S'$ is linearly independent from $\overrightarrow{R}$ and $\overrightarrow{S}$ in case $\tau = 1$,*
  *$S'$ is linearly independent from $\overrightarrow{S}$ in case $\tau \in \{2, 3\}$;* *($\tau$.2)*
- **or** *$F'$ is Type-$\tau$ "bilinearly" independent (Def. 3.2) from $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$.* *($\tau$.T)*

$(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$-**über**$_{\mathcal{G}}^{\mathsf{A}_{\mathsf{alg}}}$

$(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ ; $g_T \leftarrow e(g_1, g_2)$

$\overrightarrow{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

$\overrightarrow{\mathbf{U}} := (g_1^{R_1(\overrightarrow{x})}, \ldots, g_1^{R_r(\overrightarrow{x})})$

$\overrightarrow{\mathbf{V}} := (g_2^{S_1(\overrightarrow{x})}, \ldots, g_2^{S_s(\overrightarrow{x})})$

$\overrightarrow{\mathbf{W}} := (g_T^{F_1(\overrightarrow{x})}, \ldots, g_T^{F_f(\overrightarrow{x})})$

$(\mathbf{U}', \mathbf{V}', \mathbf{W}') \leftarrow \mathsf{A}_{\mathsf{alg}}(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$

Return $((\mathbf{U}', \mathbf{V}', \mathbf{W}') = (g_1^{R'(\overrightarrow{x})}, g_2^{S'(\overrightarrow{x})}, g_T^{F'(\overrightarrow{x})}))$

Figure 3.1: Algebraic game for the Uber assumption relative to bilinear group $\mathcal{G}$ and adversary $\mathsf{A}_{\mathsf{alg}}$, parametrized by (vectors of) or polynomials $\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S'$ and $F'$

We have argued above that if the tuple $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ is trivial then the $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$-**über** problem is trivial to solve, even with a generic algorithm. In Theorem 3.5 we now show that if the tuple is *non-trivial* then the corresponding Uber assumption holds for algebraic algorithms, as long as a type of $q$-DLog assumption holds (whose type depends on the type of bilinear group).

The (additive) security loss of the reduction depends on the degrees of the polynomials involved (as well as the group type and its order). E.g., in Type-3 groups, if $R'$ is independent of $\overrightarrow{R}$ then the probability that the reduction fails is the maximum degree of $R'$ and the components of $\overrightarrow{R}$, divided by the order of $\mathcal{G}$. In Type-1 and Type-2 groups, due to the homomorphism $\psi$, the loss depends on the maximum degree of $R'$, $\overrightarrow{R}$ and $\overrightarrow{S}$. Similar bounds hold when $S'$ is independent of $\overrightarrow{S}$ (and $\overrightarrow{R}$ for Type 1); and slightly more involved ones for the independence of $F'$. If several of $R', S'$ and $F'$ are independent then the reduction chooses the strategy that minimizes the security loss.

**Definition 3.4** (Degree of non-trivial tuple of polynomials)**.** *Let $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ be a non-trivial tuple of polynomials in $\mathbb{Z}_p[X_1, \ldots, X_m]$.*
*Define $d_{\overrightarrow{R}} := \max\{\deg R_i\}_{1 \le i \le r}$, $d_{\overrightarrow{S}} := \max\{\deg S_i\}_{1 \le i \le s}$, $d_{\overrightarrow{F}} := \max\{\deg F_i\}_{1 \le i \le f}$.*
*We define the* type-$\tau$ degree $d_\tau$ *of $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ as follows:*

- *If $(\tau.1)$ holds, let $d_{\tau.1} := \max\{\deg R', d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\}$ in case $\tau \in \{1, 2\}$ and*
  $d_{\tau.1} := \max\{\deg R', d_{\overrightarrow{R}}\}$ *in case $\tau = 3$.*
- *If $(\tau.2)$ holds, let $d_{\tau.2} := \max\{\deg S', d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\}$ in case $\tau = 1$ and*
  $d_{\tau.2} := \max\{\deg S', d_{\overrightarrow{S}}\}$ *in case $\tau \in \{2, 3\}$.*
- *If $(\tau.T)$ holds, let $d_{\tau.T} := \max\{\deg F', 2\, d_{\overrightarrow{R}}, 2\, d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$ when $\tau = 1$,*
  $d_{\tau.T} := \max\{\deg F', d_{\overrightarrow{R}} + d_{\overrightarrow{S}}, 2\, d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$ *in case $\tau = 2$ and*
  $d_{\tau.T} := \max\{\deg F', d_{\overrightarrow{R}} + d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$ *in case $\tau = 3$.*

*If $(\tau, i)$ does not hold, we set $d_{\tau,i} := \infty$ and define $d_\tau := \min\{d_{\tau.1}, d_{\tau.2}, d_{\tau.T}\}$. (By non-triviality, we have $d_\tau < \infty$.)*

**Theorem 3.5** (DLog implies Uber in the AGM)**.** *Let $\mathcal{G}$ be of type $\tau \in \{1, 2, 3\}$ and $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F') \in (\mathbb{Z}_p[X_1, \ldots, X_m])^{r+s+f+3}$ be a tuple of polynomials that is non-trivial for type $\tau$ and*

*define* $d_{\overrightarrow{R}} := \max\{\deg R_i\}$, $d_{\overrightarrow{S}} := \max\{\deg S_i\}$, $d_{\overrightarrow{F}} := \max\{\deg F_i\}$. *Let* $q, q_1, q_2$ *be such that* $q \geq \max\{d_{\overrightarrow{R}}, d_{\overrightarrow{S}}, d_{\overrightarrow{F}}/2\}$ *as well as* $q_1 \geq d_{\overrightarrow{R}}$, $q_2 \geq d_{\overrightarrow{S}}$ *and* $q_1 + q_2 \geq d_{\overrightarrow{F}}$. *If*

(Type 1) $q$-**dlog**$_{\mathcal{G}_1}$ *or* $q$-**dlog**$_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 2) $q$-**dlog**$_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 3) $(q_1, q_2)$-**dlog**$_{\mathcal{G}}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

*then* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$-**über**$_{\mathcal{G}}$ *is* $(\varepsilon', t')$*-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \leq t + o_1,$$

*where* $d_\tau$ *is the maximal degree of* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$*, as defined in Def. 3.4,*

$$o_1 := o_0 + 2 + (2\lfloor \log_2(p) \rfloor)((d_{\overrightarrow{R}} + 1)r + (d_{\overrightarrow{S}} + 1)s + (d_{\overrightarrow{F}} + 1)f + d_\tau) + rd_{\overrightarrow{R}} + sd_{\overrightarrow{S}} + fd_{\overrightarrow{F}}$$

*with* $o_0 := d_{\overrightarrow{R}} + d_{\overrightarrow{F}} + 2$ *for Types* 1 *and* 2, *and* $o_0 := d_{\overrightarrow{F}} + 1$ *for in Type* 3.

*Proof.* We give a detailed proof for Type-2 bilinear groups and then explain how to adapt it to Types 1 and 3. For $u \in \mathbb{Z}_p$ and $i \in \{1, 2, T\}$ we let $[u]_i$ denote $g_i^u$.

Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic algorithm against $\mathbf{über}_{\mathcal{G}}$ that wins with advantage $\varepsilon$ in time $t$. We construct a generic reduction with oracle access to $\mathsf{A}_{\mathsf{alg}}$, which yields an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against $q$-**dlog**$_{\mathcal{G}_2}$. There are three (non-exclusive) reasons for $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ being non-trivial, which correspond to conditions (2.1), (2.2) and (2.T) in Def. 3.3. Each condition enables a different type of reduction, of which $\mathsf{B}_{\mathsf{alg}}$ runs the one that minimizes $d_2$ from Def. 3.4.

We start with Case (2.1), that is, $R'$ is linearly independent from $\overrightarrow{R}$ and $\overrightarrow{S}$.

*Adversary* $\mathsf{B}_{\mathsf{alg}}(g_2, \mathbf{Z}_1, \ldots, \mathbf{Z}_q)$*:* On input a problem instance of game $q$-**dlog**$_{\mathcal{G}_2}$ with $\mathbf{Z}_i = [z^i]_2$, $\mathsf{B}_{\mathsf{alg}}$ simulates $\mathbf{über}_{\mathcal{G}}$ for $\mathsf{A}_{\mathsf{alg}}$. It defines $g_1 \leftarrow \psi(g_2)$ and $g_T \leftarrow e(g_1, g_2)$. Then, it picks random values $\overrightarrow{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\overrightarrow{v} \xleftarrow{\$} \mathbb{Z}_p^m$, implicitly sets $x_i := y_i z + v_i \bmod p$ and computes $\overrightarrow{\mathbf{U}} := [\overrightarrow{R}(x_1, \ldots, x_m)]_1$, $\overrightarrow{\mathbf{V}} := [\overrightarrow{S}(x_1, \ldots, x_m)]_2$, $\overrightarrow{\mathbf{W}} := [\overrightarrow{F}(x_1, \ldots, x_m)]_T$ from its $q$-DLog instance, the isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ and the pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It can do so efficiently since the total degrees of the polynomials in $\overrightarrow{R}$, $\overrightarrow{S}$ and $\overrightarrow{F}$ are bounded by $q, q$ and $2q$ respectively.[1]

Next, $\mathsf{B}_{\mathsf{alg}}$ runs $(\mathbf{U}', \mathbf{V}', \mathbf{W}') \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$. Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic, it also returns vectors and matrices $\overrightarrow{\mu}, \overrightarrow{\nu}, \overrightarrow{\zeta}, \overrightarrow{\delta}, A = (\alpha_{i,j})_{i,j}$, $\Gamma = (\gamma_{i,j})_{i,j}$ such that

$$\mathbf{U}' = \textstyle\prod_i \mathbf{U}_i^{\mu_i} \cdot \prod_i \psi(\mathbf{V}_i)^{\nu_i} \tag{3.1a}$$
$$\mathbf{V}' = \textstyle\prod_i \mathbf{V}_i^{\eta_i} \tag{3.1b}$$
$$\mathbf{W}' = \textstyle\prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{\gamma_{i,j}} \cdot \prod_i \mathbf{W}_i^{\delta_i}. \tag{3.1c}$$

$\mathsf{B}_{\mathsf{alg}}$ then computes the following multivariate polynomial, which corresponds to the exponents of (3.1a):

$$P_1(\overrightarrow{X}) = R'(\overrightarrow{X}) - \textstyle\sum_{i=1}^r \mu_i R_i(\overrightarrow{X}) - \sum_{i=1}^s \nu_i S_i(\overrightarrow{X}), \tag{3.2}$$

[1] E.g., $\mathsf{B}_{\mathsf{alg}}$ can compute $[x_1^q]_1 = [(y_1 z + v_1)^q]_1$ as $\prod_i \psi(\mathbf{Z}_i)^{\binom{q}{i} y_1^i v_1^{q-i}}$ and $[x_1^{2q}]_T$ as $e\big(\prod_i \psi(\mathbf{Z}_i)^{\binom{q}{i} y_1^i v_1^{q-i}}, \prod_i \mathbf{Z}_i^{\binom{q}{i} y_1^i v_1^{q-i}}\big)$ and similarly for terms in more variables.

which is non-zero because in Case (2.1) $R'$ is independent from $\overrightarrow{R}$ and $\overrightarrow{S}$. From $P_1$, it defines the univariate polynomial

$$Q_1(Z) := P_1(y_1 Z + v_1, \ldots, y_m Z + v_m). \tag{3.3}$$

If $Q_1$ is the zero polynomial then $\mathsf{B}_{\mathsf{alg}}$ aborts. (∗)

Else, it factors $Q_1$ to obtain its roots $z_1, \ldots$ (of which there are at most $\max\{\deg R', d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\}$; we analyse the degree of $Q_1$ below). If for one of them we have $g_2^{z_i} = \mathbf{Z}$, then $\mathsf{B}_{\mathsf{alg}}$ returns $z_i$.

We analyze $\mathsf{B}_{\mathsf{alg}}$'s success probability. First note that $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates game $\mathbf{\ddot{u}ber}_{\mathcal{G}}$, as the values $x_i$ are uniformly distributed in $\mathbb{Z}_p$ and $\overrightarrow{\mathbf{U}}$, $\overrightarrow{\mathbf{V}}$ and $\overrightarrow{\mathbf{W}}$ are correctly computed. Moreover, if $\mathsf{B}_{\mathsf{alg}}$ does not abort in (∗) and $\mathsf{A}_{\mathsf{alg}}$ wins game $\mathbf{\ddot{u}ber}_{\mathcal{G}}$, then $\mathbf{U}' = [R'(\overrightarrow{x})]_1$. On the other hand,

$$\mathbf{U}' = \textstyle\prod_i \mathbf{U}_i^{\mu_i} \cdot \psi(\prod_i \mathbf{V}_i^{\nu_i}) = \left[\sum_i \mu_i R_i(\overrightarrow{x}) + \sum_i \nu_i S_i(\overrightarrow{x})\right]_1.$$

Together, this means that $P_1(\overrightarrow{x}) \equiv_p 0$ and since $x_i \equiv_p y_i z + v_i$, moreover $Q_1(z) \equiv_p 0$. By factoring $Q_1$, reduction $\mathsf{B}_{\mathsf{alg}}$ finds thus the solution $z$.

It remains to bound the probability that $\mathsf{B}_{\mathsf{alg}}$ aborts in (∗), that is, the event that $0 \equiv Q_1(Z) = P_1(y_1 Z + v_1, \ldots, y_m Z + v_m)$. Interpreting $Q_1$ as an element from $(\mathbb{Z}_p[Y_1, \ldots, Y_m, V_1, \ldots, V_m])[Z]$, Lemma 2.2 yields that its maximal coefficient is a polynomial $Q_1^{\max}$ in $Y_1, \ldots, Y_m$ whose degree is the same as the maximal (total) degree $d$ of $P_1$. From $P_1 \not\equiv 0$ and $P_1(\overrightarrow{x}) = 0$, we have $d > 0$.

We note that the values $y_1 z, \ldots, y_m z$ are completely hidden from $\mathsf{A}_{\mathsf{alg}}$ because they are "one-time-padded" with $v_1, \ldots, v_m$, respectively. This means that the values $(\overrightarrow{\mu}, \overrightarrow{\nu})$ returned by $\mathsf{A}_{\mathsf{alg}}$ are independent from $\overrightarrow{y}$. Since $\overrightarrow{y}$ is moreover independent from $R'$, $\overrightarrow{R}$ and $\overrightarrow{S}$, it is also independent from $P_1, Q_1$ and $Q_1^{\max}$. The probability that $Q_1 \equiv 0$ is thus upper-bounded by the probability that its maximal coefficient $Q_1^{\max}(\overrightarrow{y}) \equiv_p 0$ when evaluated at the random point $\overrightarrow{y}$. By the Schwartz-Zippel lemma, the probability that $Q_1(Z) \equiv 0$ is thus upper-bounded by $\frac{d}{p-1}$. The degree $d$ of $Q_1$ (and thus of $Q_1^{\max}$) is upper-bounded by the total degrees of $P_1$, which is $\max\{d'_R, d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\} = d_{2.1}$ in Def. 3.4. $\mathsf{B}_{\mathsf{alg}}$ thus aborts in line (*) with probability at most $\frac{d_{2,1}}{p-1}$.

Case (2.2), that is, $S'$ is linearly independent from $\overrightarrow{S}$, follows completely analogously, but with $d = d_{2.2} = \max\{d_{S'}, d_{\overrightarrow{S}}\}$.

Case (2.T), when $F'$ is type-2-independent of $\overrightarrow{R}$, $\overrightarrow{S}$ and $\overrightarrow{F}$, is also analogous; we highlight the necessary changes: From $\mathsf{A}_{\mathsf{alg}}$'s representation $(A = (\alpha_{i,j}), \Gamma = (\gamma_{i,j}), \overrightarrow{\delta}) \in Z_p^{r\times s} \times Z_p^{s\times s} \times Z_p^f$ for $\mathbf{W}'$ (see (3.1c)), that is,

$$\begin{aligned}\mathbf{W}' &= \textstyle\prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_i \prod_j e(\psi(\mathbf{V}_i), \mathbf{V}_j)^{\gamma_{i,j}} \cdot \prod_i \mathbf{W}_i^{\delta_i} \\ &= \left[\sum_i \sum_j \alpha_{i,j} R_i(\overrightarrow{x}) S_j(\overrightarrow{x}) + \sum_i \sum_j \gamma_{i,j} S_i(\overrightarrow{x}) S_j(\overrightarrow{x}) + \sum_i \delta_i F_i(\overrightarrow{x})\right]_T. \end{aligned} \tag{3.4}$$

Analogously to (3.2) we define

$$\begin{aligned} P_T(\overrightarrow{X}) := F'(\overrightarrow{X}) &- \textstyle\sum_{i=1}^r \sum_{j=1}^s \alpha_{i,j} R_i(\overrightarrow{X}) S_j(\overrightarrow{X}) \\ &- \textstyle\sum_{i=1}^s \sum_{j=1}^s \gamma_{i,j} S_i(\overrightarrow{X}) S_j(\overrightarrow{X}) - \sum_{i=1}^f \delta_i F_i(\overrightarrow{X}), \end{aligned} \tag{3.5}$$

which is of degree at most $d_{2.T} := \max\{\deg F', d_{\overrightarrow{R}} + d_{\overrightarrow{S}}, 2 \cdot d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$. Polynomial $P_T$ is non-zero by Type-2-independence of $F'$ (Def. 3.2). The reduction also computes $Q_T(Z) := P_T(y_1 Z + v_1, \ldots, y_m Z + v_m)$.

If $\mathsf{A}_{\mathsf{alg}}$ wins then $\mathbf{W}' = [F'(\overrightarrow{x})]_T$, which together with (3.4) implies that $P_T(\overrightarrow{x}) \equiv_p 0$ and thus $Q_T(z) \equiv_p 0$. Reduction $\mathsf{B}_{\mathsf{alg}}$ can find $z$ by factoring $Q_T$; unless $Q_T(Z) \equiv 0$, which by an analysis

analogous to the one for case (2.1) happens with probability $\frac{d_{2.T}}{p-1}$. (We detail the reduction for the case where $\mathbf{W}'$ is independent in the proof of Theorem 3.9, which proves a more general statement.)

Theorem 3.5 for Type-2 groups follows since $\mathcal{A}q\text{-}\mathbf{dlog}\,\mathcal{G}_2, \mathsf{B}_{\mathsf{alg}} \geq \mathcal{A}\mathbf{\ddot{u}ber}\,\mathcal{G}, \mathsf{A}_{\mathsf{alg}} - \Pr[\mathsf{B}_{\mathsf{alg}}\text{ aborts}]$ and $\mathsf{B}_{\mathsf{alg}}$ follows the type of reduction that minimizes its abort probability to $\min\left\{\frac{d_{2.1}}{p-1}, \frac{d_{2.2}}{p-1}, \frac{d_{2.T}}{p-1}\right\} = \frac{d_2}{p-1}$.

**Groups of Type 1 and 3.** The reduction for bilinear groups of Type 1 to $q\text{-}\mathbf{dlog}_{\mathcal{G}_2}$ is almost the same proof. The only change is that for Case (1.T) the polynomial $P_T$ in (3.5) has an extra term $-\sum_{i=1}^{r}\sum_{j=1}^{r}\beta_{i,j}R_i(\overrightarrow{X})R_j(\overrightarrow{X})$, because of the representation of $\mathbf{W}'$ in Type-1 groups (see Def. 2.4); the degree of $P_T$ is then bounded by $\max\{\deg F', 2\,d_{\overrightarrow{R}}, 2\,d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$. Analogously for Case (1.2), $S'$ can now depend on $\overrightarrow{S}$ as well as $\overrightarrow{R}$. The reduction for Type-1 groups to $q\text{-}\mathbf{dlog}_{\mathcal{G}_1}$ is completely symmetric by swapping the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ and replacing $\psi$ by $\phi$.

The reduction for Type-3 groups relies on the $(q_1, q_2)\text{-}\mathbf{dlog}_{\mathcal{G}}$ assumption, as it requires $\{[z_i]_1\}_{i=1}^{q_1}$ and $\{[z_i]_2\}_{i=1}^{q_2}$ to simulate $\{[R_i(\overrightarrow{x})]_1\}_{i=1}^{r}$ and $\{[S_i(\overrightarrow{x})]_2\}_{i=1}^{s}$. without using any homomorphism $\phi$ or $\psi$. Apart from this, the proof is again analogous. (We treat the Type-3 case in detail in the proof of Theorem 3.9.) In Section 3.9 we detail the analysis of the running times of these reductions. □

Using Lemmas 2.5 and 2.9 we obtain the following corollary to Theorem 3.5:

**Corollary 3.6.** *Let $\mathcal{G}$ be of type $\tau$ and $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ be non-trivial for $\tau$ of maximal degree $d_\tau$. Then $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')\text{-}\mathbf{\ddot{u}ber}_{\mathcal{G}}$ is $(\frac{(o+o_1+1+q)^2q}{p-1} + \frac{d_\tau}{p-1}, o)$-secure in the generic bilinear group model.*

**Comparison to previous GGM results.** Boneh, Boyen and Goh [BBG05, Theorem A.2] claim that the decisional Uber assumption for the particular case $r = s$ and $f = 0$ is $(\frac{(o+2r+2)^2q}{2p}, o)$-secure in the generic group model, and with the same reasoning, one can obtain the more general bound $(\frac{(o+r+s+f+2)^2q}{2p}, o)$. Note that the loss in their bound is only linear in the maximum degree while ours *cubic*. Our looser bound is a result of our reduction, whereas Boneh, Boyen and Goh prove their bound directly in the GGM.[2]

## 3.2 The Flexible Uber Assumption

Boyen [Boy08] generalizes the Uber assumption framework to *flexible* assumptions, where the adversary can define the target polynomials ($R'$, $S'$ and $F'$ in Fig. 3.1) itself, conditioned on the solution not being trivially computable from the instance, for *non-triviality* as in Def. 3.3. In Sect. 3.4 we consider this kind of flexible Uber assumption in our generalization to rational fractions and thereby cover assumptions like $q$-strong Diffie-Hellman [BB08].

For polynomials, we generalize this further by allowing the adversary to also (adaptively) choose the polynomials that constitute the challenge. The adversary is provided with an oracle that takes input a value $i \in \{1, 2, T\}$ and a polynomial $P(\overrightarrow{X})$ of the adversary's choice, and returns $g_i^{P(\overrightarrow{x})}$, where $\overrightarrow{x}$ is the secret value chosen during the game. The adversary then wins if it returns polynomials $(R^*, S^*, F^*)$, which are independent from its queries, and $(g_1^{R^*(\overrightarrow{x})}, g_2^{S^*(\overrightarrow{x})}, g_T^{F^*(\overrightarrow{x})})$. The game for this flexible Uber assumption is specified in Fig. 3.2.

**Theorem 3.7.** *Let $m \geq 1$, let $\mathcal{G}$ be a bilinear-group of type $\tau \in \{1, 2, 3\}$ and consider an adversary $\mathsf{A}_{\mathsf{alg}}$ in game $m\text{-}\mathbf{f\text{-}\ddot{u}ber}_{\mathcal{G}}$. Let $d'_1, d'_2, d'_T, d^*_1, d^*_2, d^*_T$ be such that $\mathsf{A}_{\mathsf{alg}}$'s queries $(i, P(\overrightarrow{X}))$ satisfy*

[2] We did not consider the bound from [Boy08, Theorem 1], as it is an incorrect copy of the one in [BBG05].

$m$-**f-über**$_{\mathcal{G}}^{\mathsf{A}_{\mathsf{alg}}}$

01 $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T \leftarrow \emptyset$

02 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ ; $g_T \leftarrow e(g_1, g_2)$

03 $\overrightarrow{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

04 $((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}()$

05 Return $\Big((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) = (g_1^{R^*(\overrightarrow{x})}, g_2^{S^*(\overrightarrow{x})}, g_T^{F^*(\overrightarrow{x})})$
$\wedge\ (\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ non-trivial for type of $\mathcal{G}\Big)$

$\underline{\mathsf{O}}(i, P(\overrightarrow{X}))$

06 $\mathcal{Q}_i = \mathcal{Q}_i \cup \{P\}$

07 Return $g_i^{P(\overrightarrow{x})}$

Figure 3.2: Algebraic game for the flexible Uber assumption

$\deg P \le d_i'$ *and its output values* $R^*, S^*, F^*$ *satisfy* $\deg R^* \le d_1^*$, $\deg S^* \le d_2^*$, $\deg F^* \le d_T^*$. *Let* $q, q_1, q_2$ *be such that* $q \ge \max\{d_1', d_2', d_T'/2\}$ *as well as* $q_1 \ge d_1'$, $q_2 \ge d_2'$ *and* $q_1 + q_2 \ge d_T'$. *If*

(Type 1) $q$-**dlog**$_{\mathcal{G}_1}$ *or* $q$-**dlog**$_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 2) $q$-**dlog**$_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 3) $(q_1, q_2)$-**dlog**$_{\mathcal{G}}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

*then* $m$-**f-über**$_{\mathcal{G}}$ *is* $(\varepsilon', t')$*-secure in the AGM with*

$$\varepsilon' \le \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

*where* $d_\tau$ *is as in Def. 3.4 after the following replacements:* $d_{\overrightarrow{R}} \leftarrow d_1'$, $d_{\overrightarrow{S}} \leftarrow d_2'$, $d_{\overrightarrow{F}} \leftarrow d_F'$, $\deg R' \leftarrow d_1^*$, $\deg S' \leftarrow d_2^*$ *and* $\deg F' \leftarrow d_T^*$.

*Proof sketch.* Inspecting the proof of Theorem 3.5, note that the values $[R_i(\overrightarrow{x})]_1$, $[S_i(\overrightarrow{x})]_2$ and $[F_i(\overrightarrow{x})]_T$ need not be known in advance and can be computed by the reduction at any point, as long as the degrees of $R_i$ and $S_i$ are bounded by $q$ and those of $F_i$ by $2q$. The adversary could thus specify the polynomials via oracle calls and the reduction can compute $\mathbf{U}_i$, $\mathbf{V}_i$ and $\mathbf{F}_i$ on the fly.

Likewise, the polynomials $P_1$, $P_2$ and $P_T$ (and their univariate counterparts which the reduction factors) are only defined after $\mathsf{A}_{\mathsf{alg}}$ stops; therefore, $R'$, $S'$ and $F'$, from which they are defined, need only be known then. The proof of Theorem 3.5 is thus adapted to prove Theorem 3.7 in a very straightforward way. □

## 3.3 The Ruber Assumption

Reconsider the Uber assumption in Fig. 3.1, but now let $\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S'$ and $F'$ be *rational fractions* over $\mathbb{Z}_p$ rather than polynomials. We will show that even this generalization of the Uber assumption is implied by $q$-DLog assumptions. We start with introducing some notation. We view a rational fraction as defined by two polynomials, its numerator and its denominator, and assume that the fraction is reduced. For a rational fraction $R \in \mathbb{Z}_p(X_1, \ldots, X_m)$, we denote its numerator by $\hat{R}$ and its denominator by $\check{R}$. That is $\hat{R}, \check{R} \in \mathbb{Z}_p[X_1, \ldots, X_m]$ are such that $R = \hat{R}/\check{R}$. As rational fractions are not defined everywhere, we modify the game from Fig. 3.1 so the adversary wins should the experiment choose an input $\overrightarrow{x}$ for which one of the rational fractions is not defined. The rational-fraction uber game is given in Fig. 3.3.

$(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$-**rüber**$^{\mathsf{A_{alg}}}_{\mathcal{G}}$ // $R_i = \hat{R}_i/\check{R}_i$, for $\hat{R}_i, \check{R}_i \in Z_p[\overrightarrow{X}]$, etc

01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ ; $g_T \leftarrow e(g_1, g_2)$

02 $\overrightarrow{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

03 If for some $i$: $\check{R}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{S}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{F}_i(\overrightarrow{x}) \equiv_p 0$ then return 1

04 If $\check{R}'(\overrightarrow{x}) \equiv_p 0$ or $\check{S}'(\overrightarrow{x}) \equiv_p 0$ or $\check{F}'(\overrightarrow{x}) \equiv_p 0$ then return 1

05 $\overrightarrow{\mathbf{U}} := (g_1^{R_1(\overrightarrow{x})}, \ldots, g_1^{R_r(\overrightarrow{x})})$ ; $\overrightarrow{\mathbf{V}} := (g_2^{S_1(\overrightarrow{x})}, \ldots, g_2^{S_s(\overrightarrow{x})})$ ; $\overrightarrow{\mathbf{W}} := (g_T^{F_1(\overrightarrow{x})}, \ldots, g_T^{F_f(\overrightarrow{x})})$

06 $(\mathbf{U}', \mathbf{V}', \mathbf{W}') \xleftarrow{\$} \mathsf{A_{alg}}(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$

07 Return $((\mathbf{U}', \mathbf{V}', \mathbf{W}') = (g_1^{R'(\overrightarrow{x})}, g_2^{S'(\overrightarrow{x})}, g_T^{F'(\overrightarrow{x})}))$

Figure 3.3: Algebraic game for the Uber assumption relative to bilinear group $\mathcal{G}$ and adversary $\mathsf{A_{alg}}$, parametrized by (vectors of) rational fractions $\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S'$ and $F'$

For a vector of rational fractions $\overrightarrow{R} \in \mathbb{Z}_p(X_1, \ldots, X_m)^r$, we define its *common denominator* $\mathrm{Den}(\overrightarrow{R})$ as a least common multiple of the denominators of the components of $\overrightarrow{R}$. In particular, fix an algorithm LCM that given a set of polynomials returns a least common multiple of them. Then we define:

$$\mathrm{Den}(\overrightarrow{R}) = \mathrm{Den}(\hat{R}_1/\check{R}_1, \ldots, \hat{R}_r/\check{R}_r) := \mathrm{LCM}\{\check{R}_1, \ldots, \check{R}_r\}.$$

We let $\check{d}_{\overrightarrow{R}}$ denote the degree of $\mathrm{Den}(\overrightarrow{R})$ and $d_{\overrightarrow{R}}$ denote the maximal degree of the elements of $\overrightarrow{R}$, that is $d_{\overrightarrow{R}} := \max\{\deg(R_1), \ldots, \deg(R_r)\}$. Note that this integer could be negative and is lower bounded by $-\check{d}_{\overrightarrow{R}}$. The security loss in Theorem 3.9 depends on the type of the bilinear group, the reason for the tuple $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ being non-trivial, as well as the degrees of the numerators and denominators of the involved rational fractions. We summarize this in the following technical definition.

**Definition 3.8** (Degree of non-trivial tuple of rational fractions)**.** *Let $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ be a non-trivial tuple whose elements are rational fractions in $\mathbb{Z}_p(X_1, \ldots, X_m)$. Let $d_{\mathrm{den}} := \check{d}_{\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F}\|R'\|S'\|F'}$. We define the* type-$\tau$ degree $d_\tau$ *of $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ as follows, distinguishing the kinds of non-triviality defined in Def. 3.3.*

(Type 1)
- *If (1.1) holds, let* $d_{1.1} := d_{\mathrm{den}} + \check{d}_{R'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{S}} + \max\{d_{R'}, d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\}$
- *if (1.2) holds, let* $d_{1.2} := d_{\mathrm{den}} + \check{d}_{S'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{S}} + \max\{d_{S'}, d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\}$
- *if (1.T) holds,* $d_{1.T} := d_{\mathrm{den}} + \check{d}_{F'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F}} + \check{d}_{\overrightarrow{R}\|\overrightarrow{S}} + \max\{d_{F'}, 2d_{\overrightarrow{S}}, 2d_{\overrightarrow{R}}, d_{\overrightarrow{F}}\}$

(Type 2)
- *If (2.1) holds, let* $d_{2.1} := d_{\mathrm{den}} + \check{d}_{R'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{S}} + \max\{d_{R'}, d_{\overrightarrow{R}}, d_{\overrightarrow{S}}\}$
- *if (2.2) holds, let* $d_{2.2} := d_{\mathrm{den}} + \check{d}_{S'} + \check{d}_{\overrightarrow{S}} + \max\{d_{S'}, d_{\overrightarrow{S}}\}$
- *if (2.T) holds,* $d_{2.T} := d_{\mathrm{den}} + \check{d}_{F'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F}} + \check{d}_{\overrightarrow{S}} + \max\{d_{F'}, 2d_{\overrightarrow{S}}, d_{\overrightarrow{R}} + d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$

(Type 3)
- *If (3.1) holds, let* $d_{3.1} := d_{\mathrm{den}} + \check{d}_{R'} + \check{d}_{\overrightarrow{R}} + \max\{d_{R'}, d_{\overrightarrow{R}}\}$
- *if (3.2) holds, let* $d_{3.2} := d_{\mathrm{den}} + \check{d}_{S'} + \check{d}_{\overrightarrow{S}} + \max\{d_{S'}, d_{\overrightarrow{S}}\}$
- *if (3.T) holds,* $d_{3.T} := d_{\mathrm{den}} + \check{d}_{F'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{F}} + \check{d}_{\overrightarrow{S}} + \max\{d_{F'}, d_{\overrightarrow{R}} + d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\}$

*If* $(\tau, i)$ *does not hold, set* $d_{\tau,i} := \infty$. *Define* $d_\tau := \min\{d_{\tau.1}, d_{\tau.2}, d_{\tau.T}\}$.

**Theorem 3.9** (DLog implies Uber for rational fractions in the AGM)**.** *Let* $\mathcal{G}$ *be a bilinear group of type* $\tau \in \{1,2,3\}$ *and let* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F') \in (\mathbb{Z}_p(X_1, \ldots, X_m))^{r+s+f+3}$ *be a tuple of rational fractions that is non-trivial for type* $\tau$ *(Def. 3.3). Let* $q, q_1$ *and* $q_2$ *be such that* $q \geq \check{d}_{\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F}} + \max\{d_{\overrightarrow{R}}, d_{\overrightarrow{S}}, d_{\overrightarrow{F}}/2\}$ *and* $q_1 \geq \check{d}_{\overrightarrow{R}\|\overrightarrow{F}} + d_{\overrightarrow{R}}$ *and* $q_2 \geq \check{d}_{\overrightarrow{S}} + d_{\overrightarrow{S}}$ *as well as* $q_1 + q_2 \geq \check{d}_{\overrightarrow{R}\|\overrightarrow{F}} + \check{d}_{\overrightarrow{S}} + d_{\overrightarrow{F}}$. *If*

(Type 1) $q$-$\mathbf{dlog}_{\mathcal{G}_1}$ *or* $q$-$\mathbf{dlog}_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 2) $q$-$\mathbf{dlog}_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 3) $(q_1, q_2)$-$\mathbf{dlog}_{\mathcal{G}}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

*then* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$-$\mathbf{r\ddot{u}ber}_{\mathcal{G}}$*, as defined in Fig. 3.3, is* $(\varepsilon', t')$*-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

*where* $d_\tau$ *is the maximal degree of* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$*, as defined in Def. 3.8.*

The proof extends the ideas used to prove Theorem 3.5 by employing a technique from [BB08]. Consider a group of Type 1 or 2. The reduction computes $D := \mathrm{Den}(\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F})$, a least common multiple of the denominators of the instance. Given a $q$-DLog instance $g_2, g_2^z, g_2^{z^2}, \ldots$, it first implicitly sets $x_i := y_i z + v_i \bmod p$, then it checks whether any denominator evaluates to zero at $\overrightarrow{x}$ (this entails the additive loss $d_{\mathrm{den}}$). Then it computes a new random generator $h_2 := g_2^{D(y_1 z + v_1, \ldots, y_m z + v_m)}$ and $h_1 := \psi(h_2)$. For rational fractions $S_i = \hat{S}_i / \check{S}_i$, it then uses $h_1, h_2$ to compute the Uber challenge elements $h_2^{S_i(\overrightarrow{x})}$ as $g_2^{\overline{S}(\overrightarrow{x})}$ for the *polynomial* $\overline{S}(\overrightarrow{X}) := (\hat{S}_i \cdot D / \check{S}_i)(\overrightarrow{X})$, and likewise for $R_i$ and $F_i$. This explains the lower bound on $q$ in the theorem statement. When the adversary returns a group element $h_i^{R'(\overrightarrow{x})}$ so that $R'$ is non-trivial, then from the algebraic representations of this element we can define a polynomial (which with overwhelming probability is non-zero) that vanishes at $z$. The difference here is that we expand by the denominator of $R'$ in order to obtain a polynomial. The degree of this polynomial is bounded by the values in Def. 3.8, which also bound the failure probability of the reduction. In Type-3 groups, the reduction can set $h_1 := g_1^{\mathrm{Den}(\overrightarrow{R}\|\overrightarrow{F})(\overrightarrow{x})}$ and $h_1 := g_2^{\mathrm{Den}(\overrightarrow{S})(\overrightarrow{x})}$, which leads to better bounds. We detail this case in our proof of Theorem 3.9, which can be found in Section 3.10.

## 3.4 The Ruber Assumption for Flexible Targets

For rational fractions, we can also define a flexible generalization, where the adversary can choose the target polynomials $R', S'$ and $F'$ in Fig. 3.1 itself, conditioned on the tuple $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R', S', F')$ being non-trivial. The game is specified in Fig. 3.4. This extension covers assumptions such as the $q$-strong DH assumption by Boneh and Boyen [BB08], which they proved secure in the generic group model. A $q$-SDH adversary is given $(g_i, g_i^z, g_i^{z^2}, \ldots, g_i^{z^q})$ for $i = 1, 2$ and must compute $(g_1^{(z+c)^{-1}}, c)$ for any $c \in Z_p \setminus \{-z\}$ of its choice. This is an instance of the flexible game in Fig. 3.4 when setting $m = 1$, $r = s = q + 1$, $f = 0$ and $R_i(X) = S_i(X) = X^{i-1}$, and the adversary returns $R^*(X) = 1/(X + c)$, $S^*(X) = F^*(X) = 0$.

**Theorem 3.10** (DLog implies flexible-target Uber for rational fractions in the AGM)**.** *Let* $\mathcal{G}$ *be a bilinear group of type* $\tau \in \{1,2,3\}$ *and let* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}) \in (\mathbb{Z}_p(X_1, \ldots, X_m))^{r+s+f}$ *be a tuple of rational fractions.*

$(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$-**f-rüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$

01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ ; $g_T \leftarrow e(g_1, g_2)$

02 $\overrightarrow{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$

03 If for some $i$: $\check{R}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{S}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{F}_i(\overrightarrow{x}) \equiv_p 0$ then return 1

04 $\overrightarrow{\mathbf{U}} := (g_1^{R_1(\overrightarrow{x})}, \ldots, g_1^{R_r(\overrightarrow{x})})$ ; $\overrightarrow{\mathbf{V}} := (g_2^{S_1(\overrightarrow{x})}, \ldots, g_2^{S_s(\overrightarrow{x})})$ ; $\overrightarrow{\mathbf{W}} := (g_T^{F_1(\overrightarrow{x})}, \ldots, g_T^{F_f(\overrightarrow{x})})$

05 $((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$

06 Return $\Big((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) = (g_1^{R^*(\overrightarrow{x})}, g_2^{S^*(\overrightarrow{x})}, g_T^{F^*(\overrightarrow{x})})$
$\wedge\ (\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R^*, S^*, F^*)$ non-trivial for type of $\mathcal{G}\Big)$

Figure 3.4: Algebraic game for the flexible-targets Uber assumption

*Consider an adversary* $\mathsf{A}_{\mathsf{alg}}$ *in game* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$-**f-rüber** *(Fig. 3.4) and let* $d_1^*, d_2^*, d_T^*, \check{d}_1^*, \check{d}_2^*, \check{d}_T^*$ *be such that* $\mathsf{A}_{\mathsf{alg}}$*'s outputs* $R^*, S^*, F^*$ *satisfy* $\deg R^* \le d_1^*$, $\deg S^* \le d_2^*$, $\deg F^* \le d_T^*$, $\deg \check{R}^* \le \check{d}_1^*$, $\deg \check{S}^* \le \check{d}_2^*$ *and* $\deg \check{F}^* \le \check{d}_T^*$.

*Let* $q, q_1$ *and* $q_2$ *be such that* $q \ge \check{d}_{\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F}} + \max\{d_{\overrightarrow{R}}, d_{\overrightarrow{S}}, d_{\overrightarrow{F}}/2\}$ *and let* $q_1 \ge \check{d}_{\overrightarrow{R}\|\overrightarrow{F}} + d_{\overrightarrow{R}}$ *and* $q_2 \ge \check{d}_{\overrightarrow{S}} + d_{\overrightarrow{S}}$ *and* $q_1 + q_2 \ge \check{d}_{\overrightarrow{R}\|\overrightarrow{F}} + \check{d}_{\overrightarrow{S}} + d_{\overrightarrow{F}}$, *where* $\check{d}_{\overrightarrow{R}} = \check{d}_{(\hat{R}_1/\check{R}_1, \ldots, \hat{R}_r/\check{R}_r)} = \deg \mathrm{LCM}\{\check{R}_1, \ldots, \check{R}_r\}$. *If*

(Type 1) $q$-**dlog**$_{\mathcal{G}_1}$ *or* $q$-**dlog**$_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 2) $q$-**dlog**$_{\mathcal{G}_2}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

(Type 3) $(q_1, q_2)$-**dlog**$_{\mathcal{G}}$ *is* $(\varepsilon, t)$*-secure in the AGM,*

*then* $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$-**f-rüber**$_{\mathcal{G}}$ *is* $(\varepsilon', t')$*-secure in the AGM with*

$$\varepsilon' \le \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

*where* $d_\tau$ *is defined as in Def. 3.8, except for defining* $d_{\mathrm{den}} := \check{d}_{\overrightarrow{R}\|\overrightarrow{S}\|\overrightarrow{F}}$ *and replacing* $\check{d}_{R'}, \check{d}_{S'}, \check{d}_{F'}$, $d_{R'}, d_{S'}$, *and* $d_{F'}$ *by* $\check{d}_1^*, \check{d}_2^*, \check{d}_T^*, d_1^*, d_2^*$, *and* $d_T^*$), *respectively.*

*Proof sketch.* Much in the way the proof of Theorem 3.5 is adapted to Theorem 3.7, Theorem 3.10 is proved similarly to Theorem 3.9. Since $P_1, P_2$ and $P_T$ are only defined once the adversary returns its rational fractions $R^*, S^*, F^*$, they need not be known in advance. (Note that, unlike for polynomials (Theorem 3.7), the instance $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$ does have to be fixed, as the reduction uses it to set up the generators $h_1$ and $h_2$.) A difference to Theorem 3.10 is the value $d_{\mathrm{den}}$ in the security loss, which is now smaller since the experiment need not check the denominators of the target fractions. □

## 3.5 Uber Assumptions with Decisional Oracles

In this section we show that we can provide the adversary, essentially *for free*, with an oracle that checks whether the logarithms of given group elements satisfy any polynomial relation. In more detail, the adversary is given access to an oracle that takes as input a polynomial $P \in Z_p[X_1, \ldots, X_n]$ and group elements $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ (from any group $\mathbb{G}_1, \mathbb{G}_2$ or $\mathbb{G}_T$) and checks whether $P(\log \mathbf{Y}_1, \ldots, \log \mathbf{Y}_n) \equiv_p 0$. Decisional oracles can be added to any type of Uber assumption;

$(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$-**f-drüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$

01 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ ; $g_T \leftarrow e(g_1, g_2)$
02 $\overrightarrow{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$
03 If for some $i$: $\check{R}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{S}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{F}_i(\overrightarrow{x}) \equiv_p 0$
04 then return 1
05 $\overrightarrow{\mathbf{U}} := (g_1^{R_1(\overrightarrow{x})}, \ldots, g_1^{R_r(\overrightarrow{x})})$
06 $\overrightarrow{\mathbf{V}} := (g_2^{S_1(\overrightarrow{x})}, \ldots, g_2^{S_s(\overrightarrow{x})})$
07 $\overrightarrow{\mathbf{W}} := (g_T^{F_1(\overrightarrow{x})}, \ldots, g_T^{F_f(\overrightarrow{x})})$
08 $((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$
09 Return $\Big((\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) = (g_1^{R^*(\overrightarrow{x})}, g_2^{S^*(\overrightarrow{x})}, g_T^{F^*(\overrightarrow{x})})$
$\wedge\, (\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F}, R^*, S^*, F^*)$ non-trivial for type of $\mathcal{G}\Big)$

$\underline{\mathsf{O}}(P(\overrightarrow{X}), (\mathbf{Y}_1, \ldots, \mathbf{Y}_n))$
10 For $i = 1, \ldots, n$ do
11 let $\iota_i \in \{1, 2, T\}$ s.t. $\mathbf{Y}_i \in \mathbb{G}_{\iota_i}$
12 $y_i \leftarrow \log_{g_{\iota_i}} \mathbf{Y}_i$
13 Return $(P(\overrightarrow{y}) \equiv_p 0)$

Figure 3.5: Algebraic game for the flexible-targets Uber assumption with decisional oracles

for concreteness, we extend the most general variant from the previous section. The game **f-drüber** ("d" for decisional oracles) is defined in Fig. 3.5. This extension covers assumptions such as Gap Diffie-Hellman (DH) [OP01], where the adversary must solve a DH instance while being given an oracle that checks whether a triple $(\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3)$ is a DH tuple, i.e., $\mathbf{Y}_1^{\log \mathbf{Y}_2} = \mathbf{Y}_3$. This oracle is a special case of the one in Fig. 3.5, when called with $P(X_1, X_2, X_3) := X_1 X_2 - X_3$.

**Theorem 3.11** (DLog implies flexible-target Uber for rational fractions with decisional oracles in the AGM)**.** *The statement of Theorem 3.10 holds when* **f-rüber** *is replaced by* **f-drüber**.

*Proof sketch.* The reduction $\mathsf{B}_{\mathsf{alg}}$ from $(\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{F})$-**f-drüber** to $q$-DLog (or $(q_1, q_2)$-DLog) works as for Theorem 3.10 (as detailed in the proof of Theorem 3.9), except that $\mathsf{B}_{\mathsf{alg}}$ must also answer $\mathsf{A}_{\mathsf{alg}}$'s oracle queries, which we describe in the following for Type-3 groups.

As for Theorem 3.10, $\mathsf{B}_{\mathsf{alg}}$, on input $(\overrightarrow{\mathbf{Y}}, \overrightarrow{\mathbf{Z}})$ with $\mathbf{Y}_i = [z^i]_1$ and $\mathbf{Z}_j = [z^j]_2$, for $0 \le i \le q_1$ and $0 \le j \le q_2$, computes LCMs of denominators $D := \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S} \| \overrightarrow{F})$, $D_1 := \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{F})$ and $D_2 := \mathrm{Den}(\overrightarrow{S})$. It picks $\overrightarrow{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\overrightarrow{v} \xleftarrow{\$} \mathbb{Z}_p^m$, implicitly sets $x_i := y_i z + v_i \bmod p$ and checks if $D(\overrightarrow{x}) \equiv_p 0$. If so, the reduction derives the corresponding univariate polynomial and finds $z$. Otherwise it computes $h_1 := [D_1(\overrightarrow{x})]_1$, $h_2 := [D_2(\overrightarrow{x})]_2$ (note that $D_1(\overrightarrow{x})$ and $D_2(\overrightarrow{x})$ are non-zero), $\mathbf{U}_i = [(D_1 \cdot R_i)(\overrightarrow{x})]_1$, $\mathbf{V}_i = [(D_2 \cdot S_i(\overrightarrow{x})]_2$ and $\mathbf{W}_i = [(D_1 \cdot D_2 \cdot F_i)(\overrightarrow{x})]_T$.

Consider a query $\mathsf{O}(P, (\mathbf{Y}_1, \ldots, \mathbf{Y}_n))$ for some $n$ and $P \in Z_p[X_1, \ldots, X_n]$, and $\mathbf{Y}_i \in \mathbb{G}_{\iota_i}$ for $\iota_i \in \{1, 2, T\}$. Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic, it provides representations of the group elements $\mathbf{Y}_i$ with respect to its input $(\overrightarrow{\mathbf{U}}, \overrightarrow{\mathbf{V}}, \overrightarrow{\mathbf{W}})$; in particular, for each $\mathbf{Y}_i$, depending on the group, it provides $\overrightarrow{\mu}_i$ or $\overrightarrow{\eta}_i$ or $(A_i, \overrightarrow{\delta}_i)$ such that:

$$(\mathbf{Y}_i \in \mathbb{G}_1) \quad \mathbf{Y}_i = \textstyle\prod_{j=1}^r \mathbf{U}_j^{\mu_{i,j}} = \left[\sum_{j=1}^r \mu_{i,j}(D_1 \cdot R_j)(\overrightarrow{x})\right]_1 =: [Q_i(z)]_1$$

$$(\mathbf{Y}_i \in \mathbb{G}_2) \quad \mathbf{Y}_i = \textstyle\prod_{j=1}^s \mathbf{V}_j^{\eta_{i,j}} = \left[\sum_{j=1}^s \eta_{i,j}(D_2 \cdot S_j)(\overrightarrow{x})\right]_2 =: [Q_i(z)]_2$$

$$\begin{aligned}(\mathbf{Y}_i \in \mathbb{G}_T) \quad \mathbf{Y}_i &= \textstyle\prod_{j=1}^r \prod_{k=1}^s e(\mathbf{U}_j, \mathbf{V}_k)^{\alpha_{i,j,k}} \cdot \prod_{j=1}^f \mathbf{W}_j^{\delta_{i,j}} \\ &= \left[\sum_{j=1}^r \sum_{k=1}^s \alpha_{i,j,k}(D_1 \cdot R_j)(\overrightarrow{x})\,(D_2 \cdot S_j)(\overrightarrow{x}) + \sum_{j=1}^f \delta_{i,j}(D_1 \cdot D_2 \cdot F_j)(\overrightarrow{x})\right]_T =: [Q_i(z)]_T,\end{aligned}$$

where $D_1 \cdot R_j$ as well as $D_2 \cdot S_j$ and $D_1 \cdot D_2 \cdot F_j$ are multivariate polynomials (not rational fractions) and $Q_i$ is the polynomial defined by replacing $X_i$ by $y_i Z + v_i$. Let $D_i'(Z)$ be defined from $D_i(\vec{X})$ analogously. Then we have $\log_{g_{\iota_i}} \mathbf{Y}_i = Q_i(z)$ and furthermore $\log_{h_{\iota_i}} \mathbf{Y}_i = Q_i(z)/D_{\iota_i}(z)$, where $D_T := D_1 \cdot D_2$.

