M2M authentication and privacy design of new efficient cryptographic primitives and protocols

Amira Barki

Par **Amira BARKI**

Mécanismes cryptographiques conciliant authentification et respect de la vie privée dans le contexte du M2M

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC

Soutenue le 16 décembre 2016
Spécialité : Technologies de l'Information et des Systèmes : Unité de recherche Heudyasic (UMR-7253)
D2337

UTC - École Doctorale *Sciences pour l'ingénieur*

Mécanismes cryptographiques conciliant authentification et respect de la vie privée dans le contexte du M2M

Thèse

présentée et soutenue le 16/12/2016

pour l'obtention du

Doctorat de l'Université de Technologie de Compiègne
Spécialité : Technologies de l'Information et des Systèmes

par

Amira Barki

devant le jury composé de :

Directeur de thèse :	
Abdelmadjid Bouabdallah	Professeur des Universités, Université de Technologie de Compiègne
Co-encadrants :	
Saïd Gharout	Ingénieur de recherche, Orange Labs
Jacques Traoré	Ingénieur de recherche, Orange Labs
Rapporteurs :	
Gildas Avoine	Professeur des Universités, Université de Rennes 1
Louis Goubin	Professeur des Universités, Université de Versailles
Examinateurs :	
Hervé Chabanne	Professeur associé, Télécom ParisTech / Morpho
Maryline Laurent	Professeur des Universités, Télécom SudParis
Aziz Moukrim	Professeur des Universités, Université de Technologie de Compiègne

To my mother,
To the memory of my father,
who have always supported me.

Acknowledgments

Avant de commencer ce mémoire, je tiens à remercier toutes les personnes qui ont contribué, de près ou de loin, à la réussite de cette thèse.

Je souhaite tout d'abord remercier Gildas Avoine et Louis Goubin pour avoir accepté d'être rapporteurs de mes travaux de thèse. J'adresse également mes remerciements à Hervé Chabanne, Maryline Laurent et Aziz Moukrim pour m'avoir fait l'honneur de faire partie de mon jury de thèse.

Je voudrais aussi remercier mon directeur de thèse académique, Abdelmadjid, pour la liberté de travail qu'il m'a laissée pendant ces trois années. En outre, je suis très reconnaissante envers mes deux directeurs de thèse côté Orange, Saïd et Jacques, avec qui j'ai beaucoup apprécié travailler. Je les remercie notamment pour la confiance qu'ils m'ont témoignée, leur disponibilité, leurs remarques constructives et les précieux conseils qu'ils m'ont offerts tout au long de cette thèse, sans jamais rien m'imposer.

J'ai effectué cette thèse à Orange Labs et plus précisément au sein des deux équipes DSS à Paris et NPS à Caen. Je tiens à cet effet à remercier Jean-François et Sébastien pour leur bonne humeur et la bonne ambiance qu'ils maintiennent au niveau des deux équipes. Je souhaite par ailleurs remercier Solenn, avec qui j'ai eu le plaisir de travailler mais aussi de découvrir de nouveaux pays et cultures. J'ai aussi eu l'occasion de travailler avec Nicolas, que je remercie pour sa diligence. Je voudrais également remercier Marc pour avoir pris le temps de relire une partie de ce manuscrit et pour ses remarques pertinentes qui ont permis de l'améliorer. Je tiens également à remercier les différents membres des équipes NPS, DSS mais aussi NST pour tout ce qu'ils ont pu m'apporter et plus particulièrement les doctorants et post-docs, à savoir Alex, Alicia, Bastien, Baptiste, Benjamin, Donald, Ghada, Julien, Marie, Maxime, Mohamed, Olivier, Pierre, Quentin, Xiao et Yacine (j'espère que je n'oublie personne !). Merci pour les différentes discussions, rigolades et les bons moments que nous avons pu partager...

Je remercie également mes ami(e)s de longue date qui ont su être là quand j'en avais besoin (certains malgré les milliers de kilomètres qui nous séparent, je pense en particulier à Marwa) et qui m'ont permis de décompresser en découvrant de nouvelles activités ou endroits, ou tout simplement en partageant de bons moments ensemble. Ainsi, je remercie Amine, Hajer, Imen, Maroua, Mohamed, Mouna, Rania, Rim et Salma.