To answer the oracle query, $\mathsf{B}_{\mathsf{alg}}$ must therefore determine whether the function $P(Q_1/D_{i_1}, \ldots, Q_n/D_{i_n})$ vanishes at $z$. Since $D_1(z), D_2(z) \not\equiv_p 0$, this is the case precisely when $\overline{P} := D_1^d \cdot D_2^d \cdot P(Q_1/D_{i_1}, \ldots, Q_n/D_{i_n})$ vanishes at $z$, where $d$ is the maximal degree of $P$. Note that $\overline{P}$ is a polynomial. The reduction distinguishes 3 cases:

1. $\overline{P} \equiv 0$: in this case, the oracle replies 1.
2. $\overline{P} \not\equiv 0$: in this case, $\mathsf{B}_{\mathsf{alg}}$ factorizes $\overline{P}$ to find its roots $z_1, \ldots$, checks whether $\mathbf{Z}_1 = g^{z_i}$ for some $i$. If this is the case, it stops and returns the solution $z_i$ to its $(q_1, q_2)$-DLog instance.
3. Else, the oracle replies 0.

Correctness of the simulation is immediate, since the correct oracle reply is 1 if and only if $\overline{P}(z) \equiv_p 0$. □

## 3.6 The Flexible Gegenuber Assumption

In this section, we show how to extend the Uber framework even further, by letting the adversary generate its own generators (for the outputs), yielding the *GeGenUber* assumption. Consider the LRSW assumption [LRSW99] in Type-1 bilinear groups: given $(X = g^x, Y = g^y)$ (which can be viewed as a signature verification key [CL04]) and an oracle, which on input (a message) $m \in \mathbb{Z}_p$ returns (a signature) $(g^a, g^{ay}, g^{a(x+mxy)})$ for a random $a \xleftarrow{\$} \mathbb{Z}_p$, it is infeasible to return (a signature on a fresh message) $((g^{a^*}, g^{a^*y}, g^{a^*(x+m^*xy)}, m^*))$ for any $a^*$ and $m^*$ different from the queried values. Since the adversary need not return the value $a^*$, this cannot be cast into the Uber framework. Associating the values $a_1, \ldots, a_\ell$ chosen by the signing oracle to formal variables $A_1, \ldots, A_\ell$, in the Uber framework $\vec{X}$ would correspond to $(X, Y, A_1, \ldots, A_\ell)$ and signing queries to the polynomials $A_i$, $A_i Y$ and $A_i X + m_i A_i XY$. Now the adversary can choose a fresh generator $g^* := g^{a^*}$ and must return $((g^*)^{R_i^*(\vec{X})})_{i=1}^3$ for $R_1^* \equiv 1, R_2^* = Y$ and $R_3^* = X + m^* XY$ for some $m^* \in \mathbb{Z}_p^*$ of its choice.

Our last generalization now extends the flexible Uber assumption from Sect. 3.2 by letting the adversary generate its own generators $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, resp., and return polynomials $R^*, S^*$ and $F^*$, as well as $(\mathbf{U}^{R^*(\vec{x})}, \mathbf{V}^{S^*(\vec{x})}, \mathbf{W}^{F^*(\vec{x})})$. The game $m$-**gegenüber** is defined in Fig. 3.6. This additional freedom for the adversary induces a necessary change in the definition of non-triviality, as illustrated by the following simple (univariate) example: after the challenger chooses $x \xleftarrow{\$} Z_p$, the adversary makes queries $R_1 := X$ and $R_2 := X^3$ and receives $\mathbf{U}_1$ and $\mathbf{U}_2$. For all Uber assumptions so far, the polynomial $R^* := X^2$ would be considered non-trivial. However, in game **gegenüber** the adversary could return $\mathbf{U} := \mathbf{U}_1 = g^x$ and $\mathbf{U}^* := \mathbf{U}_2 = g^{x^3} = \mathbf{U}^{R^*(x)}$.

Whereas until now the target polynomial $R^*$ was not allowed to be a linear combination of the queried polynomials $P_1, \ldots, P_\ell$ (or of products of such polynomials, depending on the group types), for the Gegenuber assumption we also need to exclude fractions of such linear combinations (such as $X^3/X$ in the example above) to thwart trivial attacks.

For a family $E$ of polynomials, we denote by $\mathrm{Span}(E)$ all linear combinations of elements of $E$, which we extend to fractions as

$$\mathrm{FrSp}(E) := \{\hat{P}/\check{P} \mid (\hat{P}, \check{P}) \in \mathrm{Span}(E) \times (\mathrm{Span}(E) \setminus \{0\})\}.$$

Moreover, by $E_1 * E_2$ we denote the set $\{P_1 \cdot P_2 \mid (P_1, P_2) \in E_1 \times E_2\}$.

$m$-**gegenüber**$^{\mathsf{A}_{\mathsf{alg}}}_{\mathcal{G}}$

01 $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T \leftarrow \emptyset$
02 $(g_1, g_2) \xleftarrow{\$} \mathsf{GenSamp}$ ; $g_T \leftarrow e(g_1, g_2)$
03 $\overrightarrow{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$
04 $((\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot,\cdot)}()$
05 Return $\Big(\mathbf{U} \neq 1 \wedge \mathbf{V} \neq 1 \wedge \mathbf{W} \neq 1$
$\wedge\ (\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) = (\mathbf{U}^{R^*(\overrightarrow{x})}, \mathbf{V}^{S^*(\overrightarrow{x})}, \mathbf{W}^{F^*(\overrightarrow{x})})$
$\wedge\ (\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ gegenuber-non-trivial for type of $\mathcal{G}\Big)$

$\mathsf{O}(i, P(\overrightarrow{X}))$
06 $\mathcal{Q}_i = \mathcal{Q}_i \cup \{P\}$
07 Return $g_i^{P(\overrightarrow{x})}$

Figure 3.6: Algebraic game for the flexible Gegenuber assumption

**Definition 3.12** (Non-triviality for Gegenuber assumption)**.** *Let $\mathcal{Q}_1, \mathcal{Q}_2$ and $\mathcal{Q}_T$ be sets of polynomials and let $R^*, S^*$ and $F^*$ be polynomials. We say that $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ is* gegenuber-non-trivial *for groups of type $\tau$, if the following holds:*

- **Either** $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$ *for* $\tau \in \{1, 2\}$ *and* $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1)$ *for* $\tau = 3$, *(*$\tau$*.1)*
- **or** $S^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$ *for* $\tau = 1$ *and* $S^* \notin \mathrm{FrSp}(\mathcal{Q}_2)$ *for* $\tau \in \{2, 3\}$, *(*$\tau$*.2)*
- **or** $F^* \notin \mathrm{FrSp}(\mathcal{Q}_T \cup (\mathcal{Q}_1 \cup \mathcal{Q}_2) * (\mathcal{Q}_1 \cup \mathcal{Q}_2))$ *for* $\tau = 1$ *(1.*$T$*)*
  $F^* \notin \mathrm{FrSp}(\mathcal{Q}_T \cup (\mathcal{Q}_1 \cup \mathcal{Q}_2) * \mathcal{Q}_2)$ *for* $\tau = 2$ *(2.*$T$*)*
  $F^* \notin \mathrm{FrSp}(\mathcal{Q}_T \cup (\mathcal{Q}_1 * \mathcal{Q}_2))$ *for* $\tau = 3$. *(3.*$T$*)*

**Definition 3.13** (Degree of non-triviality for Gegenuber assumption)**.** *Let $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ be a non-trivial tuple of polynomials in $\mathbb{Z}_p[X_1, \ldots, X_m]$. For $i \in \{1, 2, T\}$ define $d'_i := \max\{\deg P\}_{P \in \mathcal{Q}_i}$. We define the* type-$\tau$ degree $d_\tau$ *of $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ as follows:*

- *If ($\tau$.1) holds, let* $d_{\tau.1} := \max\{1, \deg R^*\} \cdot \max\{d'_1, d'_2\}$ *in case* $\tau \in \{1, 2\}$ *and*
  $d_{\tau.1} := \max\{1, \deg R^*\} \cdot d'_1$ *in case* $\tau = 3$.
- *If ($\tau$.2) holds, let* $d_{\tau.2} := \max\{1, \deg S^*\} \cdot \max\{d'_1, d'_2\}$ *in case* $\tau = 1$ *and*
  $d_{\tau.2} := \max\{1, \deg S^*\} \cdot d'_2$ *in case* $\tau \in \{2, 3\}$.
- *If ($\tau.T$) holds, let* $d_{\tau.T} := \max\{1, \deg F^*\} \cdot \max\{2\,d'_1, 2\,d'_2, d'_T\}$ *for* $\tau = 1$
  $d_{\tau.T} := \max\{1, \deg F^*\} \cdot \max\{d'_1 + d'_2, 2\,d'_2, d'_T\}$ *for* $\tau = 2$,
  $d_{\tau.T} := \max\{1, \deg F^*\} \cdot \max\{d'_1 + d'_2, d'_T\}$ *for* $\tau = 3$.

*If $(\tau, i)$ does not hold, set $d_{\tau,i} := \infty$. Define $d_\tau := \min\{d_{\tau.1}, d_{\tau.2}, d_{\tau.T}\}$.*

Note that for all Uber variants, the adversary only outputs one element per group $\mathbb{G}_i$, which is without loss of generality, as a vector of group elements would be non-trivial if at least one component is non-trivial. We defined the Gegenuber assumption analogously, so one might wonder how this covers LRSW, where the adversary must output group elements corresponding to two polynomials $Y$ and $X + m^*XY$. The reason is that LRSW holds even if the adversary only has to output the latter polynomial (the former is required to verify the validity of a solution using the pairing): In the GGM this follows from LRSW being an instance of Gegenuber, Theorem 3.14 (see below for both) and Lemmas 2.5 and 2.9.

To show that LRSW is gegenuber-non-trivial, consider the set of queries an adversary can make, namely:

$$\mathcal{Q} := \{1, X, Y, \{A_i, A_iY, A_iX + m_iA_iXY\}_{i=1}^{\ell}\}.$$

To prove that the stronger variant of LRSW satisfies non-triviality as defined in Def. 3.12, we show that for $0 \neq m^* \notin \{m_1, \ldots, m_\ell\}$: $R^* := X + m^*XY \notin \mathrm{FrSp}(\mathcal{Q})$.

For the sake of contradiction, assume that for some $\check{P}, \hat{P} \in \mathrm{Span}(\mathcal{Q})$, $\check{P} \not\equiv 0$:

$$(X + m^*XY)\check{P} = \hat{P}. \tag{3.6}$$

Since $\hat{P} \in \mathrm{Span}(\mathcal{Q})$, its total degree in $X$ and $Y$ is at most 2, which implies that $\check{P}$ must be of degree 0 in $X$ and $Y$. We can thus write $\check{P}$ as $\eta + \sum_{j=1}^{\ell} \alpha_j A_j$ for some $\eta$ and $\vec{\alpha}$. Since $X$ is a factor of the left-hand side of (3.6), $\hat{P}$ cannot have terms without $X$ and must therefore be of the form $\hat{P} = \xi X + \sum_{j=1}^{\ell} \mu_j A_j \, (X + m_j XY)$ for some $\xi$ and $\vec{\mu}$. Equation (3.6) becomes thus

$$(X + m^*XY) \left(\eta + \textstyle\sum_{j=1}^{\ell} \alpha_j A_j\right) = \xi X + \sum_{j=1}^{\ell} \mu_j A_j \, (X + m_j XY) \, .$$

By equating coefficients, we get: $\eta = \xi$ (from coeff. $X$) and $m^*\eta \equiv_p 0$ (from $XY$) and for all $j \in [1, \ell]$: $\alpha_j = \mu_j$ (from $A_j X$) and $\alpha_j m^* \equiv_p \mu_j m_j$ (from $A_j XY$). Since $m^* \neq 0$, we have $\eta = \xi = 0$. Furthermore, if $\alpha_j \neq 0$ for some $j$, then $m_j = m^*$, meaning it was not a valid solution. If $\alpha_j = 0$ for all $j$ then $\check{P} \equiv 0$, which shows such $\hat{P}$ and $\check{P}$ do not exist and thus $X + m^*XY \notin \mathrm{FrSp}(\mathcal{Q})$.

Another assumption covered by this definition is the AWFCDH assumption 2.8 in type 2 (and therefore, it implies the hardness for the assumption in type 3): It corresponds to the case $R = R^* = S^* = X$ and $\mathbf{U}^* = \psi(\mathbf{V}^*)$.

**Theorem 3.14.** *Let $m \geq 1$, let $\mathcal{G}$ be a bilinear group of type $\tau \in \{1, 2, 3\}$ and let $\mathsf{A}_{\mathsf{alg}}$ be an adversary in game $m$-**gegenüber**$_{\mathcal{G}}$. Let $d_1', d_2', d_T', d_1^*, d_2^*, d_T^*$ be such that $\mathsf{A}_{\mathsf{alg}}$'s queries $(i, P(\vec{X}))$ satisfy $\deg P \leq d_i'$ and its output satisfies $\deg R^* \leq d_1^*$, $\deg S^* \leq d_2^*$, $\deg F^* \leq d_T^*$. Let $q, q_1, q_2$ be such that $q \geq \max\{d_1', d_2', d_T'/2\}$, as well as $q_1 \geq d_1'$, $q_2 \geq d_2'$ and $q_1 + q_2 \geq d_T'$. If*

(Type 1) *$q$-**dlog**$_{\mathcal{G}_1}$ or $q$-**dlog**$_{\mathcal{G}_2}$ is $(\varepsilon, t)$-secure in the AGM,*

(Type 2) *$q$-**dlog**$_{\mathcal{G}_2}$ is $(\varepsilon, t)$-secure in the AGM,*

(Type 3) *$(q_1, q_2)$-**dlog**$_{\mathcal{G}}$ is $(\varepsilon, t)$-secure in the AGM,*

*then **gegenüber**$_{\mathcal{G}}$ is $(\varepsilon', t')$-secure in the AGM with*

$$\varepsilon' \leq \varepsilon + \tfrac{d_\tau}{p-1} \quad \text{and} \quad t' \approx t,$$

*with $d_\tau$ from Def. 3.13 after replacing $\deg R^*$, $\deg S^*$ and $\deg F^*$ by $d_1^*, d_2^*$ and $d_T^*$, respectively.*

The proof can be found in Section 3.11. It is an adaptation of the one for the flexible uberassumption, which adapts the proof of Theorem 3.5. We highlight the main difference for non-triviality of type (2.1). Recall that in the proof Theorem 3.5, the adversary returns a solution $\mathbf{U}'$ and its algebraic representation so that (3.1a) holds. From this and the fact that $\mathbf{U}' = [R'(\vec{x})]_1$, we derived the polynomial $P_1$ in (3.2), which is non-zero by non-triviality.

In the proof of Theorem 3.14, the adversary also outputs a new generator $\mathbf{U}$, whose algebraic representation yields a polynomial $Q \in \mathrm{Span}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$ so that $\mathbf{U} = [Q(\vec{x})]_1$. For a valid solution $\mathbf{U}^*$, we thus have $\mathbf{U}^* = [R^*(\vec{x}) \cdot Q(\vec{x})]_1$. On the other hand, the representation of $\mathbf{U}^*$ yields $\mathbf{U}^* = [Q^*(\vec{x})]_1$ for some $Q^* \in \mathrm{Span}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$. We thus have that $P_1(\vec{X}) := R^*(\vec{X}) \cdot Q(\vec{X}) - Q^*(\vec{X})$ vanishes at $\vec{x}$. By non-triviality, $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$, which implies $P_1 \not\equiv 0$, so the reduction can find the roots of $P_1(y_1 Z + x_1, \ldots, y_n Z + x_n)$ and solve $q$-DLog.

## 3.7 Separation of $(q+1)$-DL from $q$-DL

Now that we have shown that every Uber assumption falls into a (minimal) class of assumptions that are equivalent to $q$-DLog, we show that these classes can be separated according to their parameter $q$. We prove that, assuming that $q$-DLog is hard, there does not exist an algebraic reduction from $q$-DLog to $(q+1)$-DLog. In particular, we show that if there exists a reduction $\mathsf{R}_{\mathsf{alg}}$ that has access to a $(q+1)$-DLog (algebraic) adversary $\mathsf{A}_{\mathsf{alg}}$ and can solve $q$-DLog, then there exists a meta-reduction that uses $\mathsf{R}_{\mathsf{alg}}$ to break $q$-DLog. In the following, we use the notation $\mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ to denote that $\mathsf{R}_{\mathsf{alg}}$ has complete access to $\mathsf{A}_{\mathsf{alg}}$'s internal state. In particular, $\mathsf{R}_{\mathsf{alg}}$ is allowed to rewind $\mathsf{A}_{\mathsf{alg}}$ to any point of an execution and run $\mathsf{A}_{\mathsf{alg}}$ on any choice of random coins as many times as it wants.

**Theorem 3.15.** *Let $\mathcal{G}_i$ be a group of prime order $p$. There exists an algorithm $\mathsf{M}$ such that the following holds. Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic algorithm s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that $(t,\epsilon)$-breaks $(q+1)$-**dlog**$_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ is an algorithm that $(t',\epsilon')$-breaks $q$-**dlog**$_{\mathcal{G}_i}$. If $t \geq 2\,(2q+1)\lceil \log_2 p \rceil$ then $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ $(t',\epsilon')$-breaks $q$-**dlog**$_{\mathcal{G}_i}$.*

We start with a proof overview. Consider a reduction $\mathsf{R}_{\mathsf{alg}}$, which on input a $q$-DLog instance $(g, g^x, \ldots, g^{x^q})$ can run an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ multiple times on $(q+1)$-DLog instances $(\mathbf{Z}, \mathbf{Z}^y, \ldots, \mathbf{Z}^{y^{q+1}})$; that is, $\mathsf{R}_{\mathsf{alg}}$ can choose a new generator $\mathbf{Z}$ and a new problem solution $y$. Since $\mathsf{R}_{\mathsf{alg}}$ is algebraic, it outputs a representation of the group elements composing its $(q+1)$-DLog instance in terms of the received $q$-DLog instance. We distinguish two cases: if (a) $y$ is independent from $x$ then the representation reveals $y$, which means that a meta-reduction $\mathsf{M}$ can simulate a successful $\mathsf{A}_{\mathsf{alg}}$ to $\mathsf{R}_{\mathsf{alg}}$, and the latter must thus find $x$. On the other hand, if (b) $y$ depends on $x$, then this yields a non-trivial equation in $x$, which the meta-reduction can solve and thereby (without needing to simulate $\mathsf{A}_{\mathsf{alg}}$) solve the $q$-DLog instance.

To simplify the probability analysis, we let $\mathsf{M}$ simulate $\mathsf{A}_{\mathsf{alg}}$ even when $\mathsf{R}_{\mathsf{alg}}$ behaves as in the second case (as it can compute $y$ from $x$). To correctly argue about the probability distributions, we ensure that any malformed instance provided by the algebraic reduction $\mathsf{R}_{\mathsf{alg}}$ is detected. We will use the following lemma in the proof of Theorem 3.15:

**Lemma 3.16.** *Let $q \geq 1$, let $F \in \mathbb{Z}_p(X)$ and let $0 \not\equiv P \in \mathbb{Z}_p[X]$ be of degree at most $q$. If $F^{q+1} \cdot P$ is a polynomial and of degree at most $q$, then $F$ is constant.*

*Proof.* Let $\hat{F}, \check{F} \in \mathbb{Z}_p[X]$ be coprime such that $F = \hat{F}/\check{F}$. Then $\hat{F}^{q+1}$ and $\check{F}^{q+1}$ are coprime as well. From this and the premise that $\hat{F}^{q+1} \cdot P/\check{F}^{q+1}$ is a polynomial, we get that $\check{F}^{q+1}$ divides $P$, and thus $(q+1) \cdot \deg \check{F} \leq \deg P$. Since the latter is at most $q$, we have $\deg \check{F} = 0$.

Furthermore, we assumed that $q \geq \deg(F^{q+1} \cdot P) = (q+1) \cdot \deg \hat{F} + \deg P$, and thus $\deg \hat{F} = 0$. Together, this means $F$ is constant. □

*Proof of Theorem 3.15.* Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic algorithm s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that $(t,\epsilon)$-breaks $(q+1)$-**dlog**$_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ is an algorithm that $(t',\epsilon')$-breaks $q$-**dlog**$_{\mathcal{G}_i}$. In the following, we describe a meta-reduction $\mathsf{M}$ s.t. $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ $(t',\epsilon')$-breaks $q$-**dlog**$_{\mathcal{G}_i}$.

$\mathsf{M}(g, \mathbf{X}_1, \ldots, \mathbf{X}_q)$**:** Run $\mathsf{R}_{\mathsf{alg}}$ on the received $q$-DLog instance $(g, g^x, \ldots, g^{x^q})$. Whenever $\mathsf{R}_{\mathsf{alg}}$ runs adversary $\mathsf{A}_{\mathsf{alg}}$ on $(q+1)$-DLog input $(\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_{q+1})$, do the following. Let $\overrightarrow{z}_i = (z_{i,0}, \ldots, z_{i,q})$ for $0 \leq i \leq q+1$ be the representation vectors for $\mathbf{Z}_0, \ldots, \mathbf{Z}_{q+1}$ provided by $\mathsf{R}_{\mathsf{alg}}$; that is, $\mathbf{Z}_i = \prod_{j=0}^{q} (g^{x^j})^{z_{i,j}}$.

If $\mathbf{Z}_0 = 1$ then return $\perp$. $(*)$

Else define

$$P_i(X) := \sum_{j=0}^{q} z_{i,j} X^j \quad \text{for } 0 \le i \le q+1 \quad \text{and} \tag{3.7}$$

$$Q_i := P_{i+1}P_0 - P_i P_1 \quad \text{for } 0 \le i \le q. \tag{3.8}$$

$\mathsf{M}$ now distinguishes two cases:

(a) $Q_i \equiv 0$ for all $i \in [0,q]$: Then (as we argue below) $P_1/P_0 \equiv c$, that is, a constant polynomial. $\mathsf{M}$ returns $c$ as $\mathsf{A}_{\mathsf{alg}}$'s output.

(b) For some $k \in [0,q]$: $Q_k \not\equiv 0$: Compute the roots $x_1, \ldots$ of $Q_k$ and check if for some $j$: $g^{x_j} = \mathbf{X}_1$. If not, then return $\perp$ as $\mathsf{A}_{\mathsf{alg}}$'s output. $(**)$
Else let $y := P_1(x_j)/P_0(x_j) \bmod p$. ($1 \neq \mathbf{Z}_0 = g^{P_0(x_j)}$, thus $P_0(x_j) \not\equiv_p 0$.) If for some $i \in [1, q+1] : \mathbf{Z}_i \neq \mathbf{Z}_0^{y^i}$, return $\perp$ as $\mathsf{A}_{\mathsf{alg}}$'s output. $(***)$
Else return $y$.

**Correctness of simulation.** We now argue that $\mathsf{M}$ always correctly simulates an adversary $\mathsf{A}_{\mathsf{alg}}$ that solves $(q+1)$-DLog if it received a correct instance, and returns $\perp$ otherwise. Consider the case where $\mathsf{R}_{\mathsf{alg}}$ provides a valid $(q+1)$-DLog instance, that is $\mathbf{Z}_0 \neq 1$ and

$$\exists\, y \in \mathbb{Z}_p^* \; \forall\, i \in [1, q+1] : \mathbf{Z}_i = \mathbf{Z}_0^{y^i}. \tag{3.9}$$

By (3.7), we have $\mathbf{Z}_i = g^{P_i(x)}$ for all $i$. Since $\mathbf{Z}_0 \neq 1$, $\mathsf{M}$ does not stop in line $(*)$ and $P_0(x) \not\equiv_p 0$. Moreover, from (3.9) we have $y \equiv_p P_1(x)/P_0(x)$ and

$$P_{i+1}(x) \equiv_p P_1(x)/P_0(x) \cdot P_i(x), \tag{3.10}$$

in other words, $Q_i(x) \equiv_p 0$ for all $i \in [0,q]$.

In case (a), $Q_i \equiv 0$, and thus, letting $F := P_1/P_0$, we have (by definition (3.8)) $P_{i+1} = F \cdot P_i$ for all $i$, and by induction:

$$\forall\, i \in [0, q+1] : P_i = F^i \cdot P_0,$$

and in particular $P_{q+1} = F^{q+1} \cdot P_0$. Since $P_{q+1}$ and $P_0$ are polynomials of degree at most $q$, by Lemma 3.16 we get $F \equiv c$ for $c \in \mathbb{Z}_p$. The meta-reduction $\mathsf{M}$ thus returns $c \equiv_p F(x) \equiv_p P_1(x)/P_0(x) \equiv_p y$.

In case (b), since $Q_k \not\equiv 0$ but, by (3.10), $Q_k(x) \equiv_p 0$, the meta-reduction finds $x$ and $\mathsf{M}$ does not stop in line $(**)$ and neither in line $(***)$, since $y \equiv_p P_1(x)/P_0(x)$.

Now consider the case that $\mathsf{R}_{\mathsf{alg}}$ sends a malformed instance: if $\mathbf{Z}_0$ is not a generator then $\mathsf{A}_{\mathsf{alg}}$ returns $\perp$ in line $(*)$. Assume $\mathbf{Z}_0$ is a generator (and thus $P_0(x) \not\equiv_p 0$), but (3.9) is not satisfied. Using the algebraic representations of $\mathbf{Z}_0, \ldots, \mathbf{Z}_{q+1}$, this is equivalent to

$$\forall\, y \in \mathbb{Z}_p^* \; \exists\, k \in [1, q+1] : P_k(x) \not\equiv_p y^k P_0(x). \tag{3.11}$$

We first show that the meta-reduction $\mathsf{M}$ goes to case (b): Indeed, if $Q_i \equiv 0$ for all $i \in [0,q]$, then $P_{i+1}(x) \equiv_p P_1(x)/P_0(x) \cdot P_i(x)$ for all $i$ and, by induction, $P_i(x) \equiv_p (P_1(x)/P_0(x))^i \cdot P_0(x)$, which, setting $y := P_1(x)/P_0(x) \bmod p$, contradicts (3.11).

Let $k \in [0,q]$ be such that $Q_k \not\equiv 0$. If $x$ is not among the roots of $Q_k$, then $\mathsf{M}$ returns $\perp$ in line $(**)$. Otherwise, it sets $y := P_1(x)/P_0(x) \bmod p$. If for some $i \in [1, q+1]$: $P_i(x) \not\equiv_p y^i P_0(x)$ then $\mathsf{M}$ returns $\perp$ in line $(***)$. If not then this contradicts (3.11). Therefore, the simulation will return $\perp$ on invalid inputs.

For the simulation of $\mathsf{A}_{\mathsf{alg}}$ the meta-reduction needs to compute at most $2q+1$ exponentiations, each of which require at most $2\lceil \log_2 p \rceil$ group operations using square and multiply. The simulation of

$\mathsf{A}_{\mathsf{alg}}$ is thus perfect and takes at most $t$ steps. The meta-reduction $\mathsf{M}$ succeeds in winning $q$-$\mathbf{dlog}_{\mathcal{G}_i}$ whenever $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}(\mathsf{A}_{\mathsf{alg}})$ wins $q$-$\mathbf{dlog}_{\mathcal{G}_i}$. Therefore, we obtain

$$\Pr\left[q\text{-}\mathbf{dlog}^{\mathsf{M}}_{\mathcal{G}_i} = 1\right] = \Pr\left[q\text{-}\mathbf{dlog}^{\mathsf{B}}_{\mathcal{G}_i} = 1\right] \geq \varepsilon'.$$

Moreover, the running time of $\mathsf{M}$ is that of $\mathsf{B}$, i.e., $t'$. This completes the proof. □

A similar result can be shown for $(q_1, q_2)$-$\mathbf{dlog}_{\mathcal{G}}$, that is, $(q_1, q_2)$-$\mathbf{dlog}_{\mathcal{G}}$ is not implied by $(q_1', q_2')$-$\mathbf{dlog}_{\mathcal{G}}$ if $q_1' > q_1$ or $q_2' > q_2$.

## 3.8 Separation of 2-One-More DL from $q$-DL

We conclude with showing that "one-more"-discrete logarithm (OMDL) assumptions fall outside of our $q$-DLog taxonomy. While it is known that there is no black-box reduction from DLog to OMDL [BMV08], we show that there is no algebraic reduction either, even one for algebraic adversaries.

To obtain the strongest possible impossibility result, we show that for no $q \in \mathbb{N}$, there exists an algebraic reduction from $q$-DLog (a stronger assumption than DLog) to 2-OMDL (the weakest variant of OMDL assumptions), unless $q$-DLog is easy. The game for 2-OMDL is depicted in Fig. 3.7. The proof uses the same high-level idea as for Theorem 3.17. If the representation of the group elements the reduction gives to the adversary is independent of its own $q$-DLog challenge, then the meta-reduction can directly simulate the adversary. Else, they depend on the $q$-DLog challenge in a way that allows the meta-reduction to derive a $q$-DLog solution. Compared to the previous section, we now restrict the algebraic reduction to only have black-box access (according to our notion of black-box) to the adversary. This is because a reduction that can choose the random coins of the adversary in a non-uniform (adaptive) way can make the simulation of the adversary by our meta-reduction fail.

We need to define the adversary's behavior, that is, its oracle call, beforehand; in particular, it must not depend on the type of representations obtained from the reduction, which makes the proof more complicated and restricts the simulation to adversaries that can fail with negligible probability. The adversary, after receiving a 2-OMDL challenge $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, makes a query $(\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2})$ for random $r_1, r_2$ to its DLog oracle and then returns the 2-OMDL solution $(\log_{\mathbf{Y}_0}\mathbf{Y}_1, \log_{\mathbf{Y}_0}\mathbf{Y}_2)$. We now show how the meta-reduction simulates this adversary.

Since the reduction is algebraic, it provides with its 2-OMDL instance representations $\overrightarrow{z}_i = (z_{i,0}, \ldots, z_{i,q})$ in terms of its $q$-DLog challenge $(g, g^x, \ldots, g^{x^q})$, such that $\log_g \mathbf{Y}_i \equiv_p \sum_{j=0}^{q} z_{i,j}x^j$. From the reply $y$ to the adversary's single oracle query, we get the following equation: $0 = \log_g(\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}) - \log_g \mathbf{Y}_0^{y} \equiv_p \sum_{j=0}^{q}(r_1 z_{1,j} + r_2 z_{2,j} - y z_{0,j})x^j$.

The $q$-DLog challenge $x$ is thus the root of the polynomial with coefficients $a_j := (r_1 z_{1,j} + r_2 z_{2,j} - y z_{0,j}) \bmod p$, and the meta-reduction can find $x$ if one of these coefficients is non-zero. Using $x$, it can then compute $\log_g \mathbf{Y}_i$ for all $i$ from the representations and from that the OMDL solution $(\log_{\mathbf{Y}_0}\mathbf{Y}_1, \log_{\mathbf{Y}_0}\mathbf{Y}_2)$.

If, on the other hand, $a_j = 0$ for some $j$ then by plugging in the definition of $y$, we get another polynomial which vanishes at $x$. We then show that, due to the randomizers $r_1$ and $r_2$, with overwhelming probability, the coefficients of this polynomial are non-zero – unless for some $c_1, c_2$ we have $\overrightarrow{z}_1 = c_1\overrightarrow{z}_0$ and $\overrightarrow{z}_2 = c_2\overrightarrow{z}_0$. But in this case $(c_1, c_2)$ is the solution to the 2-OMDL instance and the meta-reduction can therefore finish the simulation of the adversary.

**Theorem 3.17.** *Let $\mathcal{G}_i$ be a group of prime order $p$. There exists an algorithm $\mathsf{M}$ such that the following holds: Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic reduction s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that $(t, \epsilon)$-breaks 2-$\mathbf{omdl}_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}^{\mathsf{A}_{\mathsf{alg}}}$ is an algorithm that $(t', \epsilon')$-breaks $q$-$\mathbf{dlog}_{\mathcal{G}_i}$. If $t \geq (6 + 2q)\lceil \log_2 p \rceil + 1$ and $\epsilon \leq 1 - 1/p$ then $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ $(t', \epsilon')$-breaks $q$-$\mathbf{dlog}_{\mathcal{G}_i}$.*

| $\underline{\text{2-omdl}}_{\mathcal{G}_i}^{\mathsf{A}_{\mathsf{alg}}}$ | $\underline{\mathsf{O}}(\mathbf{Z})$ |
|---|---|
| 00 $Q \leftarrow 0$ | 05 if $Q = 0$ then |
| 01 $g \xleftarrow{\$} \mathsf{GenSamp}_i$ | 06 $\quad Q \leftarrow 1$ |
| 02 $y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p^*$ | 07 $\quad$ Return $\log_g \mathbf{Z}$ |
| 03 $(y_1^*, y_2^*) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}^{\mathsf{O}(\cdot)}(g, g^{y_1}, g^{y_2})$ | 08 Return $\perp$ |
| 04 Return $((y_1^*, y_2^*) = (y_1, y_2))$ | |

Figure 3.7: Game for 2-one-more discrete logarithm 2-$\mathbf{omdl}_{\mathcal{G}_i}$ relative to bilinear group $\mathcal{G}_i, i \in \{1, 2\}$ and adversary $\mathsf{A}_{\mathsf{alg}}$

*Proof.* Let $\mathsf{R}_{\mathsf{alg}}$ be an algebraic reduction s.t. for every algorithm $\mathsf{A}_{\mathsf{alg}}$ that $(t, \epsilon)$-breaks 2-$\mathbf{omdl}_{\mathcal{G}_i}$, $\mathsf{B} = \mathsf{R}_{\mathsf{alg}}^{\mathsf{A}_{\mathsf{alg}}}$ is an algorithm that $(t', \epsilon')$-breaks $q$-$\mathbf{dlog}_{\mathcal{G}_i}$.

We first specify a hypothetical algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ which ($t = (6 + 2q) \cdot \lceil \log_2 p \rceil + 1$, $\epsilon = 1 - 1/p$)-breaks 2-OMDL: on input $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, if $\mathbf{Y}_0$ is not a generator, it returns $\perp$. Else, it chooses two uniform values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and queries its oracle $\mathsf{O}(\cdot)$ on $\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$, providing representation $(0, r_1, r_2)$ in basis $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$. If it does not obtain the correct answer $\log_{\mathbf{Y}_0} \mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$, it returns $\perp$. Else it returns the 2-OMDL solution $\log_{\mathbf{Y}_0} \mathbf{Y}_1$ and $\log_{\mathbf{Y}_0} \mathbf{Y}_2$ with probability $1 - 1/p$.

We now construct a meta-reduction $\mathsf{M}$, s.t. $\mathsf{M}^{\mathsf{R}_{\mathsf{alg}}}$ $(t', \epsilon')$-breaks $q$-$\mathbf{dlog}_{\mathcal{G}_i}$, as follows. On input a $q$-$\mathbf{dlog}_{\mathcal{G}_i}$ instance $(g, g^x, \ldots, g^{x^q})$, the meta-reduction runs $\mathsf{R}_{\mathsf{alg}}$ on $(g, g^x, \ldots, g^{x^q})$. Every time $\mathsf{R}_{\mathsf{alg}}$ invokes 2-OMDL adversary $\mathsf{A}_{\mathsf{alg}}$ on a challenge $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, $\mathsf{M}$ simulates $\mathsf{A}_{\mathsf{alg}}$ as follows.

If $\mathbf{Y}_0$ is not a generator, it returns $\perp$. Since $\mathsf{R}_{\mathsf{alg}}$ is algebraic, along with $(\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2)$, it provides corresponding representation vectors $\vec{z}_i = (z_{i,0}, \ldots, z_{i,q}) \in \mathbb{Z}_p^{q+1}$ for $i \in [0, 2]$ such that

$$\mathbf{Y}_i = \prod_{j=0}^{q} (g^{x^j})^{z_{i,j}} = g^{\sum_{j=0}^{q} z_{i,j} x^j} \quad \text{for } i \in [0, 2] \ . \tag{3.12}$$

Let $k$ be such that $z_{0,k} \neq 0$ (which exists, since $\mathbf{Y}_0 \neq 1$). The meta-reduction chooses $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and queries $\mathsf{O}(\mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2})$ to obtain $y$. If $\mathbf{Y}_0^y \neq \mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$ then $\mathsf{M}$ returns $\perp$. Otherwise, let

$$\begin{aligned} d_{i,j} &:= z_{i,k} z_{0,j} - z_{i,j} z_{0,k} \bmod p \\ a_j &:= y z_{0,j} - r_1 z_{1,j} - r_2 z_{2,j} \bmod p \quad \text{for } 1 \leq i \leq 2 \text{ and } 1 \leq j \leq q \ . \end{aligned}$$

$\mathsf{M}$ distinguishes three cases and simulates $\mathsf{A}_{\mathsf{alg}}$ accordingly:

(a) $d_{i,j} = 0$ for all $i, j$: in this case, $\mathsf{M}$ computes (modulo $p$) $z_{1,k} z_{0,k}^{-1}$ and $z_{2,k} z_{0,k}^{-1}$, which $\mathsf{A}_{\mathsf{alg}}$ returns as the OMDL solution (we argue correctness below).

(b) $a_j \neq 0$ for some $j$: in this case, $\mathsf{M}$ factors the polynomial with coefficients $a_0, \ldots, a_q$. If $x$ is among its roots then $\mathsf{M}$ computes the following modulo $p$ and lets $\mathsf{A}_{\mathsf{alg}}$ return it:

$$\left(\textstyle\sum_{j=0}^{q} z_{1,j} x^j\right)\left(\sum_{j=0}^{q} z_{0,j} x^j\right)^{-1} \quad \text{and} \quad \left(\sum_{j=0}^{q} z_{2,j} x^j\right)\left(\sum_{j=0}^{q} z_{0,j} x^j\right)^{-1} \tag{3.13}$$

(since $((\log_g \mathbf{Y}_1)(\log_g \mathbf{Y}_0)^{-1}, (\log_g \mathbf{Y}_2)(\log_g \mathbf{Y}_0)^{-1}) \equiv_p (\log_{\mathbf{Y}_0} \mathbf{Y}_1, \log_{\mathbf{Y}_0} \mathbf{Y}_2)$, by (3.12), this is thus the OMDL solution).

(c) For the remaining case, let $i^*, j^*$ be such that $d_{i^*,j^*} \neq 0$. In this case, $\mathsf{M}$ factors the polynomial with coefficients $d_{i^*,0}, \ldots, d_{i^*,q}$. If $x$ is among its roots then $\mathsf{A}_{\mathsf{alg}}$ returns the values from (3.13). Otherwise $\mathsf{M}$ returns $\perp$.

When $\mathsf{R}_{\mathsf{alg}}$ stops and returns $x$ then $\mathsf{M}$ also stops and returns $x$.

We now show that the meta-reduction $\mathsf{M}$ correctly simulates every run of $\mathsf{A}_{\mathsf{alg}}$ that $\mathsf{R}_{\mathsf{alg}}$ invokes, in particular, that $\mathsf{A}_{\mathsf{alg}}$ solves the OMDL instance with probability at least $1 - 1/p$. First note that $\mathbf{Y}_0^y = \mathbf{Y}_1^{r_1}\mathbf{Y}_2^{r_2}$, which together with (3.12) yields:

$$0 \equiv_p \sum_{j=0}^{q}(yz_{0,j} - r_1z_{1,j} - r_2z_{2,j})x^j \equiv_p \sum_{j=0}^{q} a_jx^j =: P_{\overrightarrow{a}}(x) \ . \tag{3.14}$$

(where $P_{\overrightarrow{a}}$ is the polynomial with coefficients $(a_0, \ldots, a_q)$).
In Case (a), we have that $(z_{1,k}z_{0,k}^{-1}, z_{2,k}z_{0,k}^{-1})$ is the OMDL solution since for $i = 1, 2$:

$$\mathbf{Y}_0^{z_{i,k}z_{0,k}^{-1}} \overset{(3.12)}{=} g^{\sum_{j=0}^{q} z_{i,k}z_{0,k}^{-1}z_{0,j}x^j} \overset{d_{i,j}=0}{=} g^{\sum_{j=0}^{q} z_{i,j}x^j} \overset{(3.12)}{=} \mathbf{Y}_i \ .$$

In Case (b), the polynomial $P_{\overrightarrow{a}}$ defined in (3.14) is non-zero. Since by (3.14), $x$ is one of its roots, $\mathsf{M}$ can find it and use it to compute $\log_{\mathbf{Y}_0}\mathbf{Y}_i$ via (3.13).
In Case (c), we have $a_k = 0$ and thus (by definition of $a_k$):

$$yz_{0,k} \equiv_p r_1z_{1,k} + r_2z_{2,k} \ . \tag{3.15}$$

Multiplying (3.14) by $z_{0,k}$ we get

$$\sum_{j=0}^{q}(yz_{0,k}z_{0,j} - r_1z_{1,j}z_{0,k} - r_2z_{2,j}z_{0,k})x^j \equiv_p 0 \ ,$$

which, when substituting $yz_{0,k}$ by the right-hand side of (3.15), yields

$$r_1 \sum_{j=0}^{q}(z_{1,k}z_{0,j} - z_{1,j}z_{0,k})x^j + r_2 \sum_{j=0}^{q}(z_{2,k}z_{0,j} - z_{2,j}z_{0,k})x^j \equiv_p 0 \ .$$

Using $d_{i,j} \equiv_p z_{i,k}z_{0,j} - z_{i,j}z_{0,k}$, this can be written as

$$r_1 \sum_{j=0}^{q} d_{1,j}x^j + r_2 \sum_{j=0}^{q} d_{2,j}x^j \equiv_p 0 \ . \tag{3.16}$$

Recall that in Case (c), for some $i^*, j^*$: $d_{i^*,j^*} \neq 0$. If $D := \sum_{j=0}^{q} d_{i^*,j}x^j \equiv_p 0$, then $\mathsf{M}$ finds $x$ by factoring $P_{\overrightarrow{d}_{i^*}}$ and thus finishes the simulation. In the remaining case we have $D \not\equiv_p 0$; moreover, the values $x$ and $\overrightarrow{d}_{i^*}$ are independent of $r_{i^*}$, which was chosen after the reduction (implicitly) defined $\overrightarrow{d}_{i^*}$. Equation (3.16), that is, $r_{i^*}D \equiv_p -r_{3-i^*}\sum_{j=0}^{q} d_{3-i^*,j}x^j$, thus only holds with probability $1/p$. The meta-reduction therefore makes $\mathsf{A}_{\mathsf{alg}}$ return $\perp$ on correct inputs with probability at most $1/p$.

Finally, we show that the simulation of $\mathsf{A}_{\mathsf{alg}}$ takes at most $t$ steps. Computing $\mathbf{Y}_0^y$, $\mathbf{Y}_1^{r_1}$, and $\mathbf{Y}_2^{r_2}$ via square and multiply requires $3 \cdot 2\lceil\log_2(p)\rceil$ group operations, and computing $\mathbf{Y}_1^{r_1} \cdot \mathbf{Y}_2^{r_2}$ takes one. Checking if the roots of the polynomial $\sum_{i=0}^{q} a_iX^i$ (in Case (b)) or $\sum_{j=0}^{q} d_{i^*,j}X^j$ (in Case (c)) are equal to $x$ requires $q \cdot 2\lceil\log_2(p)\rceil$ group operations.

$\mathsf{M}$ succeeds in winning $q$-$\mathbf{dlog}_{\mathcal{G}_i}$ whenever $\mathsf{B}_{\mathsf{alg}} = \mathsf{R}_{\mathsf{alg}}^{\mathsf{A}_{\mathsf{alg}}}$ wins $q$-$\mathbf{dlog}_{\mathcal{G}_i}$. Therefore, we obtain

$$\Pr\left[q\text{-}\mathbf{dlog}_{\mathcal{G}_i}^{\mathsf{M}} = 1\right] = \Pr\left[q\text{-}\mathbf{dlog}_{\mathcal{G}_i}^{\mathsf{B}} = 1\right] \geq \varepsilon' \ .$$

Moreover, the running time of $\mathsf{M}$ is that of $\mathsf{B}$, i.e., $t'$. This completes the proof. □

## 3.9 Running Time of the Generic Reduction of Theorem 3.5

**Fact 3.18.** *Let $\mathbb{G}$ a group of order $p$. The square-and-multiply algorithm in $\mathbb{G}$ takes as input a group element $\mathbf{X} \in \mathbb{G}$ and a scalar $a \in \mathbb{Z}_p$ and returns $\mathbf{X}^a$ after computing at most $2\lceil\log(p)\rceil$ group operations.*

**Analysis.** We give an upper bound on the running time of computation, in terms of bilinear group operations, of the reduction $\mathsf{B}$ in Theorem 3.5. Let $o_{\exp}$ denote the number of group operations required to compute an exponentiation.

First, $\mathsf{B}$ computes the "core" elements, that is, $g_T^{z^k}$ for $0 \le k \le d_{\overrightarrow{T}}$ and $g_1^{z^k}$ for $0 \le k \le d_{\overrightarrow{R}}$ in Types 1 and 2, which requires $d_{\overrightarrow{T}} + 1$ computations of $e$ and (for Types 1 and 2) $d_{\overrightarrow{R}} + 1$ computations of the morphism $\psi$.

Let $i \in \{1, \ldots, r\}$, $j \in \{1, \ldots, s\}$, and $k \in \{1, \ldots, f\}$. Then we denote

$$R_i(y_1 Z + x_1, \ldots, y_m Z + x_m) = \sum_{\ell=0}^{d_{\overrightarrow{R}}} \lambda_{R_i,\ell} Z^\ell,$$
$$S_j(y_1 Z + x_1, \ldots, y_m Z + x_m) = \sum_{\ell=0}^{d_{\overrightarrow{S}}} \lambda_{S_j,\ell} Z^\ell, \text{ and}$$
$$F_k(y_1 Z + x_1, \ldots, y_m Z + x_m) = \sum_{\ell=0}^{d_{\overrightarrow{F}}} \lambda_{F_k,\ell} Z^\ell.$$

Next, $\mathsf{B}$ computes the monomials $\left(g_1^{z^\ell}\right)^{\lambda_{R_i,\ell}} = g_1^{\lambda_{R_i,\ell} z^i}$, for all $\ell \in \{0, \ldots, d_{\overrightarrow{R}}\}$, as well as $\left(g_2^{z^\ell}\right)^{\lambda_{S_j,\ell}} = g_2^{\lambda_{S_j,\ell} z^\ell}$ for $\ell \in \{0, \ldots, d_{\overrightarrow{S}}\}$, and $\left(g_T^{z^\ell}\right)^{\lambda_{F_k,\ell}} = g_T^{\lambda_{F_k,\ell} z^\ell}$ for $\ell \in \{0, \ldots, d_{\overrightarrow{F}}\}$. This requires at most $(rd_{\overrightarrow{R}} + sd_{\overrightarrow{S}} + fd_{\overrightarrow{F}} + 3)o_{\exp}$ group operations. Finally, $\mathsf{B}$ computes, for all $i \in \{0, \ldots, r\}$, all $j \in \{1, \ldots, s\}$, and $k \in \{1, \ldots, f\}$:

$$\prod_{\ell=0}^{d_{\overrightarrow{R}}} g_1^{\lambda_{R_i,\ell} z^\ell} = g_1^{R_i(y_1 z + x_1, \ldots, y_m z + x_m)},$$
$$\prod_{\ell=0}^{d_{\overrightarrow{S}}} g_2^{\lambda_{S_j,\ell} z^\ell} = g_2^{S_j(y_1 z + x_1, \ldots, y_m z + x_m)}, \text{ and}$$
$$\prod_{\ell=0}^{d_{\overrightarrow{F}}} g_T^{\lambda_{F_k,\ell} z^\ell} = g_T^{F_k(y_1 z + x_1, \ldots, y_m z + x_m)},$$

which requires $rd_{\overrightarrow{R}} + sd_{\overrightarrow{S}} + fd_{\overrightarrow{F}}$ group operations. This concludes the computation of $\mathsf{A}_{\mathsf{alg}}$'s input.

After computing $Q_1$ (or $Q_2$ or $Q_T$ depending on the case) from $\mathsf{A}_{\mathsf{alg}}$'s output, and its roots $z_1, \ldots,$ reduction $B$ verifies $g^{z_i} \stackrel{?}{=} g^z$. Since the polynomial has at most $d_\tau$ roots, this last step requires at most $d_\tau \cdot o_{\exp}$ group operations.

We can therefore upper-bound the number of group operations performed by $\mathsf{B}^{\mathsf{A}}$ by $t + o_1$ with $o_1 := o_0 + 2 + ((d_{\overrightarrow{R}} + 1)r + (d_{\overrightarrow{S}} + 1)s + (d_{\overrightarrow{F}} + 1)f + d_\tau)o_{\exp} + rd_{\overrightarrow{R}} + sd_{\overrightarrow{S}} + fd_{\overrightarrow{F}}$ with $o_0 := d_{\overrightarrow{R}} + d_{\overrightarrow{F}} + 2$ for Types 1 and 2, and $o_0 := d_{\overrightarrow{F}} + 1$ for Type 3. By applying Fact 3.18, we obtain the time bound claimed in Theorem 3.5.

## 3.10 Proof of Theorem 3.9

We will use the following lemma in the proof of Theorem 3.9.

**Lemma 3.19.** *Let $n$ be an integer, $(P_i)_{i=1}^n \in (\mathbb{Z}_p[X_1, \ldots, X_m])^n$ and $\overrightarrow{x} \in \mathbb{Z}_p^m$. Then $P_i(\overrightarrow{x}) \equiv_p 0$ for some $i \in [1, n]$ if and only if $\mathrm{LCM}(P_1, \ldots, P_n)(\overrightarrow{x}) \equiv_p 0$.*

*Proof.* Let $i$ be such that $P_i(\overrightarrow{x}) \equiv_p 0$. Since $P_i$ divides $\mathrm{LCM}(P_1, \ldots, P_n)$, we have

$$\mathrm{LCM}(P_1, \ldots, P_n)(\overrightarrow{x}) \equiv_p 0.$$

Now suppose $\mathrm{LCM}(P_1,\ldots,P_n)(\vec{x}) \equiv_p 0$. Since $\mathrm{LCM}(P_1,\ldots,P_n)$ divides $\prod_{i=1}^n P_i$, we have $\prod_{i=1}^n P_i(\vec{x}) \equiv_p 0$. Because $\mathbb{Z}_p$ is an integer domain, this implies that $P_i(\vec{x}) \equiv_p 0$ for some $i$. □

Whereas in the proof of Theorem 3.5 we focused on Type-2 groups, for the sake of variation, we give the details for Type-3 groups, and then explain what changes for Types 1 and 2. Again, for $i \in \{1,2,T\}$, we denote $g_i^u$ by $[u]_i$ for $u \in \mathbb{Z}_p$.

Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic algorithm against $\mathbf{rüber}_{\mathcal{G}}$, defined by a tuple $(\vec{R}, \vec{S}, \vec{F}, R', S', F')$ of (vectors of) rational fractions, that wins with advantage $\varepsilon$ in time $t$. We construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against $(q_1,q_2)$-$\mathbf{dlog}_{\mathcal{G}}$. Depending on why the tuple is non-trivial (conditions (3.1), (3.2) and/or (3.T) in Def. 3.3), $\mathsf{B}_{\mathsf{alg}}$ uses a different strategy, minimizing $d_3$ in Def. 3.8.

We detail the most complex case, which is (3.T); that is, $F'$ is Type-3 independent from $(\vec{R}, \vec{S}, \vec{F})$.

*Adversary* $\mathsf{B}_{\mathsf{alg}}(g_1, \mathbf{Y}_1,\ldots,\mathbf{Y}_{q_1}, g_2, \mathbf{Z}_1,\ldots,\mathbf{Z}_{q_2})$*:* On input a $(q_1,q_2)$-DLog instance with $\mathbf{Y}_i = [z^i]_1$ and $\mathbf{Z}_i = [z^i]_2$, reduction $\mathsf{B}_{\mathsf{alg}}$ simulates $\mathbf{rüber}_{\mathcal{G}}$ for $\mathsf{A}_{\mathsf{alg}}$.

It first computes a least common multiple $D := \mathrm{Den}(\vec{R}\|\vec{S}\|\vec{F}\|R'\|S'\|F')$ of the denominators of $\vec{R}, \vec{S}, \vec{F}, R', S', F'$. It then picks random values $\vec{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\vec{v} \xleftarrow{\$} \mathbb{Z}_p^m$ and uses them to implicitly define $x_i := y_i z + v_i \bmod p$.

If $D'(Z) := D(y_1 Z + v_1, \ldots, y_m Z + v_m)$ is the zero polynomial, it aborts. (∗)

Else, for each root $z_i$ of $D'$, the reduction checks whether $g_1^{z_i} = \mathbf{Y}_1$; if such $z_i$ exists, it stops the simulation and outputs $z_i$. (♯)

If $\mathsf{B}_{\mathsf{alg}}$ has not stopped then $D(\vec{x}) \not\equiv_p 0$. It now computes a least common multiple $D_1 := \mathrm{Den}(\vec{R}\|\vec{F})$ of the denominators of $\vec{R}$ and $\vec{F}$ and $D_2 := \mathrm{Den}(\vec{S})$.

From its $(q_1,q_2)$-DLog instance, it then computes $h_1 := g_1^{D_1(\vec{x})}$, $h_2 := g_2^{D_2(\vec{x})}$. Note that for all $i \in \{1,2\}$, $h_i \neq 1$ (since $D_i$ is a divisor of $D$, and $D(\vec{x}) \not\equiv_p 0$).

For $1 \le i \le r$, $\mathsf{B}_{\mathsf{alg}}$ computes $\mathbf{U}_i := [(\hat{R}_i \cdot D_1/\check{R}_i)(\vec{x})]_1 = h_1^{R_i(\vec{x})}$ (where $D_1/\check{R}_i$ is a polynomial by construction of $D_1$). This can be computed from $\mathbf{Y}_1,\ldots,\mathbf{Y}_{q_1}$, since $q_1 \ge \deg(D_1) + d_{\vec{R}}$. Likewise, for $1 \le i \le s$, $\mathsf{B}_{\mathsf{alg}}$ computes $\mathbf{V}_i := [(\hat{S}_i \cdot D_2/\check{S}_i)(\vec{x})]_2 = h_2^{S_i(\vec{x})}$ from $\mathbf{Z}_1,\ldots,\mathbf{Z}_{q_2}$. Finally, $\mathsf{B}_{\mathsf{alg}}$ computes $e(g_1,g_2)^{z^i} = [z^i]_T$ for $1 \le i \le q_1+q_2$ from its instance, from which it then computes $\mathbf{W}_i := [D_2 \cdot \hat{F}_i \cdot D_1/\check{F}_i(\vec{x})]_T = e(h_1,h_2)^{F_i(\vec{x})}$ for $1 \le i \le f$.

With these values, $\mathsf{B}_{\mathsf{alg}}$ runs $(\mathbf{U}',\mathbf{V}',\mathbf{W}') \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}(\vec{\mathbf{U}}, \vec{\mathbf{V}}, \vec{\mathbf{W}})$, which also returns representations $\vec{\mu}$ for $\mathbf{U}'$, $\vec{\eta}$ for $\mathbf{V}'$ and $(A, \vec{\delta})$ for $\mathbf{W}'$ (so that (3.1) holds with $\nu_i = \gamma_{i,j} = 0$ for all $i,j$).

$\mathsf{B}_{\mathsf{alg}}$ defines the following polynomial:

$$P_T := \check{F}' \cdot D_1 \cdot D_2 \cdot (F' - \textstyle\sum_{i=1}^r \sum_{j=1}^s \alpha_{i,j} R_i S_j - \sum_{i=k}^f \delta_k F_k). \tag{3.17}$$

Note that $P_T$ is indeed a polynomial, because $\check{F}' \cdot F'$, as well as $D_1 \cdot R_i$, $D_2 \cdot S_j$ and $D_1 \cdot F_k$ for all $i,j,k$ are all polynomials.

From $P_T$, the reduction defines $P_T'(Z) := P_T(y_1 Z + v_1, \ldots, y_m Z + v_m)$. If $P_T'$ is the zero polynomial then $\mathsf{B}_{\mathsf{alg}}$ aborts. (∗∗)

Else, it factors $P_T'$ to obtain its roots $z_1,\ldots$ If for one of them we have $g_1^{z_i} = \mathbf{Y}_1$, then $\mathsf{B}_{\mathsf{alg}}$ returns $z_i$. (♯♯)

We analyze $\mathsf{B}_{\mathsf{alg}}$'s success probability. First note that $\mathsf{B}_{\mathsf{alg}}$ solves the DLog challenge if it stops in line ($\sharp$) and it fails if it aborts in line ($*$). Otherwise, $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates game **rüber** (Fig. 3.3), as the values $x_i$ are uniformly distributed in $Z_p$, and the tests in lines 03 and 04 in **rüber** are all done: For some $i$: $\check{R}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{S}_i(\overrightarrow{x}) \equiv_p 0$ or $\check{F}_i(\overrightarrow{x}) \equiv_p 0$, or $\check{R}'(\overrightarrow{x}) \equiv_p 0$ or $\check{S}'(\overrightarrow{x}) \equiv_p 0$ or $\check{F}'(\overrightarrow{x}) \equiv_p 0$ if and only if a least common denominator of all these polynomials vanishes at $\overrightarrow{x}$, i.e. $D(\overrightarrow{x}) \equiv_p 0$ (by Lemma 3.19), and $\mathsf{B}_{\mathsf{alg}}$ only proceeds if the latter is not the case. In this case $\mathsf{A}_{\mathsf{alg}}$'s inputs, $\overrightarrow{\mathbf{U}}$, $\overrightarrow{\mathbf{V}}$ and $\overrightarrow{\mathbf{W}}$ are correctly computed.

If $\mathsf{B}_{\mathsf{alg}}$ does not stop in line ($**$) either and $\mathsf{A}_{\mathsf{alg}}$ is successful then

$$\mathbf{W}' = e(h_1, h_2)^{F'(\overrightarrow{x})} = \left[D_1(\overrightarrow{x})D_2(\overrightarrow{x})F'(\overrightarrow{x})\right]_T . \tag{3.18}$$

On the other hand, from $\mathsf{A}_{\mathsf{alg}}$'s representation $(A, \overrightarrow{\delta})$ of $\mathbf{W}'$ and from the definitions of $\mathbf{U}_i$, $\mathbf{V}_j$ and $\mathbf{W}_k$ we have:

$$\begin{aligned}\mathbf{W}' &= \textstyle\prod_i \prod_j e(\mathbf{U}_i, \mathbf{V}_j)^{\alpha_{i,j}} \cdot \prod_k \mathbf{W}_k^{\delta_k} \\ &= \left[\textstyle\sum_i \sum_j \alpha_{i,j} (\hat{R}_i \cdot D_1/\check{R}_i)(\overrightarrow{x})\, (\hat{S}_j \cdot D_2/\check{S}_j)(\overrightarrow{x}) + \sum_k \delta_k (D_2 \cdot \hat{F}_k \cdot D_1/\check{F}_k)(\overrightarrow{x})\right]_T .\end{aligned} \tag{3.19}$$

Equating the representations of $\mathbf{W}'$ in (3.18) and (3.19) in base $e(g_1, g_2)$ yields

$$(D_1 \cdot D_2 \cdot F' - \textstyle\sum_i \sum_j \alpha_{i,j}\, \hat{R}_i \cdot D_1/\check{R}_i \cdot \hat{S}_j \cdot D_2/\check{S}_j + \sum_k \delta_k\, D_2 \cdot \hat{F}_k \cdot D_1/\check{F}_k)(\overrightarrow{x}) \equiv_p 0 . \tag{3.20}$$

Multiplying the above by $\check{F}'(\overrightarrow{x})$ yields $P_T(\overrightarrow{x}) \equiv_p 0$ and since $x_i \equiv_p y_i z + v_i$, we have $P'_T(z) \equiv_p 0$ as well. By factoring $P'_T$, reduction $\mathsf{B}_{\mathsf{alg}}$ finds thus the solution $z$.

We have shown that unless $\mathsf{B}_{\mathsf{alg}}$ aborts in lines ($*$) or ($**$), it finds the $(q_1, q_2)$-DLog solution whenever $\mathsf{A}_{\mathsf{alg}}$ wins **rüber**. In the remainder of the proof we bound the probability that $\mathsf{B}_{\mathsf{alg}}$ aborts. It aborts if and only if the following polynomial is zero: $Q(Z) := (D' \cdot P'_T)(Z) = (D \cdot P_T)(y_1 Z + v_1, \ldots, y_m Z + v_m)$. It thus suffices to upper-bound the probability that the coefficient of maximal degree of this polynomial is zero. By Lemma 2.2, this coefficient can be represented as a polynomial $Q^{\max}$ in variables $(Y_1, \ldots, Y_m)$ that is of the same degree as $D \cdot P_T$, which we bound as follows (recall that $d_{\text{den}} = \deg D$; cf. Def. 3.8):

$$\begin{aligned}\deg(D \cdot P_T) &\le d_{\text{den}} + \deg \check{F}' + \deg D_1 + \deg D_2 + \\ &\qquad \max\{\deg F', \deg R_i + \deg S_j, \deg F_k\}_{1\le i\le r, 1\le j\le s, 1\le k\le f} \\ &= d_{\text{den}} + \check{d}_{F'} + \check{d}_{\overrightarrow{R}\|\overrightarrow{F}} + \check{d}_{\overrightarrow{S}} + \max\{d_{F'}, d_{\overrightarrow{R}} + d_{\overrightarrow{S}}, d_{\overrightarrow{F}}\} \\ &= d_{3.T}\end{aligned}$$

As the values $y_1 z, \ldots, y_m z$ are completely hidden from $\mathsf{A}_{\mathsf{alg}}$ (they are "one-time-padded" with $v_1, \ldots, v_m$, resp.), the values $(A, \overrightarrow{\delta})$ returned by $\mathsf{A}_{\mathsf{alg}}$ are independent from $\overrightarrow{y}$. Since $\overrightarrow{y}$ is moreover independent from $F'$, $\overrightarrow{R}$, $\overrightarrow{S}$ and $\overrightarrow{F}$, it is also independent from $P_T, D, Q$ and $Q^{\max}$. The probability that $Q \equiv 0$ is thus upper-bounded by the probability that $Q^{\max}(\overrightarrow{y}) \equiv_p 0$ when evaluated at the random point $\overrightarrow{y}$. By Lemma 2.1 (Schwartz-Zippel), the probability that $Q^{\max}(\overrightarrow{y}) \equiv_p 0$ is thus upper-bounded by $\frac{d_{3.T}}{p-1}$.

We turn to the remaining cases (3.1) and (3.2), which follow similarly, except that $P_T$ in (3.17) is replaced by different polynomials $P$. Moreover, we can optimize the loss of the security reduction, by reducing the degree of $P$. Note that the two properties which are used in the proof are:

- $P$ is a non-zero polynomial, and
- if $\mathsf{A}_{\mathsf{alg}}$ wins the game then $P(\overrightarrow{x}) \equiv_p 0$.

In Case (3.1), that is, when $R'$ is linearly independent from $\overrightarrow{R}$, from $\mathsf{A}_{\mathsf{alg}}$'s output $\mathbf{U}' = h_1^{R'(\overrightarrow{x})} = [D_1(\overrightarrow{x})\, R'(\overrightarrow{x})]_1$ and its representation $\overrightarrow{\mu}$ of $\mathbf{U}' = \prod_i \mathbf{U}_i^{\mu_i} = [\sum_i (\hat{R}_i \cdot D_1/\check{R}_i)(\overrightarrow{x})]_1$, we get that the following function vanishes at $\overrightarrow{x}$ (which corresponds to (3.20) above):

$$D_1 \cdot R' - \textstyle\sum_i \mu_i \, \hat{R}_i \cdot D_1/\check{R} \ .$$

Multiplying this by $\check{R}'$ yields a polynomial. In contrast to case (3.T), we can moreover divide by $D_1/\mathrm{Den}(\overrightarrow{R})$ and still obtain a polynomial. We thus define:

$$P_1 := \check{R}' \cdot \mathrm{Den}(\overrightarrow{R}) \cdot (R' - \textstyle\sum_{i=1}^{r} \mu_i R_i) \ .$$

Since $P_1$ is of degree at most $\check{d}_{R'} + \check{d}_{\overrightarrow{R}} + \max\{d_{R'}, d_{\overrightarrow{R}}\}$, the probability of $\mathsf{B}_{\mathsf{alg}}$ aborting is $\frac{d_{3.1}}{p-1}$.

For Case (3.2), from $\mathsf{A}_{\mathsf{alg}}$'s representation $\overrightarrow{\eta}$ of $\mathbf{V}'$, we analogously define

$$P_2 := \check{S}' \cdot \mathrm{Den}(\overrightarrow{S}) \cdot (S' - \textstyle\sum_{i=1}^{s} \eta_i S_i) \ ,$$

which is of degree at most $\check{d}_{S'} + \check{d}_{\overrightarrow{S}} + \max\{d_{S'}, d_{\overrightarrow{S}}\} = d_{3.2}$.

The theorem for Type-3 groups now follows because

$$\mathbf{Adv}^{(q_1,q_2)\text{-}\mathbf{dlog}}_{\mathcal{G},\mathsf{B}_{\mathsf{alg}}} \geq \mathbf{Adv}^{\mathbf{rüber}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \Pr[\mathsf{B}_{\mathsf{alg}} \text{ aborts}]$$

and $\mathsf{B}_{\mathsf{alg}}$ follows the strategy that minimizes its abort probability to $\min\{\frac{d_{3.1}}{p-1}, \frac{d_{3.2}}{p-1}, \frac{d_{3.T}}{p-1}\} = \frac{d_3}{p-1}$.

**Groups of Type 1 and 2.** The reductions for bilinear groups of Types 1 and 2 to $q$-$\mathbf{dlog}_{\mathcal{G}_2}$ work analogously; the main difference is that we can only redefine $g_2$ as $h_2$, since $h_2$ defines $h_1 = \psi(h_2)$. This means that instead of $D_1$ and $D_2$ as in the proof above, we define $D_{1,2} := \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S} \| \overrightarrow{F})$ and set $h_2 := g_2^{D_{1,2}(\overrightarrow{x})}$. For Type-2 groups, let $(\overrightarrow{\mu}, \overrightarrow{\nu})$, $\overrightarrow{\eta}$ and $(A, \Gamma, \overrightarrow{\delta})$ be $\mathsf{A}_{\mathsf{alg}}$'s representations of $\mathbf{U}'$, $\mathbf{V}'$ and $\mathbf{W}'$, respectively. Then $\mathsf{B}_{\mathsf{alg}}$ defines the polynomials $P_1, P_2$ and $P_T$ as follows:

$$\begin{aligned}
P_1 &:= \check{R}' \cdot \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S}) \cdot (R' - \textstyle\sum_{i=1}^{r} \mu_i R_i - \sum_{i=1}^{s} \nu_i S_i) \\
P_2 &:= \check{S}' \cdot \mathrm{Den}(\overrightarrow{S}) \cdot (S' - \textstyle\sum_{i=1}^{s} \eta_i S_i) \\
P_T &:= \check{F}' \cdot \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S} \| \overrightarrow{F}) \cdot \mathrm{Den}(\overrightarrow{S}) \cdot \\
&\qquad (F' - \textstyle\sum_{i=1}^{r} \sum_{j=1}^{s} \alpha_{i,j} \, R_i \cdot S_j - \sum_{i=1}^{s} \sum_{j=1}^{s} \gamma_{i,j} \, S_i \cdot S_j - \sum_{i=1}^{f} \delta_i F_i) \ .
\end{aligned}$$

As for the case $(3.T)$ above, by applying Lemmas 2.2 and 2.1 to $D \cdot P_1, D \cdot P_2$, and $D \cdot P_T$, we deduce that the probability of aborting is bounded by $\frac{d_2}{p-1}$.

For Type-1 groups, let $(\overrightarrow{\mu}, \overrightarrow{\nu}), (\overrightarrow{\eta}, \overrightarrow{\zeta})$ and $(A, B, \Gamma, \overrightarrow{\delta})$ be $\mathsf{A}_{\mathsf{alg}}$'s representations of $\mathbf{U}'$, $\mathbf{V}'$ and $\mathbf{W}'$, respectively. Then $\mathsf{B}_{\mathsf{alg}}$ defines the polynomials $P_1, P_2$ and $P_T$ as follows:

$$\begin{aligned}
P_1 &:= \check{R}' \cdot \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S}) \cdot (R' - \textstyle\sum_{i=1}^{r} \mu_i R_i - \sum_{i=1}^{s} \nu_i S_i) \\
P_2 &:= \check{S}' \cdot \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S}) \cdot (S' - \textstyle\sum_{i=1}^{r} \zeta_i R_i - \sum_{i=1}^{s} \eta_i S_i) \\
P_T &:= \check{F}' \cdot \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S} \| \overrightarrow{F}) \cdot \mathrm{Den}(\overrightarrow{R} \| \overrightarrow{S}) \cdot (F' - \textstyle\sum_{i=1}^{r} \sum_{j=1}^{s} \alpha_{i,j} \, R_i \cdot S_j \\
&\qquad - \textstyle\sum_{i=1}^{r} \sum_{j=1}^{r} \beta_{i,j} \, R_i \cdot R_j - \sum_{i=1}^{s} \sum_{j=1}^{s} \gamma_{i,j} \, S_i \cdot S_j - \sum_{i=1}^{f} \delta_i F_i) \ .
\end{aligned}$$

By an analysis analogous to the above, the abort probability is bounded by $\frac{d_1}{p-1}$.

## 3.11 Proof of Theorem 3.14

We give a detailed proof for Type-2 bilinear groups. Let $\mathsf{A}_{\mathsf{alg}}$ be an algebraic algorithm against $\mathbf{gegenüber}_{\mathcal{G}}$ that wins with advantage $\epsilon$ in time $t$. We construct a generic reduction with oracle access to $\mathsf{A}_{\mathsf{alg}}$, which yields an algebraic adversary $\mathsf{B}_{\mathsf{alg}}$ against $q\text{-}\mathbf{dlog}_{\mathcal{G}_2}$. There are three (non-exclusive) types of reasons that $(\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_T, R^*, S^*, F^*)$ is non-trivial; that is $(2.i)$ from Def. 3.12 holds for some $i \in \{1, 2, T\}$. Each condition enables a different type of reduction, of which $\mathsf{B}_{\mathsf{alg}}$ runs the one that minimizes $d_\tau$. We start with Case (2.1), that is, $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$.

*Adversary* $\mathsf{B}_{\mathsf{alg}}(g_2, \mathbf{Z}_1, \ldots, \mathbf{Z}_q)$*:* On input a problem instance of game $q\text{-}\mathbf{dlog}_{\mathcal{G}_2}$ with $\mathbf{Z}_i = [z^i]_2$, $\mathsf{B}_{\mathsf{alg}}$ defines $g_1 \leftarrow \psi(g_2)$ and $g_T \leftarrow e(g_1, g_2)$. Then, it picks random values $\overrightarrow{y} \xleftarrow{\$} (\mathbb{Z}_p^*)^m$ and $\overrightarrow{v} \xleftarrow{\$} \mathbb{Z}_p^m$, sets $x_i := y_i z + v_i \bmod p$ (implicitly), and runs $((\mathbf{U}_1, \mathbf{V}_1, \mathbf{W}_1, \mathbf{U}_2, \mathbf{V}_2, \mathbf{W}_2), (R^*, S^*, F^*)) \xleftarrow{\$} \mathsf{A}_{\mathsf{alg}}{}^{\mathsf{O}(\cdot,\cdot)}()$. Oracle calls $\mathsf{O}(i, P(\overrightarrow{X}))$ are answered by computing and returning $\mathbf{Y}_{P,i} := [P(x_1, \ldots, x_m)]_i$ from the $q$-DLog instance, the morphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ and the pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathsf{B}_{\mathsf{alg}}$ can do so efficiently since the total degree of the polynomials in $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_T$ are bounded by $q, q$ and $2q$, respectively.

Since $\mathsf{A}_{\mathsf{alg}}$ is algebraic, for all $k_1, k_2, k_3 \in \{1, 2\}$ it also returns vectors and matrices $\overrightarrow{\mu}^{(k_1)}$, $\overrightarrow{\nu}^{(k_1)}$, $\overrightarrow{\zeta}^{(k_2)}$, $\overrightarrow{\delta}^{(k_3)}$, $A^{(k_3)} = (\alpha_{j,k}^{(k_3)})_{j,k}$, $\Gamma^{(k_3)} = (\gamma_{j,k}^{(k_3)})_{j,k}$ respectively indexed by $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_1$, $\mathcal{Q}_T$, $\mathcal{Q}_1 \times \mathcal{Q}_2$ and $\mathcal{Q}_2 \times \mathcal{Q}_2$ such that

$$\mathbf{U}_{k_1} = \textstyle\prod_{R \in \mathcal{Q}_1} \mathbf{Y}_{R,1}^{\mu_R^{(k_1)}} \cdot \prod_{R \in \mathcal{Q}_2} \psi(\mathbf{Y}_{R,2})^{\nu_R^{(k_1)}} \tag{3.21a}$$

$$\mathbf{V}_{k_2} = \textstyle\prod_{R \in \mathcal{Q}_2} \mathbf{Y}_{R,2}^{\zeta_R^{(k_2)}} \tag{3.21b}$$

$$\mathbf{W}_{k_3} = \textstyle\prod_{R \in \mathcal{Q}_1} \prod_{S \in \mathcal{Q}_2} e(\mathbf{Y}_{R,1}, \mathbf{Y}_{S,2})^{\alpha_{R,S}^{(k_3)}} \tag{3.21c}$$

$$\cdot \textstyle\prod_{R \in \mathcal{Q}_2} \prod_{S \in \mathcal{Q}_2} e(\psi(\mathbf{Y}_{R,2}), \mathbf{Y}_{S,2})^{\gamma_{R,S}^{(k_3)}} \cdot \prod_{R \in \mathcal{Q}_T} \mathbf{Y}_{R,T}^{\delta_R^{(k_3)}}. \tag{3.21d}$$

$\mathsf{B}_{\mathsf{alg}}$ then computes the following multivariate polynomial, which corresponds to the logarithm of $\mathbf{U}_1^{R^*(\overrightarrow{x})} \cdot \mathbf{U}_2^{-1}$ in base $g_1$:

$$P_1(\overrightarrow{X}) := R^*(\overrightarrow{X})\big(\textstyle\sum_{R \in \mathcal{Q}_1} \mu_R^{(1)} R(\overrightarrow{X}) + \sum_{R \in \mathcal{Q}_2} \nu_R^{(1)} R(\overrightarrow{X})\big) - \sum_{R \in \mathcal{Q}_1} \mu_R^{(2)} R(\overrightarrow{X}) - \sum_{R \in \mathcal{Q}_2} \nu_R^{(2)} R(\overrightarrow{X}).$$

Since in Case (2.1) we have $R^* \notin \mathrm{FrSp}(\mathcal{Q}_1 \cup \mathcal{Q}_2)$, the polynomial $P_1$ is non-zero. From $P_1$, the reduction defines $Q_1(Z) := P_1(y_1 Z + v_1, \ldots, y_m Z + v_m)$. If $Q_1$ is the zero polynomial then $\mathsf{B}_{\mathsf{alg}}$ aborts. $(*)$

Else, it factors $Q_1$ to obtain its roots $z_1, \ldots$ (of which there are at most $\max\{\deg R^*, 1\} \cdot \max\{d_1', d_2'\}$). If for one of them we have $g_2^{z_i} = \mathbf{Z}$, then $\mathsf{B}_{\mathsf{alg}}$ returns $z_i$.

We analyze $\mathsf{B}_{\mathsf{alg}}$'s success probability. First note that $\mathsf{B}_{\mathsf{alg}}$ perfectly simulates game $\mathbf{gegenüber}_{\mathcal{G}}$, as the values $x_i$ are uniformly distributed in $Z_p$ and oracle calls are correctly computed.

Moreover, if $\mathsf{B}_{\mathsf{alg}}$ does not abort in $(*)$ and $\mathsf{A}_{\mathsf{alg}}$ wins game $\mathbf{gegenüber}_{\mathcal{G}}$, then $\mathbf{U}_1^{R^*(\overrightarrow{x})} \cdot \mathbf{U}_2^{-1} = 1$. Substituting the right-hand side of (3.21a) for $\mathbf{U}_1$ and $\mathbf{U}_2$ in this equation and considering the discrete logarithm in base $g_1$, this yields $P_1(\overrightarrow{x}) \equiv_p 0$. Since $x_i = y_i z + v_i$, we moreover have $Q_1(z) = 0$. By factoring $Q_1$, reduction $\mathsf{B}_{\mathsf{alg}}$ finds thus the solution $z$.

It remains to bound the probability that $\mathsf{B}_{\mathsf{alg}}$ aborts in $(*)$, that is, the probability that $0 \equiv Q_1(Z) = P_1(y_1 Z + v_1, \ldots, y_m Z + v_m)$. The analysis is analogous to all previous theorems:

We upper-bound the probability that the coefficient of maximal degree, say $d$, is zero. By Lemma 2.2, this coefficient can be represented as a polynomial $Q_1^{\max}$ in variables $(Y_1, \ldots, Y_m)$ of the same degree $d$. The values $y_1 z, \ldots, y_m z$ are completely hidden from $\mathsf{A}_{\mathsf{alg}}$ because they are "one-time-padded" with $v_1, \ldots, v_m$, respectively. This means that the values $\overrightarrow{\mu}^{(1)}$, $\overrightarrow{\mu}^{(2)}$, $\overrightarrow{\nu}^{(1)}$ nad $\overrightarrow{\nu}^{(2)}$ returned by $\mathsf{A}_{\mathsf{alg}}$ are independent from $\overrightarrow{y}$. Since $\overrightarrow{y}$ is moreover independent from $\overrightarrow{R^*}$, $\mathcal{Q}_1$ and $\mathcal{Q}_2$, $\mathcal{Q}_T$, it is also independent from $P_1, Q_1$ and $Q_1^{\max}$. The probability that $Q_1 \equiv 0$ is thus upper-bounded by the probability that $Q_1^{\max}(\overrightarrow{y}) \equiv_p 0$ when evaluated at the random point $\overrightarrow{y}$. By the Schwartz-Zippel lemma, the probability that $Q_1(Z) \equiv 0$ is thus upper-bounded by $\frac{d}{p-1}$. The degree $d$ of $Q_1$ (and thus of $Q_1^{\max}$) is upper-bounded by the total degrees of $P_1$, which is $\max\{1, \deg R^*\} \cdot \max\{d_1', d_2'\} = d_{2.1}$ in Def. 3.13. $\mathsf{B}_{\mathsf{alg}}$ thus aborts in line (*) with probability at most $\frac{d_{2,1}}{p-1}$.

Cases (2.2) (where we have $S^* \notin \mathrm{FrSp}(\mathcal{Q}_2)$) and $(2.T)$ are treated analogously. Theorem 3.14 for Type-2 groups now follows because

$$\mathbf{Adv}^{q\text{-}\mathbf{dlog}}_{\mathcal{G}_2,\mathsf{B}_{\mathsf{alg}}} \geq \mathbf{Adv}^{\mathbf{gegenüber}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}} - \Pr[\mathsf{B}_{\mathsf{alg}} \text{ aborts}]$$

and $\mathsf{B}_{\mathsf{alg}}$ follows the strategy that minimizes its abort probability to $\min\left\{\frac{d_{2.1}}{p-1}, \frac{d_{2.2}}{p-1}, \frac{d_{2.T}}{p-1}\right\} = \frac{d_2}{p-1}$.

The proofs for groups of Type 1 and Type 3 are done by analogous adaptations of Theorem 3.5 as just shown for Type 2.

# Chapter 4

# Signatures on randomizable ciphertexts

*This work was published in the proceedings of the 2020 SCN Conference. It was completed with co-author Georg Fuchsbauer.*

---

For clarity reasons, we are using additive notations for the group operations (except for the group target).

**Overview.** We observe in this chapter that SoRC can be seen as *signatures on equivalence classes* (JoC'19), another primitive with many applications to anonymous authentication, and that SoRC provide better anonymity guarantees. We first strengthen the unforgeability notion for SoRC and then give a scheme that provably achieves it in the generic group model. Signatures in our scheme consist of 4 bilinear-group elements, which is considerably more efficient than prior schemes.

## 4.1 Definitions

We start with the definition of a *signatures on randomizable ciphertexts* scheme, which consists of a randomizable public-key encryption scheme and a signature scheme, whose signatures are computed and verified on pairs (encryption key, ciphertext). In addition, there is an algorithm $\mathsf{Adapt}$, which lets one adapt a signature on a ciphertext to any randomization of the latter.

### Syntax

**Definition 4.1.** *We denote by $\mathcal{PP}$ the set of public parameters, and for $\mathsf{p} \in \mathcal{PP}$ we let $\mathcal{M}_\mathsf{p}$ be the set of messages, $\mathcal{DK}_\mathsf{p}$ the set of decryption keys, $\mathcal{EK}_\mathsf{p}$, the set of encryption keys, $\mathcal{C}_\mathsf{p}$ the set of ciphertexts, $\mathcal{R}_\mathsf{p}$ the set of ciphertext randomness, $\mathcal{SK}_\mathsf{p}$ the set of signature keys, $\mathcal{VK}_\mathsf{p}$ the set of verification keys and $\mathcal{S}_\mathsf{p}$ the set of signatures.*

*A scheme of* signatures on randomizable ciphertexts $\mathsf{S}$ *consists of the following probabilistic algorithms, of which all except* $\mathsf{Setup}$ *are implicitly parameterized by an element* $\mathsf{p} \in \mathcal{PP}$*.*

$$\mathsf{Setup}\colon \mathbb{N} \to \mathcal{PP} \tag{4.1}$$

$$\mathsf{KeyGen}\colon \emptyset \to \mathcal{DK}_\mathsf{p} \times \mathcal{EK}_\mathsf{p} \qquad \mathsf{SKeyGen}\colon \emptyset \to \mathcal{SK}_\mathsf{p} \times \mathcal{VK}_\mathsf{p} \tag{4.2}$$

$$\mathsf{Enc}\colon \mathcal{EK}_\mathsf{p} \times \mathcal{M}_\mathsf{p} \times \mathcal{R}_\mathsf{p} \to \mathcal{C}_\mathsf{p} \qquad \mathsf{Sign}\colon \mathcal{SK}_\mathsf{p} \times \mathcal{EK}_\mathsf{p} \times \mathcal{C}_\mathsf{p} \to \mathcal{S}_\mathsf{p} \tag{4.3}$$

$$\mathsf{Rndmz}\colon \mathcal{EK}_\mathsf{p} \times \mathcal{C}_\mathsf{p} \times \mathcal{R}_\mathsf{p} \to \mathcal{C}_\mathsf{p} \qquad \mathsf{Verify}\colon \mathcal{VK}_\mathsf{p} \times \mathcal{EK}_\mathsf{p} \times \mathcal{C}_\mathsf{p} \times \mathcal{S}_\mathsf{p} \to \{0,1\} \tag{4.4}$$

$$\mathsf{Dec}\colon \mathcal{DK}_\mathsf{p} \times \mathcal{C}_\mathsf{p} \to \mathcal{M}_\mathsf{p} \qquad \mathsf{Adapt}\colon \mathcal{S}_\mathsf{p} \times \mathcal{R}_\mathsf{p} \to \mathcal{S}_\mathsf{p} \tag{4.5}$$

| $\mathbf{IND\text{-}CPA}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, b)$ : | $\mathbf{CL\text{-}HID}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, b)$ : |
|---|---|
| 01 $\mathsf{p} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ | 01 $\mathsf{p} \xleftarrow{\$} \mathsf{Setup}(\lambda)$ |
| 02 $(dk, ek) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{p})$ | 02 $(dk, ek) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{p})$ |
| 03 $(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}(ek)$ | 03 $(c, st) \xleftarrow{\$} \mathcal{A}(ek)$ |
| 04 $r \xleftarrow{\$} \mathcal{R}_\mathsf{p}$ | 04 $c_0 \xleftarrow{\$} \mathcal{C}_\mathsf{p}$ |
| 05 $c := \mathsf{Enc}(ek, m_b, r)$ | 05 $r \xleftarrow{\$} \mathcal{R}_\mathsf{p}$ ; $c_1 := \mathsf{Rndmz}(ek, c, r)$ |
| 06 $b' \xleftarrow{\$} \mathcal{A}(st, c)$ | 06 $b' \xleftarrow{\$} \mathcal{A}(st, c_b)$ |
| 07 Return $b'$ | 07 Return $b'$ |

Figure 4.1: Games for ciphertext-indistinguishability and class-hiding

We define the equivalence class $[c]_{ek}$ of a ciphertext $c$ under encryption key $ek$ as all randomizations of $c$, that is, $[c]_{ek} := \{c' \mid \exists\, r \in \mathcal{R}_\mathsf{p} : c' = \mathsf{Rndmz}(ek, c, r)\}$.

### Correctness and security definitions

**Correctness** of SoRC requires that the encryption scheme and the signature scheme are correct.

**Definition 4.2.** *A SoRC scheme is* correct *if for all* $\mathsf{p} \in \mathcal{PP}$*, for all pairs* $(ek, dk)$ *and* $(sk, vk)$ *in the range of* $\mathsf{KeyGen}(\mathsf{p})$ *and* $\mathsf{SKeyGen}(\mathsf{p})$*, respectively, and all* $m \in \mathcal{M}_\mathsf{p}$*,* $r \in \mathcal{R}_\mathsf{p}$ *and* $c \in \mathcal{C}_\mathsf{p}$*:*

$$\mathsf{Dec}(dk, \mathsf{Enc}(ek, m, r)) = m \qquad \textit{and} \qquad \Pr\left[\mathsf{Verify}(vk, ek, c, \mathsf{Sign}(sk, ek, c)) = 1\right] = 1\ . \tag{4.6}$$

Note that together with signature-adaptation (Def. 4.5 below), this implies that adapted signatures verify as well. We also require that the encryption scheme satisfies the standard security notion.

**Definition 4.3.** *Let game* **IND-CPA** *be as defined in Fig. 4.1. A SoRC scheme* $\mathcal{SRC}$ *is* IND-CPA secure *if for all p.p.t. adversary* $\mathcal{A}$ *the following function is negligible in* $\lambda$*:*

$$\left|\Pr\left[\mathbf{IND\text{-}CPA}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, 1) = 1\right] - \Pr\left[\mathbf{IND\text{-}CPA}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, 0) = 1\right|\ .$$

**Class-hiding** is a property of equivalence-class signatures that states that given a representative of an equivalence class, then a random member of that class is indistinguishable from a random element of the whole space. We give a stronger definition, which we call *fully class-hiding* (analogously to full anonymity). Whereas in the original notion [FHS19, Def. 18], the representative is uniformly picked by the experiment, in our notion it is chosen by the adversary.

**Definition 4.4.** *Let game* **CL-HID** *be as defined in Fig. 4.1. A SoRC scheme is* fully class-hiding *if for all p.p.t. adversary* $\mathcal{A}$*, the following function is negligible in* $\lambda$*:*

$$\left|\Pr\left[\mathbf{CL\text{-}HID}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, 1) = 1\right] - \Pr\left[\mathbf{CL\text{-}HID}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, 0) = 1\right]\right|\ .$$

**Signature-adaptation** requires that signatures that have been adapted to a randomization of the signed ciphertext are distributed like fresh signatures on the randomized ciphertext. A strengthening is the following variant, which also holds for maliciously generated verification keys [FHS19, Def. 20].

**Definition 4.5.** *A SoRC scheme is* signature-adaptable (under malicious keys) *if for all* $\mathsf{p} \in \mathcal{PP}$*, all* $(vk, ek, c, sig) \in \mathcal{VK}_\mathsf{p} \times \mathcal{EK}_\mathsf{p} \times \mathcal{C}_\mathsf{p} \times \mathcal{S}_\mathsf{p}$ *that satisfy* $\mathsf{Verify}(vk, ek, c, sig) = 1$ *and all* $r \in \mathcal{R}_\mathsf{p}$*, the output of* $\mathsf{Adapt}(sig, r)$ *is uniformly distributed over the set*

$$\left\{sig' \in \mathcal{S}_\mathsf{p} \mid \mathsf{Verify}(vk, ek, \mathsf{Rndmz}(ek, c, r), sig') = 1\right\}\ .$$

$\mathbf{EUF}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda)$ :
01 $Q := \emptyset$ ; $\mathsf{p} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
02 $(sk, vk) \xleftarrow{\$} \mathsf{SKeyGen}(\mathsf{p})$
03 $\big((ek^*, c^*), sig^*\big) \xleftarrow{\$} \mathcal{A}_2^{\mathtt{Sign}(sk,\cdot,\cdot)}(vk)$
04 Return $\big(\mathsf{Verify}(vk, ek^*, c^*, sig^*) = 1 \wedge (ek^*, c^*) \notin Q\big)$

$\mathtt{Sign}(sk, ek, c)$
01 $Q := Q \cup \{ek\} \times [c]_{ek}$
02 Return $\mathsf{Sign}(sk, ek, c)$

Figure 4.2: Unforgeability game

Note that if Sign outputs a uniform element in the set of valid signatures (which is the case in the ECS scheme from [FHS19] and our scheme) then Def. 4.5 implies that for all honestly generated $(sk, vk)$ and all $ek, c$ and $r$ the outputs of the following two procedures are distributed equivalently:

$$\mathsf{Adapt}(\mathsf{Sign}(sk, ek, c), r') \qquad \text{and} \qquad \mathsf{Sign}(sk, ek, \mathsf{Rndmz}(ek, c, r')) \ . \tag{4.7}$$

Together, full class-hiding and signature-adaptability under malicious keys imply that for an adversary that creates a signature verification key as well as a ciphertext and a signature on it, a randomization of this ciphertext together with an adapted signature looks like a random ciphertext with a fresh signature on it. (In contrast, for equivalence-class signatures, this was only true if the signed message was not chosen by the adversary [FHS19].)

**Unforgeability.** Finally, we present our strengthened notion of unforgeability, which is defined w.r.t. keys and equivalence classes. That is, after the adversary queries a signature for $(ek, c)$, all tuples $(ek, c')$ with $c' \in [c]_{ek}$ (that is, $c'$ encrypts the same message as $c$ under $ek$) are added to a set $Q$ of signed objects. The adversary's goal is to produce a signature on a pair $(ek^*, c^*)$ that is not contained in $Q$. (In the original definition [BFPV11], $Q$ would contain the equivalence classes of $c$ under *all* encryption keys, i.e., all encryptions of the plaintext of $c$ under all keys.)

**Definition 4.6.** *Let* **EUF** *be the game defined in Fig. 4.2. A SoRC scheme is* unforgeable *is for all p.p.t. adversary $\mathcal{A}$ the following function is negligible in $\lambda$:*

$$\Pr\left[\mathbf{EUF}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda) = 1\right] \ .$$

## 4.2 Instantiation

Our instantiation of SoRC is given in Fig. 4.3. Its signatures sign ElGamal ciphertexts $(C_0, C_1)$, and the signature elements $(Z, S, \hat{S})$ constitute a structure-preserving signature on $(C_0, C_1)$ similar to the optimal scheme from [AGHO11]. (And removing $G$ from the definition of $Z$ would yield the equivalence-class scheme from [FHS19]: note that, without $G$, multiplying $Z$ by $r$ yields a signature on the message $r \cdot (C_0, C_1)$.) The new element $T$ in our scheme allows for adaptation of signatures to randomizations of the signed ciphertext. Randomization implicitly defines the following equivalence classes: for $P \in \mathcal{EK}_{\mathsf{p}}$ and $(C_0, C_1), (C_0', C_1') \in \mathcal{C}_{\mathsf{p}}$:

$$(C_0', C_1') \in [(C_0, C_1)]_P \iff \exists\, r \in \mathbb{Z}_p : (C_0', C_1') = (C_0 + rG,\ C_1 + rP) \ .$$

## 4.3 Security of our scheme

Correctness of our scheme follows by inspection. Moreover, ElGamal encryption [ElG85] satisfies IND-CPA if the decisional Diffie-Hellman (DDH) assumption holds for BGGen.

**Property 4.7.** *If DDH holds for BGGen then the scheme in Fig. 4.3 is fully class-hiding (Def. 4.4).*

$\mathsf{Setup}(1^\lambda)$: Return $\mathsf{p} = (p, \mathbb{G}, G, \hat{\mathbb{G}}, \hat{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, which define $\mathcal{M}_\mathsf{p} := \mathbb{G}$, $\mathcal{C}_\mathsf{p} := \mathbb{G}^2$, $\mathcal{R}_\mathsf{p} := \mathbb{Z}_p$, $\mathcal{SK}_\mathsf{p} := (\mathbb{Z}_p^*)^2$, $\mathcal{VK}_\mathsf{p} := (\hat{\mathbb{G}}^*)^2$, $\mathcal{EK}_\mathsf{p} := \mathbb{G}^*$, $\mathcal{DK}_\mathsf{p} := \mathbb{Z}_p^*$ and $\mathcal{S}_\mathsf{p} := \mathbb{G} \times \mathbb{G}^* \times \hat{\mathbb{G}}^* \times \mathbb{G}$.

$\mathsf{KeyGen}(\mathsf{p})$: Parse $\mathsf{p}$ as $(p, \mathbb{G}, G, \hat{\mathbb{G}}, \hat{G}, \mathbb{G}_T, e)$
$dk := d \xleftarrow{\$} \mathbb{Z}_p^*$ ; $ek = P = dG$ ; return $(dk, ek)$

$\mathsf{Enc}(P, M, r)$: Return $(rG, M + rP)$

$\mathsf{Dec}(d, (C_0, C_1))$: Return $M := C_1 - dC_0$

$\mathsf{Rndmz}(P, (C_0, C_1), r')$: Return $(C_0 + r'G, C_1 + r'P)$

$\mathsf{SKeyGen}(\mathsf{p})$: Parse $\mathsf{p}$ as $(p, \mathbb{G}, G, \hat{\mathbb{G}}, \hat{G}, \mathbb{G}_T, e)$
$sk := (x_0, x_1) \xleftarrow{\$} (\mathbb{Z}_p^*)^2$ ; $vk := (\hat{X}_0 = x_0\hat{G}, \hat{X}_1 = x_1\hat{G})$ ; return $(sk, vk)$

$\mathsf{Sign}((x_0, x_1), P, (C_0, C_1))$: $s \xleftarrow{\$} \mathbb{Z}_p^*$ ; return $(Z, S, \hat{S}, T)$ with

$$Z := \frac{1}{s}\big(G + x_0C_0 + x_1C_1\big) \qquad S := sG \qquad \hat{S} := s\hat{G} \qquad T := \frac{1}{s}\big(x_0G + x_1P\big) \tag{4.8}$$

$\mathsf{Adapt}\big((Z, S, \hat{S}, T), r'\big)$: $s' \xleftarrow{\$} \mathbb{Z}_p^*$ ; return $(Z', S', \hat{S}', T')$ with

$$Z' := \frac{1}{s'}\big(Z + r'T\big) \qquad S' := s'S \qquad \hat{S}' := s'\hat{S} \qquad T' := \frac{1}{s'}T \tag{4.9}$$

$\mathsf{Verify}((\hat{X}_0, \hat{X}_1), P, (C_0, C_1), (Z, S, \hat{S}, T))$: Return 0 if $P = 0$ or $S = 0$; return 1 if the following equations hold and 0 otherwise:

$$e(Z, \hat{S}) = e(G, \hat{G})e(C_0, \hat{X}_0)e(C_1, \hat{X}_1) \qquad e(G, \hat{S}) = e(S, \hat{G}) \tag{4.10}$$
$$e(T, \hat{S}) = e(G, \hat{X}_0)e(P, \hat{X}_1) \tag{4.11}$$

Figure 4.3: Our instantiation $\mathcal{SRC}$ of SoRC

*Proof.* We first recall the game **DDH**, which formalizes the DDH assumption in Fig. 4.4 (left). Next, we instantiate **CL-HID** with our scheme from Fig. 4.3 and rewrite it in Fig. 4.4 (right). In particular, instead of choosing $c_0 \xleftarrow{\$} \mathbb{G}^2$, we compute it as $c + c_0'$ for a uniform $c_0' \xleftarrow{\$} \mathbb{G}^2$.

Let $\mathcal{A}$ be an adversary against **CL-HID**. We define an adversary $\mathcal{B}$ against **DDH**, which upon receiving a challenge $(P, R, S)$, sends $P$ to $\mathcal{A}$ to get $c$ and then sends $c + (R, S)$ to $\mathcal{A}$. Finally, $\mathcal{B}$ returns $\mathcal{A}$'s output $b'$.

| $\mathbf{DDH}^{\mathcal{B}}_{\mathsf{BGGen}}(\lambda, b)$ : | $\mathbf{CL\text{-}HID}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, b)$ : |
|---|---|
| 01 $(p, \mathbb{G}, G, \hat{\mathbb{G}}, \hat{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathsf{BGGen}(\lambda)$ | 01 $\mathsf{p} \xleftarrow{\$} \mathsf{Setup}(\lambda)$ |
| 02 $P \xleftarrow{\$} \mathbb{G}$ | 02 $(d, P) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{p})$ |
| 03 $r \xleftarrow{\$} Z_p$ | 03 $(c, st) \xleftarrow{\$} \mathcal{A}(P)$ |
| 04 $S_1 := rP$ | 04 $c_0' \xleftarrow{\$} \mathbb{G} \times \mathbb{G}$ |
| 05 $S_0 \xleftarrow{\$} \mathbb{G}$ | 05 $r \xleftarrow{\$} \mathcal{R}_\mathsf{p}$ ; $c_1' := (rG, rP)$ |
| 06 $b' \xleftarrow{\$} \mathcal{B}(\mathsf{p}, (P, rG, S_b))$ ; Return $b'$ | 06 $b' \xleftarrow{\$} \mathcal{A}(st, c + c_b')$ ; Return $b'$ |

Figure 4.4: Games for decisional Diffie-Hellman and class-hiding instantiated with $\mathcal{SRC}$ from Fig. 4.3

Since for all $\lambda$ and $b$ we have that $\mathbf{DDH}^{\mathcal{B}^{\mathcal{A}}}_{\mathcal{SRC}}(\lambda, b)$ and $\mathbf{CL\text{-}HID}^{\mathcal{A}}_{\mathcal{SRC}}(\lambda, b)$ follow the same distribution, $\mathcal{B}$'s advantage in breaking DDH is the same as $\mathcal{A}$'s advantage in breaking full class-hiding. □

**Property 4.8.** *The SoRC scheme in Fig. 4.3 is signature-adaptable under malicious keys (Def. 4.5).*

*Proof.* Let $\mathsf{p} = (p, \mathbb{G}, G, \hat{\mathbb{G}}, \hat{G}, \mathbb{G}_T, e) \in \mathcal{PP}$, let $vk = (x_0\hat{G}, x_1\hat{G})$, $ek = dG$, $C_0 = c_0G$, $C_1 = c_1G$ and $sig = (Z = zG, S = sG, \hat{S} = \hat{s}\hat{G}, T = tG)$ be such that $\mathsf{Verify}(vk, ek, (C_0, C_1), sig) = 1$. Taking the logarithms in basis $e(G, \hat{G})$ of the verification equations yields $\hat{s} = s$ and, using this,

$$zs = z\hat{s} = 1 + c_0x_0 + c_1x_1 \tag{4.12}$$
$$ts = t\hat{s} = x_0 + dx_1 \tag{4.13}$$

Let us now consider a uniform random element $sig' = (Z' = z'G, S' = s'G, \hat{S}' = \hat{s}'\hat{G}, T' = t'G)$ from the set $\{sig' \in \mathcal{S}_{\mathsf{p}} \mid \mathsf{Verify}(vk, ek, \mathsf{Rndmz}(ek, c, r), sig') = 1\}$. Again considering logarithms of the verification equation yields $\hat{s}' = s'$ and

$$z's' = 1 + (c_0 + r)x_0 + (c_1 + rd)x_1 = 1 + c_0x_0 + c_1x_1 + r(x_0 + dx_1) \overset{(4.12),(4.13)}{=} zs + rts \tag{4.14}$$
$$t's' = x_0 + dx_1 \overset{(4.13)}{=} ts \tag{4.15}$$

Moreover, by signature validity, we have $s \neq 0$ and $s' \neq 0$. We thus have $Z' = \frac{s}{s'}(Z + rT)$ and $T' = \frac{s}{s'}T$, as well as $S' = \frac{s'}{s}S$ and $\hat{S}' = \frac{s'}{s}\hat{S}$ (since $\hat{s} = s$ and $\hat{s}' = s'$). In other words, $sig'$ is a uniform element from the set $\{(\frac{1}{s^*}(Z + rT), s^*S, s^*\hat{S}, T' = \frac{1}{s^*}T) \mid s^* \in Z_p^*\}$. Since $\mathsf{Adapt}(sig, r)$ outputs a uniform random element from that set, this concludes the proof. □

## Proof of unforgeability

Our main technical result is to prove that our scheme satisfies unforgeability (Def. 4.6) in the generic group model [Sho97] for asymmetric ("Type-3") bilinear groups (for which there are no efficiently computable homomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$). In this model, the adversary is only given *handles* of group elements, which are just uniform random strings. To perform group operations, it uses an oracle to which it can submit handles and is given back the handle of the sum, inversion, etc of the group elements for which it submitted handles.