Pour finir, je remercie ma famille qui a toujours cru en moi, m'a soutenue et accompagnée, tout en me permettant de suivre la voie que je désirais. Merci pour vos conseils avisés et amour inconditionnel, sans quoi je ne serais pas là où j'en suis aujourd'hui.

Abstract

Machine to Machine (M2M) applications are increasingly being deployed so as to enable a better management of resources and provide users with greater comfort, convenience as well as peace of mind. Unfortunately, they also entail serious security and privacy concerns that users are either underestimating or unaware of. In this thesis, we focus on M2M security, and particularly on the authentication and privacy issues of M2M applications involving a SIM card.

In the first part of this thesis, we design five new cryptographic primitives, that are of independent interest, and formally prove that they meet the expected security requirements. More precisely, they consist of a partially blind signature scheme, a sequential aggregate Message Authentication Codes (MAC) scheme, an algebraic MAC scheme and two pre-Direct Anonymous Attestation (pre-DAA) schemes. Some of the proposed schemes aim to achieve a particular property that was not provided by previous constructions, as it is the case of our partially blind signature scheme which enables multiple presentations of the same signature in an unlinkable way, whereas others intend to improve the efficiency of state-of-the-art schemes. Furthermore, our five schemes do not require the user's device to compute pairings. Thus, they are suitable for resource constrained environments such as SIM cards.

In a second part, we rely on these primitives to propose new privacy-preserving protocols. More specifically, we design a private eCash system where the user can settle expenses of different amounts using a single reusable payment token. Its implementation on a standard NFC-enabled SIM card confirms that it is quite efficient, even for real world applications such as electronic Toll (eToll). In this particular use case, a payment can be performed in just 205 ms. We also propose a protocol enabling anonymous authentication and identification of an embedded SIM (eSIM) to a Discovery Server (DS), which is an MNO independent third party that allows linking of an eSIM to the relevant MNO. Thereby, eSIMs can be remotely provisioned with their new network profiles while protecting users' privacy against a malicious discovery server. Furthermore, we rely on our algebraic MAC scheme to build a practical Keyed-Verification Anonymous Credentials (KVAC) system which can be easily turned into an efficient public key anonymous credentials system. Finally, based on our sequential aggregate MAC scheme, we introduce a remote electronic voting system that is coercion-resistant and practical for real polls. The security of our protocols is formally proven in the Random Oracle Model (ROM) under classical computational assumptions.

Contents

Abbreviations

3GPP	**3**rd **G**eneration **P**artnership **P**roject
BB	**B**oneh-**B**oyen
DAA	**D**irect **A**nonymous **A**ttestation
DCR	**D**ecisional **C**omposite **R**esiduosity
DDH	**D**ecisional **D**iffie-**H**ellman
DL	**D**iscrete **L**ogarithm
eCash	**e**lectronic **C**ash
EID	**E**mbedded UICC **ID**entifier
eSIM	**e**mbedded **S**ubscription **I**dentity **M**odule
eToll	**e**lectronic **T**oll
ETSI	**E**uropean **T**elecommunications **S**tandards **I**nstitute
eUICC	**e**mbedded **U**niversal **I**ntegrated **C**ircuit **C**ard
EUM	**E**mbedded **UICC** **M**anufacturer
EVC	**E**lectric **V**ehicle **C**harging
GGM	**G**eneric **G**roup **M**odel
GP	**G**lobal**P**latform
GSMA	**G**lobal **S**ystem for **M**obile communications **A**ssociation
ISO	**I**nternational **S**tandards **O**rganization
KVAC	**K**eyed-**V**erification **A**nonymous **C**redentials
LRSW	**L**ysyanskaya, **R**ivest, **S**ahai and **W**olf
M2M	**M**achine **To** **M**achine
MAC	**M**essage **A**uthentication **C**ode
MNO	**M**obile **N**etwork **O**perator
NFC	**N**ear **F**ield **C**ommunication
NIZKPK	**N**on-**I**nteractive **Z**ero-**K**nowledge **P**roof of **K**nowledge
OMDL	**O**ne-**M**ore **D**iscrete **L**ogarithm
OTA	**O**ver-**T**he-**A**ir
q-SDH	**q**-**S**trong **D**iffie-**H**ellman
q-DDHI	**q**-**D**ecisional **D**iffie-**H**ellman **I**nversion
ROM	**R**andom **O**racle **M**odel
RSA	**R**ivest, **S**hamir and **A**dleman
SD	**S**ecurity **D**omain
SE	**S**ecure **E**lement
SIM	**S**ubscription **I**dentity **M**odule
SM-DP+	**S**ubscription **M**anager-**D**ata **P**reparation+
SM-DS	**S**ubscription **M**anager-**D**iscovery **S**erver
UICC	**U**niversal **I**ntegrated **C**ircuit **C**ard
WSN	**W**ireless **S**ensor **N**etwork
ZKPK	**Z**ero-**K**nowledge **P**roof of **K**nowledge