**Theorem 4.9.** *A generic adversary $\mathcal{A}$ that computes at most $q$ group operations and makes up to $k$ queries to its signature oracle cannot win the game $\mathbf{EUF}^{\mathcal{A}}_{\mathsf{S}}(\lambda)$ from Fig. 4.2 for $\mathsf{S}$ defined in Fig. 4.3 with probability greater than $2^{-\lambda}\,(2k+1)\,(q+3k+3)^2$.*

*Proof.* We consider an adversary that only uses generic group operations on the group elements it receives. After getting a verification key $(\hat{X}_0 = x_0\hat{G}, \hat{X}_1 = x_1\hat{G})$ and signatures $(Z_i, S_i, \hat{S}_i, T_i)_{i=1}^k$ computed with randomness $s_i$ on queries $\left((P^{(i)}, (C_0^{(i)}, C_1^{(i)}))\right)_{i=1}^k$, the adversary outputs an encryption key $P^{(k+1)}$, a ciphertext $(C_0^{(k+1)}, C_1^{(k+1)})$ and a signature $(Z^*, S^*, \hat{S}^*, T^*)$ for them. As it must compute any new group element by combining received group elements, it must choose coefficients $\psi^{(i)}, \psi^{(i)}_{z,1}, \ldots, \psi^{(i)}_{z,i-1}, \psi^{(i)}_{s,1}, \ldots, \psi^{(i)}_{s,i-1}, \psi^{(i)}_{t,1}, \ldots, \psi^{(i)}_{t,i-1}, \gamma^{(i)}, \gamma^{(i)}_{z,1}, \ldots, \gamma^{(i)}_{z,i-1}, \gamma^{(i)}_{s,1}, \ldots, \gamma^{(i)}_{s,i-1}, \gamma^{(i)}_{t,1}, \ldots, \gamma^{(i)}_{t,i-1}, \kappa^{(i)}, \kappa^{(i)}_{z,1}, \ldots, \kappa^{(i)}_{z,i-1}, \kappa^{(i)}_{s,1}, \ldots, \kappa^{(i)}_{s,i-1}, \kappa^{(i)}_{t,1}, \ldots, \kappa^{(i)}_{t,i-1}$ for all $i \in \{1, \ldots, k+1\}$, as well as $\sigma, \sigma_{z,1}, \ldots, \sigma_{z,k}, \sigma_{s,1}, \ldots, \sigma_{s,k}, \sigma_{t,1}, \ldots, \sigma_{t,k}$, $\tau, \tau_{z,1}, \ldots, \tau_{z,k}, \tau_{s,1}, \ldots, \tau_{s,k}, \tau_{t,1}, \ldots, \tau_{t,k}$, $\zeta$,

$\zeta_{z,1},\ldots,\zeta_{z,k},\zeta_{s,1},\ldots,\zeta_{s,k},\zeta_{t,1},\ldots,\zeta_{t,k}$, $\phi,\phi_0,\phi_1,\phi_{s,1},\ldots,\phi_{s,k}$, which define

$$P^{(i)} = \psi^{(i)}G + \sum_{j=1}^{i-1}\left(\psi^{(i)}_{z,j}Z_j + \psi^{(i)}_{s,j}S_j + \psi^{(i)}_{t,j}T_j\right) \quad Z^* = \zeta G + \sum_{j=1}^{k}\left(\zeta_{z,j}Z_j + \zeta_{s,j}S_j + \zeta_{t,j}T_j\right) \tag{4.16}$$

$$C_0^{(i)} = \gamma^{(i)}G + \sum_{j=1}^{i-1}\left(\gamma^{(i)}_{z,j}Z_j + \gamma^{(i)}_{s,j}S_j + \gamma^{(i)}_{t,j}T_j\right) \quad S^* = \sigma G + \sum_{j=1}^{k}\left(\sigma_{z,j}Z_j + \sigma_{s,j}S_j + \sigma_{t,j}T_j\right) \tag{4.17}$$

$$C_1^{(i)} = \kappa^{(i)}G + \sum_{j=1}^{i-1}\left(\kappa^{(i)}_{z,j}Z_j + \kappa^{(i)}_{s,j}S_j + \kappa^{(i)}_{t,j}T_j\right) \quad T^* = \tau G + \sum_{j=1}^{k}\left(\tau_{z,j}Z_j + \tau_{s,j}S_j + \tau_{t,j}T_j\right) \tag{4.18}$$

$$\hat{S}^* = \phi\hat{G} + \phi_0\hat{X}_0 + \phi_1\hat{X}_1 + \sum_{j=1}^{k}\phi_{s,j}\hat{S}_j \tag{4.19}$$

Using this, we can write, for all $1 \le i \le k$, the discrete logarithms $z_i$ and $t_i$ in basis $G$ of the elements $Z_i = \frac{1}{s_i}(G + x_0C_0^{(i)} + x_1C_1^{(i)})$ and $T_i = \frac{1}{s_i}(x_0G + x_1P^{(i)})$ from the oracle answers.

$$z_i = \frac{1}{s_i}\Big(1 + x_0\Big(\gamma^{(i)} + \sum_{j=1}^{i-1}\left(\gamma^{(i)}_{z,j}z_j + \gamma^{(i)}_{s,j}s_j + \gamma^{(i)}_{t,j}t_j\right)\Big) + x_1\Big(\kappa^{(i)} + \sum_{j=1}^{i-1}\left(\kappa^{(i)}_{z,j}z_j + \kappa^{(i)}_{s,j}s_j + \kappa^{(i)}_{t,j}t_j\right)\Big)\Big) \tag{4.20}$$

$$t_i = \frac{1}{s_i}\Big(x_0 + x_1\Big(\psi^{(i)} + \sum_{j=1}^{i-1}\left(\psi^{(i)}_{z,j}z_j + \psi^{(i)}_{s,j}s_j + \psi^{(i)}_{t,j}t_j\right)\Big)\Big) \tag{4.21}$$

We interpret these values as multivariate rational fractions in variables $x_0, x_1, s_1, \ldots, s_k$. A successful forgery $(Z^*, S^*, \hat{S}^*, T^*)$ on $(P^{(k+1)}, (C_0^{(k+1)}, C_1^{(k+1)}))$ satisfies the verification equations

$$e(Z^*,\hat{S}^*) = e(G,\hat{G})e(C_0^{(k+1)},\hat{X}_0)e(C_1^{(k+1)},\hat{X}_1) \qquad e(G,\hat{S}^*) = e(S^*,\hat{G}) \tag{4.22}$$

$$e(T^*,\hat{S}^*) = e(G,\hat{X}_0)e(P^{(k+1)},\hat{X}_1) \tag{4.23}$$

Using the coefficients defined above and considering the logarithms in base $e(G,\hat{G})$ we obtain:

$$\Big(\zeta + \sum_{j=1}^{k}\left(\zeta_{z,j}z_j + \zeta_{s,j}s_j + \zeta_{t,j}t_j\right)\Big)\Big(\phi + \phi_0x_0 + \phi_1x_1 + \sum_{i=1}^{k}\phi_{s,i}s_i\Big) = 1 + x_0c_0^{(k+1)} + x_1c_1^{(k+1)} \tag{4.24}$$

$$\phi + \phi_0x_0 + \phi_1x_1 + \sum_{i=1}^{k}\phi_{s,i}s_i = \sigma + \sum_{j=1}^{k}\left(\sigma_{z,j}z_j + \sigma_{s,j}s_j + \sigma_{t,j}t_j\right) \tag{4.25}$$

$$\Big(\tau + \sum_{j=1}^{k}\left(\tau_{z,j}z_j + \tau_{s,j}s_j + \tau_{t,j}t_j\right)\Big)\Big(\phi + \phi_0x_0 + \phi_1x_1 + \sum_{i=1}^{k}\phi_{s,i}s_i\Big) = x_0 + x_1d^{(k+1)} \tag{4.26}$$

$$\text{where for all } i \in \{1,\ldots,k+1\} : c_0^{(i)} = \log C_0^{(i)} = \gamma^{(i)} + \sum_{j=1}^{i-1}\left(\gamma^{(i)}_{z,j}z_j + \gamma^{(i)}_{s,j}s_j + \gamma^{(i)}_{t,j}t_j\right), \tag{4.27}$$

$$c_1^{(i)} = \kappa^{(i)} + \sum_{j=1}^{i-1}\left(\kappa^{(i)}_{z,j}z_j + \kappa^{(i)}_{s,j}s_j + \kappa^{(i)}_{t,j}t_j\right) \text{ and } d^{(i)} = \log P^{(i)} = \psi^{(i)} + \sum_{j=1}^{i-1}\left(\psi^{(i)}_{z,j}z_j + \psi^{(i)}_{s,j}s_j + \psi^{(i)}_{t,j}t_j\right).$$

We follow the standard proof technique for results in the generic group model and now consider an "ideal" game in which the challenger treats all the (handles of) group elements as elements of $\mathbb{Z}_p(s_1,\ldots,s_k,x_0,x_1)$, that is, rational fractions whose indeterminates represent the secret values chosen by the challenger.

We first show that in the ideal game if the adversary's output satisfies the verification equations, then the second winning condition, $(P^{(k+1)}, (C_0^{(k+1)}, C_1^{(k+1)})) \notin Q$, is not satisfied, which demonstrates that the ideal game cannot be won. We then compute the statistical distance from the adversary's point of view between the real and the ideal game at the end of the proof.

In the ideal game we thus interpret the three equalities (4.24), (4.25) and (4.26) as polynomial equalities over the field $\mathbb{Z}_p(s_1,\ldots,s_k,x_0,x_1)$. More precisely, we consider the equalities in the ring $\mathbb{Z}_p(s_1,\ldots,s_k)[x_0,x_1]$, that is, the polynomial ring with $x_0$ and $x_1$ as indeterminates over the field $\mathbb{Z}_p(s_1,\ldots,s_k)$. (Note that this interpretation is possible because $x_0$ and $x_1$ never appear in the denominators of any expressions.) As one of our proof techniques, we will also consider the equalities over the ring factored by $(x_0,x_1)$, the ideal generated by $x_0$ and $x_1$:[1]

$$\mathbb{Z}_p(s_1,\ldots,s_k)[x_0,x_1]/(x_0,x_1) \cong \mathbb{Z}_p(s_1,\ldots,s_k)\ .$$

From (4.20) and (4.21), over this quotient we have $z_i = \frac{1}{s_i}$ and $t_i = 0$ and thus (4.24)–(4.26) become

$$\Big(\zeta + \sum_{j=1}^{k}\big(\zeta_{z,j}\frac{1}{s_j} + \zeta_{s,j}s_j\big)\Big)\Big(\phi + \sum_{i=1}^{k}\phi_{s,i}s_i\Big) = 1 \tag{4.28}$$

$$\phi + \sum_{i=1}^{k}\phi_{s,i}s_i = \sigma + \sum_{i=1}^{k}\big(\sigma_{z,i}\frac{1}{s_i} + \sigma_{s,i}s_i\big) \tag{4.29}$$

$$\Big(\tau + \sum_{i=1}^{k}\big(\tau_{z,i}\frac{1}{s_i} + \tau_{s,i}s_i\big)\Big)\Big(\phi + \sum_{i=1}^{k}\phi_{s,i}s_i\Big) = 0 \tag{4.30}$$

We first consider (4.29). By equating coefficients, we deduce:

$$\phi = \sigma \qquad \forall i \in \{1,\ldots,k\}:\ \phi_{s,i} = \sigma_{s,i} \quad \text{and} \quad \sigma_{z,i} = 0 \tag{4.31}$$

We now turn to (4.28) and first notice that

$$\Big(\phi + \sum_{i=1}^{k}\phi_{s,i}s_i\Big) \neq 0\ , \tag{4.32}$$

because it is a factor of a non-zero product in (4.28). We next consider the degrees of the factors in (4.28), using the fact that the degree of a product is the sum of the degrees of the factors. Let $i \in \{1,\ldots,k\}$. Since $\deg_{s_i}(1) = 0$ and $\deg_{s_i}\big(\phi + \sum_{i=1}^{k}\phi_{s,i}s_i\big) \geq 0$, we have $\deg_{s_i}\big(\zeta + \sum_{j=1}^{k}(\zeta_{z,j}\frac{1}{s_j} + \zeta_{s,j}s_j)\big) \leq 0$, from which we get

$$\forall\, i \in \{1,\ldots,k\} : \zeta_{s,i} = 0\ . \tag{4.33}$$

(Note that $\deg_{s_i}\big(\frac{1}{s_i} + s_i\big) = \deg_{s_i}\big(\frac{1+s_i^2}{s_i}\big) = 1$.) We next show that there is at most one $\phi_{s,i}$ that is non-zero. Suppose there exist $i_1 \neq i_2 \in \{1,\ldots,k\}$ such that $\phi_{s,i_1} \neq 0$ and $\phi_{s,i_2} \neq 0$. This implies that $\deg_{s_{i_1}}\big(\phi + \sum_{i=1}^{k}\phi_{s,i}s_i\big) = \deg_{s_{i_2}}\big(\phi + \sum_{i=1}^{k}\phi_{s,i}s_i\big) = 1$. By considering these degrees in (4.28), the left factor must be of degree $-1$, that is (recall that $\zeta_{s,i} = 0$ for all $i$ by (4.33)):

$$\deg_{s_{i_1}}\Big(\zeta + \sum_{j=1}^{k}\zeta_{z,j}\frac{1}{s_j}\Big) = -1 \quad \text{and} \quad \deg_{s_{i_2}}\Big(\zeta + \sum_{j=1}^{k}\zeta_{z,j}\frac{1}{s_j}\Big) = -1\ . \tag{4.34}$$

This is a contradiction since the former implies that $\zeta_{z,i_1} \neq 0$, while the latter implies that $\zeta_{z,i_1} = 0$, as we show next. Consider the expression $\deg_{s_{i_2}}\big((\zeta + \sum_{j=1,j\neq i_2}^{k}\zeta_{z,j}\frac{1}{s_j})s_{i_2} + \zeta_{z,i_2}\big) = \deg_{s_{i_2}}\big((\zeta +$

[1] Considering an equation of rational fractions over this quotient can also be seen as simply setting $x_0 = x_1 = 0$. Everything we infer about the coefficients from these modified equations is also valid for the original equation, since these must hold for all values $(x_0,x_1,s_1,\ldots,s_k)$ and so in particular for $(0,0,s_1,\ldots,s_k)$.

Yet another interpretation when equating coefficients in equations modulo $(x_0,x_1)$ is that one equates coefficients only of monomials that do not contain $x_0$ or $x_1$.

$\sum_{j=1}^{k} \zeta_{z,j}\frac{1}{s_j})s_{i_2}) = -1 + \deg_{s_{i_2}}(s_{i_2}) = 0$ , by using (4.34). This implies $(\zeta + \sum_{j=1,j\neq i_2}^{k} \zeta_{z,j}\frac{1}{s_j}) = 0$ and thus $\zeta_{z,i_1} = 0$, which was our goal.

Therefore, there exists $i_0$ such that, for all $i \neq i_0$, $\phi_{s,i} = 0$ and by (4.31):

$$\forall\, i \in \{1,\ldots,k\} \setminus \{i_0\} : \sigma_{s,i} = \phi_{s,i} = 0\ . \tag{4.35}$$

Together with (4.33), this means that we can rewrite (4.28) as $(\zeta + \sum_{j=1}^{k} \zeta_{z,j}\frac{1}{s_j})(\phi + \phi_{s,i_0}s_{i_0}) = 1$. Since for all $i \neq i_0$, $s_i$ does not appear in 1, we have

$$\forall\, i \in \{1,\ldots,k\} \setminus \{i_0\} : \zeta_{z,i} = 0\ . \tag{4.36}$$

We now consider equation (4.25) modulo $(x_1)$. Since, by (4.31), $\phi = \sigma$ and $\phi_{s,i} = \sigma_{s,i}$ for all $i$, two terms cancel on both sides. Moreover, by (4.31), $\sigma_{z,i} = 0$ for all $i$ and thus, using $t_i \bmod (x_1) = \frac{x_0}{s_i}$ for all $i$, yields

$$\phi_0 x_0 = \sum_{i=1}^{k} \sigma_{t,i}\frac{x_0}{s_i}\ . \tag{4.37}$$

By identifying coefficients, we deduce that

$$\forall\, i \in \{1,\ldots,k\} : \sigma_{t,i} = \phi_0 = 0\ . \tag{4.38}$$

Using all of this in the original equation (4.25) (that is, "putting back" $x_1$ in (4.37) and applying (4.38)) yields $\phi_1 x_1 = 0$ and thus

$$\phi_1 = 0\ . \tag{4.39}$$

We now turn to (4.30), in which by (4.32) we have $(\tau + \sum_{i=1}^{k}(\tau_{z,i}\frac{1}{s_i} + \tau_{s,i}s_i)) = 0$. From this we get by equating coefficients:

$$\forall\, i \in \{1,\ldots,k\} : \tau_{z,i} = \tau_{s,i} = \tau = 0\ . \tag{4.40}$$

Going back to equation (4.26) and applying the latter, as well as (4.38), (4.39) and (4.35) yields

$$\Big(\sum_{i=1}^{k} \tau_{t,i}t_i\Big)(\phi + \phi_{s,i_0}s_{i_0}) = x_0 + x_1\Big(\psi^{(k+1)} + \sum_{j=1}^{k}\big(\psi_{z,j}^{(k+1)}z_j + \psi_{s,j}^{(k+1)}s_j + \psi_{t,j}^{(k+1)}t_j\big)\Big)\ . \tag{4.41}$$

Computing this modulo $(x_1)$ and recalling $t_i \bmod (x_1) = \frac{x_0}{s_i}$ yields $(\sum_{i=1}^{k} \tau_{t,i}\frac{x_0}{s_i})(\phi + \phi_{s,i_0}s_{i_0}) = x_0$, and thus

$$\sum_{i=1}^{k} \phi\tau_{t,i}\frac{x_0}{s_i} + \sum_{i=1,i\neq i_0}^{k} \phi_{s,i_0}\tau_{t,i}s_{i_0}\frac{x_0}{s_i} + \phi_{s,i_0}\tau_{t,i_0}x_0 = x_0\ . \tag{4.42}$$

By equating the coefficients for $x_0$, we deduce that

$$\phi_{s,i_0}\tau_{t,i_0} = 1 \quad (\text{and thus } \phi_{s,i_0} \neq 0 \text{ and } \tau_{t,i_0} \neq 0)\,. \tag{4.43}$$

Moreover, for all $i \in \{1,\ldots,k\} \setminus \{i_0\}$, we deduce $\phi_{s,i_0}\tau_{t,i} = 0$ and $\phi\tau_{t,i_0} = 0$, which by applying (4.43) to both yields

$$\forall\, i \in \{1,\ldots,k\} \setminus \{i_0\} : \tau_{t,i} = 0 \quad \text{and} \quad \phi = 0\ . \tag{4.44}$$

Using this, the left-hand side of (4.41) becomes $\phi_{s,i_0}\tau_{t,i_0}t_{i_0}s_{i_0}$, which, applying (4.43) and (4.21), becomes $\frac{1}{s_{i_0}}(x_0 + x_1 d^{(i_0)})s_{i_0}$. This means that (4.41) becomes $x_0 + x_1 d^{(i_0)} = x_0 + x_1 d^{(k+1)}$, which

implies $x_1(d^{(i_0)} - d^{(k+1)}) = 0$. Since a polynomial ring over a integral domain such as $\mathbb{Z}_p(s_1, \ldots, s_k)$ is an integral domain, and $x_1 \neq 0$, the last equality implies $d^{(i_0)} = d^{(k+1)}$. This means

$$P^{(i_0)} = P^{(k+1)} \ , \tag{4.45}$$

that is, the encryption key of the forgery is the same as used in the $i_0$-th query. We next show that the ciphertext $(C_0^{(k+1)}, C_1^{(k+1)})$ of the forgery is a randomization of the one from the $i_0$-th query.

Consider equation (4.28). Since $\zeta_{z,i} = 0$ for $i \neq i_0$ (by (4.36)), all $\zeta_{s,i} = 0$ (by (4.33)), $\phi = 0$ (by (4.44)) and $\phi_{s,i} = 0$ for $i \neq i_0$ (by (4.35)), it simplifies to

$$\Big(\zeta + \zeta_{z,i_0}\frac{1}{s_{i_0}}\Big)\phi_{s,i_0}s_{i_0} = \zeta\phi_{s,i_0}s_{i_0} + \zeta_{z,i_0}\phi_{s,i_0} = 1 \ , \tag{4.46}$$

from which we deduce

$$\zeta_{z,i_0}\phi_{s,i_0} = 1 \quad \text{and} \quad \zeta = 0 \ . \tag{4.47}$$

We now consider (4.24) modulo $(x_1)$ and apply what we have deduced so far, that is $\zeta = 0$ by (4.47), the coefficients previously mentioned above (4.46) and $\phi_0 = 0$ by (4.38). The left-hand side of (4.24) modulo $(x_1)$ becomes thus $\Big(\zeta_{z,i_0}z_{i_0} + \sum_{j=1}^{k}\zeta_{t,j}t_j\Big)\phi_{s,i_0}s_{i_0} \bmod (x_1)$. Using moreover (4.47), we get that (4.24) modulo $(x_1)$ becomes

$$\begin{aligned} z_{i_0}s_{i_0} + \Big(\sum_{j=1}^{k}\zeta_{t,j}t_j\Big)\phi_{s,i_0}s_{i_0} \bmod (x_1) \\ = 1 + x_0\Big(\gamma^{(k+1)} + \sum_{j=1}^{k}\big(\gamma_{z,j}^{(k+1)}z_j + \gamma_{s,j}^{(k+1)}s_j + \gamma_{t,j}^{(k+1)}t_j\big)\Big) \bmod (x_1) \ , \end{aligned} \tag{4.48}$$

and using $z_i \bmod (x_1) = \frac{1+c_0^{(i)}x_0}{s_i} \bmod (x_1)$ and $t_i \bmod (x_1) = \frac{x_0}{s_i}$ for all $i$ (cf. (4.20) and (4.21)) we get

$$\begin{aligned} (1 + c_0^{(i_0)})x_0 + \Big(\sum_{j=1}^{k}\zeta_{t,j}\frac{x_0}{s_j}\Big)\phi_{s,i_0}s_{i_0} \bmod (x_1) \\ = 1 + x_0\Big(\gamma^{(k+1)} + \sum_{j=1}^{k}\Big(\gamma_{z,j}^{(k+1)}\frac{1+c_0^{(j)}x_0}{s_j} + \gamma_{s,j}^{(k+1)}s_j + \gamma_{t,j}^{(k+1)}\frac{x_0}{s_j}\Big)\Big) \bmod (x_1) \ . \end{aligned} \tag{4.49}$$

Let $i > i_0$ and let us consider the monomials of degree $-1$ in $s_i$ and degree 0 in $s_j$, for all $j > i$. Note that all monomials of $c_0^{(j)} = \gamma^{(j)} + \sum_{\ell=1}^{j-1}\big(\gamma_{z,\ell}^{(j)}z_\ell + \gamma_{s,\ell}^{(j)}s_\ell + \gamma_{t,\ell}^{(j)}t_\ell\big)$ are of degree 0 in $s_\ell$, for $\ell \geq j$. Therefore, we do not consider any $\frac{c_0^{(j)}}{s_j}$ for $j < i$ (because they do not contain the term $s_i$) nor $\frac{c_0^{(j)}}{s_j}$ for $j > i$ (since the contained monomials are of degree $-1$ in $s_j$ for $j > i$). For the monomials of degree $-1$ in $s_i$ and degree 0 in $s_j$ for $j > i$ in (4.49) we thus have

$$\forall\, i > i_0 : \ \frac{\zeta_{t,i}x_0\phi_{s,i_0}s_{i_0}}{s_i} = x_0\Big(\gamma_{z,i}^{(k+1)}\frac{1+c_0^{(i)}x_0}{s_i} + \gamma_{t,i}^{(k+1)}\frac{x_0}{s_i}\Big) \bmod (x_1) = 0 \ .$$

Multiplying by $s_i$ yields $\zeta_{t,i}x_0\phi_{s,i_0}s_{i_0} - x_0\big(\gamma_{z,i}^{(k+1)}(1 + x_0c_0^{(i)}) + \gamma_{t,i}^{(k+1)}x_0\big) \bmod (x_1) = 0$ and after reordering the monomials according to their degree in $x_0$ we get

$$\forall\, i > i_0 : -x_0^2\big(\gamma_{z,i}^{(k+1)}c_0^{(i)} + \gamma_{t,i}^{(k+1)}\big) + x_0\big(\zeta_{t,i}\phi_{s,i_0}s_{i_0} - \gamma_{z,i}^{(k+1)}\big) \bmod (x_1) = 0 \ . \tag{4.50}$$

Considering the linear coefficient in $x_0$, and recalling that $\phi_{s,i_0} \neq 0$ by (4.43), we deduce

$$\forall\, i > i_0 :\ \gamma_{z,i}^{(k+1)} = \zeta_{t,i} = 0\ . \tag{4.51}$$

Applying this to equation (4.50) yields $x_0^2\,\gamma_{t,i}^{(k+1)} \bmod (x_1) = 0$ for all $i > i_0$, and therefore

$$\forall\, i > i_0 :\ \gamma_{t,i}^{(k+1)} = 0\ . \tag{4.52}$$

Since by (4.51) and (4.52) for all $i > i_0 : \zeta_{t,i} = \gamma_{z,i}^{(k+1)} = \gamma_{t,i}^{(k+1)} = 0$, we can rewrite (4.48) as

$$\begin{aligned} &z_{i_0} s_{i_0} + \Big(\sum_{i=1}^{i_0} \zeta_{t,i} t_i\Big)\phi_{s,i_0} s_{i_0} \bmod (x_1) \\ &\qquad = 1 + x_0\Big(\gamma^{(k+1)} + \sum_{i=1}^{i_0} \big(\gamma_{z,i}^{(k+1)} z_i + \gamma_{t,i}^{(k+1)} t_i\big) + \sum_{i=1}^{k} \gamma_{s,i}^{(k+1)} s_i\Big) \bmod (x_1)\ . \end{aligned} \tag{4.53}$$

For $i > i_0$, from the coefficients of $x_0 s_i$ we get $\gamma_{s,i}^{(k+1)} = 0$. Applying this, (4.51) and (4.52) to (4.27) yields

$$c_0^{(k+1)} = \gamma^{(k+1)} + \sum_{i=1}^{i_0} \big(\gamma_{z,i}^{(k+1)} z_i + \gamma_{s,i}^{(k+1)} s_i + \gamma_{t,i}^{(k+1)} t_i\big)\ ; \tag{4.54}$$

and the right-hand side of (4.53) becomes $1 + x_0\Big(\gamma^{(k+1)} + \sum_{i=1}^{i_0} \big(\gamma_{z,i}^{(k+1)} z_i + \gamma_{s,i}^{(k+1)} s_i + \gamma_{t,i}^{(k+1)} \frac{x_0}{s_i}\big)\Big) \bmod (x_1)$. Since $z_i \bmod (x_1) = \frac{1 + x_0 c_0^{(i)}}{s_i} \bmod (x_1)$ and $t_i \bmod (x_1) = \frac{x_0}{s_i}$, for all $i$, (4.53) becomes

$$\begin{aligned} &1 + x_0 c_0^{(i_0)} + \Big(\sum_{i=1}^{i_0} \zeta_{t,i} \frac{x_0}{s_i}\Big)\phi_{s,i_0} s_{i_0} \bmod (x_1) \\ &\qquad = 1 + x_0\Big(\gamma^{(k+1)} + \sum_{i=1}^{i_0} \Big(\gamma_{z,i}^{(k+1)} \frac{1 + x_0 c_0^{(i)}}{s_i} + \gamma_{s,i}^{(k+1)} s_i + \gamma_{t,i}^{(k+1)} \frac{x_0}{s_i}\Big)\Big) \bmod (x_1)\ . \end{aligned} \tag{4.55}$$

We will now look at the coefficients of $s_{i_0}$ and of $\frac{1}{s_{i_0}}$. For this, we first note that for $j \geq i$ no $s_j$ appears in $c_0^{(i)}$ (cf. (4.27)) and therefore for all $i \leq i_0 : c_0^{(i)}$ is constant in $s_{i_0}$. From the coefficients of $s_{i_0}$ and of $\frac{1}{s_{i_0}}$ we thus get, respectively:

$$\phi_{s,i_0} \sum_{i=1}^{i_0-1} \zeta_{t,i} \frac{x_0}{s_i} = x_0 \gamma_{s,i_0}^{(k+1)} \tag{4.56}$$

$$0 = x_0\big(\gamma_{z,i_0}^{(k+1)}(1 + x_0 c_0^{(i_0)}) + \gamma_{t,i_0}^{(k+1)} x_0\big) \bmod (x_1) \tag{4.57}$$

From (4.56) we get $\gamma_{s,i_0}^{(k+1)} = 0$ and, since $\phi_{s,i_0} \neq 0$ by (4.43),

$$\forall i < i_0 : \zeta_{t,i} = 0\ , \tag{4.58}$$

and from (4.57) we get $\gamma_{z,i_0}^{(k+1)} = 0$ (from the coefficient of $x_0$) and therefore $\gamma_{t,i_0}^{(k+1)} = 0$. Together, this lets us rewrite (4.54) as

$$c_0^{(k+1)} = \gamma^{(k+1)} + \sum_{i=1}^{i_0-1} \big(\gamma_{z,i}^{(k+1)} z_i + \gamma_{s,i}^{(k+1)} s_i + \gamma_{t,i}^{(k+1)} t_i\big)\big)\ . \tag{4.59}$$

Recall that $\hat{S}^* = \phi\hat{G} + \phi_0\hat{X}_0 + \phi_1\hat{X}_1 + \sum_{j=1}^{k} \phi_{s,j}\hat{S}_j$ and $Z^* = \zeta G + \sum_{j=1}^{k} (\zeta_{z,j}Z_j + \zeta_{s,j}S_j + \zeta_{t,j}T_j)$. By (4.44), (4.38), (4.39) and (4.35) we have $\hat{S}^* = \phi_{s,i_0}\hat{S}_{i_0}$ Moreover, by (4.47), (4.36), (4.33), (4.51) and (4.58) we have $Z^* = \zeta_{z,i_0}Z_{i_0} + \zeta_{t,i_0}T_{i_0}$. We can now rewrite (4.24) as:

$$(\zeta_{z,i_0}z_{i_0} + \zeta_{t,i_0}t_{i_0})(\phi_{s,i_0}s_{i_0}) = 1 + x_0c_0^{(k+1)} + x_1c_1^{(k+1)} \ .$$

Since, by (4.47), $\zeta_{z,i_0}\phi_{s,i_0} = 1$ and plugging in the definitions of $z_{i_0}$ and $t_{i_0}$, this yields

$$\begin{aligned} 1 + x_0c_0^{(i_0)} + x_1c_1^{(i_0)} + \zeta_{t,i_0}\phi_{s,i_0}(x_0 + x_1d^{(i_0)}) &= 1 + x_0c_0^{(k+1)} + x_1c_1^{(k+1)} \ , \quad \text{and thus} \\ x_0(c_0^{(i_0)} + \zeta_{t,i_0}\phi_{s,i_0} - c_0^{(k+1)}) &= -x_1(c_1^{(i_0)} + \zeta_{t,i_0}\phi_{s,i_0}d^{(i_0)} - c_1^{(k+1)}) \ . \end{aligned} \tag{4.60}$$

By considering the above modulo $(x_1)$, plugging in the definition of $c_0^{(i)}$ from (4.27) and using (4.59), we get

$$\begin{aligned} 0 &= \zeta_{t,i_0}\phi_{s,i_0} + c_0^{(i_0)} - c_0^{(k+1)} \bmod (x_1) \\ &= \zeta_{t,i_0}\phi_{s,i_0} + \gamma^{(i_0)} - \gamma^{(k+1)} + \sum_{j=1}^{i_0-1} ((\gamma_{z,j}^{(i_0)} - \gamma_{z,j}^{(k+1)})z_j + (\gamma_{s,j}^{(i_0)} - \gamma_{s,j}^{(k+1)})s_j + (\gamma_{t,j}^{(i_0)} - \gamma_{t,j}^{(k+1)})t_j) \\ &\qquad \bmod (x_1) \\ &= \zeta_{t,i_0}\phi_{s,i_0} + \gamma^{(i_0)} - \gamma^{(k+1)} \\ &\quad + \sum_{j=1}^{i_0-1} \left((\gamma_{z,j}^{(i_0)} - \gamma_{z,j}^{(k+1)})\frac{(1 + x_0c_0^{(j)})}{s_j} + (\gamma_{s,j}^{(i_0)} - \gamma_{s,j}^{(k+1)})s_j + (\gamma_{t,j}^{(i_0)} - \gamma_{t,j}^{(k+1)})\frac{x_0}{s_j}\right) \bmod (x_1) \ . \end{aligned} \tag{4.61}$$

Taking the above modulo $(x_0)$ we get

$$\zeta_{t,i_0}\phi_{s,i_0} + \gamma^{(i_0)} - \gamma^{(k+1)} + \sum_{j=1}^{i_0-1} ((\gamma_{z,j}^{(i_0)} - \gamma_{z,j}^{(k+1)})\frac{1}{s_j} + (\gamma_{s,j}^{(i_0)} - \gamma_{s,j}^{(k+1)})s_j) \bmod (x_0, x_1) = 0 \ .$$

By looking at the coefficients of the constant monomial and of $\frac{1}{s_i}$ and $s_i$ for all $i < i_0$, we deduce the following:

$$\zeta_{t,i_0}\phi_{s,i_0} + \gamma^{(i_0)} - \gamma^{(k+1)} = 0 \tag{4.62}$$

$$\forall i < i_0 : \ \gamma_{z,i}^{(i_0)} - \gamma_{z,i}^{(k+1)} = 0 \quad \text{and} \quad \gamma_{s,i}^{(i_0)} - \gamma_{s,i}^{(k+1)} = 0 \tag{4.63}$$

This lets us rewrite (4.61) as $\sum_{j=1}^{i_0-1} (\gamma_{t,j}^{(i_0)} - \gamma_{t,j}^{(k+1)})\frac{x_0}{s_j} \bmod (x_1) = 0$, and equating the coefficients of $\frac{x_0}{s_j}$ for all $j < i_0$ yields

$$\forall i < i_0 : \gamma_{t,i}^{(i_0)} = \gamma_{t,i}^{(k+1)} \ . \tag{4.64}$$

Applying (4.62), (4.63) and (4.64) to (4.59) yields

$$c_0^{(k+1)} = \zeta_{t,i_0}\phi_{s,i_0} + \gamma^{(i_0)} + \sum_{i=1}^{i_0-1} (\gamma_{z,i}^{(i_0)}z_i + \gamma_{s,i}^{(i_0)}s_i + \gamma_{t,i}^{(i_0)}t_i)) \ . \tag{4.65}$$

Recalling the definition of $c_0^{(i_0)}$ form (4.27), we can conclude that:

$$c_0^{(k+1)} = \zeta_{t,i_0}\phi_{s,i_0} + c_0^{(i_0)} \ . \tag{4.66}$$

Therefore (4.60) becomes $0 = -x_1(c_1^{(i_0)} + \zeta_{t,i_0}\phi_{s,i_0}d^{(i_0)} - c_1^{(k+1)})$, in other words

$$c_1^{(k+1)} = \zeta_{t,i_0}\phi_{s,i_0}d^{(i_0)} + c_1^{(i_0)} \ . \tag{4.67}$$

The last two equations mean that $(C_0^{(k+1)}, C_1^{(k+1)}) = (C_0^{(i_0)} + rG,\ C_1^{(i_0)} + rP^{(i_0)})$, for $r = \zeta_{t,i_0}\phi_{s,i_0}$, which together with (4.45) means that

$$(P^{(k+1)}, (C_0^{(k+1)}, C_1^{(k+1)})) \in \{P^{(i_0)}\} \times [(C_0^{(i_0)}, C_1^{(i_0)})]_{P^{(i_0)}} \subseteq Q \ .$$

We have thus shown that in the "ideal" model, the attacker cannot win the game. It remains to upper-bound the statistical distance from the adversary point of view between these two models.

**Difference between ideal and real game.** We start with upper-bounding the degree of the denominators and numerators of the rational fractions that can be generated by the adversary.

We first show that by induction on the number of queries $k$, that all the elements returned by the challenger in the ideal game are divisors of $\prod_{i=1}^{k} s_i$. In the base case, when no queries are made, no $s_i$ appears and the elements returned by the adversary are polynomials. For the induction step, assume the statement holds for $\ell$ queries. Consider the reply to the $(\ell+1)$-th query: $S_{\ell+1}$ and $\hat{S}_{\ell+1}$ are monomials; $Z_{\ell+1}$ and $T_{\ell+1}$ are sums of polynomials and elements output by the adversary divided by $s_{\ell+1}$. Using the induction hypothesis on the adversary's outputs, we deduce that the denominators divide $\prod_{i=1}^{\ell+1} s_i$. □

Similarly, we can show that the numerators of each element output by the challenger can be written as a sum of divisors of $x_0^{k+1}\, x_1^{k+1}\, \prod_{i=1}^{k} s_i$.

The "ideal" model and the generic group model differ if and only if two elements are distinct as rational fractions but identical as (handle of a) group element. That is, if we evaluate two different rational fractions at scalar values $x_0, x_1, s_1, \ldots, s_k$ and obtain the same result.

Any such equality of rational fractions generated during the game can be rewritten as a polynomial equation of degree $3k + k + 2$ ($3k + 2$ upper-bounding the degree of the numerator and $k$ that of the denominator). Because the values $x_0, x_1, s_1, \ldots, s_k$ are uniformly random (and hidden from the adversary), the Schwartz-Zippel lemma [Sch80] yields that the probability of this equality holding is at most $\frac{4k+2}{p-1}$.

If the adversary computes at most $q$ group operations, then there are at most $q + 3 + 3k$ group elements, where 3 comes from the generator and the verification key, and $3k$ corresponds to the answers to the signing queries (note that $\hat{S}$ and $S$ correspond to the same monomial). There are therefore

$$\tfrac{1}{2}(q + 3k + 3)(q + 3k + 2)$$

pairs of rational fractions. Using the union bound, we conclude that the adversary can distinguish the two models with probability at most $\frac{4k+2}{2(p-1)}(q + 3k + 3)(q + 3k + 2) < \frac{2k+1}{2^\lambda}(q + 3k + 3)^2$, since $p - 1 > 2^\lambda$, which is the bound claimed by the theorem. □

**Generalization of our scheme.** We conclude by mentioning that our scheme easily generalizes to ElGamal encryptions of vectors of group elements without increasing the size of signatures: for an encryption key $(P_1, \ldots, P_n)$ and a signing key $(x_0, \ldots, x_n)$, a ciphertext consisting of $C_0 = rG$ and $C_i = M_i + rP_i$ for $1 \le i \le n$, a signature on randomizable ciphertexts is defined as:

$$Z := \frac{1}{s}\Big(G + \sum_{i=0}^{n} x_i C_i\Big) \qquad S := sG \qquad \hat{S} := s\hat{G} \qquad T := \frac{1}{s}\Big(x_0 G + \sum_{i=1}^{n} x_i P_i\Big) \tag{4.68}$$

# Chapter 5

# Transferable E-cash: A Cleaner Model and the First Practical Instantiation

*This is a joint-work with Georg Fuchsbauer and Chen Qian.*

---

**Overview.** In this chapter we first revisit the model for transferable e-cash, proposing simpler yet stronger security definitions and then give the first concrete instantiation of the primitive, basing it on bilinear groups, and analyze its concrete efficiency.

## 5.1 Security Models

### Algorithms and protocols

An e-cash scheme is set up by running ParamGen and the bank generating its key pair via BKeyGen. The bank maintains a list of users $\mathcal{UL}$ and a list of deposited coins $\mathcal{DCL}$. Users run the protocol Register with the bank to obtain their secret key, while their public keys are added to $\mathcal{UL}$. With her secret key a user can run Withdraw with the bank to obtain coins, which she can transfer to others via the protocol Spend.

Spend is also used when a user deposits a coin at the bank. After receiving a coin, the bank runs CheckDS on the coin and the previously deposited coins in $\mathcal{DCL}$, which determines whether to accept the coin. If so, it is added to $\mathcal{DCL}$; if not (in case of double-spending), CheckDS returns the public key of the accused user and a proof $\Pi$, which can be verified using VfyGuilt.

$\mathsf{ParamGen}(1^\lambda)$, on input the security parameter $\lambda$ in unary, outputs public parameters $par$, which are an implicit input to all of the following algorithms and interactive protocols.

$\mathsf{BKeyGen}\,(par)$ is executed by the bank $\mathcal{B}$ and outputs its public key $pk_{\mathcal{B}}$ and its secret key $sk_{\mathcal{B}} = (sk_{\mathcal{W}}, sk_{\mathcal{D}}, sk_{\mathcal{CK}})$, where $sk_{\mathcal{W}}$ is used to issue coins in Withdraw and to register users in Register; $sk_{\mathcal{D}}$ is used as the secret key of the receiver when coins are deposited via Spend; and $sk_{\mathcal{CK}}$ is used for CheckDS.

$\mathsf{Register}\langle\mathcal{B}(sk_{\mathcal{W}}),\mathcal{U}(pk_{\mathcal{B}})\rangle$ is a protocol between the bank and a user. The user obtains a secret key $sk$ and the bank gets $pk$, which it adds to $\mathcal{UL}$. In case of error, they both obtain $\perp$.

$\mathsf{Withdraw}\langle\mathcal{B}(sk_{\mathcal{W}}),\mathcal{U}(sk_{\mathcal{U}},pk_{\mathcal{B}})\rangle$ is run between the bank and a user, who outputs a coin $c$ (or $\perp$), while the bank outputs $ok$ (in which case it debits the user's account) or $\perp$.

$\mathsf{Spend}\langle\mathcal{U}(c,sk,pk_{\mathcal{B}}),\mathcal{U}'(sk',pk_{\mathcal{B}})\rangle$ is run between two users and lets $\mathcal{U}$ spend a coin $c$ to $\mathcal{U}'$ (who could be the bank). $\mathcal{U}'$ outputs a coin $c'$ (or $\perp$), while $\mathcal{U}$ outputs $ok$ (or $\perp$).

$\mathsf{CheckDS}\,(sk_{\mathcal{CK}},\mathcal{UL},\mathcal{DCL},c)$, run by the bank, takes as input its checking key, the lists of registered users $\mathcal{UL}$ and of deposited coins $\mathcal{DCL}$ and a coin $c$. It outputs an updated list $\mathcal{DCL}$ (when the coin is accepted) or a user public key $pk_{\mathcal{U}}$ and an incrimination proof $\Pi$.

$\mathsf{VfyGuilt}\,(pk_{\mathcal{U}},\Pi)$ can be executed by anyone. It takes a user public key and an incrimination proof and returns 1 (acceptance of $\Pi$) or 0 (rejection).

Note that we define a transferable e-cash scheme as stateless: there is no common state information shared between the algorithms. This means that a coin withdrawn will be the same, whether it was the first or the $n$-th coin the bank issues to a specific user. Moreover, when a user $\mathcal{U}'$ receives a coin from a user $\mathcal{U}$, then the transferred coin will only depend on the original coin (not on other coins received by $\mathcal{U}'$ or coins transferred by $\mathcal{U}$). Thus, the bank and the users need not store anything about past transactions for transfer; the coin itself must be sufficient.

In particular, the bank can separate withdrawing from depositing, in that CheckDS, used during deposit, need not be aware of the withdrawn coins.

### Correctness properties

These properties were not stated in previous models. We believe they are important, as they preclude schemes that satisfy security notions by not doing anything.

Let $par$ be an output of $\mathsf{ParamGen}(1^\lambda)$ and $(sk_{\mathcal{B}} = (sk_{\mathcal{W}},sk_{\mathcal{D}},sk_{\mathcal{CK}}),pk_{\mathcal{B}})$ be output by $\mathsf{BKeyGen}(par)$ and let $sk$ and $sk'$ be two user outputs of Register. Then the following holds (while correctness of CheckDS and VfyGuilt is implied by the security definitions below):

- none of the outputs is $\perp$;
- any execution of $\mathsf{Register}\langle\mathcal{B}(sk_{\mathcal{W}}),\mathcal{U}(pk_{\mathcal{B}})\rangle$ yields output $pk$ for $\mathcal{B}$ and $sk$ for $\mathcal{U}$;
- any execution of $\mathsf{Withdraw}\langle\mathcal{B}(sk_{\mathcal{W}}),\mathcal{U}(sk,pk_{\mathcal{B}})\rangle$ yields $ok$ for $\mathcal{B}$ and $c$ for $\mathcal{U}$;
- in an execution of $\mathsf{Spend}\langle\mathcal{U}(c,sk,pk_{\mathcal{B}}),\mathcal{U}'(sk',pk_{\mathcal{B}})\rangle$, no party outputs $\perp$;
- and $sk_{\mathcal{D}}$ works as a user secret key $sk'$.

### Security definitions

#### Global variables

In our security games, we store all information about users and their keys in the user list $\mathcal{UL}$. Its entries are of the form $(pk_i,sk_i,uds_i)$, where $uds_i$ indicates how many times user $\mathcal{U}_i$ has double-spent.

In the coin list $\mathcal{CL}$, we keep information about the coins created in the system. For each withdrawn or spent coin $c$, we store a tuple $(owner,c,cds,origin)$, where $owner$ stores the index $i$ of the user who withdrew or received the coin (we do not store coins withdrawn or received by the adversary). We also include $cds$, which counts how often this *specific instance* of the coin has been spent. In $origin$ we write "$\mathcal{B}$" if the coin was issued by the honest bank and "$\mathcal{A}$" if it originates from the adversary; if the coin was originally spent by the challenger itself, we store a pointer indicating which original coin this transferred coin corresponds to. Finally, we maintain a list of deposited coins $\mathcal{DCL}$.

### Oracles

We now define oracles used in the security definitions, which differ depending on whether the adversary impersonates a corrupt bank or users. If during the oracle execution an algorithm fails (i.e., it outputs $\perp$) then the oracle also stops. Otherwise the call to the oracle is considered successful; a successful deposit oracle call must also not detect any double-spending.

**Registration and corruption of users.** The adversary can instruct the creation of honest users and either play the role of the bank during registration, or passively observe registration. It can moreover "spy" on users, meaning it can learn the user's secret key. This will strengthen yet simplify our anonymity games compared to [BCFK15], where once the adversary had learned the secret key of a user (by "corrupting" her), the user could not be a challenge user in the anonymity games anymore (*selfless anonymity*, while we achieve *full* anonymity).

`BRegist`() plays the bank side of Register and interacts with $\mathcal{A}$. If successful, it adds $(pk, \perp, uds = 0)$ to $\mathcal{UL}$ (where $uds$ is the number of double-spends).

`URegist`() plays the user side of the Register protocol when the bank is controlled by the adversary. Upon successful execution, it adds $(pk, sk, 0)$ to $\mathcal{UL}$.

`Regist`() plays both parties in the Register protocol and adds $(pk, sk, 0)$ to $\mathcal{UL}$.

`Spy`$(i)$, for $i \leq |\mathcal{UL}|$, returns user $i$'s secret key $sk_i$.

**Withdrawal oracles.** The adversary can either withdraw a coin from the bank, play the role of the bank, or passively observe a withdrawal.

`BWith`() plays the bank side of the Withdraw protocol. Coins withdrawn by $\mathcal{A}$ (and thus unknown to the experiment) are not added to the coin list $\mathcal{CL}$.

`UWith`$(i)$ plays user $i$ in Withdraw when the bank is controlled by the adversary. Upon obtaining a coin $c$, it adds $(owner = i, c, cds = 0, origin = \mathcal{A})$ to $\mathcal{CL}$.

`With`$(i)$ simulates a Withdraw protocol execution playing both $\mathcal{B}$ and user $i$. It adds $(owner = i, c, cds = 0, origin = \mathcal{B})$ to $\mathcal{CL}$.

### Spend and deposit oracles.

`Rcv`$(i)$ makes user $i$ receive a coin from $\mathcal{A}$. The oracle plays the role of $\mathcal{U}'$ with user $i$'s secret key in the Spend protocol. It adds a new entry $(owner = i, c, cds = 0, origin = \mathcal{A})$ to $\mathcal{CL}$.

`Spd`$(j)$ spends the coin from the $j$-th entry $(owner_j, c_j, cds_j, origin_j)$ in $\mathcal{CL}$ to $\mathcal{A}$, who could be impersonating a user, or the bank during a deposit. The oracle plays user $\mathcal{U}$ in the Spend protocol with secret key $sk_{owner_j}$. It increments the coin spend counter $cds_j$ by 1. If afterwards $cds_j > 1$, then the owner's double-spending counter $uds_{owner_j}$ is incremented by 1.

`S&R`$(j, i)$ spends the $j$-th coin in $\mathcal{CL}$ to user $i$. It runs $(ok, c) \leftarrow \mathsf{Spend}\langle \mathcal{U}(c_j, sk_{owner_j}, pk_{\mathcal{B}}), \mathcal{U}'(sk_i, pk_{\mathcal{B}}) \rangle$ and adds $(owner = i, c, cds = 0, pointer = j)$ to $\mathcal{CL}$. It increments the coin spend counter $cds_j$ by 1. If afterwards $cds_j > 1$, then $uds_{owner_j}$ is incremented by 1.

`BDepo`() lets $\mathcal{A}$ deposit a coin. It runs $\mathcal{U}'$ in Spend using the bank's secret key $sk_{\mathcal{D}}$ with the adversary playing $\mathcal{U}$. If successful, it runs CheckDS on the received coin and updates $\mathcal{DCL}$ accordingly; else it outputs a pair $(pk, \Pi)$.

$\mathbf{Expt}^{\mathtt{sound}}_{\mathcal{A}}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$; $pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$(b, i_1, i_2) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
If $b = 0$ then run $\mathtt{UWith}(i_1)$ with $\mathcal{A}$
Else run $\mathtt{Rcv}(i_1)$ with $\mathcal{A}$
If this outputs $\perp$ then return 0
Run $\mathtt{S\&R}(1, i_2)$; if one party outputs $\perp$ then return 1
Return 0

Figure 5.1: Game for *soundness* (protecting users from financial loss)

$\mathtt{Depo}(j)$, the honest deposit oracle, runs $\mathsf{Spend}$ between the owner of the $j$-th coin in $\mathcal{CL}$and an honest bank. If successful, it increments $cds_j$ by 1; if afterwards $cds_j > 1$, it also increments $uds_{owner_j}$. It runs $\mathsf{CheckDS}$ on the received coin and either updates $\mathcal{DCL}$ or returns a pair $(pk, \Pi)$.

(Note that no oracle "$\mathtt{UDepo}$" is required, since $\mathtt{Spd}$ lets the adversarial bank have an honest user deposit a coin.)

### Economic properties

We distinguish two types of security properties of transferable e-cash schemes. Besides anonymity notions, economic properties ensure that neither the bank nor users will incur an economic loss when participating in the system.

**Soundness.** If an honest user accepted a coin during a withdrawal or a transfer, then she is guaranteed that the coin will be accepted by others, either honest customers when transferring, or the bank when depositing. The game is formalized in Fig. 5.1 where $i_2$ plays the role of the receiver of a spending or the bank. For convenience, we define probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ to be stateful in all our security games.

**Definition 5.1** (Soundness)**.** *A transferable e-cash system is* sound *if for any PPT $\mathcal{A}$, we have* $\mathbf{Adv}^{\mathtt{sound}}_{\mathcal{A}}(\lambda) := \Pr[\mathbf{Expt}^{\mathtt{sound}}_{\mathcal{A}}(\lambda) = 1]$ *is negligible in $\lambda$.*

#### Unforgeability

This notion can be seen as a merge of *unforgeability* and *user identification* from [BCFK15] (which were not consistent as we will explain in Sect. 5.2). The notion protects the bank, ensuring that no (coalition of) users can spend more coins than the number of coins they withdrew.

It also guarantees that whenever a coin is deposited and refused by $\mathsf{CheckDS}$, it also returns the identity of a registered user, who is accused of double-spending. (*Exculpability*, below, ensures that no innocent user will be accused.) The game is formalized in Fig. 5.2 and lets the adversary impersonate all users.

**Definition 5.2** (Unforgeability)**.** *A transferable e-cash system is* unforgeable *if* $\mathbf{Adv}^{\mathtt{unforg}}_{\mathcal{A}}(\lambda) := \Pr[\mathbf{Expt}^{\mathtt{unforg}}_{\mathcal{A}}(\lambda) = 1]$ *is negligible in $\lambda$ for any PPT $\mathcal{A}$.*

**Exculpability.** This notion, a.k.a. *non-frameability*, ensures that the bank, even when colluding with malicious users, cannot wrongly accuse an honest user of double-spending. Specifically, it guarantees that an adversarial bank cannot produce a double-spending proof $\Pi^*$ that verifies for the public key of an honest user $i^*$ that has never double-spent. The game is formalized as in Fig. 5.3.

$\mathbf{Expt}^{\mathtt{unforg}}_{\mathcal{A}}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$; $(sk_{\mathcal{B}}, pk_{\mathcal{B}}) \leftarrow \mathsf{BKeyGen}(par)$
$\mathcal{A}^{\mathtt{BRegist},\mathtt{BWith},\mathtt{BDepo}}(par, pk_{\mathcal{B}})$
If in a BDepo call, CheckDS does not return a coin list:
Return 1 if any of the following hold:
– CheckDS outputs $\perp$
– CheckDS outputs $(pk, \Pi)$ and $\mathsf{VfyGuilt}(pk, \Pi) = 0$
– CheckDS outputs $(pk, \Pi)$ and $pk \notin \mathcal{UL}$
Let $q_W$ be the number of calls to BWith
If $q_W < |\mathcal{DCL}|$, then return 1
Return 0

Figure 5.2: Game for *unforgeability* (protecting the bank from financial loss)

$\mathbf{Expt}^{\mathtt{excul}}_{\mathcal{A}}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$; $pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$(i^*, \Pi^*) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy},\mathtt{UWith},\mathtt{Rcv},\mathtt{Spd},\mathtt{S\&R},\mathtt{UDepo}}(par)$
Return 1 if **all** of the following hold:
– $\mathsf{VfyGuilt}(pk_{i^*}, \Pi^*) = 1$
– There was no call $\mathtt{Spy}(i^*)$
– $uds_{i^*} = 0$
Return 0

Figure 5.3: Game for *exculpability* (protecting honest users from accusation)

**Definition 5.3** (Exculpability)**.** *A transferable e-cash system is* exculpable *if* $\mathbf{Adv}^{\mathtt{excul}}_{\mathcal{A}}(\lambda) := \Pr[\mathbf{Expt}^{\mathtt{excul}}_{\mathcal{A}}(\lambda) = 1]$ *is negligible in* $\lambda$ *for any PPT* $\mathcal{A}$*.*

### Anonymity properties

We will not follow previous anonymity notions [BCF+11, BCFK15], but introduce new ones which precisely distinguish between the adversary's capabilities, in particular, whether it is able to detect double-spending. When the adversary impersonates the bank, we consider two cases: user anonymity and coin anonymity (and explain why this distinction is necessary).

As transferred coins necessarily grow in size [CP93], we can only guarantee indistinguishability of *compatible* coins. We therefore define $\mathtt{comp}(c_1, c_2) = 1$ iff $\mathtt{size}(c_1) = \mathtt{size}(c_2)$, where the $\mathtt{size}(c) = 1$ after $c$ was withdrawn and it increases by 1 after each transfer.

**Coin anonymity.** This notion is closest to (and implies) the anonymity notion of classical e-cash: an adversary, who also impersonates the bank, issues two coins to the challenger and when she later receives them (via a deposit in classical e-cash), she should not be able to associate them to their issuances. In transferable e-cash, we allow the adversary to determine two series of honest users via which the coins are respectively transfered before being given back to the adversary.

The experiment is specified on the left of Fig. 5.4: users $i_0^{(0)}$ and $i_0^{(1)}$ withdraw a coin from the adversarial bank, user $i_0^{(0)}$ passes it to $i_1^{(0)}$, who passes it to $i_2^{(0)}$, etc., In the end, the last users of the two chains spend the coins to the adversary, but the order in which this happens depends on a random bit $b$, which the adversary must decide.

**User anonymity.** Coin anonymity required that users who transfer the coin are honest. If one of the users through which the coin passes colluded with the bank, there would be a trivial attack: after receiving the two challenge coins, the bank simulates the deposit of one of them and the deposit of the coin intercepted by the colluding user. If a double-spending is detected, it knows that the received coin corresponds to the sequence of users which the colluder was part of.

$\mathbf{Expt}^{\texttt{c-an}}_{\mathcal{A},b}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$i_0^{(0)} \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$; run $\texttt{UWith}(i_0^{(0)})$ with $\mathcal{A}$
$i_0^{(1)} \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$; run $\texttt{UWith}(i_0^{(1)})$ with $\mathcal{A}$
$\big((i_1^{(0)},\ldots,i_{k_0}^{(0)}),(i_1^{(1)},\ldots,i_{k_1}^{(1)})\big) \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
If $k_0 \neq k_1$ then abort
For $j = 1,\ldots,k_0$:
  Run $\texttt{S\&R}\big(2j-1, i_j^{(0)}\big)$
  Run $\texttt{S\&R}\big(2j, i_j^{(1)}\big)$
Run $\texttt{Spd}(2k_0+1+b)$ with $\mathcal{A}$
Run $\texttt{Spd}(2k_0+2-b)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

$\mathbf{Expt}^{\texttt{u-an}}_{\mathcal{A},b}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$(i_0^{(0)}, i_0^{(1)}) \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
Run $\texttt{Rcv}(i_b)$ with $\mathcal{A}$
$\big((i_1^{(0)},\ldots,i_{k_0}^{(0)}),(i_1^{(1)},\ldots,i_{k_1}^{(1)})\big) \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
If $k_0 \neq k_1$ then abort
For $j = 1,\ldots,k_0$:
  Run $\texttt{S\&R}\big(j, i_j^{(b)}\big)$
Run $\texttt{Spd}(k_0+1)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

Figure 5.4: Games for *coin* and *user anonymity* (protecting users from a malicious bank)

Since double-spending detection is an essential feature of e-cash, attacks of this kind are impossible to prevent. However, we still want to guarantee that, while the bank can trace coins, the involved *users* remain anonymous. We formalize this in the game on the right of Fig. 5.4, where, in contrast to coin anonymity, there is only one coin and the adversary must distinguish the sequence of users through which the coin passes before returning to her. In contrast to coin anonymity, we now allow the coin to already have some "history", rather than being freshly withdrawn.

**Coin transparency.** This notion is in some sense the strongest anonymity notion and it implies that a user that transfers a coin cannot recognize it if at some point she receives it again. Note that this notion (as well as *coin anonymity*) is not even achieved by physical cash, as banknotes can be marked by users (or the bank). As the bank can necessarily trace coins (for double-spending detection), it is assumed to be honest for this notion. Actually, only the detection key $sk_{\mathcal{CK}}$ must remain hidden from the adversary, while $sk_{\mathcal{W}}$ and $sk_{\mathcal{D}}$ can be given.

The game formalizing this notion, specified in Fig. 5.5, is analogous to coin anonymity, except that the challenge coins are not freshly withdrawn; instead, the adversary spends two coins of its choice to users of its choice, both are passed through a sequence of users of the adversary's choice and one of them is returned to the adversary.

There is another trivial attack that we need to exclude: the adversary could deposit the coin that is returned to him and one, say the first, of the coins he initially transfered to an honest user. Now if the deposit does not succeed because of double-spending, the adversary knows that it was the first coin that was returned to him. Again, this attack is unavoidable due to the necessity of double-spending detection. It is a design choice that lies outside of our model to implement sufficient deterrence from double-spending, so it would exceed the utility of breaking anonymity.

This is the reason why the game aborts if the adversary deposits twice a coin from the set of "challenge coins" (consisting of the two coins the adversary transfers and the one it receives). The variable *ctr* counts how many times a coin from this set was deposited. Note also that because $\mathcal{A}$ has $sk_{\mathcal{W}}$, and can therefore create unregistered users, we do not consider $\mathcal{UL}$ in this game.

**Definition 5.4** (Anonymity)**.** *For* $\texttt{x} \in \{\texttt{c-an}, \texttt{u-an}, \texttt{c-tr}\}$ *a transferable e-cash scheme satisfies* x *if* $\mathbf{Adv}^{\texttt{x}}_{\mathcal{A}}(\lambda) := \Pr[\mathbf{Expt}^{\texttt{x}}_{\mathcal{A},1}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\texttt{x}}_{\mathcal{A},0}(\lambda) = 1]$ *is negligible in* $\lambda$ *for any PPT adversary* $\mathcal{A}$*.*

$\mathbf{Expt}^{\texttt{c-tr}}_{\mathcal{A},b}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$; $((sk_{\mathcal{W}}, sk_{\mathcal{D}}, sk_{\mathcal{CK}}), pk_{\mathcal{B}}) \leftarrow \mathsf{BKeyGen}(par)$
$\mathcal{DCL}' \leftarrow \emptyset$ // *lists the challenge coins*
$ctr \leftarrow 0$ // *counts how often a challenge coin was deposited*
$i^{(0)} \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{BDepo}',\texttt{Spy}}(par, pk_{\mathcal{B}}, sk_{\mathcal{W}}, sk_{\mathcal{D}})$
// $\texttt{BDepo}'$ *uses* $\mathsf{CheckDS}'(\cdot,\cdot,\cdot,\cdot,\mathcal{DCL}')$ *(see below) instead of* $\mathsf{CheckDS}$
Run $\texttt{Rcv}(i^{(0)})$ with $\mathcal{A}$; let $c_0$ be the received coin stored in $\mathcal{CL}[1]$
$x_0 \leftarrow \mathsf{CheckDS}(sk_{\mathcal{CK}}, \emptyset, \mathcal{CL}, c_0)$
If $x_0 = \perp$ then $ctr \leftarrow ctr + 1$ //$c_0$ *had been deposited*
$\mathcal{DCL}' \leftarrow \mathsf{CheckDS}(sk_{\mathcal{CK}}, \emptyset, \emptyset, c_0)$ //*add* $c_0$ *to list of challenge coins*
$i^{(1)} \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{BDepo},\texttt{Spy}}$
Run $\texttt{Rcv}(i^{(1)})$ with $\mathcal{A}$; let $c_1$ be the received coin stored in $\mathcal{CL}[2]$
$x_1 \leftarrow \mathsf{CheckDS}(sk_{\mathcal{CK}}, \emptyset, \mathcal{CL}, c_1)$
If $x_1 = \perp$ then $ctr \leftarrow ctr + 1$ //$c_1$ *had been deposited*
If $\texttt{comp}(c_0, c_1) \neq 1$ then abort
$x_2 \leftarrow \mathsf{CheckDS}(sk_{\mathcal{CK}}, \emptyset, \mathcal{DCL}', c_1)$ //*add* $c_1$ *to list of challenge coins*
If $x_2 \neq \perp$ then $\mathcal{DCL}' \leftarrow x_2$ // ($c_1$ *could be a double-spending of* $c_0$)
$\left((i_1^{(0)}, \ldots, i_{k_0}^{(0)}), (i_1^{(1)}, \ldots, i_{k_1}^{(1)})\right) \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{BDepo}',\texttt{Spy}}$
If $k_0 \neq k_1$ then abort
If $(k_b \neq 0)$ then run $\texttt{S\&R}\left(b+1, i_1^{(b)}\right)$ // *spend coin* $c_b$ *to user* $i_1^{(b)}$ ...
For $j = 2, \ldots, k_0$: // ... *the received coin is placed in* $\mathcal{CL}[3]$
Run $\texttt{S\&R}\left(j+1, i_j^{(b)}\right)$ // *spend coins consecutively*
Run $\texttt{Spd}(k_0 + 2)$ with $\mathcal{A}$ // *and transfer it back to* $\mathcal{A}$
$b^* \leftarrow \mathcal{A}^{\texttt{BDepo}'}$
Return $b^*$

$\mathsf{CheckDS}'\left(sk_{\mathcal{CK}}, \mathcal{UL}, \mathcal{DCL}, c, \mathcal{DCL}'\right)$: // used by $\texttt{BDepo}'$
$x \leftarrow \mathsf{CheckDS}\left(sk_{\mathcal{CK}}, \emptyset, \mathcal{DCL}', c\right)$
If $x = \perp$: // *the deposited coin* $c$ *is a double-spending of* $c_0$ *or* $c_1$
$ctr \leftarrow ctr + 1$
If $ctr > 1$ then abort
Output $\mathsf{CheckDS}\left(sk_{\mathcal{CK}}, \emptyset, \mathcal{DCL}, c\right)$

Figure 5.5: Game for *coin transparency* (protecting users from malicious users)

## 5.2 Comparison with previous work

### Model comparison

In order to justify our new model, we start with discussing a security vulnerability of the previous model [BCFK15].

**Issues with economical notions.** As already pointed out in Sect. 5.1, the *correctness properties* were missing in previous models.

**No soundness guarantees.** In none of the previous models was there a security notion that guaranteed that an honest customer could successfully transfer a coin to another honest user or the bank, even if the coin was obtained regularly.

**Fuzzy definition of "unsuccessful deposit".** Previous models defined a protocol called "Deposit", which we separated it into an interactive (Spend) and a static part (CheckDS). In their definition of unforgeability, the authors [BCFK15] use the concept of "successful deposit", which was not clearly defined, as an "unsuccessful deposit" could mean one of the following:

- The bank detects a double-spending and provides a proof accusing the cheater (who could be different from the depositer).
- The customer did not follow the protocol (e.g., by sending a malformed coin), in which case we cannot expect a proof of guilt from the bank.
- The customer followed the protocol but using a coin that was double-spent (either earlier or during deposit); however, the bank does not obtain a valid proof of guilt and outputs $\perp$.

After a careful reading of the definitions in [BCFK15] we concluded that the authors do not distinguish the second and the third case. This is a serious issue, since the second case cannot be avoided, while the third case *should* be avoided so the bank does not lose money without being able to accuse the cheater. This is now guaranteed by our unforgeability notion in Def. 5.2.

### An error in a proof in BCFK15

The authors of [BCFK15] claim that their scheme satisfies the notion `StR-fa` defined in [BCF+11] (after having discovered an error in the `StR-fa` proof of the scheme of that paper). To achieve this anonymity property (the hardest property to achieve, as they notice), they use malleable signatures, a primitive with very strong security guarantees. The ensure that whenever the adversary, after obtaining simulated signatures, outputs a valid message/signature pair $(m, \sigma)$, it must have derived the pair from received signatures. Formally, there exists an extractor that can extract a transformation from $\sigma$ that links $m$ to the messages for which the adversary queried signatures.

However, in the definition of `StR-fa` [BCF+11] the adversary receives $sk_{\mathcal{W}}$ (as in our `c-tr` notion), which in their instantiation [BCFK15] contains the signing key for the malleable signature scheme. Using this, the adversary can thus easily compute a fresh signature from which no extractor can recover a transformation explaining the signed message.

For this reason we claim that our scheme is the first to satisfy the "spirit" of the `StR-fa` notion. We believe that the error in the proof was a consequence of the complexity of `StR-fa` and that our notion `c-tr` is simpler and more intuitive.

## 5.3 Instantiation

### Overview

The bank creates money and the validates new users in the system. Digital signatures can be used for these two functions: the bank signs the key of a new user, who can then prove that he is registered; and during a coin issuing, the bank signs a message $M_{sn}$ that is associated to the initial serial-number (SN) component $sn_0$ of a coin, which makes coins unforgeable.

After a coin has been transferred $k$ times, its core consists of an SN list $sn_0, sn_1, \ldots, sn_k$, together with a list of tags $tag_1, \ldots, tag_k$ (for a freshly withdrawn coin, we have $k = 0$). When a user spends such a coin, the receiver generates a fresh SN component $sn_{k+1}$, from which the spender must generate a tag $tag_{k+1}$ that is also associated with her public key and the last serial number $sn_k$ (which she generated when she received the coin.)

These tags allow the bank to identify the fraudster in case of double-spending, while they preserve the anonymity of honest users. A coin moreover contains the users' public key w.r.t. which the tags were created, as well as certificates on them issued by the bank. To provide anonymity, all these components are not given in the clear, but as a zero-knowledge proof of knowledge. That is, a coin is a proof of knowledge of its serial number, its tags, the user public keys and their certificates and ensures that all of them are consistent.

As we use a commit-and-prove proof system, all these values are included in the coin as commitments. Recall that a coin also includes a signature by the bank on (a message related to)

the initial SN component. In order to achieve anonymity towards the bank (*coin anonymity*), the bank must sign this message blindly, which is achieved by using the $\mathsf{SigCm}$ functionality: the user sends a commitment to the serial number, and the bank computes a committed signature on the committed value.

Finally, the bank needs to be able to *detect* whether a double-spending occurred and *identify* the user that committed it. One way would be to give the serial numbers and the tags (which protect the anonymity of honest users) in the clear. This would yield a scheme that satisfies *coin anonymity* and *user anonymity* (note that in these two notions the bank is corrupted). However, *coin transparency*, the most intricate anonymity notion, would not be achieved, since the owner of a coin could easily recognize it when she receives it again by looking at its serial number.

It is *coin transparency* that requires to hide the serial numbers (and the associated tags), and moreover to use a randomizable proof system, since the appearance of a coin needs to completely change after every transfer. To hide the serial numbers and the tags from users, but still provide access to them by the bank, we add encryptions of them to the coin. However, these must smoothly interoperate with the randomization of the coin, which is why we require rerandomizable encryption that can be tied into the machinery of updating of the proofs, which is necessary every time the ciphertexts and the commitments contained in a coin are refreshed.

## Technical description

### Primitives used.

The basis of our scheme is a randomizable extractable NIZK commit-and-prove scheme

$$\mathsf{C} = (\mathsf{C.Setup}, \mathsf{C.Cm}, \mathsf{C.RdCm}, \mathsf{C.ExSetup}, \mathsf{C.Extr}, \mathsf{C.Prv}, \mathsf{C.Verify}, \mathsf{C.AdptPrf}),$$

to which we add compatible schemes: an $\mathcal{M}$-structure-preserving signature scheme $\mathsf{S} = (\mathsf{S.Setup}, \mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$, admitting an $\mathcal{M}$-commuting signature add-on $\mathsf{SigCm}$, as well as an $\mathcal{M}'$-structure-preserving signature scheme $\mathsf{S}'$; furthermore a double-spending tag scheme

$$\mathsf{T} = (\mathsf{T.Setup}, \mathsf{T.KeyGen}, \mathsf{T.SGen}, \mathsf{T.SGen}_{\text{init}}, \mathsf{T.SVfy}, \mathsf{T.SVfy}_{\text{init}}, \mathsf{T.TGen}, \mathsf{T.TVfy}, \mathsf{T.Detect}, \mathsf{T.VfyGuilt}),$$

as well as two randomizable encryption schemes $\mathsf{E} = (\mathsf{E.KeyGen}, \mathsf{E.Enc}, \mathsf{E.ReRand}, \mathsf{E.Dec}, \mathsf{E.Verify}, \mathsf{E.AdptPrf})$, and $\mathsf{E}'$. (We require $\mathsf{E}'$ to satisfy RCCA security, whereas $\mathsf{E}$ need only be IACR-secure).

### Auxiliary functions.

In order to simplify the description of our scheme, we will first define several auxiliary functions. To randomize a given tuple of commitments and ciphertext, and proofs for them (and adapting the proofs to the randomizations), we use an algorithm $\mathsf{Rand}$, which internally runs $\mathsf{C.RdCm}$, $\mathsf{E.ReRand}$, $\mathsf{C.AdptPrf}$ and $\mathsf{E.AdptPrf}$ with the same randomness.

Assuming the equations for $\mathsf{T.SVfy}(\cdot,\cdot,\cdot) = 1$, $\mathsf{T.SVfy}_{\text{init}}(\cdot,\cdot,\cdot) = 1$ and $\mathsf{T.TVfy}(\cdot,\cdot,\cdot,\cdot,\cdot) = 1$, as well as $\mathsf{E.Verify}(pk,\cdot,\cdot,c) = 1$ are all in the set $\mathcal{E}$ of equations supported by the proof system, we define the following:

$\mathsf{C.Prv}_{\text{sn,init}}$ proves that a committed initial serial number $sn$ has been honestly generated w.r.t. a committed key $pk_{\mathsf{T}}$ and a committed message $M$ (given the used randomness $\rho_{pk}$, $\rho_{sn}$ and $\rho_M$), while $\mathsf{C.Verify}_{\text{sn,init}}$ verifies such proofs. $\mathsf{C.Prv}_{\text{sn}}$ and $\mathsf{C.Verify}_{\text{sn}}$ do the same for non-initial serial numbers (for which there are no messages, but which require a proof of well-formedness instead).

$\mathsf{C.Prv}_{\text{sn,init}}(ck, pk_{\mathsf{T}}, sn, M, \rho_{pk}, \rho_{sn}, \rho_M)$:

– Return $\pi \leftarrow \mathsf{C.Prv}(ck, \mathsf{T.SVfy}_{\text{init}}(\cdot,\cdot,\cdot) = 1, (pk_{\mathsf{T}}, \rho_{pk}), (sn, \rho_{sn}), (M, \rho_M))$

$\mathsf{C.Verify}_{\text{sn,init}}(ck, c_{pk}, c_{sn}, c_M, \pi_{sn})$:

– Return $(\mathsf{C.Verify}(ck, \mathsf{T.SVfy}_{\text{init}}(\cdot,\cdot,\cdot) = 1, c_{pk}, c_{sn}, c_M, \pi_{sn}))$

$\mathsf{C.Prv}_{\text{sn}}(ck, pk_{\mathsf{T}}, sn, sn\text{-}pf, \rho_{pk}, \rho_{sn}, \rho_{sn\text{-}pf})$:

– $\pi \leftarrow \mathsf{C.Prv}(ck, \mathsf{T.SVfy}(\cdot,\cdot,\cdot) = 1, (pk_{\mathsf{T}}, \rho_{pk}), (sn, \rho_{sn}), (sn\text{-}pf, \rho_{sn\text{-}pf}))$

– Return $(\pi, \mathsf{C.Cm}(ck, sn\text{-}pf, \rho_{sn\text{-}pf}))$

$\mathsf{C.Verify}_{\text{sn}}(ck, c_{pk}, c_{sn}, \tilde{\pi}_{sn} = (\pi_{sn}, c_{sn\text{-}pf}))$:

– Return $\mathsf{C.Verify}(ck, \mathsf{T.SVfy}(\cdot,\cdot,\cdot) = 1, c_{pk}, c_{sn}, c_{sn\text{-}pf}, \pi_{sn})$

$\mathsf{C.Prv}_{\text{tag}}$ produces a proof that a committed $tag$ has been correctly generated w.r.t. committed serial numbers $sn$ and $sn'$; and $\mathsf{C.Verify}_{\text{tag}}$ verifies such proofs.

$\mathsf{C.Prv}_{\text{tag}}(ck, pk_{\mathsf{T}}, sn, sn', tag, \rho_{pk}, \rho_{sn}, \rho'_{sn}, \rho_{tag}, t\text{-}pf, \rho_{t\text{-}pf})$

– $\pi \leftarrow \mathsf{C.Prv}(ck, \mathsf{T.TVfy}(\cdot,\cdot,\cdot,\cdot,\cdot) = 1, (pk_{\mathsf{T}}, \rho_{pk}), (sn, \rho_{sn}), (sn', \rho'_{sn}), (tag, \rho_{tag}), (t\text{-}pf, \rho_{t\text{-}pf}))$

– Return $(\pi, \mathsf{C.Cm}(ck, t\text{-}pf, \rho_{t\text{-}pf}))$

$\mathsf{C.Verify}_{\text{tag}}(ck, c_{pk}, c_{sn}, c'_{sn}, c_{tag}, \pi_{tag} = (\pi, c_{t\text{-}pf}))$:

– Return $\mathsf{C.Verify}(ck, \mathsf{T.TVfy}(\cdot,\cdot,\cdot,\cdot) = 1, c_{pk}, c_{sn}, c'_{sn}, c_{tag}, c_{t\text{-}pf}, \pi)$

$\mathsf{C.E.Prv}_{\text{enc}}$ proves that a ciphertext $\tilde{c}$ of $M$ and $\mathsf{C.Cm}(ck, M, \rho_M)$ contain the same message; $\mathsf{C.E.Verify}_{\text{enc}}$ verifies such proofs. (Note that the output of $\mathsf{C.E.Prv}_{\text{enc}}$ is the same $\pi$ as in the input of $\mathsf{E.AdptPrf}$.)

$\mathsf{C.E.Prv}_{\text{enc}}(ck, ek, M, \rho_M, \nu_M, \tilde{c})$:

– $\rho_\nu \xleftarrow{\$} \mathcal{R}$; $\pi \leftarrow \mathsf{C.Prv}(ck, \mathsf{E.Verify}\ (ek,\cdot,\cdot,\tilde{c}) = 1, (M, \rho_M), (\nu_M, \rho_\nu))$

– Return $(\pi, \mathsf{C.Cm}(ck, \nu_M, \rho_\nu))$

$\mathsf{C.E.Verify}_{\text{enc}}(ck, ek, c_M, \tilde{c}_M, \tilde{\pi}_{\text{eq}} = (\pi_{\text{eq}}, c_\nu))$:

– Return $\mathsf{C.Verify}(ck, \mathsf{E.Verify}(ek,\cdot,\cdot,\tilde{c}_M) = 1, c_M, c_\nu, \pi_{\text{eq}})$

**Components of the coin.**

There are two types of components, the *initial* components $\mathcal{C}_{\text{init}}$, and the *standard* components $\mathcal{C}_{\text{std}}$. The first is of the form

$$coin_{\text{init}} = (c^0_{pk}, c^0_{cert}, \pi^0_{cert}, c^0_{sn}, \pi^0_{sn}, \epsilon, \epsilon, c_M, c^0_\sigma, \pi^0_\sigma, \tilde{c}^0_{sn}, \tilde{\pi}^0_{sn}, \epsilon, \epsilon), \tag{5.1}$$

consisting of commitments (the $c$-values) to the withdrawer's key $pk$, her certificate $cert$, the initial serial number $sn$ and the related message $M$, the bank's signature $\sigma$ on $M$, and an encryption $\tilde{c}_{sn}$ of $sn$. It also contains proofs $\pi_{cert}$ and $\pi_{sn}$ of validity of $cert$ and $sn$ and a proof $\tilde{\pi}_{sn}$ that $c_{sn}$ and $\tilde{c}_{sn}$ contain the same value. We use $\epsilon$ to pad so that both types of component have the same format. Validity of an initial component is verified w.r.t. an encryption key (that can be for either $\mathsf{E}$ or $\mathsf{E}'$) and two signature verification keys for $\mathsf{S}$ and $\mathsf{S}'$:

$\mathsf{VER}_{\text{init}}(ek, vk, vk', coin_{\text{init}})$: Return 1 iff the following hold: // $coin_{\text{init}}$ *as in* (5.1)

- $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M, c^0_\sigma, \pi^0_\sigma)$
- $\mathsf{C.Verify}(ck, \mathsf{S}'.\mathsf{Verify}(vk', \cdot, \cdot) = 1, c^0_{pk}, c^0_{cert}, \pi^0_{cert})$
- $\mathsf{C.Verify}_{\text{sn,init}}(ck, c^0_{pk}, c^0_{sn}, c_M, \pi^0_{sn}) \wedge \mathsf{C.E}'.\mathsf{Verify}_{\text{enc}}(ck, ek, c^0_{sn}, \tilde{c}^0_{sn}, \tilde{\pi}^0_{sn})$

Standard components of a coin are of the form

$$coin_{\text{std}} = (c^i_{pk}, c^i_{cert}, \pi^i_{cert}, c^i_{sn}, \pi^i_{sn}, c^i_{tag}, \pi^i_{tag}, \epsilon, \epsilon, \epsilon, \tilde{c}^i_{sn}, \tilde{\pi}^i_{sn}, \tilde{c}^i_{tag}, \tilde{\pi}^i_{tag}), \tag{5.2}$$

and instead of $M$ and the bank's signature they contain a commitment $c_{tag}$ and an encryption $\tilde{c}_{tag}$ of the tag produced by the spender (and a proof $\pi_{tag}$ of validity and $\tilde{\pi}_{tag}$ proving that the values in $c_{tag}$ and $\tilde{c}_{tag}$ are equal). A coin is verified by checking the validity and consistency of each two consecutive components. If the first is an initial component then the values $\underline{c^{i-1}_{tag}}, \underline{\pi^{i-1}_{tag}}, \underline{\tilde{c}^{i-1}_{tag}}$ and $\underline{\tilde{\pi}^{i-1}_{tag}}$ are $\epsilon$; if it is a standard component then $c_M, c^{i-1}_\sigma$ and $\pi^{i-1}_\sigma$ are $\epsilon$.

$\mathsf{VER}_{\text{body}}(ek, vk_{\mathcal{B}}, (c^{i-1}_{pk}, c^{i-1}_{cert}, \pi^{i-1}_{cert}, c^{i-1}_{sn}, \pi^{i-1}_{sn}, \underline{c^{i-1}_{tag}}, \underline{\pi^{i-1}_{tag}}, c_M, c^{i-1}_\sigma, \pi^{i-1}_\sigma, \tilde{c}^{i-1}_{sn}, \tilde{\pi}^{i-1}_{sn}, \underline{\tilde{c}^{i-1}_{tag}}, \underline{\tilde{\pi}^{i-1}_{tag}}), coin_{\text{std}})$: // $coin_{\text{std}}$ *as in* (5.2)

Return 1 iff the following hold:

- $\mathsf{C.Verify}(ck, \mathsf{S}'.\mathsf{Verify}(vk_{\mathcal{B}}, \cdot, \cdot) = 1, c^i_{pk}, c^i_{cert}, \pi^i_{cert})$
- $\mathsf{C.Verify}_{\text{sn}}(ck, c^i_{pk}, c^i_{sn}, \pi^i_{sn}) \wedge \mathsf{C.Verify}_{\text{tag}}(ck, c^{i-1}_{pk}, c^{i-1}_{sn}, c^i_{sn}, c^i_{tag}, \pi^i_{tag})$
- $\mathsf{C.E.Verify}_{\text{enc}}(ck, ek, c^i_{sn}, \tilde{c}^i_{sn}, \tilde{\pi}^i_{sn}) \wedge \mathsf{C.E.Verify}_{\text{enc}}(ck, ek, c^i_{tag}, \tilde{c}^i_{tag}, \tilde{\pi}^i_{tag})$

**Our scheme.**

Using the above, we now give the formal definition of our transferable e-cash scheme. (Recall that $par$ is an implicit input to all algorithms.)

$\underline{\mathsf{ParamGen}}(1^\lambda)$:

- $Gr \leftarrow \mathsf{BGGen}(1^\lambda)$
- $par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
- $par_{\mathsf{S}'} \leftarrow \mathsf{S}'.\mathsf{Setup}(Gr)$

- $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
- $ck \leftarrow \mathsf{C.Setup}(Gr)$
- Return $par = (1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)$

$\underline{\mathsf{BKeyGen}}()$:

- Parse $par$ as $(1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)$
- $(sk, vk) \leftarrow \mathsf{S.KeyGen}(par_{\mathsf{S}})$
- $(sk', vk') \leftarrow \mathsf{S}'.\mathsf{KeyGen}(par_{\mathsf{S}'})$
- $(ek_{\text{init}}, dk_{\text{init}}) \leftarrow \mathsf{E}'.\mathsf{KeyGen}(Gr)$
- $(ek, dk) \leftarrow \mathsf{E.KeyGen}(Gr)$
- $(sk_{\mathsf{T}}, pk_{\mathsf{T}}) \leftarrow \mathsf{T.KeyGen}(par_{\mathsf{T}})$
- $cert \leftarrow \mathsf{S}'.\mathsf{Sign}(sk', pk_{\mathsf{T}})$
- Return $(sk_{\mathcal{W}} = (sk, sk'), sk_{\mathcal{CK}} = (dk_{\text{init}}, dk),$
  $sk_{\mathcal{D}} = (cert, pk_{\mathsf{T}}, sk_{\mathsf{T}}), pk_{\mathcal{B}} = (ek_{\text{init}}, ek, vk, vk'))$

$\underline{\mathsf{Register}}\langle \mathcal{B}(sk_{\mathcal{W}} = (sk, sk')), \mathcal{U}(pk_{\mathcal{B}} = (ek_{\text{init}}, ek, vk, vk'))\rangle$:

$\mathcal{U}$: $(sk_{\mathsf{T}}, pk_{\mathsf{T}}) \leftarrow \mathsf{T.KeyGen}(1^\lambda)$; send $pk_{\mathsf{T}}$ to $\mathcal{B}$

$\mathcal{B}$: $cert_{\mathcal{U}} \leftarrow \mathsf{S}'.\mathsf{Sign}(sk', pk_{\mathsf{T}})$; send $cert_{\mathcal{U}}$ to $\mathcal{U}$; output $pk_{\mathsf{T}}$

$\mathcal{U}$: If $\mathsf{S}'.\mathsf{Verify}(vk', pk_{\mathsf{T}}, cert_{\mathcal{U}}) = 1$, output $sk_{\mathcal{U}} \leftarrow (cert_{\mathcal{U}}, pk_{\mathsf{T}}, sk_{\mathsf{T}})$; else $\perp$

$\underline{\mathsf{Withdraw}}\langle \mathcal{B}(sk_{\mathcal{W}} = (sk, sk'), pk_{\mathcal{B}} = (ek_{\text{init}}, ek, vk, vk')),$
$\mathcal{U}(sk_{\mathcal{U}} = (cert_{\mathcal{U}}, pk_{\mathsf{T}}, sk_{\mathsf{T}}), pk_{\mathcal{B}})\rangle$:

$\mathcal{U}$:
- $n \xleftarrow{\$} \mathcal{N}$; $\rho_{sn}, \rho_{cert}, \rho_{pk}, \rho_M \xleftarrow{\$} \mathcal{R}$
- $(sn, M_{sn}) \leftarrow \mathsf{T.SGen}_{\text{init}}(sk_{\mathsf{T}}, n)$
- $c_{cert} \leftarrow \mathsf{C.Cm}(ck, cert_{\mathcal{U}}, \rho_{cert})$
- $c_{sn} \leftarrow \mathsf{C.Cm}(ck, sn, \rho_{sn})$
- $c_{pk} \leftarrow \mathsf{C.Cm}(ck, pk_{\mathsf{T}}, \rho_{pk})$
- $c_M \leftarrow \mathsf{C.Cm}(ck, M_{sn}, \rho_M)$
- $\pi_{cert} \leftarrow \mathsf{C.Prv}(ck, \mathsf{S}'.\mathsf{Verify}(vk', \cdot, \cdot) = 1, (pk_{\mathsf{T}}, \rho_{pk}), (cert_{\mathcal{U}}, \rho_{cert}))$
- $\pi_{sn} \leftarrow \mathsf{C.Prv}_{\text{sn,init}}(ck, pk_{\mathsf{T}}, sn, M_{sn}, \rho_{pk}, \rho_{sn}, \rho_M)$
- Send $(c_{pk}, c_{cert}, \pi_{cert}, c_{sn}, c_M, \pi_{sn})$ to $\mathcal{B}$

$\mathcal{B}$:
- if $\mathsf{C.Verify}(ck, \mathsf{S}'.\mathsf{Verify}(vk', \cdot, \cdot) = 1, c_{pk}, c_{cert}, \pi_{cert})$ or
  $\mathsf{C.Verify}_{\text{sn,init}}(ck, c_{pk}, c_{sn}, c_M, \pi_{sn})$ fail then abort and output $\perp$.
- $(c_\sigma, \pi_\sigma) \leftarrow \mathsf{SigCm}(ck, sk, c_M)$; send $(c_\sigma, \pi_\sigma)$ to $\mathcal{U}'$; return $ok$

$\mathcal{U}$:
- if $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M, c_\sigma, \pi_\sigma)$ fails, abort and output $\perp$.
- $\nu_{sn} \xleftarrow{\$} \mathcal{R}$
- $\tilde{c}_{sn} \leftarrow \mathsf{E}'.\mathsf{Enc}(ek_{\text{init}}, sn, \nu_{sn})$

– $\tilde{\pi}_{sn} \leftarrow \mathsf{C.E'.Prv}_{\mathrm{enc}}(ck, ek_{\mathrm{init}}, sn, \rho_{sn}, \nu_{sn}, \tilde{c}_{sn})$
– $\rho'_{pk}, \rho'_{cert}, \rho'_{sn}, \rho'_{M}, \rho'_{\sigma}, \nu'_{sn}, \rho'_{\tilde{\pi},sn} \xleftarrow{\$} \mathcal{R}$ *//since $\tilde{\pi}_{sn}$ contains a commitment, we also sample randomness for it*
– $c^0 \leftarrow \mathsf{Rand}((c_{pk}, c_{cert}, \pi_{cert}, c_{sn}, \pi_{sn}, c_M, c_\sigma, \pi_\sigma, \tilde{c}_{sn}, \tilde{\pi}_{sn}), (\rho'_{pk}, \rho'_{cert}, \rho'_{sn}, \rho'_{M}, \rho'_{\sigma}, \nu'_{sn}, \rho'_{\tilde{\pi},sn}))$
– Output $(c^0, n, sn, \rho_{sn} + \rho'_{sn}, \rho_{pk} + \rho'_{pk})$

$\underline{\mathsf{Spend}}\langle \mathcal{U}(c, sk_{\mathcal{U}} = (cert, pk_{\mathsf{T}}, sk_{\mathsf{T}}), pk_{\mathcal{B}} = (ek_{\mathrm{init}}, ek, vk, vk')), \mathcal{U}'(sk'_{\mathcal{U}} = (cert', pk'_{tag}, sk'_{tag}), pk_{\mathcal{B}})\rangle$:

$\mathcal{U}'$:
– $n' \xleftarrow{\$} \mathcal{N}$; $\rho'_{sn}, \rho'_{cert}, \rho'_{pk}, \rho'_{sn\text{-}pf}, \nu'_{sn} \xleftarrow{\$} \mathcal{R}$
– $(sn', sn\text{-}pf') \leftarrow \mathsf{T.SGen}(par_{\mathsf{T}}, sk'_{tag}, n')$
– $c'_{cert} \leftarrow \mathsf{C.Cm}(ck, cert', \rho'_{cert})$
– $c'_{pk} \leftarrow \mathsf{C.Cm}(ck, pk', \rho'_{pk})$
– $c'_{sn} \leftarrow \mathsf{C.Cm}(ck, sn', \rho'_{sn})$
– $c'_{sn\text{-}pf} \leftarrow \mathsf{C.Cm}(ck, sn\text{-}pf', \rho'_{sn\text{-}pf})$
– $\tilde{c}'_{sn} \leftarrow \mathsf{E.Enc}(ek, sn', \nu'_{sn})$
– $\pi'_{cert} \leftarrow \mathsf{C.Prv}(ck, \mathsf{S.Verify}(vk_{\mathcal{B}}, \cdot, \cdot) = 1, (pk'_{tag}, \rho'_{pk}), (cert', \rho'_{cert}))$
– $\pi'_{sn} \leftarrow \mathsf{C.Prv}_{\mathrm{sn}}(ck, pk'_{\mathsf{T}}, sn', sn\text{-}pf, \rho'_{pk}, \rho'_{sn}, \rho'_{sn\text{-}pf})$
– $\tilde{\pi}'_{sn} \leftarrow \mathsf{C.E.Prv}_{\mathrm{enc}}(ck, ek, sn', \rho'_{sn}, \nu'_{sn}, \tilde{c}'_{sn})$
– Send $(sn', \rho'_{sn})$ to $\mathcal{U}$

$\mathcal{U}$:
– Parse $c$ as $(c^0, (c^j = (c^j_{pk}, c^j_{cert}, \pi^j_{cert}, c^j_{sn}, \pi^j_{sn}, c^j_{tag}, \pi^j_{tag}, \tilde{c}^j_{sn}, \tilde{c}^j_{tag}, \tilde{\pi}^j_{sn}, \tilde{\pi}^j_{tag}))^i_{j=1}, n, sn, \rho_{sn}, \rho_{pk})$
– $\rho_{tag}, \nu_{tag}, \rho_{t\text{-}pf} \xleftarrow{\$} \mathcal{R}$
– $(tag, t\text{-}pf) \leftarrow \mathsf{T.TGen}(par_{\mathsf{T}}, sk_{\mathsf{T}}, n, sn')$
– $c_{tag} \leftarrow \mathsf{C.Cm}(ck, tag, \rho_{tag})$
– $\tilde{c}_{tag} \leftarrow \mathsf{E.Enc}(ek, tag, \nu_{tag})$
– $\pi_{tag} \leftarrow \mathsf{C.Prv}_{tag}(ck, pk_{\mathsf{T}}, sn, sn', tag, t\text{-}pf, \rho_{pk}, \rho_{sn}, \rho'_{sn}, \rho_{tag}, \rho_{t\text{-}pf})$
– $\tilde{\pi}_{tag} \leftarrow \mathsf{C.E.Prv}_{\mathrm{enc}}(ck, ek, tag, \rho_{tag}, \nu_{tag}, \tilde{c}_{tag})$
– Send $c' = (c^0, (c^j)^i_{j=1}, c_{tag}, \pi_{tag}, \tilde{c}_{tag}, \tilde{\pi}_{tag})$ to $\mathcal{U}'$; output $ok$

$\mathcal{U}'$:
– If any of the following fail then abort and output $\perp$:
  – $\mathsf{VER}_{\mathrm{init}}(ek_{\mathrm{init}}, c^0)$
  – $\mathsf{VER}_{\mathrm{body}}(ek, vk, vk', c^{j-1}, c^j)$, for $j = 1, \ldots, i$
  – $\mathsf{C.Verify}_{\mathrm{tag}}(ck, c^i_{pk}, c^i_{sn}, c'_{sn}, c_{tag}, \pi_{tag})$
  – $\mathsf{C.E.Verify}_{\mathrm{enc}}(ck, ek, c_{tag}, \tilde{c}_{tag}, \tilde{\pi}_{tag})$
– pick uniformly at random $\overrightarrow{\rho'}$
– $c_{\mathrm{rand}} \leftarrow \mathsf{Rand}(((c^j)^i_{j=0}, c'_{pk}, c'_{cert}, \pi'_{cert}, c'_{sn}, \pi'_{sn}, c_{tag}, \pi_{tag}, \tilde{c}'_{sn}, \tilde{\pi}'_{sn}, \tilde{c}'_{tag}, \tilde{\pi}'_{tag}), \overrightarrow{\rho'})$
– Output $(c_{\mathrm{rand}}, n', sn', \rho'_{sn} + (\overrightarrow{\rho'})_{sn'}, \rho'_{pk} + (\overrightarrow{\rho'})_{pk'})$

$\underline{\mathsf{CheckDS}}(sk_{\mathcal{CK}} = (dk_{\mathrm{init}}, dk), \mathcal{DCL}, \mathcal{UL}, c)$:

- Parse $c$ as $\big(c^0 = (c^0_{pk}, c^0_{cert}, \pi^0_{cert}, c^0_{sn}, \pi^0_{sn}, c^0_M, c_\sigma, \pi_\sigma, \tilde{c}^0_{sn}, \tilde{\pi}^0_{sn}),$
  $(c^j = (c^j_{pk}, c^j_{cert}, \pi^j_{cert}, c^j_{sn}, \pi^j_{sn}, c^j_{tag}, \pi^j_{tag}, \tilde{c}^j_{sn}, \tilde{\pi}^j_{sn}, \tilde{c}^j_{tag}, \tilde{\pi}^j_{tag}))^i_{j=1}, n, \mathrm{sn}, \rho_{sn}, \rho_{pk}\big)$
- $\overrightarrow{sn} \leftarrow (\mathsf{E.Dec}(dk_{\text{init}}, \tilde{c}^0_{sn}), \mathsf{E.Dec}(dk, \tilde{c}^1_{sn}), \ldots, \mathsf{E.Dec}(dk, \tilde{c}^i_{sn}))$
- $\overrightarrow{tag} \leftarrow (\mathsf{E.Dec}(dk, \tilde{c}^1_{tag}), \ldots, \mathsf{E.Dec}(dk, \tilde{c}^i_{tag}))$
- If for all $(\overrightarrow{sn'}, \overrightarrow{tag'}) \in \mathcal{DCL}$: $(\overrightarrow{sn})_0 \neq (\overrightarrow{sn'})_0$

  then return $\mathcal{DCL} \parallel (\overrightarrow{sn}, \overrightarrow{tag})$
- Else let $j$ be minimal so that $(\overrightarrow{sn})_j \neq (\overrightarrow{sn'})_j$
- $(pk_{\mathsf{T}}, \Pi) \leftarrow \mathsf{T.Detect}\big((\overrightarrow{sn})_j, (\overrightarrow{sn'})_j, (\overrightarrow{tag})_j, (\overrightarrow{tag'})_j, \mathcal{UL}\big)$
- Return $(pk_{\mathsf{T}}, \Pi)$

$\underline{\mathsf{VfyGuilt}}(pk_{\mathsf{T}}, \Pi)$: Return $\mathsf{T.VfyGuilt}(pk_{\mathsf{T}}, \Pi)$

### Correctness and security analysis

**Theorem 5.5.** *Our transferable e-cash scheme satisfies all* **correctness** *properties and is perfectly* **sound**.

The first four correctness properties follow in a straightforward way from the correctness properties of $\mathsf{S}$, $\mathsf{S}'$ and $\mathsf{C}$, and verifiability of $\mathsf{T}$. The fifth property follows from the fact that $sk_{\mathcal{D}}$ has the form of a customer secret key.

Because a user verifies the validity of all components of a coin before accepting it, perfect soundness of our scheme is a direct consequence of the correctness properties of $\mathsf{S}$, $\mathsf{S}'$ and $\mathsf{C}$, and in particular perfect soundness of $\mathsf{C}$, as well as verifiability of $\mathsf{T}$.

The detailed proofs of the following theorems can be found in Appendix A.1. We omit the proofs for `u-an` and `c-tr` as they are analogous to the one for `c-an`.

**Theorem 5.6.** *Let $\mathcal{N}$ be the nonce space and $\mathcal{S}$ be the space of signatures of scheme $\mathsf{S}$. Let $\mathsf{A}$ be an adversary that wins the* **unforgeability** *game with advantage $\epsilon$ and makes at most $d$ calls to* `BDepo`*. Suppose that $\mathsf{C}$ is perfectly sound and $(\mathcal{M} \cup \mathcal{S})$-extractable. Then there exist adversaries against the unforgeability of the signature schemes $\mathsf{S}$ and $\mathsf{S}'$ with advantages $\epsilon_{\text{sig}}$ and $\epsilon'_{\text{sig}}$, resp., such that*

$$\epsilon \leq \epsilon_{\text{sig}} + \epsilon'_{\text{sig}} + d^2/|\mathcal{N}|.$$

Assume that during the adversary's deposits the bank never picks the same final nonce twice. (The probability that there is a collision is at most $d^2/|\mathcal{N}|$.) In this case, there are two ways for the adversary to win:
(1) $\mathsf{CheckDS}$ outputs $\perp$, or an invalid proof, or an unregistered user: Suppose that, during a `BDepo` call for a coin $c$, $\mathsf{CheckDS}$ does not return a coin list. Recall that, by assumption, the final part (chosen by the bank at deposit) of the serial number of $c$ is fresh. Since $\mathsf{CheckDS}$ runs $\mathsf{T.Detect}$, by soundness of $\mathsf{C}$ and two-extractability of $\mathsf{T}$, this will output a pair $(pk, \Pi)$, such that $\mathsf{VfyGuilt}(pk, \Pi) = 1$. Since a coin contains a commitment to a certificate for the used tag key (and proofs of validity), we can, again by soundness of $\mathsf{C}$, extract an $\mathsf{S}'$-signature on $pk$. Now if $pk$ is not in $\mathcal{UL}$, then it was never signed by the bank, and $\mathsf{A}$ has thus broken unforgeability of $\mathsf{S}'$.
(2) $q_W < |\mathcal{DCL}|$: If the adversary creates a valid coin that has not been withdrawn, then by soundness of $\mathsf{C}$, we can extract a signature by the bank on a new initial serial number and therefore break unforgeability of $\mathsf{S}$.