Notation

$x \xleftarrow{R} X$ or $x \in_R X$	x is chosen uniformly at random from the set X
$\mathbb{Z}_p$	Set of positive integers less than $p-1$
$\{0,1\}^*$	Set of all binary strings of arbitrary finite length
$\{0,1\}^k$	Set of all binary strings of length k
$\mathbb{G}$	Cyclic group of prime order
$1_{\mathbb{G}}$	Identity element of $\mathbb{G}$
k	Security parameter
pp	Public parameters
$\vec{m}$	The vector $(m_1, \ldots, m_n)$
$\{g_i\}_{i=1}^{l}$	The set $\{g_1, g_2, \ldots, g_l\}$
$\mathcal{H}(m,t)$	The hash of the concatenation of m and t
$\texttt{Pr}[x]$	Probability of the event x
$q \vert p-1$	q divides $p-1$
$a \leftarrow b$	a is assigned the value b
$\texttt{Protocol}(\mathcal{U}(sk), \mathcal{V}(pp))$	Interactive protocol between $\mathcal{U}$ and $\mathcal{V}$ which takes as input sk and pp
$k\mathbb{G}_i$	k exponentiations in the group $\mathbb{G}_i$
$k\mathbb{G}_i^j$	k j-multi-exponentiations in the group $\mathbb{G}_i$
ID_U	Identifier of a user
$\mathcal{U}$	User
$\mathcal{I}$	Issuer
$\mathcal{V}$	Verifier
$\mathcal{A}$	Adversary
$\mathcal{B}$	Reduction
$\mathcal{C}$	Challenger
$\mathcal{A}^{\mathcal{O}_1}(x)$	The adversary $\mathcal{A}$ knows x and can query the oracle $\mathcal{O}_1$

Introduction

1.1. Contexte

Le machine to machine (M2M) est un concept désignant tous les systèmes intégrant des équipements connectés et capables de communiquer entre eux, ou avec un serveur distant, d'une manière autonome (sans intervention humaine). Ces communications permettent la remontée de différents évènements, la collecte de données et parfois même le déclenchement de certaines commandes en fonction des données collectées. Ainsi, le M2M offre plus de confort et permet une utilisation optimale des ressources (matérielles, énergétiques et humaines). A ce jour, 5 milliards d'équipements sont déjà connectés au réseau mobile et d'après les estimations d'Ericsson [Eri11] ainsi que de Cisco [Eva11], leur nombre devrait atteindre 50 milliards d'ici 2020.

Toutefois, les applications M2M ne présentent pas que des avantages pour les utilisateurs. Ces dernières peuvent engendrer des problèmes de sécurité, voire porter atteinte à la vie privée des détenteurs d'équipements M2M. Ces différents problèmes pourraient à terme entraver le déploiement massif de certaines applications. Par ailleurs, la nature des environnements M2M et le caractère personnel des données échangées rend les applications M2M particulièrement vulnérables à diverses attaques qui peuvent être initiées soit par un fournisseur de services malveillant, soit un attaquant externe, voire un utilisateur malhonnête. En outre, compte tenu de l'ubiquité des équipements M2M, une quantité importante d'informations sensibles est susceptible d'être collectée, ce qui, dans le cas où des mesures de sécurité idoines ne seraient pas mises en place, pourrait engendrer des atteintes significatives à la vie privée des utilisateurs. En effet, les informations recueillies pourraient révéler des données à caractère personnel tels que les habitudes de l'utilisateur, son état de santé, etc. Ces problèmes de sécurité et de vie privée sont amplifiés dans le contexte du M2M, notamment en raison de l'hétérogénéité des équipements M2M qui sont, pour la plupart, peu onéreux et donc limités en ressources (énergie, capacité de calcul, de stockage, etc.). En outre, certaines applications, tel que le télépéage par exemple, ont des contraintes temporelles assez strictes. De telles limitations semblent de facto exclure les mécanismes de sécurité relevant de la cryptographie à clé publique qui sont particulièrement coûteux en termes de temps de calcul et d'espace mémoire.