**Theorem 5.7.** *Let* $\mathsf{A}$ *be an adversary that wins the game* **exculpability** *with advantage* $\epsilon$ *and makes* $u$ *calls to the oracle* `URegist`*. Then there exist adversaries against mode-indistinguishability of* $\mathsf{C}$ *and tag-exculpability of* $\mathsf{T}$ *with advantages* $\epsilon_{\text{m-ind}}$ *and* $\epsilon_{\text{t-exc}}$*, resp., such that*

$$\epsilon \;\leq\; u \cdot \epsilon_{\text{t-exc}} + \epsilon_{\text{m-ind}}.$$

An incrimination proof in our e-cash scheme is simply an incrimination proof of the tag scheme $\mathsf{T}$. Thus, if the reduction correctly guesses the user $u$ that will be wrongfully incriminated by $\mathsf{A}$ (which it can with probability $1/u$), then we can construct an adversary against exculpability of $\mathsf{T}$. The term $\epsilon_{\text{m-ind}}$ comes from the fact that we first need to switch $\mathsf{C}$ to hiding mode, so we can simulate $\pi_{sn}$ and $\pi_{tag}$ for the target user, since the oracles $O_1$ and $O_2$ in the game for tag exculpability (see Fig. 2.3) do not return *sn-pf* and *t-pf*.

**Theorem 5.8.** *Let* $\mathsf{A}$ *be an adversary that wins the* **coin anonymity** *game (*`c-an`*) with advantage* $\epsilon$ *and let* $k$ *be an upper-bound on the number of users transferring the challenge coins. Then there exist adversaries against mode-indistinguishability of* $\mathsf{C}$ *and tag-anonymity of* $\mathsf{T}$ *with advantages* $\epsilon_{\text{m-ind}}$ *and* $\epsilon_{\text{t-an}}$*, resp., such that*

$$\epsilon \;\leq\; 2\,(\epsilon_{\text{m-ind}} + (k+1)\,\epsilon_{\text{t-an}}).$$

**Theorem 5.9.** *Let* $\mathsf{A}$ *be an adversary that wins the* **user anonymity** *game (*`u-an`*) with advantage* $\epsilon$ *and let* $k$ *be a bound on the number of users transferring the challenge coin. Then there exist adversaries against mode-indistinguishability of* $\mathsf{C}$ *and tag-anonymity of* $\mathsf{T}$ *with advantages* $\epsilon_{\text{m-ind}}$ *and* $\epsilon_{\text{t-an}}$*, resp., such that*

$$\epsilon \;\leq\; 2\,\epsilon_{\text{m-ind}} + (k+1)\,\epsilon_{\text{t-an}}.$$

In the proof of both theorems, we first define a hybrid game in which the commitment key is switched to hiding mode (hence the loss $\epsilon_{\text{m-ind}}$, which occurs twice for $b = 0$ and $b = 1$). All commitments are then perfectly hiding and the only information available to the adversary are the serial numbers and tags. (They are encrypted in the coin, but the adversary, impersonating the bank, can decrypt them.)

We then argue that, by tag anonymity of $\mathsf{T}$, the adversary cannot link a user to a pair $(sn, tag)$, even when it knows the users' secret keys. We define a sequence of $k+1$ hybrid games (as $k$ transfers involve $k+1$ users); going through the user vector output by the adversary, we can switch, one-by-one all users from the first two the second vector. Each switching can be detected by the adversary with probability at most $\epsilon_{\text{t-an}}$. Note the additional factor 2 for $\epsilon_{\text{t-an}}$ in game `c-an`, which is due to the fact that there are two coins in witch we switch users, whereas there is only one in game `u-an`.

**Theorem 5.10.** *Let* $\mathsf{A}$ *be an adversary that wins the* **coin-transparency** *game (*`c-tr`*) with advantage* $\epsilon$*, let* $\ell$ *be the size of the challenge coins, and* $k$ *be an upper-bound on the number of users transferring the challenge coins. Then there exist adversaries against mode-indistinguishability of* $\mathsf{C}$*, tag-anonymity of* $\mathsf{T}$*, IACR-security of* $\mathsf{E}$ *and RCCA-security of* $\mathsf{E}'$ *with advantages* $\epsilon_{\text{m-ind}}$*,* $\epsilon_{\text{t-an}}$*,* $\epsilon_{\text{iacr}}$ *and* $\epsilon_{\text{rcca}}$*, resp., such that*

$$\epsilon \;\leq\; 2\,\epsilon_{\text{m-ind}} + (k+1)\,\epsilon_{\text{t-an}} + 2\,\ell\,\epsilon_{\text{iacr}} + \epsilon_{\text{rcca}}.$$

The crucial difference to the previous anonymity theorems is that the bank is honest (which makes this strong notion possible). We therefore must rely on the security of the encryptions, for which the reduction thus does not know the decryption key. At the same time, the reduction must be able to detect double-spendings, when the adversary deposits coins. Since we use RCCA encryption, the reduction can do so by using its own decryption oracle.

As for `c-an` and `u-an`, the reduction first makes all commitments perfectly hiding and proofs perfectly simulatable (which loses $\epsilon_{\text{m-ind}}$ twice). Since all ciphertexts in the challenge coin given to the adversary are randomized, the reduction can replace all of them, except the initial one, by IACR-security of $\mathsf{E}$. (Note that in the game these ciphertexts never need to be decrypted.) The factor $2\ell$ is due to the fact that there are at most $\ell$ encryptions of SN/tag pairs. Finally, replacing the initial ciphertext (the one that enables detection of double-spending) can be done by a reduction to RCCA-security of $\mathsf{E}'$: the oracle $\mathtt{Depo}'$ can be simulated by using the reduction's own oracles $\mathsf{Dec}$ and $\mathsf{GDec}$ oracles (depending on whether $\mathtt{Depo}'$ is called before or after the reduction receives the challenge ciphertext) in the RCCA-security game. Note that, when during a simulation of $\mathsf{CheckDS}$, oracle $\mathsf{GDec}$ outputs `replay`, the reduction knows that a challenge coin was deposited, and thus increases *ctr*.

## 5.4 Instantiation of the building blocks and efficiency

The instantiations we use are all proven secure in the standard model under non-interactive hardness assumptions.

**Commitments and proofs.** The commit-and-prove system $\mathsf{C}$ will be instantiated with Groth-Sahai proofs [GS08], of which we use the instantiation based on SXDH.

**Theorem 5.11** ([GS08])**.** *The Groth-Sahai scheme with values $\mathcal{V} := \mathbb{Z}_p \cup \mathbb{G}_1 \cup \mathbb{G}_2$ is perfectly complete, sound and randomizable; it is $(\mathbb{G}_1 \cup \mathbb{G}_2)$-extractable, mode-indistinguishable assuming SXDH, and perfectly hiding in the hiding mode.*

We note that moreover, all our proofs can be made zero-knowledge [GS08], because all pairing-product equations we use are homogeneous (i.e., the right-hand term is the neutral element). We have (efficient) extractability, as we only need to efficiently extract group elements from commitments (and not scalars) in our reductions. (Note that for information-theoretic arguments concerning soundness, $\mathsf{Extr}$ can also be inefficient.)

**Signature schemes.** For efficiency and type-compatibility reasons, we use two different signature schemes. The first one needs to support the functionality $\mathsf{SigCm}$, which implies a specific format of messages. The second scheme is less restrictive, which simplifies the description of our scheme and makes its instantiation more efficient. While all our other components rely on standard assumptions, the following scheme is secure under a non-interactive $q$-type assumption defined in [AFG+10].

**Theorem 5.12.** *The signature scheme from [AFG+10] with message space $\mathcal{M} := \{(g^m, \hat{g}^m) \mid m \in \mathbb{Z}_p\}$ is (strongly) unforgeable assuming q-ADHSDH and AWFCDH (defined in Section 2.3), and it supports the $\mathsf{SigCm}$ functionality [Fuc11].*

**Theorem 5.13.** *The signature scheme described in [AGHO11, Section 5] is structure-preserving with message space $\mathcal{M}' := \mathbb{G}_2$ and (strongly) unforgeable assuming SXDH.*

**Randomizable encryption schemes.** To instantiate the RCCA-secure scheme $\mathsf{E}'$ we follow the approach from [LPQ17]. Their construction is only for one group element, but by adapting the scheme, it can support encryption of a vector in $\mathbb{G}^n$ for arbitrary $n$.

In our e-cash scheme, we need to encrypt a vector in $\mathbb{G}^2$, and since it is not clear whether more recent efficient schemes like [FFHR19] can be adapted to this, we give an explicit construction, which we detail in Appendix A.2.

Recall that the RCCA-secure scheme $\mathsf{E}'$ is only used to encrypt the initial part of the serial number; using a less efficient scheme does thus not have a big impact on the efficiency of our scheme.

From all other ciphertexts contained in a coin (which are under scheme $\mathsf{E}$) we only require IACR security, which standard ElGamal encryption satisfies under DDH. Thus, we instantiate $\mathsf{E}$ with ElGamal vector encryption. (Note that our instantiation of $\mathsf{E}'$ is also built on top of ElGamal). We prove the following in the appendix.

**Theorem 5.14.** *Assuming SXDH, our randomizable encryption scheme in Appendix A.2 is RCCA-secure and the one in Appendix A.2 is IACR-secure.*

**Double-spending tags.** We will use a scheme that builds on the one given in [BCFK15]. We have optimized the size of the tags and made explicit all the functionalities not given in the previous version. We defer this to Appendix A.2.

### Efficiency analysis

For a group $G \in \{\mathbb{G}, \hat{\mathbb{G}}, \mathbb{Z}_p\}$, let $|G|$ denote the size of one element of $G$. Let $c_{\text{btsrap}}$ denote the coin output by $\mathcal{U}$ at the end of the Withdraw protocol (which corresponds to $c_{\text{init}}$ plus secret values, like $n$, $\rho_{sn}$, etc., to be used when transferring the coin), and let $c_{\text{std}}$ one (non-initial) component of the coin. We conclude by summarizing the characteristics of our scheme in the following table and refer to Appendix A.3 for the details of our analysis.

After $k$ transfers the size of a coin is $|c_{\text{btsrap}}| + k|c_{\text{std}}|$.

| | | | |
|---|---|---|---|
| $\lvert sk_{\mathcal{B}}\rvert$ | $9\lvert\mathbb{Z}_p\rvert + 2\lvert\mathbb{G}\rvert + 2\lvert\hat{\mathbb{G}}\rvert$ | $\lvert\Pi_{\text{guilt}}\rvert$ | $2\lvert\mathbb{G}\rvert$ |
| $\lvert pk_{\mathcal{B}}\rvert$ | $15\lvert\mathbb{G}\rvert + 8\lvert\hat{\mathbb{G}}\rvert$ | $\lvert c_{\text{btstrap}}\rvert$ | $6\lvert\mathbb{Z}_p\rvert + 147\lvert\mathbb{G}\rvert + 125\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert sk_{\mathcal{U}}\rvert$ | $\lvert\mathbb{Z}_p\rvert + 2\lvert\mathbb{G}\rvert + 2\lvert\hat{\mathbb{G}}\rvert$ | $\lvert c_{\text{std}}\rvert$ | $54\lvert\mathbb{G}\rvert + 50\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert pk_{\mathcal{U}}\rvert$ | $\lvert\hat{\mathbb{G}}\rvert$ | $\lvert(\overrightarrow{sn}, \overrightarrow{tag})\rvert$ | $(4t+2)\lvert\mathbb{G}\rvert$ |

# Bibliography


[ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587. IEEE Computer Society Press, May 2015.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, Heidelberg, August 2011.

[AGO11] Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, December 2011.

[AHK20] Thomas Agrikola, Dennis Hofheinz, and Julia Kastner. On instantiating the algebraic group model from falsifiable assumptions. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 96–126. Springer, Heidelberg, May 2020.

[AW98] Claude Deschamps André Warusfel, François Moulin. *Mathématiques 1ère année : Cours et exercices corrigés*. Editions Dunod, 1998.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BB15] Joao MM Barguil and Paulo SLM Barreto. Security issues in Sarkar's e-cash protocol. *Information Processing Letters*, 115(11):801–803, 2015.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[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, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.

[BCF+11] Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 206–223. Springer, Heidelberg, July 2011.

[BCFK15] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. Anonymous transferable E-cash. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 101–124. Springer, Heidelberg, March / April 2015.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, Heidelberg, March 2008.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, Heidelberg, August 2009.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87. Springer, Heidelberg, March 2009.

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, Heidelberg, March 2011.

[BFPV13] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short blind signatures. *Journal of computer security*, 21(5):627–661, 2013.

[BHKS18] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 405–434. Springer, Heidelberg, December 2018.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013.

[Bla08] Marina Blanton. Improved conditional e-payments. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 08*, volume 5037 of *LNCS*, pages 188–206. Springer, Heidelberg, June 2008.

[BMV08] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the "one-more" computational problems. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 71–87. Springer, Heidelberg, April 2008.

[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, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008.

[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*, pages 62–73. ACM Press, November 1993.

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/2004/331.

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

[Bra00] Stefan Brands. *Rethinking public-key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.

[CCFG16] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1614–1625. ACM Press, October 2016.

[CFL19] Véronique Cortier, Alicia Filipiak, and Joseph Lallemand. Beleniosvs: Secrecy and verifiability against a corrupted voting device. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 367–36714. IEEE, 2019.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Heidelberg, August 1990.

[CG05] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, 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, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 482–497. Springer, Heidelberg, May 2007.

[CG08] Sébastien Canard and Aline Gouget. Anonymity in transferable e-cash. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 08*, volume 5037 of *LNCS*, pages 207–223. Springer, Heidelberg, June 2008.

[CGG19] Véronique Cortier, Pierrick Gaudry, and Stephane Glondu. Belenios: a simple private and verifiable electronic voting system. In *Foundations of Security, Protocols, and Equational Reasoning*, pages 214–238. Springer, 2019.

[CGT08] Sébastien Canard, Aline Gouget, and Jacques Traoré. Improvement of efficiency in (unconditional) anonymous transferable e-cash. In Gene Tsudik, editor, *FC 2008*, volume 5143 of *LNCS*, pages 202–214. Springer, Heidelberg, January 2008.

[Cha83] David Chaum. Blind signature system. In David Chaum, editor, *CRYPTO'83*, page 153. Plenum Press, New York, USA, 1983.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Heidelberg, April 2012.

[CKLM14] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *CSF 2014*, pages 199–213. IEEE, 2014.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Heidelberg, August 2003.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, 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, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

[CL19] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 535–555. Springer, Heidelberg, March 2019.

[CLM07] Jan Camenisch, Anna Lysyanskaya, and Mira Meyerovich. Endorsed e-cash. In *2007 IEEE Symposium on Security and Privacy*, pages 101–115. IEEE Computer Society Press, May 2007.

[CM14] Melissa Chase and Sarah Meiklejohn. Déjà Q: Using dual systems to revisit q-type assumptions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 622–639. Springer, Heidelberg, May 2014.

[CMM16] Melissa Chase, Mary Maller, and Sarah Meiklejohn. Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 655–681. Springer, Heidelberg, December 2016.

[CP93] David Chaum and Torben P. Pedersen. Transferred cash grows in size. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 390–407. Springer, Heidelberg, May 1993.

[CPST15] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible E-cash made practical. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 77–100. Springer, Heidelberg, March / April 2015.

[DH76] Whitfield Diffie and Martin 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, Part II*, volume 9986 of *LNCS*, pages 547–576. Springer, Heidelberg, October / November 2016.

[DHS15] David Derler, Christian Hanser, and Daniel Slamanig. A new approach to efficient revocable attribute-based anonymous credentials. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 57–74. Springer, Heidelberg, December 2015.

[DL77] Richard A DeMillo and Richard J Lipton. A probabilistic remark on algebraic program testing. Technical report, Georgia Inst of Tech Atlanta School of Information and Computer Science, 1977.

[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*, pages 551–565. ACM Press, April 2018.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[EO95] Tony Eng and Tatsuaki Okamoto. Single-term divisible electronic coins. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 306–319. Springer, Heidelberg, May 1995.

[FFHR19] Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. Cryptology ePrint Archive, Report 2019/955, 2019. https://eprint.iacr.org/2019/955.

[FG18] Georg Fuchsbauer and Romain Gay. Weakly secure equivalence-class signatures from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 153–183. Springer, Heidelberg, March 2018.

[FGKO17] Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi. Access control encryption for equality, comparison, and more. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 88–118. Springer, Heidelberg, March 2017.

[FHKS16] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Heidelberg, August / September 2016.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, Heidelberg, December 2009.

[Fri] David D. Friedman. Blog of david d. friedman. http://daviddfriedman.blogspot.co.uk/2014/09/bitcoin-anonymous-ecash-and-strong.html.

[Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Heidelberg, May 2011.

[FY93] Matthew K. Franklin and Moti Yung. Secure and efficient off-line digital money (extended abstract). In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *ICALP'93*, volume 700 of *LNCS*, pages 265–276. Springer, 1993.

[GG17] Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 66–96. Springer, Heidelberg, December 2017.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[HPP20] Chloé Hébant, Duong Hieu Phan, and David Pointcheval. Linearly-homomorphic signatures and scalable mix-nets. In *PKC 2020*, LNCS. Springer, 2020.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511. Springer, Heidelberg, December 2014.

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 63–93. Springer, Heidelberg, December 2019.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

[Lip19] Helger Lipmaa. Simulation-extractable SNARKs revisited. *ePrint Cryptogoly Archive, Report 2019/612*, 2019.

[Los19] Julian Loss. *New techniques for the modular analysis of digital signature schemes*. PhD thesis, Ruhr University Bochum, Germany, 2019.

[LPJY13] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 289–307. Springer, Heidelberg, August 2013.

[LPQ17] Benoît Libert, Thomas Peters, and Chen Qian. Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 247–276. Springer, Heidelberg, March 2017.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, Heidelberg, August 1999.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.

[MTT19] Taiga Mizuide, Atsushi Takayasu, and Tsuyoshi Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 169–188. Springer, Heidelberg, March 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash. 2008. bitcoin.org/bitcoin.pdf.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[Oka95] Tatsuaki Okamoto. An efficient divisible electronic cash scheme. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 438–451. Springer, Heidelberg, August 1995.

[OO90] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 481–496. Springer, Heidelberg, August 1990.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Heidelberg, August 1992.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, Heidelberg, February 2001.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sar13] Pratik Sarkar. Multiple-use transferable e-cash. *International Journal of Computer Applications*, 77(6), 2013.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[SCS07] Larry Shi, Bogdan Carbunar, and Radu Sion. Conditional e-cash. In Sven Dietrich and Rachna Dhamija, editors, *FC 2007*, volume 4886 of *LNCS*, pages 15–28. Springer, Heidelberg, February 2007.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[TG14] Kamlesh Tiwari and Phalguni Gupta. Biometrics based observer free transferable e-cash. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 63–70. ACM, 2014.

[TH16] Hitesh Tewari and Arthur Hughes. Fully anonymous transferable ecash. Cryptology ePrint Archive, Report 2016/107, 2016. http://eprint.iacr.org/2016/107.

[Tur37] Alan Turing. On computable numbers with an application to the entscheidungsproblem. 1937.

[vAE90] H. van Antwerpen and Technische Universiteit Eindhoven. *Off-line Electronic Cash*. Eindhoven University of Technology, 1990.

[vSN92] Sebastiaan H. von Solms and David Naccache. On blind signatures and perfect crimes. *Computers & Security*, 11(6):581–583, 1992.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.

[ZLG12] Jiangxiao Zhang, Zhoujun Li, and Hua Guo. Anonymous transferable conditional e-cash. In *Security and Privacy in Communication Networks*, pages 45–60. Springer, 2012.

# Appendix A

## A.1 Security proofs for the transferable e-cash scheme

Remark that some of our theorems in the appendix are more general than the ones in the body of the thesis, as they work for a scheme with only computational soundness of $\mathsf{C}$ (whereas in the body we assume perfect soundness).

### Unforgeability

**Theorem A.1.** *Suppose that there exists an adversary $\mathcal{A}$ against unforgeability with advantage $\epsilon_{\text{unforg}}$ and using at most $d$ calls to oracle* `BDepo`*. Suppose that $\mathcal{M}$ and the signature space of $\mathsf{S}$ are included in $\mathcal{V}'$. Then we can build a polynomial-time adversary $\mathcal{B}_1$ against the unforgeability of the signature scheme $\mathsf{S}$ with advantage $\epsilon_{\text{sig}}$, an adversary $\mathcal{B}_2$ against the unforgeability of $\mathsf{S}'$ with advantage $\epsilon'_{\text{sig}}$, and $\mathcal{B}_3$ and $\mathcal{B}_4$ against the soundness of the commitment scheme $\mathsf{C}$ with advantage $\epsilon_{h,1}$ and $\epsilon_{h,2}$. Then*

$$\epsilon_{\text{unforg}} \leq \epsilon_{h,1} + \epsilon_{h,2} + \epsilon_{\text{sig}} + \epsilon'_{\text{sig}} + \frac{d^2}{|\mathcal{N}|}.$$

**Sketch of proof** Note that the adversary has two possibilities to break the game: either it creates counterfeit (i.e., $q_W < |\mathcal{CL}|$), or it wins by making a deposit fail (because $\mathsf{CheckDS}$ does not output neither a list nor a valid pair with a registered user key). In our proof we will prove separately the security of these two aspects. First we will prove in property A.2, that creating counterfeit is harder than breaking the unforgeability of $\mathsf{S}$, or proving a false statement in $\mathsf{C}$. In a second step, in property A.3, we prove that if fresh nonces are picked every time during the deposits, then it is harder to make $\mathsf{Deposit}$ fail than breaking the unforgeability of S', or proving a false statement in $\mathsf{C}$.

We first recall the security games involved. The unforgeability against the e-cash system:

**$\mathbf{Expt}_{\mathcal{A}}^{\texttt{unforg}}(\lambda)$:**
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$; $(sk_{\mathcal{B}}, pk_{\mathcal{B}}) \leftarrow \mathsf{BKeyGen}(par)$
$\mathcal{A}^{\texttt{BRegist},\texttt{BWith},\texttt{BDepo}}(par, pk_{\mathcal{B}})$
If in a `BDepo` call, $\mathsf{CheckDS}$ does not return a coin list
Return 1 if any of the following hold:
- $\mathsf{CheckDS}$ did not output a pair $(pk, \Pi)$
- $\mathsf{VfyGuilt}(pk, \Pi) = 0$
- $pk \notin \mathcal{UL}$

Let $q_W$ be the number of calls to `BWith`
If $q_W < |\mathcal{DCL}|$ then return 1
Return 0

and the unforgeability game against a signature scheme:

$par \leftarrow \mathsf{S.Setup}\left(1^{\lambda}\right)$
$(sk, vk) \leftarrow \mathsf{S.KeyGen}\,(par)$
$Q := \emptyset$
$(m, \sigma) \leftarrow \mathcal{B}^{\mathsf{S.Sign}^*(sk,\cdot,Q)}\,(par, vk)$
Return $m \notin Q \wedge \mathsf{S.Verify}\,(vk, m, \sigma)$

Oracle: $\mathsf{S.Sign}^*\,(sk, m, Q)$:
$Q := Q \cup \{m\}$
Return $\mathsf{S.Sign}\,(sk, m)$

Finally, the soundness of the commitment scheme:

$\mathbf{Expt}_{\mathcal{B}}^{\mathtt{Soundness}}\,(\lambda)$:
$(ck, xk) \leftarrow \mathsf{C.ExSetup}(1^{\lambda})$
$Q := \emptyset$; $(E, c_1, \ldots, c_n) \leftarrow \mathcal{B}_2\,(ck)$
Return $\mathsf{C.Verify}\,(ck, E, c_1, \ldots, c_n) \wedge$
$\neg E\,(\mathsf{C.Extr}\,(xk, c_1), \ldots, \mathsf{C.Extr}\,(xk, c_n))$

Let $E_{\text{unforg}}$ be the event that $\mathcal{A}$ wins the game. Then it means that, at some point after a call to BDepo, CheckDS did not output a list or $q_W < |\mathcal{DCL}|$. We partition $E_{\text{unforg}}$ as follows:

- $E_{\text{Decrypt-fails}}$: In CheckDS, a decryption fails or does not output any serial number and tag, when it is supposed to;
- $E_{\text{same}}$: In CheckDS, we do not find any $j$ such that $\overrightarrow{sn}_j \neq \overrightarrow{sn}'_j$;
- $E_{\text{DDS-fails}}$: In CheckDS, T.Detect does not output any $(pk_{\mathsf{T}}, \Pi_G)$;
- $E_{\text{incorrect}}$: CheckDS outputs $(pk_{i^*}, \Pi_G)$ such that $\mathsf{VfyGuilt}\,(pk_{i^*}, \Pi_G) = 0$;
- $E_{\text{not-register}}$: $pk_{i^*} \notin \mathcal{UL}$;
- $E_{\text{counterfeit}}$: $q_W < |\mathcal{DCL}|$;

We can first build the adversary $\mathcal{B}_1$ against the unforgeability of S, this adversary will bet on $E_{\text{counterfeit}}$: $q_W < |\mathcal{DCL}|$ (id est $\mathcal{A}$ creates valid money). It means that $\mathcal{A}$ has forged a commited signature for a fresh serial number and thus forged a signature for a fresh serial number or a false ZK-proof for the equation S.Verify (And in this second case $\mathcal{B}_3$ will break soundness). One may notice that to simulate SigCm, $\mathcal{B}_1$ needs $xk$ and SmSigCm.

Adversary $\mathcal{B}_1^{\mathcal{A},\mathsf{S.Sign}^*(sk,\cdot)}\,(par_{\mathsf{S}}, vk)$:
Extract $Gr$ from $par_{\mathsf{S}}$
$(ck, xk) \leftarrow \mathsf{C.ExSetup}\,(Gr)$
$par_{\mathsf{S}'} \leftarrow \mathsf{S}'.\mathsf{Setup}\,(Gr)$
$(sk', vk') \leftarrow \mathsf{S}'.\mathsf{KeyGen}\,(par_{\mathsf{S}'})$
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}\,(Gr)$
$par \leftarrow \left(1^{\lambda}, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck\right)$
$\mathtt{Coins_D} := \emptyset$
$\mathtt{Coins_W} := \emptyset$
$(ek, dk) \leftarrow \mathsf{E.KeyGen}\,(Gr)$
$(ek_{\text{init}}, dk_{\text{init}}) \leftarrow \mathsf{E}'.\mathsf{KeyGen}\,(Gr)$

$pk_{\mathcal{B}} \leftarrow (ek_{\text{init}}, ek, vk, vk')$
$(sk_{\mathsf{T}}, pk_{\mathsf{T}}) \leftarrow \mathsf{T.KeyGen}(Gr)$
$sk_{\mathcal{D}} \leftarrow (\epsilon, sk_{\mathsf{T}}, pk_{\mathsf{T}})$
Execute $\mathcal{A}^{\texttt{BRegist},\texttt{BWith}^*,\texttt{BDepo}}(par, pk_{\mathcal{B}})$
In each call of `BWith`, add the coin received to the list $\texttt{Coins}_{\texttt{W}}$
In each call of `BDepo`, add the coin received to the list $\texttt{Coins}_{\texttt{D}}$
$\texttt{BWith}^*$ is similar to `BWith`,
  except instead of using $\mathsf{SigCm}$ we use $\mathsf{SigCm}^*$:
  $\mathsf{SigCm}^*(c)$:
    $m \leftarrow \mathsf{C.Extr}(xk, c)$
    Use the oracle to obtain $\Sigma \leftarrow \mathsf{S.Sign}^*(sk, m)$
    $(c_\sigma, \pi) \leftarrow \mathsf{SmSigCm}(xk, vk, c, \Sigma)$
    Return $(c_\sigma, \pi)$
Let $q_W$ be the number of successful calls to `BWith` respectively
If $q_W \geq |\mathcal{DCL}|$ then abort
Let $D := \emptyset$
Let $W := \emptyset$
For $c \in \texttt{Coins}_{\texttt{W}}$:
  Parse $c$ as $(c^0, (c^j)_{j=1}^i, n, sn, \rho_{sn}, \rho_{pk})$
  Parse $c^0$ as $\left(c^0_{pk}, c^0_{cert}, \pi^0_{cert}, c^0_{sn}, \pi^0_{sn}, c_M, c^0_\sigma, \pi^0_\sigma, \tilde{c}^0_{sn}, \tilde{\pi}^0_{sn}\right)$
  $M := \mathsf{C.Extr}(xk, c_M)$
  $\sigma := \mathsf{C.Extr}(xk, c^0_\sigma)$
  $W := W \cup \{(M, \sigma)\}$
For $c \in \texttt{Coins}_{\texttt{D}}$:
  Parse $c$ as $\left(c^0, (c^j)_{j=1}^i, n, sn, \rho_{sn}, \rho_{pk}\right)$
  Parse $c^0$ as $\left(c^0_{pk}, c^0_{cert}, \pi^0_{cert}, c^0_{sn}, \pi^0_{sn}, c_M, c^0_\sigma, \pi^0_\sigma, \tilde{c}^0_{sn}, \tilde{\pi}^0_{sn}\right)$
  $M := \mathsf{C.Extr}(xk, c_M)$
  $\sigma := \mathsf{C.Extr}(xk, c^0_\sigma)$
  $D := D \cup \{(M, \sigma)\}$
If $\exists (M, \sigma) \in W \setminus D$:
  Then return $(M, \sigma)$
  Else abort

We let $\epsilon$ identify the part of the secret key that is ignored in the entire game (because the bank never spent a coin). By correctness of the committed signature of $\mathsf{S}$, the simulation will be perfect. And by $\mathcal{M} \cup \mathcal{S}$-extractability of $\mathsf{C}$, we can deduce that $\mathcal{B}_1$ is efficient.

We now construct a first adversary $\mathcal{B}_3$ against soundness:

Adversary $\mathcal{B}_3^{\mathcal{A}}(ck)$:
  Extract $Gr$ from $ck$
  $par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
  $par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}(Gr)$
  $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
  $par \leftarrow \left(1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck\right)$
  $S := \emptyset$
  $\left(pk_{\mathcal{B}} = (ek_{\text{init}}, ek, vk, vk'), sk_{\mathcal{W}} = (sk, sk'), sk_{\mathcal{D}}, sk_{\mathcal{CK}}\right) \leftarrow \mathsf{BKeyGen}()$
  Execute $\mathcal{A}^{\texttt{BRegist},\texttt{BWith},\texttt{BDepo}}(par, pk_{\mathcal{B}})$
  In each call of `BDepo`, add the whole coin received in the list $\texttt{Coins}_{\texttt{D}}$
  Let $q_W$ be the number of successful calls to `BWith`

If $q_W \geq |\mathcal{DCL}|$, then abort
Let $D := \emptyset$
For $c \in \mathtt{Coins_D}$:
  Parse $c$ as $(c_0, (c_j)_{j=1}^i, n, sn, \rho_{sn}, \rho_{pk})$
  Add $c_0$ to $S$
Parse $S$ as $\left\{(c^i_{pk_\mathsf{T}}, c^i_{cert}, \pi^i_{cert}, c^i_{sn}, \pi^i_{sn}, c^i_M, c^i_\sigma, \pi^i_\sigma, \tilde{c}^i_{sn}, \tilde{\pi}^i_{sn})\right\}_{1\leq i\leq |S|}$
Parse $\tilde{\pi}^i_{sn}$ as $(c^i, \pi^i)$
Send $\Bigg( \bigwedge_{i=1}^{|S|} \Big(\mathsf{E'.Verify}\left(ek_{\text{init}}, X^i_1, X^i_2, \tilde{c}^i_{sn}\right) \wedge \mathsf{S.Verify}\left(vk, X^i_3, X^i_4\right) = 1 \wedge$
  $\mathsf{T.SVfy}_{\text{init}}\left(X^i_5, X^i_1, X^i_3\right)\Big),$
  $c^1_{sn}, c^1, c^1_M, c^1_\sigma, c^1_{pk_\mathsf{T}}, \ldots, c^{|S|}_{sn}, c^{|S|}, c^{|S|}_M, c^{|S|}_\sigma, c^{|S|}_{pk_\mathsf{T}}, \bigwedge_{i=1}^{|S|} \pi^i \wedge \pi^i_\sigma \wedge \pi^i_{sn}\Bigg)$

Let $E_{\text{sig}}$ and $E_{\text{com},1}$ be the following events: $\mathcal{B}_1$ breaks the unforgeability of the signature scheme $\mathsf{S}$; and $\mathcal{B}_3$ breaks the soundness of the commitment scheme $\mathsf{C}$.

**Property A.2.** *$E_{counterfeit} \subseteq E_{sig} \cup E_{com,1}$.*

Suppose that we are in the case $E_{\text{counterfeit}} \setminus E_{\text{sig}}$. Because the coin has been accepted during $\mathsf{Spend}$ in a $\mathtt{BDepo}$ oracle call, the proofs output in $\mathcal{B}_3$ are correct. Let $(sn^1, \nu^1, M^1, pk^1_\mathsf{T}, \sigma^1, \ldots, sn^{|S|}, \nu^{|S|}, M^{|S|}, pk^{|S|}_\mathsf{T}, \sigma^{|S|})$ be the extraction of the commitments given by $\mathcal{B}_3$ to the challenger of the soundness game. Then if there exists $i$ such that
$\mathsf{S.Verify}\left(vk, M^i, \sigma^i\right) \neq 1$, the soundness game is won by $\mathcal{B}_3$.

Suppose that $\bigwedge_{i=1}^{|S|} \mathsf{S.Verify}\left(vk, M^i, \sigma^i\right) = 1$. Since we are not in $E_{\text{sig}}$, all the $M^i$'s correspond to one coin that has been withdrawn. But only $q_W$ messages $M$ correspond to these coins. Thus, $|\{M^i\}_{i=1}^{|S|}| \leq q_W$.
If $\bigwedge_{i=1}^{|S|} \mathsf{T.SVfy}_{\text{init}}\left(pk^i_\mathsf{T}, sn^i, M^i\right) \neq 1$, then $\mathcal{B}_3$ won the soundness game.
Assume $\bigwedge_{i=1}^{|S|} \mathsf{T.SVfy}_{\text{init}}\left(pk^i_\mathsf{T}, sn^i, M^i\right) = 1$. Since $\mathsf{T}$ is bootable, we have $|\{sn^i\}_{i=1}^{|S|}| \leq |\{M^i\}_{i=1}^{|S|}|$, from which we deduce $|\{sn^i\}_{i=1}^{|S|}| \leq q_W < |\mathcal{DCL}|$ (the last inequality comes from we suppose we are in $E_{\text{counterfeit}}$).
Note that by construction of $\mathcal{DCL}$, all the initial serial numbers of his elements are differents. Let call this set $I$. By $|I| = |\mathcal{DCL}|$, we deduce $|I| > |\{sn^i\}_{i=1}^{|S|}|$. By construction $|I| = |\{\mathsf{E'.Dec}(dk_{\text{init}}, \tilde{c}^i_{sn})\}_{i=1}^{|S|}|$, then $|\{\mathsf{E'.Dec}(dk_{\text{init}}, \tilde{c}^i_{sn})\}_{i=1}^{|S|}| > |\{sn^i\}_{i=1}^{|S|}|$. Let $i_0$ be such that $\mathsf{E'.Dec}(dk_{\text{init}}, \tilde{c}^{i_0}_{sn}) \notin |\{sn^i\}_{i=1}^{|S|}|$. It means that by correctness of $\mathsf{E'}$ $\mathsf{E'.Enc}\left(pk, sn^{i_0}, \nu^{i_0}\right) \neq \tilde{c}^{i_0}_{sn}$, and thus (still because of the correctness of $\mathsf{E'}$) $\mathsf{E'.Verify}\left(pk, sn^{i_0}, \nu^{i_0}, \tilde{c}^{i_0}_{sn}\right) \neq 1$.
We deduce that $\bigwedge_{i=1}^{|S|} \mathsf{E'.Verify}\left(pk, sn^i, \nu^i, \tilde{c}^i_{sn}\right) \neq 1$, and consequently $\mathcal{B}_3$ won the soundness game.
We thus have $E_{\text{counterfeit}} \setminus E_{\text{sig}} \subseteq E_{\text{com},1}$. □
Now we build the following algorithm to break unforgeability of $\mathsf{S'}$:

Adversary $\mathcal{B}_2^{\mathcal{A},\mathsf{S'.Sign}^*(sk',\cdot)}\left(par_{\mathsf{S}'}, vk'\right)$:
  Initialize $\mathcal{UL}$ as an empty list
  Extract $Gr$ from $par_{\mathsf{S}'}$
  $(ck, xk) \leftarrow \mathsf{C.ExSetup}\left(Gr\right)$
  $par_\mathsf{S} \leftarrow \mathsf{S.Setup}\left(Gr\right)$

$(sk, vk) \leftarrow \mathsf{S.KeyGen}(Gr)$
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
$par \leftarrow \left(1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck\right)$
$(ek_{\text{init}}, dk_{\text{init}}) \leftarrow \mathsf{E}'.\mathsf{KeyGen}(Gr)$
$(ek, dk) \leftarrow \mathsf{E.KeyGen}(Gr)$
$(sk_{tag}, pk_{\mathsf{T}}) \leftarrow \mathsf{T.KeyGen}(Gr)$
$sk_{\mathcal{D}} \leftarrow (\epsilon, pk_{\mathsf{T}}, sk_{tag})$
$pk_{\mathcal{B}} \leftarrow (ek_{\text{init}}, ek, vk, vk')$
Execute $\mathcal{A}^{\mathtt{BRegist},\mathtt{BWith},\mathtt{BDepo}}(par, pk_{\mathcal{B}})$
Every time we would use the algorithm $\mathsf{S}'.\mathsf{Sign}$ in an oracle call,
  we use the oracle $\mathsf{S}'.\mathsf{Sign}^*(sk', \cdot)$, and we add the input to $\mathcal{UL}$
Let $(pk, \Pi)$ be the output of the last call to `BDepo`
If a such pair is never returned by `BDepo`, then abort
Let $c_1$ be the last coin sent by the user
Parse $c_1$ as $\left(c^0, (c^j)_{j=1}^i, n, sn, \rho_{sn}, \rho_{pk}\right)$
Let $j$ be minimal such that $(\overrightarrow{sn})_{j-1} \neq (\overrightarrow{sn'})_{j-1}$
(using the same notation as in $\mathsf{CheckDS}$)
Parse $c^{j-1}$ as $\left(c_{pk_{\mathsf{T}}}^{j-1}, c_{cert}^{j-1}, \pi_{cert}^{j-1}, c_{sn}^{j-1}, \pi_{sn}^{j-1}, \underline{c_{tag}^{j-1}}, \underline{\pi_{tag}^{j-1}}, c_M, c_\sigma^{j-1}, \pi_\sigma^{j-1},\right.$
  $\left.\tilde{c}_{sn}^{j-1}, \tilde{\pi}_{sn}^{j-1}, \underline{\tilde{c}_{tag}^{j-1}}, \underline{\tilde{\pi}_{tag}^{j-1}}\right)$
$pk_{\mathsf{T}} := \mathsf{C.Extr}\left(xk, c_{pk_{\mathsf{T}}}^{j-1}\right)$
$\sigma := \mathsf{C.Extr}\left(xk, c_{cert}^{j-1}\right)$
If $pk_{\mathsf{T}} \notin \mathcal{UL}$:
  Then return $(pk_{\mathsf{T}}, \sigma)$
Else abort

We let $\epsilon$ identify the part of the secret key that could be ignored in the protocols (as the certificate *cert* of a receiver is never used). Let $E'_{\text{sig}}$ be the event that $\mathcal{B}_2$ breaks the unforgeability of $\mathsf{S}'$.

We construct a second adversary against soundness of $\mathsf{C}$:

Adversary $\mathcal{B}_4^{\mathcal{A}}(ck)$:
  Extract $Gr$ from $ck$
  $par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
  $par_{\mathsf{S}'} \leftarrow \mathsf{S}'.\mathsf{Setup}(Gr)$
  $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
  $par \leftarrow \left(1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck\right)$
  $(pk_{\mathcal{B}} = (ek_{\text{init}}, ek, vk, vk'), sk_{\mathcal{W}} = (sk, sk'), sk_{\mathcal{D}}, sk_{\mathcal{CK}}) \leftarrow \mathsf{BKeyGen}()$
  Execute $\mathcal{A}^{\mathtt{BRegist},\mathtt{BWith},\mathtt{BDepo}}(par, pk_{\mathcal{B}})$
  Let $(pk, \Pi)$ be the output of the last call to `BDepo`
  If a such pair is never returned by `BDepo`, then abort
  Let $c$ be the last coin sent by the user and $i$ be its size
  Parse $c$ as $\left(c^0, \left(c^k\right)_{k=1}^i, n, sn, \rho_{sn}, \rho_{pk}\right)$
  Let $j$ be minimal such that $(\overrightarrow{sn})_{(j-1)} \neq (\overrightarrow{sn'})_{(j-1)}$
    (with the same notation as in $\mathsf{CheckDS}$)
  Parse $c^{(j-1)}$ as $\left(c_{pk}^{(j-1)}, c_{cert}^{(j-1)}, \pi_{cert}^{(j-1)}, c_{sn}^{(j-1)}, \pi_{sn}^{(j-1)}, \underline{c_{tag}^{(j-1)}}, \underline{\pi_{tag}^{(j-1)}}, c_M,\right.$
    $\left.c_\sigma^{(j-1)}, \pi_\sigma^{(j-1)}, \tilde{c}_{sn}^{(j-1)}, \tilde{\pi}_{sn}^{(j-1)}, \underline{\tilde{c}_{tag}^{(j-1)}}, \underline{\tilde{\pi}_{tag}^{(j-1)}}\right)$

Do the same for all $k \in \{0, \ldots, i\}$
If $j \neq 1$, then parse $\pi_{sn}^{(j-1)}$ as $\left(\pi_{sn,valid}^{(j-1)}, c_{sn\text{-}pf}^{(j-1)}\right)$
  Else $\left(\pi_{sn,valid}^{(j-1)}, c_{sn\text{-}pf}^{(j-1)}\right) \leftarrow \left(\pi_{sn}^{(j-1)}, c_M\right)$
Parse $\pi_{sn}^{j}$ as $\left(\pi_{sn,valid}^{j}, c_{sn\text{-}pf}^{j}\right)$
Parse $\pi_{tag}^{j}$ as $\left(\pi_{tag,valid}^{j}, c_{t\text{-}pf}^{j}\right)$
For all $k \in \{0, \ldots, i\}$:
  Parse $\tilde{\pi}_{sn}^{k}$ as $\left(c_{\nu_{sn}}^{k}, \pi_{sn,eq}^{k}\right)$
  Parse $\tilde{\pi}_{tag}^{k}$ as $\left(c_{\nu_{tag}}^{k}, \pi_{tag,eq}^{k}\right)$
Let $c'$ be the coin that collides with $c$ and $i'$ be its size
Parse $c'$ as $\left(c'^0, \left(c'^k\right)_{k=1}^{i'}, n', sn', \rho'_{sn}, \rho'_{pk}\right)$
Parse $c'^{(j-1)}$ as $\left(c_{pk}'^{(j-1)}, c_{cert}'^{(j-1)}, \pi_{cert}'^{(j-1)}, c_{sn}'^{(j-1)}, \pi_{sn}'^{(j-1)}, \underline{c_{tag}'^{(j-1)}}, \right.$
$\left. \underline{\pi_{tag}'^{(j-1)}}, c_M'^{(j-1)}, c_\sigma'^{(j-1)}, \pi_\sigma'^{(j-1)}, \tilde{c}_{sn}'^{(j-1)}, \tilde{\pi}_{sn}'^{(j-1)}, \underline{\tilde{c}_{tag}'^{(j-1)}}, \underline{\tilde{\pi}_{tag}'^{(j-1)}}\right)$
Do the same for all $k \in \{0, \ldots, i'\}$
Parse $\pi_{sn}'^{j}$ as $\left(\pi_{sn,valid}'^{j}, c_{sn\text{-}pf}'^{j}\right)$
Parse $\pi_{tag}'^{j}$ as $\left(\pi_{tag,valid}'^{j}, c_{t\text{-}pf}'^{j}\right)$
Parse $\tilde{\pi}_{sn}'^{(j-1)}$ as $\left(c_{\nu_{sn}}'^{(j-1)}, \pi_{sn,eq}'^{(j-1)}\right)$
Parse $\tilde{\pi}_{sn}'^{j}$ as $\left(c_{\nu_{sn}}'^{j}, \pi_{sn,eq}'^{j}\right)$
Parse $\tilde{\pi}_{tag}'^{j}$ as $\left(c_{\nu_{tag}}'^{j}, \pi_{tag,eq}'^{j}\right)$
Return $\Bigg($ $\mathsf{E}'.\mathsf{Verify}(ek_{\text{init}}, Y_0, Y_1, \tilde{c}_{sn}^0) \wedge \bigwedge_{k=1}^{i} \mathsf{E}.\mathsf{Verify}\left(ek, Y_{2k}, Y_{2k+1}, \tilde{c}_{sn}^k\right) \wedge$
$\bigwedge_{k=1}^{i} \mathsf{E}.\mathsf{Verify}\left(ek, Y_{2i+2k}, Y_{2i+2k+1}, \tilde{c}_{tag}^k\right) \wedge$
$\mathsf{S}'.\mathsf{Verify}\left(vk', X_1, X_2\right) = 1 \wedge$
$\mathsf{E}.\mathsf{Verify}\left(ek, X_5, X_6, \tilde{c}_{sn}^{(j-1)}\right) \wedge \mathsf{E}.\mathsf{Verify}\left(ek, X_7, X_8, \tilde{c}_{sn}'^{(j-1)}\right) \wedge$
$\mathsf{E}.\mathsf{Verify}\left(ek, X_9, X_{10}, \tilde{c}_{sn}^{j}\right) \wedge \mathsf{E}.\mathsf{Enc}\left(ek, X_{11}, X_{12}, \tilde{c}_{sn}'^{j}\right) \wedge$
$\mathsf{T}.\mathsf{SVfy}_{\text{all}}\left(X_1, X_5, X_{13}\right) = 1 \wedge$
$\mathsf{T}.\mathsf{SVfy}\left(X_1, X_9, X_{14}\right) = 1 \wedge \mathsf{T}.\mathsf{SVfy}\left(X_3, X_{11}, X_{15}\right) = 1 \wedge$
$\mathsf{E}.\mathsf{Verify}\left(ek, X_{16}, X_{17}, \tilde{c}_{tag}^{j}\right) \wedge \mathsf{E}.\mathsf{Enc}\left(ek, X_{18}, X_{19}, \tilde{c}_{tag}'^{j}\right) \wedge$
$\mathsf{T}.\mathsf{TVfy}\left(X_1, X_5, X_9, X_{16}, X_{20}\right) = 1 \wedge$
$\mathsf{T}.\mathsf{TVfy}\left(X_1, X_7, X_{11}, X_{18}, X_{21}\right) = 1,$
$c_{sn}^0, c_{\nu_{sn}}^0, \ldots, c_{sn}^k, c_{\nu_{sn}}^k, c_{tag}^1, c_{\nu_{tag}}^1, \ldots, c_{tag}^k, c_{\nu_{tag}}^k,$
$c_{pk}^{(j-1)}, c_{cert}^{(j-1)}, c_{pk}^{j}, c_{pk}'^{j}, c_{sn}^{(j-1)}, c_{\nu_{sn}}^{(j-1)}, c_{sn}'^{(j-1)}, c_{\nu_{sn}}'^{(j-1)}, c_{sn}^{j}, c_{\nu_{sn}}^{j}, c_{sn}'^{j}, c_{\nu_{sn}}'^{j},$
$c_{sn\text{-}pf}^{(j-1)}, c_{sn\text{-}pf}^{j}, c_{sn\text{-}pf}'^{j}, c_{tag}^{j}, c_{\nu_{tag}}^{j}, c_{tag}'^{j}, c_{\nu_{tag}}'^{j}, c_{t\text{-}pf}^{j}, c_{t\text{-}pf}'^{j},$
$\bigwedge_{k=0}^{i} \pi_{sn,eq}^{k} \bigwedge_{k=1}^{i} \pi_{tag,eq}^{k} \wedge \pi_{cert}^{(j-1)} \wedge \pi_{sn,eq}^{(j-1)} \wedge \pi_{sn,eq}'^{(j-1)} \wedge \pi_{sn,eq}^{j} \wedge \pi_{sn,eq}'^{j} \wedge$
$\pi_{sn,valid}^{(j-1)} \wedge \pi_{sn,valid}^{j} \wedge \pi_{sn,valid}'^{j} \wedge$
$\left.\pi_{tag,eq}^{j} \wedge \pi_{tag,eq}'^{j} \wedge \pi_{tag,valid}^{j} \wedge \pi_{tag,valid}'^{j}\right)$

Let $E_{\text{com},2}$ and $E_{\text{same-nonce}}$ be the events that $\mathcal{B}_4$ breaks the soundness of $\mathsf{C}$, and a same nonce is picked by the bank during two different calls to `BDepo` respectively.

**Property A.3.** $E_{\text{unforg}} \setminus E_{\text{counterfeit}} \subseteq E_{\text{same}} \cup E_{\text{com},2} \cup E'_{\text{unforg}}$.

Suppose that we are in $E_{\text{unforg}} \setminus (E_{\text{com},2} \cup E_{\text{counterfeit}} \cup E_{\text{same}})$. Because the coin has been accepted, the proofs are correct (as they are verified in the Spend protocol, during a call to BDepo). We are thus in a case where the extracted commitment will verify the equations. Let

$$(sn^0, \ldots, sn^i, tag^1, \ldots, tag^i, pk_{tag}^{(j-1)}, cert^{(j-1)}, pk_{tag}^j, pk_{tag}'^j, sn^{(j-1)}, \nu_{sn}^{(j-1)}, sn'^{(j-1)},$$
$$\nu_{sn}'^{(j-1)}, sn^j, \nu_{sn}^j, sn'^j, \nu_{sn}'^j, sn\text{-}pf^{(j-1)}, sn\text{-}pf^j, sn\text{-}pf'^j, tag^j, \nu_{tag}^j, tag^{j\prime}, \nu_{tag}^{j\prime}, t\text{-}pf^j, t\text{-}pf'^j)$$

be the extraction of the commitments given by $\mathcal{B}_4$ to the challenger of the soundness game. Because we are not in $E_{\text{com},2}$, $\mathcal{B}_4$ loses the game: for all $k \in \{1, \ldots, i\}$:

$$\mathsf{E}'.\mathsf{Verify}\left(ek_{\text{init}}, sn^0, \nu_{sn}^0, \tilde{c}_{sn}^0\right) = \mathsf{E}.\mathsf{Verify}\left(ek, sn^k, \nu_{sn}^k, \tilde{c}_{sn}^k\right) = 1,$$

and for all $k \in \{1, \ldots, i\}$:

$$\mathsf{E}.\mathsf{Verify}\left(ek, tag^k, \nu_{sn}^k, \tilde{c}_{tag}^k\right) = 1.$$

Then by the correctness of $\mathsf{E}$ and $\mathsf{E}'$, we deduce that $E_{\text{Decrypt-fails}}$ will not happen. Because we are in $E_{\text{unforg}} \setminus E_{\text{counterfeit}}$, it means that during the execution of CheckDS, it has been detected that the first serial number of $c$ was the same of another coin (here $c'$). Note that the last $sn$ of a deposited coin is generated (with the key $sk_{\mathsf{T}}$) and encrypted by the bank itself. Then because we are not in $E_{\text{same}}$, CheckDS will find a $j$, such that $\overrightarrow{sn}_j \neq \overrightarrow{sn'}_j$.

By construction $\mathsf{E}.\mathsf{Dec}\,(dk, \tilde{c}_{sn}^j) = \mathsf{E}.\mathsf{Dec}\,(dk, \tilde{c}_{sn}'^j)$, which by correctness of $\mathsf{E}$ (and because we are not in $E_{\text{com, 2}}$) means $sn^{(j-1)} = sn'^{(j-1)}$. Since:

$$\begin{aligned}
1 &= \mathsf{T}.\mathsf{SVfy}(pk_{\mathsf{T}}^j, sn^j, sn\text{-}pf^j) \\
&= \mathsf{T}.\mathsf{SVfy}(pk_{\mathsf{T}}'^j, sn'^j, sn\text{-}pf'^j) \\
&= \mathsf{T}.\mathsf{TVfy}(pk_{\mathsf{T}}^{(j-1)}, sn^{(j-1)}, sn^j, tag^j, t\text{-}pf^j) \\
&= \mathsf{T}.\mathsf{TVfy}(pk_{\mathsf{T}}'^{(j-1)}, sn^{(j-1)}, sn'^j, tag'^j, t\text{-}pf'^j) \\
&= \mathsf{T}.\mathsf{SVfy}_{\text{all}}(pk_{\mathsf{T}}^{(j-1)}, sn^{(j-1)}, sn\text{-}pf^{(j-1)}),
\end{aligned}$$

and, because $\mathsf{T}$ is SN-identifiable, we can deduce that $pk_{tag}^j = pk_{tag}'^j$.

Moreover, since $\mathsf{T}$ is two-extractable, we deduce that if $pk_{tag}^j \in \mathcal{UL}$, $E_{\text{DDSfails}}$, $E_{\text{incorrect}}$ and $E_{\text{not-register}}$ will not happen.

We proved that $(E_{\text{unforg}} \setminus E_{\text{counterfeit}} \cup E_{\text{same}} \cup E_{\text{com, 2}}) \implies pk_{tag}^j \notin \mathcal{UL}$. By noting that if $pk_{tag}^j \notin \mathcal{UL}$, and if $E_{\text{com, 2}} \cup E_{\text{same}} \cup E_{\text{counterfeit}}$ does not happen, $\mathcal{B}_2$ will win the unforgeability game against $\mathsf{S}'$. We finally deduce:

$$E_{\text{unforg}} \setminus (E_{\text{com},2} \cup E_{\text{counterfeit}}) \subseteq E_{\text{same}} \cup E'_{\text{sig}}.$$

□

Now we suppose to be in $E_{\text{same}}$, then by correctness of $\mathsf{E}$, we deduce that the serial numbers were also identical before their encryption. Then by $sn$-injectivity, it means that the nonces picked during the deposits were the same, and we are therefore in $E_{\text{same-nonce}}$. Thus $E_{\text{same}} \subseteq E_{\text{same-nonce}}$. From this inequality we deduce

$$E_{\text{unforg}} \subseteq E_{\text{com},2} \cup E_{\text{same-nonce}} \cup E'_{\text{sig}} \cup E_{\text{com},2} \cup E_{\text{sig}}.$$

By measuring the probability, we can finally conclude:

$$\epsilon \leq \epsilon_{\text{h},1} + \epsilon_{\text{h},2} + \epsilon_{\text{sig}} + \epsilon'_{\text{sig}} + \frac{d^2}{N}.$$

□

Experiment $\mathbf{Expt}_{\mathcal{B}}^{tag-exculpability}(\lambda)$:
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}\left(1^{\lambda}\right)$
$(sk_{tag}, sk_{\mathsf{T}}) \leftarrow \mathsf{T.KeyGen}\left(1^{\lambda}\right)$
$\mathcal{L} := \emptyset$
$\Pi' \leftarrow \mathcal{B}^{O_1(sk_{tag}),O_2(sk_{tag},\cdot)}(sk_{\mathsf{T}})$
Output $\mathsf{T.VfyGuilt}(sk_{\mathsf{T}}, \Pi')$

$O_1(sk)$:
$n \xleftarrow{\$} \mathcal{N}$; $T[k] := n$; $k := k + 1$
$(sn, sn\text{-}pf) \leftarrow \mathsf{T.SGen}(sk, n)$
Return $sn$

$O_2(sk, sn', i)$:
If $T[i] = \perp$, abort the oracle call
$n := T[i]$ $T[i] := \perp$
$(tag, t\text{-}pf) \leftarrow \mathsf{T.TGen}(sk, n, sn')$
Return $tag$

Figure A.1: Game for *tag exculpability* for double-spending tag schemes

## Exculpability

**Theorem A.4.** *Suppose there is an adversary $\mathcal{A}$ against the exculpability of a double-spender with advantage $\epsilon$ that uses $u$ calls to the oracle* `URegist`*. Then there exist adversaries $\mathcal{B}_1, \mathcal{B}_2$ against tag-exculpability with advantage $\epsilon_{tag}$, and against mode-hiding of $\mathsf{C}$ with advantage $\epsilon_{\text{m-ind}}$ respectively such that*

$$\epsilon \leq u\epsilon_{tag} + \epsilon_{\text{m-ind}}.$$

We start with recalling the tag-exculpability game in Fig. A.1. We construct the following adversary to break the tag-exculpability of $\mathsf{T}$.

Adversary $\mathcal{B}_1^{O_1(sk_{tag}),O_2(sk_{tag},\cdot)}(par_{\mathsf{T}}, pk_{\mathsf{T}})$:
$(ck, td) \leftarrow \mathsf{C.SmSetup}\left(1^{\lambda}\right)$
$par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}\left(1^{\lambda}\right)$
$par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}\left(1^{\lambda}\right)$
$par \leftarrow \left(1^{\lambda}, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck\right)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$u^* \xleftarrow{\$} \{1, \ldots, u\}$
$(i^*, \Pi^*) \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy},\texttt{UWith},\texttt{Rcv},\texttt{Spd},\texttt{S\&R},\texttt{UDepo}}(par, pk_{\mathcal{B}})$
At the $u^{*th}$ call of `URegist`, choose $pk_{\mathsf{T}}$ and do not use $\mathsf{T.KeyGen}$
If the adversary queries $\texttt{Spy}(u^*)$, abort
If the adversary queries `UWith`, `Rcv`, `Spd`, `S&R`, `UDepo` on $u^*$,
use $O_1$ and $O_2$, and $td$ (since $sk_{tag}$ is unknown)
If $O_2$ fails, abort the entire procedure
Output $\Pi^*$

The game is perfectly simulated from $\mathcal{A}$'s point of view, except when it calls $\texttt{Spy}(u^*)$, or makes that user double-spend, or if it detects that we are in hiding-mode (which happens with probability at most $\epsilon_{\text{m-ind}}$). Let $E_{\text{ex}}$ and $E_{tag}$ be the events that $\mathcal{A}$ wins and that $\mathcal{B}_1$ wins respectively. Suppose that we are in $E_{\text{ex}}$. That means $\mathcal{A}$ forges a proof against one of the user registered (and does not spy on her). The probability that this user is $u^*$ is at least $\frac{1}{u}$. And in this case, the following holds:

- $\mathcal{A}$ did not spy on $u^*$ or make her double-spend (in both cases we would not be in $E_{\text{ex}}$).
- $\mathsf{VfyGuilt}(pk_{\mathsf{T}}, \Pi^*) = 1$, (because we are in $E_{\text{ex}}$) we thus have $\mathsf{T.VfyGuilt}(sk_{\mathsf{T}}, \Pi^*) = 1$.

We thus deduce that

$$\Pr(E_{\text{ex}}) \leq u \Pr(E_{tag}).$$

□

### Coin anonymity (full proof)

**Theorem A.5.** *Let $\mathcal{A}$ be an adversary that wins the* **coin anonymity** *game (*c-an*) with advantage $\epsilon$ and let $k$ be an upper-bound on the number of users transferring the challenge coins. Then there exist adversaries against mode-indistinguishability of* C *and tag-anonymity of* T *with advantages $\epsilon_{\text{m-ind}}$ and $\epsilon_{\text{t-an}}$, resp., such that*

$$\epsilon \leq 2\left(\epsilon_{\text{m-ind}} + (k+1)\epsilon_{\text{t-an}}\right).$$

**Sketch of proof.** In the proof, we first define a hybrid game in which the commitment key is switched to hiding mode (hence the loss $\epsilon_{\text{m-ind}}$, which occurs twice for $b = 0$ and $b = 1$). All commitments are then perfectly hiding and the only information available to the adversary are the serial numbers and tags. (They are encrypted in the coin, but the adversary, impersonating the bank, can decrypt them.)

We then argue that, by tag anonymity of T, the adversary cannot link a user to a pair $(sn, tag)$, even when it knows the users' secret keys. We define a sequence of $k+1$ hybrid games (as $k$ transfers involve $k+1$ users); going through the user vector output by the adversary, we can switch, one-by-one all users from the first two the second vector. Each switching can be detected by the adversary with probability at most $2\epsilon_{\text{t-an}}$.
There is a technical difficulty to manage the first swap: We still don't know the second tag secret key to withdraw the first coin. Fortunately in the hiding mode, we don't send any ciphertext to the bank, but only commitments and zero-knowledge proof.

**Full proof.** We recall $\mathbf{Expt}^{\texttt{c-an}}_{\mathcal{A},0}$:

$\mathbf{Expt}^{\texttt{c-an}}_{\mathcal{A},0}(\lambda)$:
$par \leftarrow \mathsf{ParamGen}(1^\lambda)$; $pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$i_0 \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
Run $\texttt{UWith}(i_0)$ with $\mathcal{A}$
$i_1 \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
Run $\texttt{UWith}(i_1)$ with $\mathcal{A}$
$(\overrightarrow{i^{(0)}}, \overrightarrow{i^{(1)}}) \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
Let $k := |\overrightarrow{i^{(0)}}|$; if $k \neq |\overrightarrow{i^{(1)}}|$, abort the entire procedure
Then repeat the following step for $j = 1, \ldots, k$:
Run $\texttt{S\&R}(2j-1, (\overrightarrow{i^{(0)}})_j)$; Run $\texttt{S\&R}(2j, (\overrightarrow{i^{(1)}})_j)$
Run $\texttt{Spd}(2k+1+b)$ with $\mathcal{A}$
Run $\texttt{Spd}(2k+2-b)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

In the game $\mathbf{Expt}^{\texttt{c-an}}_{\mathcal{A},0,\text{hiding}}$, we will change the commitment key. If the adversary will detect it, it will break the mode-indistinguishability of C, then the distribution of the experiment will not change except with probability $\epsilon_{\text{m-ind}}$ (property A.6).

Experiment $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{hiding}}(\lambda)$:

$\boxed{Gr \leftarrow \mathsf{BGGen}(1^\lambda)}$
$\boxed{par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)}$
$\boxed{par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)}$
$\boxed{par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}(Gr)}$
$\boxed{(ck, td) \leftarrow \mathsf{C.SmSetup}(Gr)}$
$par \leftarrow \boxed{(1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)}$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$i_0 \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
Run $\mathtt{UWith}(i_0)$ with $\mathcal{A}$
$i_1 \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
Run $\mathtt{UWith}(i_1)$ with $\mathcal{A}$
$(\overrightarrow{i^{(0)}}, \overrightarrow{i^{(1)}}) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
Let $k := |\overrightarrow{i^{(0)}}|$; if $k \neq |\overrightarrow{i^{(1)}}|$, abort the entire procedure
Then repeat the following step for $j = 1, \ldots, k$:
Run $\mathtt{S\&R}(2j-1, (\overrightarrow{i^{(0)}})_j)$; Run $\mathtt{S\&R}(2j, (\overrightarrow{i^{(1)}})_j)$
Run $\mathtt{Spd}(2k+1+b)$ with $\mathcal{A}$
Run $\mathtt{Spd}(2k+2-b)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

**Property A.6.** $|\Pr(\mathrm{Expt}^{\text{c-an}}_{\mathcal{A},0}(\lambda)) - \Pr(\mathrm{Expt}^{\text{c-an}}_{\mathcal{A},0,\mathrm{hiding}}(\lambda))| \leq \epsilon_{\text{m-ind}}$.

We never use $td$ in $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{hiding}}(\lambda)$. Therefore, if we use a challenge $(Gr, ck)$, in a $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0}(\lambda)$ game. If $ck$ has been generated by $\mathsf{C.Setup}$, it will be exactly $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{hiding}}(\lambda)$, if $ck$ has been generated by $\mathsf{C.SmSetup}$, the experiment will be exactly $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{hiding}}$. This experiment can therefore be seen as a *mode-distinguisher*. □

We now define $\mathsf{C.SmPrv}_{\mathrm{sn}}, \mathsf{C.SmPrv}_{\mathrm{sn,init}}, \mathsf{C.SmPrv}_{\mathrm{tag}}, \mathsf{C.E.SmPrv}_{\mathrm{enc}}$ as analogues of $\mathsf{C.Prv}_{\mathrm{sn}}$, $\mathsf{C.Prv}_{\mathrm{sn,init}}$, $\mathsf{C.Prv}_{\mathrm{tag}}$, $\mathsf{C.E.Prv}_{\mathrm{enc}}$, except that we substitute every $\mathsf{C.Prv}$ by $\mathsf{C.SmPrv}$ and every $\mathsf{C.Cm}$ by $\mathsf{C.ZCm}$.

Now each time the challenger is using an oracle, it uses $\mathsf{C.ZCm}$ instead of $\mathsf{C.Cm}$, $\mathsf{C.SmPrv}$ instead of $\mathsf{C.Prv}$, $\mathsf{C.SmPrv}_{\mathrm{enc}}$ instead of $\mathsf{C.Prv}_{\mathrm{enc}}$ etc

We call these new oracles: $\mathtt{S\&R}_{\mathrm{ZK}}, \mathtt{UWith}_{\mathrm{ZK}}, \mathtt{Spd}_{\mathrm{ZK}}$.

Experiment $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{ZK}}(\lambda)$:

$Gr \leftarrow \mathsf{BGGen}(1^\lambda)$
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
$par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
$par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}(Gr)$
$(ck, td) \leftarrow \mathsf{C.SmSetup}(Gr)$
$par \leftarrow (1^\lambda, Gr, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$i_0 \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
Run $\boxed{\mathtt{UWith}_{\mathrm{ZK}}(i_0)}$ with $\mathcal{A}$
$i_1 \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
Run $\boxed{\mathtt{UWith}_{\mathrm{ZK}}(i_1)}$ with $\mathcal{A}$

$(\overrightarrow{i^{(0)}}, \overrightarrow{i^{(1)}}) \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
Let $k := |\overrightarrow{i^{(0)}}|$; if $k \neq |\overrightarrow{i^{(1)}}|$, abort the entire procedure
Then repeat the following step for $j = 1, \ldots, k$:
Run $\boxed{\texttt{S\&R}_{\text{ZK}}}$ $(2j-1, (\overrightarrow{i^{(0)}})_j)$; Run $\boxed{\texttt{S\&R}_{\text{ZK}}}$ $(2j, (\overrightarrow{i^{(1)}})_j)$
Run $\boxed{\texttt{Spd}_{\text{ZK}}}(2k+1+b)$ with $\mathcal{A}$
Run $\boxed{\texttt{Spd}_{\text{ZK}}}(2k+2-b)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

Because, we are in the hiding mode the following property follows directly of the *perfect zero-knowledge in hiding mode*:

**Property A.7.** $|\Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{hiding}}(\lambda)) - \Pr(\text{Expt}^{\texttt{c-an}}_{\mathcal{A},0,\text{ZK}}(\lambda))| = 0.$

First we consider the following part of $\text{Expt}^{\texttt{c-an}}_{\mathcal{A},0,\text{ZK}}(\lambda))$ with more details:

$i_0 \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
Run $\texttt{UWith}_{\text{ZK}}$ $(i_0)$ with $\mathcal{A}$
$i_1 \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
Run $\texttt{UWith}_{\text{ZK}}(i_1)$ with $\mathcal{A}$

We would like to swap the serial numbers of $i_0$ and $i_1$ by using tag-anonymity. The issue here is that in the first call to $\texttt{UWith}_{\text{ZK}}$, we do not know yet $i_1$ (because it is chosen in a second round). Fortunately, at this step we only sent $\mathcal{A}$ data that is unrelated to this serial number, because we are using $\mathsf{ZCm}$. Thus, at the end of this part, we can compute the ciphertexts of both initial coins.

We can decompose this part of the game as:

$i_0 \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
$n^{(0)} \xleftarrow{\$} \mathcal{N}$; $\rho_{sn}^{(0)}, \rho_{cert}^{(0)}, \rho_{pk}^{(0)}, \rho_M^{(0)} \xleftarrow{\$} \mathcal{R}$
$(sn^{(0)}, M_{sn}^{(0)}) \leftarrow \mathsf{T.SGen}_{\text{init}}(sk_{i_0}, n^{(0)})$
$c_{cert}^{(0)}, c_{sn}^{(0)}, c_{pk}^{(0)}, c_M^{(0)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{cert}^{(0)}, \rho_{sn}^{(0)}, \rho_{pk}^{(0)}, \rho_M^{(0)})$
$\pi_{cert}^{(0)} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S'.Verify}(vk', \cdot, \cdot) = 1, \rho_{pk}^{(0)}, \rho_{cert}^{(0)})$
$\pi_{sn}^{(0)} \leftarrow \mathsf{C.SmPrv}_{sn,\text{init}}(td, \rho_{pk}^{(0)}, \rho_{sn}^{(0)}, \rho_M^{(0)})$
Send $(c_{pk}^{(0)}, c_{cert}^{(0)}, \pi_{cert}^{(0)}, c_{sn}^{(0)}, c_M^{(0)}, \pi_{sn}^{(0)})$ to $\mathcal{A}$
Receive $(c_\sigma^{(0)}, \pi_\sigma^{(0)})$ from $\mathcal{A}$
Compute $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M^{(0)}, c_\sigma^{(0)}, \pi_\sigma)$
If it fails, output $\perp$ else continue; $\nu_{sn}^{(0)} \xleftarrow{\$} \mathcal{R}$
$\tilde{c}_{sn}^{(0)} \leftarrow \mathsf{E.Enc}(ek, sn^{(0)}, \nu_{sn}^{(0)})$
$\tilde{\pi}_{sn}^{(0)} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{sn}^{(0)}, \tilde{c}_{sn}^{(0)})$
Pick $\overrightarrow{\rho^{(0)\prime}}$ long enough to compute:
$c_1^{(0)} = (\mathsf{Rand}((c_{pk}^{(0)}, c_{cert}^{(0)}, \pi_{cert}^{(0)}, c_{sn}^{(0)}, \pi_{sn}^{(0)}, c_M^{(0)}, c_\sigma^{(0)}, \pi_\sigma^{(0)}, \tilde{c}_{sn}^{(0)}, \tilde{\pi}_{sn}^{(0)}), \overrightarrow{\rho^{(0)\prime}}),$
$n^{(0)}, sn^{(0)}, \rho_{sn}^{(0)} + (\overrightarrow{\rho^{(0)\prime}})_{sn}, \rho_{pk}^{(0)} + (\overrightarrow{\rho^{(0)\prime}})_{pk})$
$\mathcal{CL} \leftarrow [(i_0, c_1^{(0)}, 0, \mathcal{A})]$
$i_1 \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
$n^{(1)} \xleftarrow{\$} \mathcal{N}$; $\rho_{sn}^{(1)}, \rho_{cert}^{(1)}, \rho_{pk}^{(1)}, \rho_M^{(1)} \xleftarrow{\$} \mathcal{R}$, and compute:

$(sn^{(1)}, M_{sn}^{(1)}) \leftarrow \mathsf{T.SGen}_{\text{init}}(sk_{i_1}, n^{(1)})$
$c_{cert}^{(1)}, c_{sn}^{(1)}, c_{pk}^{(1)}, c_M^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{cert}^{(1)}, \rho_{sn}^{(1)}, \rho_{pk}^{(1)}, \rho_M^{(1)})$
$\pi_{cert}^{(1)} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S'.Verify}(vk', \cdot, \cdot) = 1, \rho_{pk}^{(1)}, \rho_{cert}^{(1)})$
$\pi_{sn}^{(1)} \leftarrow \mathsf{C.SmPrv}_{sn,\text{init}}(td, \rho_{pk}^{(1)}, \rho_{sn}^{(1)}, \rho_M^{(1)})$
Send $(c_{pk}^{(1)}, c_{cert}^{(1)}, \pi_{cert}^{(1)}, c_{sn}^{(1)}, c_M^{(1)}, \pi_{sn}^{(1)})$ to $\mathcal{A}$
We receive $(c_\sigma^{(1)}, \pi_\sigma^{(1)})$ from $\mathcal{A}$
Compute $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M^{(1)}, c_\sigma^{(1)}, \pi_\sigma^{(1)})$
If fails, output $\perp$ else continue; $\nu_{sn}^{(1)} \xleftarrow{\$} \mathcal{R}$
$\tilde{c}_{sn}^{(1)} \leftarrow \mathsf{E.Enc}(ek, sn^{(1)}, \nu_{sn}^{(1)})$
$\tilde{\pi}_{sn}^{(1)} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{sn}^{(1)}, \tilde{c}_{sn}^{(1)})$
Pick $\overrightarrow{\rho^{(1)\prime}}$ long enough to compute the following and output it:
$c_1^{(1)} = (\mathsf{Rand}((c_{pk}^{(1)}, c_{cert}^{(1)}, \pi_{cert}^{(1)}, c_{sn}^{(1)}, \pi_{sn}^{(1)}, c_M^{(1)}, c_\sigma^{(1)}, \pi_\sigma^{(1)}, \tilde{c}_{sn}^{(1)}, \tilde{\pi}_{sn}^{(1)}), \overrightarrow{\rho^{(1)\prime}}),$
$n^{(1)}, sn^{(1)}, \rho_{sn}^{(1)} + (\overrightarrow{\rho^{(1)\prime}})_{sn}, \rho_{pk}^{(1)} + (\overrightarrow{\rho^{(1)\prime}})_{pk})$
$\mathcal{CL}[2] \leftarrow (i_1, c_1^{(1)}, 0, \mathcal{A})$;

We can do the *sn*-computations and the encryptions at the end of this part (because they are not related to data sent to $\mathcal{A}$). We can therefore replace the previous instructions by the following algorithm `DoubleUWith`:

$\texttt{DoubleUWith}^{\mathcal{A}}$:
$i_0 \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
$\rho_{sn}^{(0)}, \rho_{cert}^{(0)}, \rho_{pk}^{(0)}, \rho_M^{(0)} \xleftarrow{\$} \mathcal{R}$; Compute:
$c_{cert}^{(0)}, c_{sn}^{(0)}, c_{pk}^{(0)}, c_M^{(0)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{cert}^{(0)}, \rho_{sn}^{(0)}, \rho_{pk}^{(0)}, \rho_M^{(0)})$
$\pi_{cert}^{(0)} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S'.Verify}(vk', \cdot, \cdot) = 1, \rho_{pk}^{(0)}, \rho_{cert}^{(0)})$
$\pi_{sn}^{(0)} \leftarrow \mathsf{C.SmPrv}_{sn,\text{init}}(td, \rho_{pk}^{(0)}, \rho_{sn}^{(0)}, \rho_M^{(0)})$
Send $(c_{pk}^{(0)}, c_{cert}^{(0)}, \pi_{cert}^{(0)}, c_{sn}^{(0)}, c_M^{(0)}, \pi_{sn}^{(0)})$ to $\mathcal{A}$
Receive $(c_\sigma^{(0)}, \pi_\sigma^{(0)})$ from $\mathcal{A}$
If $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M^{(0)}, c_\sigma^{(0)}, \pi_\sigma)$
fails, then output $\perp$ else continue
$i_1 \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
$\rho_{sn}^{(1)}, \rho_{cert}^{(1)}, \rho_{pk}^{(1)}, \rho_M^{(1)} \xleftarrow{\$} \mathcal{R}$; Compute:
$c_{cert}^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{cert}^{(1)})$
$c_{sn}^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{sn}^{(1)})$
$c_{pk}^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{pk}^{(1)})$
$c_M^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_M^{(1)})$
$\pi_{cert}^{(1)} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S'.Verify}(vk', \cdot, \cdot) = 1, \rho_{pk}^{(1)}, \rho_{cert}^{(1)})$
$\pi_{sn}^{(1)} \leftarrow \mathsf{C.SmPrv}_{sn,\text{init}}(td, \rho_{pk}^{(1)}, \rho_{sn}^{(1)}, \rho_M^{(1)})$
Send $(c_{pk}^{(1)}, c_{cert}^{(1)}, \pi_{cert}^{(1)}, c_{sn}^{(1)}, c_M^{(1)}, \pi_{sn}^{(1)})$ to $\mathcal{A}$
Receive $(c_\sigma^{(1)}, \pi_\sigma^{(1)})$ from $\mathcal{A}$
If $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M^{(1)}, c_\sigma^{(1)}, \pi_\sigma^{(1)})$
fails, then output $\perp$
$n^{(0)}, n^{(1)} \xleftarrow{\$} \mathcal{N}$; $(sn^{(0)}, M_{sn}^{(0)}) \leftarrow \mathsf{T.SGen}_{\text{init}}(sk_{i_0}, n^{(0)})$
$(sn^{(1)}, M_{sn}^{(1)}) \leftarrow \mathsf{T.SGen}_{\text{init}}(sk_{i_1}, n^{(1)})$

$\nu_{sn}^{(0)}, \nu_{sn}^{(1)} \xleftarrow{\$} \mathcal{R}$; Execute:
$\tilde{c}_{sn}^{(0)} \leftarrow \mathsf{E.Enc}(ek, sn^{(0)}, \nu_{sn}^{(0)})$; $\tilde{c}_{sn}^{(1)} \leftarrow \mathsf{E.Enc}(ek, sn^{(1)}, \nu_{sn}^{(1)})$
$\tilde{\pi}_{sn}^{(0)} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{sn}^{(0)}, \tilde{c}_{sn}^{(0)})$
$\tilde{\pi}_{sn}^{(1)} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{sn}^{(1)}, \tilde{c}_{sn}^{(1)})$
Pick uniformly at random $\overrightarrow{\rho^{(0)\prime}}, \overrightarrow{\rho^{(1)\prime}}$ long enough to compute:
$c_1^{(0)} = (\mathsf{Rand}((c_{pk}^{(0)}, c_{cert}^{(0)}, \pi_{cert}^{(0)}, c_{sn}^{(0)}, \pi_{sn}^{(0)}, c_M^{(0)}, c_\sigma^{(0)}, \pi_\sigma^{(0)}, \tilde{c}_{sn}^{(0)}, \tilde{\pi}_{sn}^{(0)}), \overrightarrow{\rho^{(0)\prime}}),$
$\quad n^{(0)}, sn^{(0)}, \rho_{sn}^{(0)} + (\overrightarrow{\rho^{(0)\prime}})_{sn}, \rho_{pk}^{(0)} + (\overrightarrow{\rho^{(0)\prime}})_{pk})$
$c_1^{(1)} = (\mathsf{Rand}((c_{pk}^{(1)}, c_{cert}^{(1)}, \pi_{cert}^{(1)}, c_{sn}^{(1)}, \pi_{sn}^{(1)}, c_M^{(1)}, c_\sigma^{(1)}, \pi_\sigma^{(1)}, \tilde{c}_{sn}^{(1)}, \tilde{\pi}_{sn}^{(1)}), \overrightarrow{\rho^{(1)\prime}}),$
$\quad n^{(1)}, sn^{(1)}, \rho_{sn}^{(1)} + (\overrightarrow{\rho^{(1)\prime}})_{sn}, \rho_{pk}^{(1)} + (\overrightarrow{\rho^{(1)\prime}})_{pk})$
$\mathcal{CL}[1] \leftarrow (i_0, c_1^{(0)}, 0, \mathcal{A})$
$\mathcal{CL}[2] \leftarrow (i_1, c_1^{(1)}, 0, \mathcal{A})$
Return $(i_0, i_1)$

To express these swaps of instruction, we define the following game.

Experiment $\mathbf{Expt}^{\texttt{c-an}}_{\mathcal{A},0,\text{ZKV2}}(\lambda)$:
$Gr \leftarrow \mathsf{BGGen}(1^\lambda)$
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
$par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
$par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}(Gr)$
$(ck, td) \leftarrow \mathsf{C.SmSetup}(Gr)$
$par \leftarrow (1^\lambda, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$\boxed{(i_0, i_1) \leftarrow \texttt{DoubleUWith}^{\mathcal{A}}}$
$(\overrightarrow{i^{(0)}}, \overrightarrow{i^{(1)}}) \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
Let $k := |\overrightarrow{i^{(0)}}|$; if $k \neq |\overrightarrow{i^{(1)}}|$, abort the entire procedure
Then repeat the following step for $j = 1, \ldots, k$:
$\quad$ Run $\texttt{S\&R}_{\text{ZK}}$ $(2j-1, (\overrightarrow{i^{(0)}})_j, )$; Run $\texttt{S\&R}_{\text{ZK}}$ $(2j, (\overrightarrow{i^{(1)}})_j)$
Run $\texttt{Spd}_{\text{ZK}}$ $(2k+1+b)$ with $\mathcal{A}$
Run $\texttt{Spd}_{\text{ZK}}$ $(2k+2-b)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

And because this swap is transparent for the adversary, it implies the following game swap:

**Property A.8.** $|\Pr(\text{Expt}^{\texttt{c-an}}_{\mathcal{A},0,\text{ZK}}(\lambda)) - \Pr(\text{Expt}^{\texttt{c-an}}_{\mathcal{A},0,\text{ZKV2}}(\lambda))| = 0.$

Now we have to swap the serial numbers. To do that we define two new procedures:

$\texttt{DoubleUWith}^{\mathcal{A}}_{\text{rev}}$:
$i_0 \leftarrow \mathcal{A}^{\texttt{URegist,Spy}}$
$\rho_{sn}^{(0)}, \rho_{cert}^{(0)}, \rho_{pk}^{(0)}, \rho_M^{(0)} \xleftarrow{\$} \mathcal{R}$; Compute:
$\quad c_{cert}^{(0)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{cert}^{(0)})$
$\quad c_{sn}^{(0)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{sn}^{(0)})$; $c_{pk}^{(0)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{pk}^{(0)})$
$\quad c_M^{(0)} \leftarrow \mathsf{C.ZCm}(ck, \rho_M^{(0)})$
$\quad \pi_{cert}^{(0)} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S}'\mathsf{.Verify}(vk', \cdot, \cdot) = 1, \rho_{pk}^{(0)}, \rho_{cert}^{(0)})$

$\pi_{sn}^{(0)} \leftarrow \mathsf{C.SmPrv}_{sn,\text{init}}(td, \rho_{pk}^{(0)}, \rho_{sn}^{(0)}, \rho_M^{(0)})$

Send $(c_{pk}^{(0)}, c_{cert}^{(0)}, \pi_{cert}^{(0)}, c_{sn}^{(0)}, c_M^{(0)}, \pi_{sn}^{(0)})$ to $\mathcal{A}$

Receive $(c_\sigma^{(0)}, \pi_\sigma^{(0)})$ from $\mathcal{A}$

If $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M^{(0)}, c_\sigma^{(0)}, \pi_\sigma)$ fails, then output $\perp$, else continue

$i_1 \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$

$\rho_{sn}^{(1)}, \rho_{cert}^{(1)}, \rho_{pk}^{(1)}, \rho_M^{(1)} \xleftarrow{\$} \mathcal{R}$; Compute:

$c_{cert}^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{cert}^{(1)})$

$c_{sn}^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{sn}^{(1)})$

$c_{pk}^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_{pk}^{(1)})$

$c_M^{(1)} \leftarrow \mathsf{C.ZCm}(ck, \rho_M^{(1)})$

$\pi_{cert}^{(1)} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S'.Verify}(vk', \cdot, \cdot) = 1, \rho_{pk}^{(1)}, \rho_{cert}^{(1)})$

$\pi_{sn}^{(1)} \leftarrow \mathsf{C.SmPrv}_{sn,\text{init}}(td, \rho_{pk}^{(1)}, \rho_{sn}^{(1)}, \rho_M^{(1)})$

Send $(c_{pk}^{(1)}, c_{cert}^{(1)}, \pi_{cert}^{(1)}, c_{sn}^{(1)}, c_M^{(1)}, \pi_{sn}^{(1)})$ to $\mathcal{A}$

We receive $(c_\sigma^{(1)}, \pi_\sigma^{(1)})$ from $\mathcal{A}$

If $\mathsf{C.Verify}(ck, \mathsf{S.Verify}(vk, \cdot, \cdot) = 1, c_M^{(1)}, c_\sigma^{(1)}, \pi_\sigma^{(1)})$ rejects then output $\perp$ else continue

$n^{(0)}, n^{(1)} \xleftarrow{\$} \mathcal{N}$; $(sn^{(0)}, M_{sn}^{(0)}) \leftarrow \mathsf{T.SGen}_{\text{init}}(\boxed{sk_{i_1}}, n^{(0)})$

$(sn^{(1)}, M_{sn}^{(1)}) \leftarrow \mathsf{T.SGen}_{\text{init}}(\boxed{sk_{i_0}}, n^{(1)})$

$\nu_{sn}^{(1)}, \nu_{sn}^{(0)} \xleftarrow{\$} \mathcal{R}$; Execute:

$\tilde{c}_{sn}^{(0)} \leftarrow \mathsf{E.Enc}(ek, sn^{(0)}, \nu_{sn}^{(0)})$; $\tilde{c}_{sn}^{(1)} \leftarrow \mathsf{E.Enc}(ek, sn^{(0)}, \nu_{sn}^{(1)})$

$\tilde{\pi}_{sn}^{(1)} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{sn}^{(1)}, \tilde{c}_{sn}^{(1)})$

$\tilde{\pi}_{sn}^{(0)} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{sn}^{(0)}, \tilde{c}_{sn}^{(0)})$

Pick uniformly at random $\overrightarrow{\rho^{(0)\prime}}, \overrightarrow{\rho^{(1)\prime}}$ long enough to compute:

$c_1^{(0)} = \Big(\mathsf{Rand}((c_{pk}^{(0)}, c_{cert}^{(0)}, \pi_{cert}^{(0)}, c_{sn}^{(0)}, \pi_{sn}^{(0)}, c_M^{(0)}, c_\sigma^{(0)}, \pi_\sigma^{(0)}, \tilde{c}_{sn}^{(0)}, \tilde{\pi}_{sn}^{(0)}), \overrightarrow{\rho^{(0)\prime}}),$
$n^{(0)}, sn^{(0)}, \rho_{sn}^{(0)} + (\overrightarrow{\rho^{(0)\prime}})_{sn}, \rho_{pk}^{(0)} + (\overrightarrow{\rho^{(0)\prime}})_{pk}\Big)$;

$c_1^{(1)} = (\mathsf{Rand}((c_{pk}^{(1)}, c_{cert}^{(1)}, \pi_{cert}^{(1)}, c_{sn}^{(1)}, \pi_{sn}^{(1)}, c_M^{(1)}, c_\sigma^{(1)}, \pi_\sigma^{(1)}, \tilde{c}_{sn}^{(1)}, \tilde{\pi}_{sn}^{(1)}), \overrightarrow{\rho^{(1)\prime}}),$
$n^{(1)}, sn^{(1)}, \rho_{sn}^{(1)} + (\overrightarrow{\rho^{(1)\prime}})_{sn}, \rho_{pk}^{(1)} + (\overrightarrow{\rho^{(1)\prime}})_{pk})$;

$\mathcal{CL}[1] \leftarrow (1, i_0, c_1^{(0)}, 0, \mathcal{A})$

$\mathcal{CL}[2] \leftarrow (i_1, c_1^{(1)}, 0, \mathcal{A})$;

$\mathtt{S\&R}_{\text{ZK,inv}}(j, i, sk_1, sk_2)$:

$c \leftarrow \mathcal{CL}[j].c$

$n' \xleftarrow{\$} \mathcal{N}$ ; $\rho'_{sn}, \rho'_{cert}, \rho'_{pk}, \rho'_{P_{sn}}, \nu'_{sn} \xleftarrow{\$} \mathcal{R}$; Compute:

$(sn', P'_{sn}) \leftarrow \mathsf{T.SGen}(par_{\mathsf{T}}, \boxed{sk_2}, n')$

$c'_{cert}, c'_{pk}, c'_{sn}, c'_{P_{sn}} \leftarrow \mathsf{C.ZCm}(ck, \rho'_{cert}, \rho'_{pk}, \rho'_{sn}, \rho'_{P_{sn}})$

$\tilde{c}'_{sn} \leftarrow \mathsf{E.Enc}(ek, sn', \nu'_{sn})$

$\pi'_{cert} \leftarrow \mathsf{C.SmPrv}(td, \mathsf{S.Verify}(vk_{\mathcal{B}}, \cdot, \cdot) = 1, \rho'_{vk}, \rho'_{pk}, \rho'_{cert})$

$\pi'_{sn} \leftarrow \mathsf{C.SmPrv}_{sn}(td, pk'_{tag}, sn', P_{sn}, \rho'_{pk}, \rho'_{sn}, \rho'_{P_{sn}})$

$\tilde{\pi}'_{sn} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho'_{sn}, \tilde{c}'_{sn})$

Parse $c$ as

$(c^0, (c^j = (c_{pk}^j, c_{cert}^j, \pi_{cert}^j, c_{sn}^j, \pi_{sn}^j, c_{tag}^j, \pi_{tag}^j, \tilde{c}_{sn}^j, \tilde{c}_{tag}^j, \tilde{\pi}_{sn}^j, \tilde{\pi}_{tag}^j))_{j=1}^i,$

$n, sn, \rho_{sn}, \rho_{pk})$
$\rho_{tag}, \nu_{tag}, \rho_{P_{tag}} \xleftarrow{\$} \mathcal{R}$
$(tag, P_{tag}) \leftarrow \mathsf{T.Gen}(par_{\mathsf{T}}, \boxed{sk_1}, n, sn')$
$c_{tag} \leftarrow \mathsf{C.ZCm}(ck, \rho_{tag})$
$c_{\tilde{tag}} \leftarrow \mathsf{E.Enc}(ek, tag, \nu_{tag})$
$\pi_{tag} \leftarrow \mathsf{C.SmPrv}_{tag}(td, pk_{tag}, sn, sn', tag, P_{tag}, \rho_{pk}, \rho_{sn}, \rho'_{sn}, \rho_{tag}, \rho_{P_{tag}})$
$\tilde{\pi}_{tag} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{tag}, \tilde{c}_{tag})$
Check $\mathsf{VER}_{\text{init}}(c^0) \wedge \bigwedge_{j=1}^{i} \mathsf{VER}_{\text{body}}(c^{j-1}, c^j) \wedge$
$\mathsf{T.Verify}(ck, c^i_{pk}, c'_{sn}, c_{tag}, \pi_{tag}) \wedge \mathsf{C.Verify}_{\text{enc}}(ck, ek, c_{tag}, \tilde{c}_{tag}, \tilde{\pi}_{tag})$
If any of them rejects then outputs $\perp$ else choose
a vector of randomness $\overrightarrow{\rho'}$ long enough to compute:
$c_{\text{rand}} \leftarrow$
$\quad \mathsf{Rand}((c^0, (c^j)^i_{j=1}, c'_{pk}, c'_{cert}, \pi'_{cert}, c'_{sn}, \pi'_{sn}, c_{tag}, \pi_{tag}, \tilde{c}'_{sn}, \tilde{\pi}'_{sn}, \tilde{c}'_{tag}, \tilde{\pi}'_{tag}), \overrightarrow{\rho'})$
$c_{\text{new}} := (c_{\text{rand}}, n', sn', \rho'_{sn} + (\overrightarrow{\rho'})_{sn'}, \rho'_{pk} + (\overrightarrow{\rho'})_{pk'})$
$\mathcal{CL}[|\mathcal{CL}| + 1] := (i, c_{\text{new}}, 0, j);$

We define $\forall l \in \{0, \ldots, k-1\}$ a new game:

Experiment $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\text{ZKV2},l}(\lambda)$:
$Gr \leftarrow \mathsf{BGGen}(1^\lambda)$
$par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
$par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
$par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}(Gr)$
$(ck, td) \leftarrow \mathsf{C.SmSetup}(Gr)$
$par \leftarrow (1^\lambda, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$\boxed{(i_0, i_1) \leftarrow \mathtt{DoubleUWith}^{\mathcal{A}}_{\text{rev}}}$
$(\overrightarrow{i^{(0)}}, \overrightarrow{i^{(1)}}) \leftarrow \mathcal{A}^{\mathtt{URegist,Spy}}$
Let $k := |\overrightarrow{i^{(0)}}|$; if $k \neq |\overrightarrow{i^{(1)}}|$, abort the entire procedure
$\boxed{\text{Consider } i_0 \text{ as } (\overrightarrow{i^{(0)}})_0, \text{ and } i_1 \text{ as } (\overrightarrow{i^{(1)}})_0}$
$\boxed{\forall b, j, sk^{(b)}_j \leftarrow \mathcal{UL}[(\overrightarrow{i^{(b)}})_j].sk}$
Then repeat the following step for $j = 1, \ldots, l$:
$\quad \boxed{\text{Run } \mathtt{S\&R}_{ZK,\text{inv}}\ (2j-1, (\overrightarrow{i^{(0)}})_j, sk^{(1)}_{j-1}, sk^{(1)}_j)}$
$\quad \boxed{\text{Run } \mathtt{S\&R}_{ZK,\text{inv}}\ (2j, (\overrightarrow{i^{(1)}})_j, sk^{(0)}_{j-1}, sk^{(0)}_j)}$
$\boxed{\text{Run } \mathtt{S\&R}_{\text{ZK,inv}}\ (2l+1, (\overrightarrow{i^{(0)}})_{l+1}, sk^{(1)}_l, sk^{(0)}_{l+1})}$
$\boxed{\text{Run } \mathtt{S\&R}_{\text{ZK,inv}}\ (2l+2, (\overrightarrow{i^{(1)}})_{l+1}, sk^{(0)}_l, sk^{(1)}_{l+1})}$
Then repeat the following step for $j = l+2, \ldots, k$:
$\quad \text{Run } \mathtt{S\&R}_{\text{ZK}}\ (2j-1, (\overrightarrow{i^{(0)}})_j);\ \text{Run } \mathtt{S\&R}_{\text{ZK}}\ (2j, (\overrightarrow{i^{(1)}})_j)$
Run $\mathtt{Spd}_{\text{ZK}}\ (2k+1+b)$ with $\mathcal{A}$
Run $\mathtt{Spd}_{\text{ZK}}\ (2k+2-b)$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$

Return $b^*$

**Property A.9.** $|\Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2}}(\lambda)) - \Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},0}(\lambda))| \leq 2\epsilon_{\text{t-an}}$.

We receive a challenge of the `tag-anon` game of $\mathsf{T}$ (and not the tag exculpability game, contrarily to the proof of theorem A.4): $par_{\mathsf{T}}$, and we use this parameter as the double spending tag primitive parameter instead of generate it ourself in the $\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2}}$ experiment. In `DoubleUWith`, we send to the `tag-anon`-challenger, the secret keys of $i_0$ and $i_1$. And we use $O_1$ to generate the serial number of $i_0$ in `DoubleUWith` and $O_2(0)$ to generate the corresponding tag in the first $\mathtt{S\&R}_{\text{ZK}}$[1]. If the challenger was in mode 0, it will not change the whole experiment. But if the challenger is in mode 1, it will replace $i_0$ by $i_1$. Let call $\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},-1}$ the game corresponding to this swap. We have just proved that

$$|\Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2}}(\lambda)) - \Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},-1}(\lambda))| \leq \epsilon_{\text{t-an}}$$

With the same strategy we replace $i_1$ by $i_0$, i.e show that:

$$|\Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},-1}(\lambda)) - \Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},0}(\lambda))| \leq \epsilon_{\text{t-an}}$$

and therefore $|\Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2}}(\lambda)) - \Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},0}(\lambda))| \leq 2\epsilon_{\text{t-an}}$.

□

The proof is completely analog for the following property which authorize us to swap multiple games.

**Property A.10.** *For all* $l \in \{0, \ldots, k-2\}$, $|\Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},l}(\lambda)) - \Pr(\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2},l+1}(\lambda))| \leq 2\epsilon_{\text{t-an}}$.

And finally we have to define a last oracle to swap the last keys (and the corresponding game):

$\mathtt{Spd}_{\text{ZK,inv}}(i, j, sk_1)$:
  Receive $(sn', \rho'_{sn})$ from $\mathcal{A}$
  $c \leftarrow \mathcal{CL}[j].c$
  Decompose $c$ as
  $(c^0, (c^j = (c^j_{pk}, c^j_{cert}, \pi^j_{cert}, c^j_{sn}, \pi^j_{sn}, c^j_{tag}, \pi^j_{tag}, \tilde{c}^j_{sn}, \tilde{c}^j_{tag}, \tilde{\pi}^j_{sn}, \tilde{\pi}^j_{tag}))^i_{j=1},$
    $n, sn, \rho_{sn}, \rho_{pk})$
  $\rho_{tag}, \nu_{tag}, \rho_{P_{tag}} \xleftarrow{\$} \mathcal{R}$
  $(tag, P_{tag}) \leftarrow \mathsf{T.Gen}(par_{\mathsf{T}}, \boxed{sk_1}, n, sn')$
  $c_{tag} \leftarrow \mathsf{C.ZCm}(ck, \rho_{tag})$
  $\tilde{c}_{tag} \leftarrow \mathsf{E.Enc}(ek, tag, \nu_{tag})$
  $\pi_{tag} \leftarrow \mathsf{C.SmPrv}_{tag}(td, pk_{tag}, sn, sn', tag, P_{tag}, \rho_{pk}, \rho_{sn}, \rho'_{sn}, \rho_{tag}, \rho_{P_{tag}})$
  $\tilde{\pi}_{tag} \leftarrow \mathsf{C.SmPrv}_{\text{enc}}(td, ek, \rho_{tag}, \tilde{c}_{tag})$
  Send $(c^0, (c^j)^i_{j=1}, c_{tag}, \pi_{tag}, \tilde{c}_{tag}, \tilde{\pi}_{tag})$ to $\mathcal{A}$

Experiment $\mathbf{Expt}^{\mathtt{c-an}}_{\mathcal{A},0,\text{ZKV2,k}}(\lambda)$:
  $Gr \leftarrow \mathsf{BGGen}(1^\lambda)$
  $par_{\mathsf{T}} \leftarrow \mathsf{T.Setup}(Gr)$
  $par_{\mathsf{S}} \leftarrow \mathsf{S.Setup}(Gr)$
  $par_{\mathsf{S}'} \leftarrow \mathsf{S}'\mathsf{.Setup}(Gr)$
  $(ck, td) \leftarrow \mathsf{C.SmSetup}(Gr)$

[1] We use the oracle only at these step, for the other serial number and tag generations, we use the secret keys (which we have generated) like in the $\text{Expt}^{\text{c-an}}_{\mathcal{A},0,\text{ZKV2}}$.

$par \leftarrow (1^\lambda, par_{\mathsf{S}}, par_{\mathsf{S}'}, par_{\mathsf{T}}, ck)$
$pk_{\mathcal{B}} \leftarrow \mathcal{A}(par)$
$(i_0, i_1) \leftarrow \mathtt{DoubleUWith}^{\mathcal{A}}_{\mathrm{rev}}$
$(\overrightarrow{i^{(0)}}, \overrightarrow{i^{(1)}}) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
Let $k := |\overrightarrow{i^{(0)}}|$; if $k \neq |\overrightarrow{i^{(1)}}|$, abort the entire procedure
Consider $i_0$ as $(\overrightarrow{i^{(0)}})_0$, and $i_1$ as $(\overrightarrow{i^{(1)}})_0$
$\forall b, j, sk_j^{(b)} \leftarrow \mathcal{UL}[(\overrightarrow{i^{(b)}})_j].sk$
Then repeat the following step for $j = 1, \ldots, \boxed{\mathrm{k}}$:
Run $\mathtt{S\&R}_{\mathrm{ZK,inv}}$ $(2j-1, (\overrightarrow{i^{(0)}})_j, \mathcal{UL}[(\overrightarrow{i^{(1)}})_{j-1}].sk, \mathcal{UL}[(\overrightarrow{i^{(1)}})_j].sk)$;
Run $\mathtt{S\&R}_{\mathrm{ZK,inv}}$ $(2j, (\overrightarrow{i^{(1)}})_j, \mathcal{UL}[(\overrightarrow{i^{(0)}})_{j-1}].sk, \mathcal{UL}[(\overrightarrow{i^{(0)}})_j].sk)$
$\boxed{\text{Run } \mathtt{Spd}_{\mathrm{ZK,inv}}\ (2k+1+b) \text{ with } \mathcal{A}}$
$\boxed{\text{Run } \mathtt{Spd}_{\mathrm{ZK,inv}}\ (2k+2-b) \text{ with } \mathcal{A}}$
$b^* \leftarrow \mathcal{A}$
Return $b^*$

Analogously to two previous properties we deduce:

**Property A.11.** $|\Pr(\mathrm{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{ZKV2},k-1}(\lambda)) - \Pr(\mathrm{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{ZKV2},k}(\lambda))| \leq 2\epsilon_{\text{t-an}}$.