Dans cette thèse, nous nous intéressons essentiellement aux applications M2M incluant un élément de sécurité, de l'anglais *Secure Element* (SE), telle qu'une carte SIM par exemple. Nous cherchons notamment à garantir deux propriétés de sécurité antinomiques de prime abord, à savoir l'authentification d'équipements M2M et la préservation de la vie privée de leurs utilisateurs, en particulier vis-à-vis de fournisseur de services. Ci-après, nous définissons brièvement ces deux propriétés avant de donner un aperçu de nos différentes contributions.

L'**authentification** est l'un des premiers objectifs de la cryptographie qui permet à une entité, dans un premier temps, de s'assurer de l'identité de son correspondant (*i.e.* qu'il est réellement celui qu'il prétend être) avant de poursuivre tout échange avec ce dernier. Généralement, un protocole d'authentification implique deux parties, mais il pourrait y en avoir plusieurs. En outre, l'authentification peut être mutuelle (*i.e.* les deux entités doivent s'authentifier l'une et

l'autre) ou unilatérale (*i.e.* une des deux entités uniquement doit s'authentifier). Actuellement, l'authentification est utilisée partout et dans la vie de tous les jours. Prenons l'exemple d'un utilisateur qui souhaiterait connaître le solde de son compte bancaire en utilisant une application sur son smartphone. Pour ce faire, ce dernier doit d'abord s'authentifier (*i.e.* prouver qu'il est bien le détenteur de ce compte, ou du moins, qu'il connaît le login/mot de passe correspondant).

Le **respect de la vie privée** est une propriété moins ordinaire, mais qui ne cesse de gagner de l'intérêt surtout avec l'arrivée prochaine du futur règlement en matière de protection des données à caractère personnel (GDPR). Celui-ci risque, en effet, d'être très contraignant pour les fournisseurs de services et les opérateurs de télécommunications quant à l'utilisation des données de leurs clients. Plus précisément, ce règlement met en avant le principe de *minimisation des données* qui stipule que seules les informations strictement nécessaires à la réalisation d'une transaction (quelle qu'en soit la nature) devraient être dévoilées (et rien de plus). Par exemple, la seule information que la société d'autoroute à besoin de connaître lorsqu'un automobiliste emprunte les voies de télépéage, c'est le montant total que ce dernier lui doit. Il n'est sans doute pas nécessaire qu'elle ait en plus accès à l'historique précis des ses différents trajets.

A ce jour, plusieurs travaux ont déjà visé à concilier ces deux propriétés. Cependant, cela s'est bien souvent fait au détriment de l'efficacité. Rares sont en effet les solutions qui réussissent à satisfaire les contraintes temporelles strictes de certaines applications et aucune ne semble être adaptée aux dispositifs limités en termes de puissance de calcul et d'espace mémoire telle que la carte SIM.