Now, by remarking that two randomized commitments of the same type in the hiding-mode will have the (exact) same distribution, we can deduce that $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0,\mathrm{ZKV2},k}(\lambda)$ is perfectly indistinguishable from $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},1,\mathrm{ZK}}(\lambda)$.
From a similar reasoning to previous ones we get that $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},1,\mathrm{ZK}}(\lambda)$ is $\epsilon_{\text{m-ind}}$ statistically close to $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},1}(\lambda)$. Finally, we deduce that $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},1}(\lambda)$ is $2(\epsilon_{\mathrm{ZK}} + (k+1)\epsilon_{\text{t-an}})$-statistically-close to $\mathbf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},0}(\lambda)$. □
Note that $\epsilon_{\text{t-an}}$ is the advantage against tag-anonymity of an adversary that is making just one call to $O_1$ and one to $O_2$.

## A.2 Instantiation of the new blocks

### Instantiation and proofs of the double spending tag scheme

We will reuse the scheme introduced in [BCFK15].

$\mathsf{T.Setup}\,(Gr)$:

- Parse $Gr$ as $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g_1, \hat{g})$
- $g_2, h_1, h_2 \xleftarrow{\$} \mathbb{G}$
- Return $(g_1 = g, g_2, h_1, h_2)$

We define $\mathcal{M} = \{(g_1^m, \hat{g}^m) \in \mathbb{G} \times \hat{\mathbb{G}}\}_{m \in \mathbb{Z}_p}$

$\mathsf{T.KeyGen}\,(par_{\mathsf{T}})$:

- $sk \xleftarrow{\$} \mathbb{Z}_p$
- Return $\left(sk_{\mathsf{T}} := sk, pk_{\mathsf{T}} := \hat{g}^{sk}\right)$

$\mathsf{T.SGen}_{\text{init}}\left(par_{\mathsf{T}}, sk_{\mathsf{T}}, n\right)$:

- $M \leftarrow g_1^n$
- $N \leftarrow g_2^{n+sk_{\mathsf{T}}}$
- $M_{sn}^{(1)} = (g_1^n, \hat{g}^n)$
- $M_{sn}^{(2)} = (g_1^{sk_{\mathsf{T}}}, \hat{g}^{sk_{\mathsf{T}}})$
- Return $\left(sn = (M, N), M_{sn} = (M_{sn}^{(1)}, M_{sn}^{(2)})\right)$

$\mathsf{T.SGen}\left(par_{\mathsf{T}}, sk_{\mathsf{T}}, n\right)$:

- $M \leftarrow g_1^n$
- $N \leftarrow g_2^{n+sk_{\mathsf{T}}}$
- $sn\text{-}pf = \hat{g}^n$
- Return $\left(sn = (M, N), sn\text{-}pf\right)$

$\mathsf{T.TGen}\left(par_{\mathsf{T}}, sk, n, sn = (M, N)\right)$:

- $M_0 \leftarrow g_1^n$
- $tag := \left(M^{sk} h_1^n, N^{sk} h_2^n\right)$
- $t\text{-}pf \leftarrow \hat{g}^n$
- Return $\left(tag := (A, B), t\text{-}pf\right)$

$\mathsf{T.Detect}\left(sn, sn', tag, tag', \mathcal{L}\right)$:

- Parse $sn$ as $(M, N)$
- Parse $sn'$ as $(M', N')$
- Parse $tag$ as $(A, B)$
- Parse $tag'$ as $(A', B')$
- $A'' := \frac{A}{A'}$
- $B'' := \frac{B}{B'}$
- $M'' := \frac{M}{M'}$
- $N'' := \frac{N}{N'}$
- If $A'' = 0_{G_1}$ then:
  - $A'' := B''$
  - $M'' := N''$
- Search $pk_{\mathsf{T}}$ in $\mathcal{L}$ such that:
  - $e\left(A'', \hat{g}\right) = e\left(M'', pk_{\mathsf{T}}\right)$
- Return $\left(pk, (A'', M'')\right)$

$\mathsf{T.VfyGuilt}\left(pk, \pi\right)$:

- Parse $\pi$ as $(A, N)$;

- Return $e\left(A,\hat{g}\right)=e\left(N,pk\right)\wedge A\neq 0_{G_1}$

$\mathsf{T.SVfy}_{\text{init}}\left(par_{\mathsf{T}},pk_{\mathsf{T}},sn,M_{sn}\right)$:

- Parse $sn$ as $(M,N)$
- Parse $M_{sn}$ as $((M_1,\hat{M_1}),(M_2,\hat{M_2}))$
- Return $e\left(M,\hat{g}\right)e(g_1^{-1},\hat{M_1})=1_{G_T}\wedge e\left(M,\hat{g}\right)e(g_2^{-1},\hat{M_2})e\left(g_2^{-1},pk_{\mathsf{T}}\right)=1_{G_T}\wedge\hat{M_2}=pk_{\mathsf{T}}\wedge e(M_1,\hat{g})=e(g_1,\hat{M_1})\wedge e(M_2,\hat{g})=e(g_1,\hat{M_2})$

$\mathsf{T.SVfy}\left(par_{\mathsf{T}},pk_{\mathsf{T}},sn,sn\text{-}pf\right)$:

- Parse $sn$ as $(M,N)$
- Return $e\left(M,\hat{g}\right)e\left(g_1^{-1},sn\text{-}pf\right)=1_{G_T}\wedge e\left(N,\hat{g}\right)e\left(g_2^{-1},sn\text{-}pf\right)e\left(g_2^{-1},pk_{\mathsf{T}}\right)=1_{G_T}$

$\mathsf{T.TVfy}\left(par_{\mathsf{T}},pk,sn,sn',tag,t\text{-}pf\right)$:

- Parse $sn$ as $(M,N)$
- Parse $tag$ as $(A,B)$
- Parse $sn'$ as $(M',N')$
- Return

$$e\left(M,\hat{g}\right)e\left(g_1^{-1},t\text{-}pf\right)=1_{G_T}\wedge e\left(A,\hat{g}^{-1}\right)e\left(M',pk\right)e\left(h_1,t\text{-}pf\right)=1_{G_T}\wedge e\left(B,\hat{g}^{-1}\right)e\left(N',pk\right)e\left(h_2,t\text{-}pf\right)=1_{G_T}$$

**Proofs**

**Theorem A.12.** *This scheme is extractable, bootable, SN-verifiable, tag-verifiable and $\mathcal{N}$ injective.*

These properties are all straightforward to show and we therefore omit the proof.

**Property A.13.** *The scheme is SN-collision-resistant.*

Let $(pk,P)$ and $(pk',P')$ such that it exists $sn$ such that:

$$\mathsf{T.SVfy}\left(par_{\mathsf{T}},pk,sn,P\right)=\mathsf{T.SVfy}\left(par,pk',sn,P'\right)=1.$$

We parse $sn$ as $(M,N)$. We can deduce the followings equations:

$$e\left(M,\hat{g}\right)=e\left(g_1,P\right)=e\left(g_1,P'\right)$$

We deduce that $P=P'$. Then we can deduce

$$e\left(M,\hat{g}\right)e\left(g_2^{-1},P\right)e\left(g_2^{-1},pk\right)=0_{G_T}=e\left(M,\hat{g}\right)e\left(g_2^{-1},P\right)e\left(g_2^{-1},pk'\right)$$

$$\implies e\left(g_2^{-1},pk\right)=e\left(g_2^{-1},pk'\right)$$

Then we can finally deduce $pk=pk'$. The reasoning is analogous for $\mathsf{T.SVfy}_{\text{init}}$. □

To prove the two other results, we will use the following lemma.

**Lemma A.14.** *If there exists an adversary $\mathcal{A}$ against Tuple with advantage $\epsilon$, then there exists an adversary $\mathcal{B}$ against DDH in $\mathbb{G}$ with advantage $\epsilon_{\text{DDH}}$ such that $\frac{\epsilon}{3K}\leq\epsilon_{\text{DDH}}$, with $K$ the number of oracles calls to $O'_b$.*

$\mathbf{Expt}^{Tuple}_{\mathcal{A},b}(\lambda)$:
$((G, g_1, q), (\hat{G}, \hat{g}, q), e)$
$g_2, h_1, h_2 \xleftarrow{\$} G$
$b' \leftarrow \mathcal{A}^{O'_b}$
Return $b = b'$

$O'_0$:
$n \xleftarrow{\$} \mathbb{Z}_q$
Return $(g_1^n, g_2^n, h_1^n, h_2^n)$
$O'_1$:
$n_1, n_2, n_3, n_4 \xleftarrow{\$} \mathbb{Z}_q$
Return $(g_1^{n_1}, g_2^{n_2}, h_1^{n_3}, h_2^{n_4})$

We define the following oracles:

$O'_{0.3}$:
$n, n_2 \xleftarrow{\$} \mathbb{Z}_q$
Return $(g_1^n, g_2^{n_2}, h_1^n, h_2^n)$

$O'_{0.7}$:
$n, n_2, n_3 \xleftarrow{\$} \mathbb{Z}_q$
Return $(g_1^n, g_2^{n_2}, h_1^{n_3}, h_2^n)$

We notice that the adversary cannot distinguish $O'_0$ and $O'_{0.3}$, $O'_{0.3}$ and $O'_{0.7}$, $O'_{0.7}$ and $O'_1$, with probability more than $\epsilon_{\text{DDH}}$, respectively, by viewing the triple $g_2, g_1^n, g_2^n$, $h_1, g_1^n, h_1^n$, $h_2, g_1^n, h_2^n$ as DDH triple. (We can compute the others elements by knowing the discrete logarithms of the elements which are not in the DDH-triple, and computing all the other oracle calls honestly). This proves the lemma. □

**Corollary A.15.** *No adversary can break* `tag-anonymity` *with an advantage better than* $6K\epsilon_{\text{DDH}}$*, where* $K$ *is the number of calls to* $O_1$*.*

$\mathbf{Expt}^{\texttt{Perfect-tag-anonymity},O'_{b'}}_{\mathcal{A},b}((\mathbb{G}, g_1, q), (\hat{\mathbb{G}}, \hat{g}, q), e)$:
$par_{\mathsf{T}} \leftarrow ((\mathbb{G}, g_1, q), (\hat{\mathbb{G}}, \hat{g}, q), e)$
$(sk_0, sk_1) \leftarrow \mathcal{A}(par_{\mathsf{T}})$
$k := 0$
$b^* \leftarrow \mathcal{A}^{O_1^{\text{perfect}}(b), O_2^{\text{perfect}}(b,\cdot,\cdot)}(par_{\mathsf{T}}, sk_0, sk_1)$
Return $(b = b^*)$

$O_1^{\text{perfect}}(sk)$:
$(g_3, g_4, g_5, g_6) \leftarrow O'_b$
$T[k] := (g_5, g_6)$
Return $\left(g_3, g_1^{sk} * g_4\right)$

$O_2^{\text{perfect}}(sk, sn', t)$:
If $t > k$ abort entire game
$(g_5, g_6) \leftarrow T[t]$
$(N, M) \leftarrow sn'$
Return $(Ng_5, Mg_6)$

We can use the previous theorem to understand that the adversary could not distinguish `Perfect-tag-anonymity` from `tag-anonymity` (except with probability $3K\epsilon_{\text{DDH}}$) In the latter game, we could replace $b'$ by $1 - b'$ without changing the distribution of the input adversary. □

**Theorem A.16.** *Let* $\mathcal{A}$ *be an adversary that wins the exculpability game with probability* $\epsilon$ *and* $K$ *oracle call to* $O_1$*, then there exist* $\mathcal{B}_1, \mathcal{B}_2$ *adversary against DDH respectively in* $\mathbb{G}$ *with advantage* $\epsilon_{\text{DDH}}$ *and in* $\hat{\mathbb{G}}$ *with advantage* $\hat{\epsilon}_{\text{DDH}}$*, such that:*

$$\epsilon \leq 3K\epsilon_{\text{DDH}} + \hat{\epsilon}_{\text{DDH}}$$

Using the same argument as in the previous corollary, we deduce by incurring a loss of $3K\epsilon_{\text{DDH}}$, we can consider that oracle calls do not yield any information to the adversary. Then after we receive a triple $(\hat{g}_1, \hat{g}_2, \hat{g}_3)$ in $\hat{\mathbb{G}}$, we send $\hat{g}_1$ as the public key. Then we receive $(N, A)$ such that

$$e(N, pk) = e(A, \hat{g})$$

with $A \neq 0_{\mathbb{G}_1}$. This means that $A = N^{\log_{g_1}(pk)}$ and we can check if $e(N, \hat{g}_3) = e(A, \hat{g}_2)$ to guess if it is a DDH triple or not. □

**Efficiency results**

We summarize all the efficiency results as follows (where "m.s.w.u" means multiscalar with unkown):

| | |
|---|---|
| $\lvert par_{\mathsf{T}}\rvert$ | $3\lvert\mathbb{G}\rvert$ |
| $\lvert sk_{\mathsf{T}}\rvert$ | $\lvert\mathbb{Z}_p\rvert$ |
| $\lvert pk_{\mathsf{T}}\rvert$ | $\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert sn\rvert = \lvert tag\rvert$ | $2\lvert\mathbb{G}\rvert$ |
| $\lvert sn\text{-}pf\rvert = \lvert t\text{-}pf\rvert$ | $\lvert\hat{\mathbb{G}}\rvert$ |
| $\pi$ | $2\lvert\mathbb{G}\rvert$ |
| Number of pairing equations in $\mathsf{T.SVfy}$ | 2 generics |
| Number of pairing equations in $\mathsf{T.SVfy}_{\text{init}}$ | 4 generics, 1 m.s.w.u in $\hat{\mathbb{G}}$ |
| Number of pairing equations in $\mathsf{T.TVfy}$ | 3 generics |
| $\lvert\pi_{sn}\rvert$ | $8\lvert\mathbb{G}\rvert + 8\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert\pi_{sn,\text{init}}\rvert$ | $16\lvert\mathbb{G}\rvert + 16\lvert\hat{\mathbb{G}}\rvert + 2\lvert\mathbb{Z}_p\rvert$ |
| $\lvert\pi_{tag}\rvert$ | $12\lvert\mathbb{G}\rvert + 12\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert\tilde{\pi}_{sn}\rvert$ | $8\lvert\mathbb{G}\rvert + 10\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert\tilde{\pi}_{tag}\rvert$ | $12\lvert\mathbb{G}\rvert + 14\lvert\hat{\mathbb{G}}\rvert$ |

## Instantiation of the encryption scheme $\mathsf{E}'$

**Construction overview.** We roughly follow the framework proposed by Chase et. al. [CKLM12]. The first part of the ciphertext is an encryption

$$\overrightarrow{C} = (c_0, c_1, \ldots, c_{n+1}) = (f^\theta, g^\theta, \{h_i^\theta \cdot m_i\}_{i=1}^n)$$

of the message vector $\overrightarrow{m} = (m_1, \ldots, m_n) \in \mathbb{G}_n$. As in [LPQ17], we use the same one-time linearly homomorphic structure-preserving signature scheme [LPJY13] $\mathsf{LHSPS} = (\mathsf{KeyGen}, \mathsf{Sign}.\mathsf{Verify})$, for which let

$$\overrightarrow{v}_1 = (f, g, 1, \ldots, 1), \overrightarrow{v}_2 = (1, 1, 1, g, h_1, \ldots, h_n),$$

the signing key of the LHSPS is composed of two linearly homomorphic signatures of $\overrightarrow{v}_1$ and $\overrightarrow{v}_2$, that is, $sk = (\sigma_{\overrightarrow{v}_1}, \sigma_{\overrightarrow{v}_2})$. Using this signing key, anyone generate the signature $\sigma_{\overrightarrow{m}}$ of any message $\overrightarrow{m} \in Span(\overrightarrow{v}_1, \overrightarrow{v}_2)$.

The second part of the ciphertext is a zero-knowledge proof for the language

$$\mathcal{L}_\vee = \{(\overrightarrow{C}, (b, \theta, \{m_i\}_{i=1}^n, \sigma_{\overrightarrow{v}})) \mid b \in \{0,1\} \vee \mathsf{LHSPS.Verify}(vk_{\mathsf{LHSPS}}, \sigma_{\overrightarrow{v}}, \overrightarrow{v}) = 1\}.$$

where $\overrightarrow{v} = (c_0^b, c_1^b, g^{1-b}, c_1^{(1-b)}, c_2^{(1-b)}, \ldots, c_{n+1}^{1-b})$. Note that when $b = 1$, $\overrightarrow{v} \in Span(\overrightarrow{v}_1, \overrightarrow{v}_2)$ means that $\log_f(c_0) = \log_g(c_1)$, then the ciphertext is a valid ciphertext. Also note that signatures on $\overrightarrow{v}$ with $b = 1$ will only be generated in the security proof.

To enable re-randomization, we generate a signature $\sigma_{\overrightarrow{w}}$ on the vector $\overrightarrow{w} = (f^b, g^b, 1, g^{(1-b)\cdot\theta}, h_1^{(1-b)\cdot\theta}, \ldots, h_n^{(1-b)\cdot\theta})$ and add a zero-knowledge proof of knowledge of the valid signature $\sigma_{\overrightarrow{w}}$. It is easy to see that with $\sigma_{\overrightarrow{w}}$, we can generate signatures for all re-randomization of the vector $\overrightarrow{v}$.

### One-time linearly homomorphic structure-preserving signature

To construct the re-randomizable CCA encryption scheme, we need the one-time linearly homomorphic structure-preserving signature.

**Definition A.17** ((One-time) linearly homomorphic structure-preserving signature [LPJY13])**.** *A one-time linearly homomorphic structure-preserving signature is tuple of* 4 *algorithms* $\mathsf{LHSPS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ *with the following specifications:*

$\mathsf{Setup}\,(Gr, n)$ *is a probabilistic algorithm taking the group parameter* $Gr$ *and an integer* $n$ *denoting the dimension of the message to be signed. It outputs the public verification key* vk *and the signature key* sk*.*

$\mathsf{Sign}\,(sk, \overrightarrow{m})$ *is a deterministic algorithm that takes the signing key* sk *and the message* $\overrightarrow{m} \in \mathbb{G}^n$*, and outputs a signature* $\sigma$*.*

$\mathsf{SignDerive}(vk, \{(w_i, \sigma^{(i)})\}_{i=1}^{\ell})$ *is a deterministic algorithm taking the verification key* vk *and* $\ell$ *pairs* $(w_i \sigma^{(i)})$ *where* $w_i \in \mathbb{Z}_p$ *and* $\sigma^{(i)}$ *is an LHSPS signature. It outputs a signature* $\sigma$ *on the message* $\overrightarrow{m} = \prod_{i=1}^{\ell} \overrightarrow{m}_i^{w_i}$*.*

$\mathsf{Verify}\,(vk, \overrightarrow{m}, \sigma)$ *is a deterministic algorithm taking the verification key* vk*, the message vector* $\overrightarrow{m}$ *and a signature* $\sigma$*. It outputs* 1 *if the signature is valid,* 0 *otherwise.*

**Definition A.18** (One-Time Unforgeability)**.** *A one-time linearly homomorphic SPS scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is secure if no adversary has non-negligible advantage in the following game:*

1. *The adversary* $\mathcal{A}$ *outputs an integer* $n$*, sends it to the challenger* $\mathcal{C}$*. The challenger generates* $(vk, sk) \leftarrow \mathsf{Setup}(1^{\lambda}, n)$ *and sends the public verification key back to* $\mathcal{A}$*.*
2. *The adversary* $\mathcal{A}$ *has access to the signing oracle*
   - $\mathsf{Sign}(sk, \cdot)$*:* $\mathcal{A}$ *can request the challenger* $\mathcal{C}$ *to sign the message vectors* $\{\overrightarrow{m}_i\}_{i=1}^{Q_s}$ *where* $Q_s$ *denotes the number of signing queries.*
3. $\mathcal{A}$ *outputs* $(\overrightarrow{m}^{\star}, \sigma_{\star})$*. The adversary wins if and only if* $\mathsf{Verify}(vk, \overrightarrow{m}^{\star}, \sigma_{\star}) = 1$ *and* $\overrightarrow{m}^{\star} \notin Span(\{\overrightarrow{m}_i\}_{i=1}^{Q_s})$*.*

We recall the following construction of the one-time linearly homomorphic structure-preserving signature scheme.

- $\mathsf{LHSPS.Setup}\,(Gr, n)$:
  1. Parse $Gr$ as $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$.
  2. Chose $\hat{g}_z, \hat{g}_r \xleftarrow{\$} \hat{\mathbb{G}}$. For $i \in \{1, \ldots, n\}$, randomly chose $\chi_i, \gamma_i$ and compute $\hat{g}_i = \hat{g}_z^{\chi_i} \hat{g}_r^{\gamma_i}$.
  3. Output the verification key $pk = (\hat{g}_z, \hat{g}_r, \{\hat{g}_i\}^n) \in \hat{\mathbb{G}}^{n+2}$ and the signing key $sk = (\{\chi_i, \gamma_i\}_{i=1}^n)$.
- $\mathsf{LHSPS.Sign}(vk, sk, \overrightarrow{M})$:
  1. Parse the verification key $vk = (\hat{g}_z, \hat{g}_r, \{\hat{g}_i\}^n) \in \hat{\mathbb{G}}^{n+2}$, the signing key $sk = (\{\chi_i, \gamma_i\}_{i=1}^n)$ and the message $\overrightarrow{M} = (M_1, \ldots, M_n) \in \mathbb{G}^n$.
  2. Output the signature $\overrightarrow{\sigma} = (z, r) \in \mathbb{G}^2$ such that $z = \prod_{i=1}^n M_i^{\chi_i}$ and $r = \prod_{i=1}^n M_i^{\gamma_i}$.

- $\mathsf{LHSPS.SignDerive}(vk, (\overrightarrow{\sigma}, \{w_i, \overrightarrow{\sigma^{(i)}}\}_{i=1}^{\ell}))$:
  1. For all $i \in \{1, \ldots, \ell\}$, parse $\sigma^{(i)}$ as $(z_i, r_i)$.
  2. Output the signature $\sigma = (\prod_{i=1}^{\ell} z_i^{w_i}, \prod_{i=1}^{\ell} r_i^{w_i})$.

- $\mathsf{LHSPS.Verify}(vk_{\mathsf{LHSPS}}, \sigma)$:
  1. Parse the signature as $\sigma = (z, r)$ and the message $\overrightarrow{M} = (M_1, \ldots, M_n)$.
  2. Return 1 iff $(M_1, \ldots, M_n) \neq (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$ and the following equation is verified.

$$e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) = \prod_{i=1}^{n} e(M_i, \hat{g}_i).$$

**Theorem A.19** ([LPJY13, Theorem 1])**.** *The above construction of a one-time linearly homomorphic structure-preserving signature scheme is unforgeable if the SXDH assumption holds in the underlying group.*

The above scheme was proven to be unforgeable under the DP assumption, which is implied by the SXDH assumption. As in the remaining part of the construction of RCCA requires SXDH to hold, we state this theorem with SXDH assumption.

**Replayable-CCA encryption scheme.**

An RCCA encryption scheme $\mathsf{E}$ consists of six PPT algorithms $\mathsf{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ReRand}, \mathsf{Dec}, \mathsf{Verify}, \mathsf{AdptPrf})$. It should verify the following specifications:

- $\mathsf{E.KeyGen}\,(Gr)$: a randomized algorithm which takes as input the group description and outputs an encryption public key $pk$ and a corresponding decryption key $dk$.
- $\mathsf{E.Enc}\,(pk, m, \nu)$: a randomized encryption algorithm which takes as input a public encryption key $pk$, a plaintext (from a plaintext space), some randomness and outputs a ciphertext.
- $\mathsf{E.ReRand}\,(pk, c, \nu)$: a randomized algorithm which takes as input a public key, a ciphertext and some randomness,. and outputs another ciphertext.
- $\mathsf{E.Dec}\,(dk, c)$: a deterministic decryption algorithm which takes a decryption key and a ciphertext, and outputs either a plaintext or an error indicator $\perp$.
- $\mathsf{E.Verify}\,(pk, m, \rho, c)$: a deterministic algorithm which takes as input a public key, a message, some randomness, and a ciphertext and outputs a bit.
- $\mathsf{E.AdptPrf}\,(ck, pk, c_M, c, (\pi, c_\nu), \nu')$ a randomized algorithm which takes as input a commitment key, an encryption public key, a commitment, an equality proof (i.e a Groth-Sahai proof and a commitment), a ciphertext, a proof, some randomness, and outputs an equality proof.

We give the following explicit construction of the RCCA scheme supporting encryption of vectors of group elements.

$\mathsf{E.KeyGen}\,(Gr)$:

1. Parse $Gr$ as $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$.
2. Choose two random group elements $f, g \xleftarrow{\$} \mathbb{G}^2$.

3. Choose random exponents $\{\alpha_i\}_{i=1}^n \xleftarrow{\$} \mathbb{Z}_p$ and compute $\{h_i\}_{i=1}^n = g^{\alpha_i}$.
4. Generate the Groth-Sahai crs $\overrightarrow{crs}_{GS}$ by choosing random $\overrightarrow{u}_1, \overrightarrow{u_2} \xleftarrow{\$} \mathbb{G}^2$ and $\hat{\overrightarrow{u}}_1, \hat{\overrightarrow{u}}_2 \xleftarrow{\$} \hat{\mathbb{G}}^2$.
5. Define two vectors $\overrightarrow{v}_1, \overrightarrow{v}_2$ such that

$$\overrightarrow{v}_1 = (f, g, 1, 1, \ldots, 1) \in \mathbb{G}^{n+2} \qquad \overrightarrow{v}_2 = (1, 1, 1, h_1, \ldots, h_n) \in \mathbb{G}^{n+2},$$

then generate two LHSPS signatures $\sigma_{\overrightarrow{v}_1}$ and $\sigma_{\overrightarrow{v}_2}$ which will be used to proof that a vector is in the $Span(\overrightarrow{v}_1, \overrightarrow{v}_2)$. together with the signing key $\overrightarrow{tk}$.
6. Output the decryption key $dk = \alpha$ and the public key

$$pk = (f, g, \{h^{\alpha_i}\}_{i=1}^n, \overrightarrow{crs}_{GS}, \sigma_{\mathsf{LHSPS}} = (\sigma_{\overrightarrow{v}_1}, \sigma_{\overrightarrow{v}_2})).$$

Notice that the LHSPS signing key $\overrightarrow{tk}$ will never be published by the key generation algorithm, it will only be used in the security proofs.

$\mathsf{E.Enc}\,(pk, m, \nu)$:

1. Randomly pick a number $\theta \in \mathbb{Z}_p$. Compute $\overrightarrow{C} = (c_0, c_1, \ldots, c_{n+1}) = (f^\theta, g^\theta, M_1 \cdot h_1^\theta, \ldots, M_n \cdot h_n^\theta)$.
2. Define the bit $b = 1$ and denote $G = g^b \in \mathbb{G}$ and $\hat{G} = \hat{g}^b \in \hat{\mathbb{G}}$.
3. Generate the Groth-Sahai proof $\pi_b$ of

$$e(G, \hat{g}) = e(g, \hat{G})$$

4. For all $i \in \{1, \ldots n+1\}$, compute $\Theta_i = c_i^b$. Compute also the Groth-Sahai $\pi_\Theta$ proof of the equations:

$$e(\Theta_i, \hat{g}) = e(c_i, \hat{G})$$

5. Define the vector $\overrightarrow{v} = (c_0^b, c_1^b, g^{1-b}, c_1^{1-b}, \ldots, c_{n+1}^{1-b})$. Generate a LHSPS signature $\sigma_{\overrightarrow{v}}$ such that $\overrightarrow{v} \in Span(\overrightarrow{v}_1, \overrightarrow{v}_2)$.
6. Compute a Groth-Sahai proof $\pi_{\overrightarrow{v}}$ of the validity of the LHSPS signature $\sigma_{\overrightarrow{v}}$.
7. To enable the re-randomization, compute

$$(F, G, \{H_i\}_{i=1}^n) = (f^b, g^b, \{h_i^b\}_{i=1}^n)$$

and Groth-Sahai proof $\pi_{FGH}$ of them.
8. Define the vector $\overrightarrow{w} = (f^b, g^b, 1, h_1^{1-b}, \ldots, h_n^{1-b})$. Compute a LHSPS signature $\sigma_{\overrightarrow{w}}$ of the fact that $\overrightarrow{w} \in Span(\overrightarrow{v}_1, \overrightarrow{v}_2)$.
9. Generate a Groth-Sahai proof $\pi_{\overrightarrow{w}}$ of the validity of LHSPS signature $\sigma_{\overrightarrow{w}}$.
10. Output the ciphertext $c = (\{c_i\}_{i=1}^n, \pi_b, \pi_\theta, \pi_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}})$.

$\mathsf{E.ReRand}\,(pk, c, \nu)$:

1. Parse $c = (\{c_i\}_{i=1}^{n+1}, \pi_b, \pi_\theta, \pi_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}})$.
2. Compute $c_0' = c_0 \cdot f^\nu$, $c_1' = c_1 \cdot g^\nu$ and for $i \in \{2, \ldots, n+1\}$, compute $c_i' = c_i \cdot h_{i-1}^\nu$.

3. We update the proof $\pi_b, \pi_\theta$ using the commitment $C_F, C_G, C_H$ in $\pi_{FGH}$ to get $\pi'_b, \pi'_\theta$.
4. We update the commitment of the LHSPS signature $C'_{\sigma_{\overrightarrow{v}}} = C_{\sigma_{\overrightarrow{v}}} \cdot C^{\nu}_{\sigma_{\overrightarrow{w}}}$ and the update the proof $\pi_{\overrightarrow{v}}$ accordingly to get $\pi'_{\overrightarrow{v}}$.
5. We re-randomize all the updated Groth-Sahai proofs
$$\pi'_b, \pi'_\theta, \pi'_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}}$$
to get the new proofs $\pi''_b, \pi''_\theta, \pi''_{\overrightarrow{v}}, \pi''_{FGH}, \pi''_{\overrightarrow{w}}$.
6. Output the new ciphertext $c' = (\{c'_i\}_{i=1}^n, \pi''_b, \pi''_\theta, \pi''_{\overrightarrow{v}}, \pi''_{FGH}, \pi''_{\overrightarrow{w}})$.

$\mathsf{E.Dec}\,(dk, c)$:

1. Parse $c$ as $\sigma = (\{c_i\}_{i=1}^n, \pi_b, \pi_\theta, \pi_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}})$.
2. Check all proofs $(\pi_b, \pi_\theta, \pi_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}})$ are valid.
3. For $i \in \{1, \dots n\}$, compute $M_i = c_{i+1}/(c_1^{\alpha_i})$.
4. Output $\{M_i\}_{i=1}^n$.

$\mathsf{E.Verify}\,(pk, \overrightarrow{m}, \nu, c)$:

1. Parse $c$ as $\sigma = (\{c_i\}_{i=1}^n, \pi_b, \pi_\theta, \pi_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}})$.
2. Verify that $\pi_b, \pi_\theta, \pi_{\overrightarrow{v}}, \pi_{FGH}, \pi_{\overrightarrow{w}}$ are all correct.
3. Verify the following pairing equations:
$$c_0 = g^\nu \qquad c_1 = f^\nu \qquad c_{i+1} = h^\nu \cdot m_i.$$
where $i \in \{1, \dots, n\}$.

$\mathsf{E.AdptPrf}\,(ck, pk, c_M, c, (\pi, c_\nu), \nu')$:

1. We just update the Groth-Sahai proof the new randomness $\nu'$ by multiplying $c'_\nu = c_\nu \cdot \hat{g}^{\nu'}$.
2. As the equality proofs consists of the following pairing equations:
$$c_0 = g^\nu \qquad c_1 = f^\nu \qquad c_{i+1} = h^\nu \cdot m_i.$$
where $i \in \{1, \dots, n\}$.

| | |
|---|---|
| Public Key | $(10+n)\mathbb{G} + 4\hat{\mathbb{G}}$ |
| Decryption Key | $n\mathbb{Z}_p$ |
| Ciphertext | $(6n+19)\mathbb{G} + (16+4n)\hat{\mathbb{G}}$ |
| Verification equations | 2 Linear + $n$ quadratic |
| size of the equality proof | $(2+2n)\mathbb{G} + (2+4n)\hat{\mathbb{G}}$ |

*of Theorem 5.14.* The completeness and the correctness of the above RCCA encryption scheme are straightforward to verify. We will focusing on the Replayable-CCA property.

We proceed by the series of hybrid games $\mathsf{Game}_0, \dots, \mathsf{Game}_5$ , we denote by $\mathsf{Adv}_i$ the advantage of the adversary $\mathcal{A}$ to win the game $\mathsf{Game}_i$.

$\mathsf{Game}_0$: We have $\mathsf{Game}_0$ is identical to the original RCCA security game and thus by definition:

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\mathtt{RCCA}}_{\mathcal{A}}(1^\lambda)$$

$\mathsf{Game}_1$: In this game, we will modify the challenge ciphertext provided to the adversary in the RCCA security game. The new challenge ciphertext is:

$$c^\star = (\{c_i^\star\}_{i=1}^n, \pi_b^\star, \pi_\theta^\star, \pi_{\overrightarrow{v}}^\star, \pi_{FGH}^\star, \pi_{\overrightarrow{w}}^\star)$$

We only modify $\pi_{\overrightarrow{v}}^\star$ and $\pi_{\overrightarrow{w}}^\star$. In stead of generating these two proofs using the signing key $(\sigma_{\overrightarrow{v}_1}, \sigma_{\overrightarrow{v}_2})$ of the LHSPS, we will use the signing key $td$ to directly compute the signatures of $\overrightarrow{v}^\star$ and $\overrightarrow{w}$, where $\overrightarrow{v}^\star = (1, 1, g, c_2, \ldots, c_{n+1})$. (Notice that the secret signing key is never used in the real game.)

As this change is only conceptional, the distribution of the challenge ciphertext is identical in $\mathsf{Game}_1$ as in $\mathsf{Game}_0$. We have $\mathsf{Adv}_1 = |Adv_0$

$\mathsf{Game}_2$: In this game, we modify the $\overrightarrow{crs}$ of the Groth-Sahai proof system. We generate two random values $\xi, \zeta \xleftarrow{\$} \mathbb{Z}_p$, then compute $\overrightarrow{u}_1, \overrightarrow{u}_2, \overrightarrow{\hat{u}}_1, \overrightarrow{\hat{u}}_2$ such that $\overrightarrow{u}_1 = \overrightarrow{u}_2^\xi$ and $\overrightarrow{\hat{u}}_1 = \overrightarrow{\hat{u}}_2^\zeta$.

Notice that this is the perfect sound setting of the Groth-Sahai proof system. $\xi$ and $\zeta$ can be use to extract the witness. Since the only difference between $\mathsf{Game}_1$ and $\mathsf{Game}_2$ is the change of $\overrightarrow{u}_1, \overrightarrow{u}_2, \overrightarrow{\hat{u}}_1, \overrightarrow{\hat{u}}_2$, the indistinguishability can be proven using the SXDH assumption. Thus, we have $\mathsf{Adv}_2 \leq \mathsf{Adv}_1 + 2 \cdot \mathsf{Adv}_{SXDH}$.

$\mathsf{Game}_3$: In this game, we modify the decryption oracle. We will add a manual verification of the underlying LHSPS for the decryption queries. To do this, since the Groth-Sahai proof is settled in the soundness mode ($\overrightarrow{u}_1 = \overrightarrow{u}^\xi$ and $\overrightarrow{\hat{u}}_1 = \overrightarrow{\hat{u}}_2^\zeta$). We can use the trapdoors $\xi, \zeta$ to extract the witness in the commitments of the Groth-Sahai proof. We extract $\overrightarrow{v}$ and $\sigma_{\overrightarrow{v}} = (z, r)$ from the proof $\pi_{\overrightarrow{v}}$. We use the signing key $td$ of the linearly homomorphic structure-preserving signature $\sigma^\dagger_{\overrightarrow{v}} = (z^\dagger, r^\dagger)$ to generate a signature $\sigma^\dagger_{\overrightarrow{v}}$ of the vector $\overrightarrow{v}$. The challenger will reject the decryption query if $\sigma^\dagger_{\overrightarrow{v}} \neq \sigma_{\overrightarrow{v}}$.

We can see that, if an adversary can distinguish $\mathsf{Game}_3$ from $\mathsf{Game}_2$ then he can forge a valid signature of the underlying LHSPS. Since the unforgeability of the LHSPS is based on the SXDH problem, we have $\mathsf{Adv}_3 \leq \mathsf{Adv}_2 + \mathsf{Adv}_{DP}(1^\lambda)$.

$\mathsf{Game}_4$: We will modify all the decryption oracles (both pre-challenge and post-challenge ones) to avoid the use of $\log_g(h_i) = \alpha_i$. After making these changes, we can modify the generation of $h_i$ to $h_i = f^{x_i} g^{y_i}$.

*Pre-challenge decryption queries:* We use the trapdoor of the Groth-Sahai proof to extract the witness of the proof, if we have $b = 0$ then we directly reject the proof.

*Post-challenge decryption queries:* We also use the trapdoor of the Groth-Sahai proof to extract the witness of the proof, if $b = 0$ and the ciphertext is not rejected by the rule of $\mathsf{Game}_3$, the challenger outputs $\mathsf{Replay}$. Additionally, both in pre-challenge and post-challenge decryption queries. Since we don't have $\alpha_i$ anymore, we decrypt the ciphertext by computing $M_i = c_{i+1}/(c_0^{x_i} \cdot c_1^{y^i})$.

We now analyse the change of the decryption oracles:

*Pre-challenge:* It is easy to see that in case of $b = 0$, the challenger only issued two LHSPS signatures of $\overrightarrow{v}_1$ and $\overrightarrow{v}_2$. And the vector $\overrightarrow{v}$ is clearly not in the span of $Span(\overrightarrow{v}_1, \overrightarrow{v}_2)$. So the adversary is statistically impossible to forge a correct signature.

*Post-challenge:* Note that the Groth-Sahai proof is in the perfect soundness setting of the Groth-Sahai proof, thus the challenger $\mathcal{C}$ can use the trapdoor to extract all the witness used in the proof. We will now separate two case:

- If $g^b = 1$, we have $\overrightarrow{v} = (c_0, c_1, 1, 1, \ldots, 1)$. But $\overrightarrow{c}$ is not rejected in the $\mathsf{Game}_3$, with a overwhelming probability, we will have $\overrightarrow{v} \in Span(\overrightarrow{v}_1)$. Thus we have $M_i = c_{i+1}/(c_0^x \cdot c_1^y)$.
- If $g^b = 0$, we have $\overrightarrow{v} = (1, 1, g, c_2, \ldots, c_{n+1})$. As the third element is $g$, $\overrightarrow{v} = \overrightarrow{v} \cdot \overrightarrow{v}_2^{\theta} \cdot \overrightarrow{w}^{\rho}$. This means that $\overrightarrow{v}$ is a randomization of $\overrightarrow{v}^{\star}$, thus we can answer *Replay* to the adversary.

$\mathsf{Game}_5$: We modify the distribution of the challenge ciphertext. Instead of choosing them as an encryption of $\overrightarrow{M}_0$ or $\overrightarrow{M}_1$. We Choose them all random elements. By the self-rerandomizability of the DDH assumption in $\mathbb{G}$, the game 5 is indistinguishable from the game 4.

During the $\mathsf{Game}_5$, as the challenge ciphertext is only random group elements, the adversary cannot have more advantage than a random guess.

□

### Instantiation of the encryption scheme E

Let $Gr = (p, \mathbb{G}, G)$.

$\mathsf{E.KeyGen}()$:

- $(dk_1, dk_2) \xleftarrow{\$} \mathbb{Z}_p^2$
- Return $((G^{dk_1}, G^{dk_2}), (dk_1, dk_2))$

$\mathsf{E.Enc}((D_1, D_2), (M_1, M_2), \nu)$:

- Return $(G^{\nu}, M_1 \cdot D_1^{\nu}, M_2 \cdot D_2^{\nu})$

$\mathsf{E.ReRand}((D_1, D_2), (C_0, C_1, C_2), \nu)$:

- Return $(C_0 \cdot G^{\nu}, C_1 \cdot D_1^{\nu}, C_2 \cdot D_2^{\nu})$

$\mathsf{E.Dec}((dk_1, dk_2), (C_0, C_1, C_2))$:

- Return $(C_0 \cdot C_1^{dk_1}, C_2 \cdot C_0^{dk_2})$

$\mathsf{E.Verify}((D_1, D_2), (M_1, M_2), \nu, (C_0, C_1, C_2))$:

- Return $(G_0 \cdot G^{\nu}, G_1 \cdot D_1^{\nu}, G_2 \cdot D_2^{\nu}) = (C_0, C_1, C_2)$

$\mathsf{E.AdptPrf}(ck, ek, (com_{M_1}, com_{M_2}), c, \tilde{\pi} = (\pi, com_{\nu}), \nu')$:

- Analog to A.2

**Property A.20.** *If there exists an adversary $\mathcal{A}$ that breaks the IACR property of the scheme with advantage $\epsilon_{\mathrm{IACR}}$, then there exists an adversary $\mathcal{B}$ that breaks SXDH with advantage $\epsilon_{\mathrm{SXDH}}$, with*

$$\epsilon_{\mathrm{IACR}} \leq 4\epsilon_{\mathrm{SXDH}}.$$

We define the following experiments:

$\mathbf{Expt}^{\mathtt{IACR}}_{\mathcal{A},b}((\mathbb{G},G,p))$:
$((P_1,P_2),(dk_1,dk_2)) \leftarrow \mathsf{KeyGen}(Gr)$
$((C_0^{(0)},C_1^{(0)},C_2^{(0)}),(C_0^{(1)},C_1^{(1)},C_2^{(1)})) \leftarrow \mathcal{A}((P_1,P_2))$
$\nu \xleftarrow{\$} \mathbb{Z}_p$
$(C_0,C_1,C_2) \leftarrow (C_0^{(b)} \cdot G^{\nu}, C_1^{(b)} \cdot P_1^{\nu}, C_2^{(b)} \cdot P_2^{\nu})$
$b' \leftarrow \mathcal{A}(C_0,C_1,C_2)$
Return $b'$

$\mathbf{Expt}^{\mathtt{IACR}V2}_{\mathcal{A},b}((\mathbb{G},G,p))$:
$((P_1,P_2),(dk_1,dk_2)) \leftarrow \mathsf{KeyGen}(Gr)$
$((C_0^{(0)},C_1^{(0)},C_2^{(0)}),(C_0^{(1)},C_1^{(1)},C_2^{(1)})) \leftarrow \mathcal{A}((P_1,P_2))$
$\nu,\nu_2 \xleftarrow{\$} \mathbb{Z}_p$
$(C_0,C_1,C_2) \leftarrow (C_0^{(b)} \cdot G^{\nu}, C_1^{(b)} \cdot P_1^{\nu_2}, C_2^{(b)} \cdot P_2^{\nu})$
$b' \leftarrow \mathcal{A}(C_0,C_1,C_2)$
Return $b'$

$\mathbf{Expt}^{\mathtt{IACR}V3}_{\mathcal{A},b}((\mathbb{G},G,p))$:
$((P_1,P_2),(dk_1,dk_2)) \leftarrow \mathsf{KeyGen}(Gr)$
$((C_0^{(0)},C_1^{(0)},C_2^{(0)}),(C_0^{(1)},C_1^{(1)},C_2^{(1)})) \leftarrow \mathcal{A}((P_1,P_2))$
$\nu,\nu_2,\nu_3 \xleftarrow{\$} \mathbb{Z}_p$
$(C_0,C_1,C_2) \leftarrow (C_0^{(b)} \cdot G^{\nu}, C_1^{(b)} \cdot P_1^{\nu_2}, C_2^{(b)} \cdot P_2^{\nu_3})$
$b' \leftarrow \mathcal{A}(C_0,C_1,C_2)$
Return $b'$

By noticing that $|\Pr(\mathbf{Expt}^{\mathtt{IACR}}_{\mathcal{A},b}((\mathbb{G},G,p))=1) - \Pr(\mathbf{Expt}^{\mathtt{IACR}V2}_{\mathcal{A},b}((\mathbb{G},G,p))=1)|$ and $|\Pr(\mathbf{Expt}^{\mathtt{IACR}V2}_{\mathcal{A},b}((\mathbb{G},G,p))=1) - \Pr(\mathbf{Expt}^{\mathtt{IACR}V3}_{\mathcal{A},b}((\mathbb{G},G,p))=1|$ are less or equal to $\epsilon_{\mathrm{SXDH}}$, and because $\mathbf{Expt}^{\mathtt{IACR}V3}_{\mathcal{A},0}((\mathbb{G},G,p))$ and $\mathbf{Expt}^{\mathtt{IACR}V3}_{\mathcal{A},1}((\mathbb{G},G,p))$ are distributed equally, we deduce

$$\epsilon_{\mathrm{IACR}} \leq 4\,\epsilon_{\mathrm{SXDH}}.$$

## A.3 Efficiency analysis of the transferable e-cash scheme

We summarize the characteristics of $\mathsf{C}$ in this table :

| | |
|---|---|
| $\lvert ck\rvert$ | $3\lvert\mathbb{G}\rvert + 3\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert\mathsf{Cm}(g_1)\rvert$ | $2\lvert\mathbb{G}\rvert$ |
| $\lvert\mathsf{Cm}(\hat{g})\rvert$ | $2\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert\mathsf{Cm}(1_{\mathbb{Z}_p})\rvert$ | $2\lvert\hat{\mathbb{G}}\rvert$ |
| **Homogen. pair. product equation with unknown in $\mathbb{G}$** | $2\lvert\hat{\mathbb{G}}\rvert$ |
| **Homogen. pair. product equation with unknown in $\hat{\mathbb{G}}$** | $2\lvert\mathbb{G}\rvert$ |
| **General homogenous pairing product equation** | $4\lvert\mathbb{G}\rvert + 4\lvert\hat{\mathbb{G}}\rvert$ |
| **M-s equation in $\mathbb{G}$ with unknown in $\mathbb{Z}_p$** | $\lvert\mathbb{G}\rvert$ |
| **Homogeneous M-s equation in $\hat{\mathbb{G}}$ with unknown in $\hat{\mathbb{G}}$** | $2\lvert\mathbb{Z}_p\rvert$ |
| **General m-s equation in $\mathbb{G}$** | $2\lvert\mathbb{G}\rvert + 4\lvert\hat{\mathbb{G}}\rvert$ |

The following table summarizes the characteristics of the two signature schemes. (Recall that $\mathcal{M}' = \hat{\mathbb{G}}$ and $\mathcal{M} = \{(g^m, \hat{g}^m) | m \in \mathbb{Z}_p\}^2$.)

| **Signature scheme** | $\mathsf{S}$ [Fuc11] | $\mathsf{S}'$ [AGHO11] |
|---|---|---|
| $\lvert par_S \rvert$ | $3\lvert\mathbb{G}\rvert$ | 0 |
| $\lvert sk \rvert$ | $\lvert\mathbb{Z}_p\rvert$ | $3\lvert\mathbb{Z}_p\rvert$ |
| $\lvert vk \rvert$ | $\lvert\mathbb{G}\rvert + \lvert\hat{\mathbb{G}}\rvert$ | $3\lvert\hat{\mathbb{G}}\rvert$ |
| $\lvert\sigma\rvert$ | $13\lvert\mathbb{G}\rvert + 9\lvert\hat{\mathbb{G}}\rvert$ | $2\lvert\mathbb{G}\rvert + \lvert\hat{\mathbb{G}}\rvert$ |
| **Nb of pairing eqs in** $S.\mathsf{Verify}$ | 12 general equations | 1 linear in $\hat{\mathbb{G}}$, 1 general |
| $\lvert\pi_\sigma\rvert$ | $48\lvert\mathbb{G}\rvert + 48\lvert\hat{\mathbb{G}}\rvert$ | $6\lvert\mathbb{G}\rvert + 4\lvert\hat{\mathbb{G}}\rvert$ |

The following table summarizes the characteristics of the the ElGamal encryption scheme (recall the message space is $\mathbb{G}^2$):

| | |
|---|---|
| $\lvert sk \rvert$ | $2\lvert\mathbb{Z}_p\rvert$ |
| $\lvert pk \rvert$ | $2\lvert\mathbb{G}\rvert$ |
| $\lvert c \rvert$ | $3\lvert\mathbb{G}\rvert$ |
| $\lvert\nu\rvert$ | $\lvert\mathbb{Z}_p\rvert$ |
| **Nb ms eqs in** $E.\mathsf{Verify}$ | 2 general equations 1 linear with unkown in $\mathbb{Z}_p$ |
| $\lvert\tilde{\pi}_{\text{eq}}\rvert$ | $5\lvert\mathbb{G}\rvert + 10\lvert\hat{\mathbb{G}}\rvert$ |

## RÉSUMÉ

Les billets électroniques transferables sont l'analogue numérique des monnaies fiducières, étant donné qu'ils donnent la possibilité aux usagers de transférer des pièces entre eux sans interagir avec la banque. L'ambition de rendre un tel système fortement anonyme, et la necessité de détecter les fraudes (en particulier les doubles-dépenses) ont longtemps rendu difficile la construction d'un tel schéma. Baldimisti et coll. (PKC'15) donnèrent une première construction, qui malheureusement se base sur une puissante primitive qui la rend impraticable. Dans cette thèse, on reconsidère les modèles de sécurité en proposant des définitions plus parcimonieuses et plus fortes, puis nous rédigeons un premier schéma ayant l'ambition d'être implantable, et analysons son efficacité. La sécurité de notre schéma reposant sur des hypothèses cryptographiques non standards, on analyse alors la pertinence d'un large ensemble d'hypothèses cryptographiques construites à partir de couplages en utilisant une technique adaptée au contexte : le modèle du groupe algébrique.

## MOTS CLÉS

monnaie numérique, anonymat, preuve à divulgation nulle, modèle du groupe algébrique, meta-hypothèses, signatures de classes d'équivalence, modèle du groupe générique.

## ABSTRACT

Transferable e-cash is the most faithful digital analog of physical cash, as it allows users to transfer coins between them without interacting with the bank. Strong anonymity requirements and the need for mechanisms to trace illegal behavior (double-spending of coins) have made instantiating the concept notoriously hard. Baldimtsi et al. (PKC'15) have given a first instantiation, which however relied on a powerful cryptographic primitive that made the scheme non-practical. In this thesis we revisit the model for transferable e-cash, proposing simpler yet stronger security definitions and then give the first concrete instantiation of the primitive, basing it on bilinear groups, and analyze its concrete efficiency. Because to build our scheme, we are using non-standard assumption in a bilinear group context, we analyze the hardness of a broad class of assumptions in a relevant context: the algebraic group model.

## KEYWORDS

e-cash, strong anonymity, randomizable NIZK proofs, algebraic group model, uber assumption, signatures on equivalence classes, generic group model.