1.2. Contributions

Primitives cryptographiques efficaces. Afin de surmonter les contraintes induites par les capacités de calcul limitées de certains dispositifs M2M et les exigences temporelles de certaines applications M2M, nous avons été conduits à introduire de nouveaux schémas cryptographiques. Nous proposons à cet effet, dans le Chapitre 4 de ce mémoire, cinq nouveaux schémas cryptographiques. Nous introduisons dans un premier temps un nouveau schéma de signature partiellement aveugle dont les signatures ont la particularité de pouvoir être "randomisées" et donc d'être intraçables y compris lorsqu'elles sont présentées à de multiples reprises au même vérificateur. Dans un deuxième temps, nous nous focalisons sur les schémas de code d'authentification de messages (MAC) algébriques qui, contrairement aux schémas MAC classiques, reposent sur des opérations dans des groupes cycliques. Dans ce contexte, nous concevons un nouveau schéma de MAC algébrique que nous avons baptisé $\mathsf{MAC}_{\mathsf{BB}}$ et qui peut, indifféremment, être utilisé pour caculer un tag sur un unique message ou un bloc de messages. Ensuite, nous proposons un nouveau schéma de MAC dont les tags peuvent être *séquentiellement* agrégés. Enfin, nous introduisons deux schémas d'attestations anonymes (DAA) pour lesquels tous les calculs sont effectués par la puce TPM (la SIM dans notre contexte) contrairement aux solutions antérieures qui nécessitent de déléguer une partie des calculs à la plateforme hôte embarquant cette puce TPM (typiquement un smartphone ou un PC). Ces cinq schémas cryptographiques sont adaptés aux environnements limités en ressources tels que les cartes SIMs dans la mesure où ils ne nécessitent aucun calcul de *couplage* (notoirement connus pour être particulièrement coûteux en termes de temps de calcul) par l'équipement M2M.

Dans le reste du mémoire, en nous appuyant sur les schémas proposés dans le chapitre 4, nous construisons quatre protocoles efficaces et préservant la vie privée des utilisateurs (*i.e.* utilisables en pratique). Ces protocoles consistent en un système de paiement anonyme, un système d'accréditations anonymes, un protocole d'authentification et d'identification anonyme

pour l'embedded SIM (eSIM) ainsi qu'un système de vote électronique *résistant à la coercition*.

Un système de paiement anonyme pratique. Chaum a introduit en 1982 [Cha83] le concept de *Monnaie électronique* (e-cash), équivalent numérique de la monnaie fiduciaire. Depuis, trois types de monnaie électronique ont été définis : transférable, divisible et compacte. Comme son nom l'indique, la monnaie électronique transférable permet, à l'instar de la monnaie fudiciaire, le transfert de pièces entre deux marchands (afin d'acheter des biens) ou entre un marchand et un utilisateur (afin de rendre la monnaie). La monnaie électronique divisible, quant à elle, offre la possibilité de convertir une pièce de valeur faciale importante (10 euros par exemple) en pièces de valeurs inférieures (5 pièces de 2 euros ou 10 pièces de 1 euro par exemple), afin d'éviter les problèmes d'appoint, courants avec la monnaie classique. Quant à la monnaie électronique compacte, elle permet à l'utilisateur de retirer plusieurs pièces en une seule fois, contrairement aux autres monnaies qui ne permettent de retirer qu'une seule pièce à la fois. Dans le chapitre 5, nous proposons un autre type de monnaie électronique que nous appelons *private eCash*. Ce nouveau type est particulièrement adapté aux applications tels que le transport public, le télépéage et la recharge de véhicules éléctriques, pour lesquelles une même et unique entité fait à la fois office de banque et de marchand. Notre système repose sur le schéma de signature partiellement aveugle introduit dans le chapitre 4. Plus précisément, une pièce de monnaie consiste en une signature partiellement aveugle portant sur une information connue uniquement par l'utilisateur. Grâce à une seule et unique pièce, un utilisateur pourra régler plusieurs "achats", dont le montant peut être variable, et ce de manière intraçable. Ces travaux ont donné lieu à la publication "*Private eCash in Practice*" [BBD+16] à FC'16.

Chaum a aussi été à l'origine des accréditations anonymes [Cha83] qui sont, depuis, largement répandues. Un système d'accréditations anonymes permet aux utilisateurs d'obtenir des accréditations certifiées (permis de conduire, carte d'étudiant, de sécurité sociale, etc.) auprès d'organisations émettrices (préfecture, mairie, université, etc.) et de *prouver* ensuite la possession de ces accréditations auprès de fournisseurs de services tout en *minimisant l'information dévoilée* (comme par exemple prouver que l'on est étudiant et que l'on a moins de 25 ans afin de bénéficier d'un tarif avantageux lors de l'entrée à un musée, et ce sans révéler sa date de naissance ni sa filière). Les accréditations anonymes ont plusieurs applications dont la monnaie électronique et le vote en ligne.

Accréditations anonymes à clés secrètes. Dans le chapitre 6, nous nous intéressons à un type particulier d'accréditations anonymes connues en anglais sous le nom de *Keyed-Verification Anonymous Credentials (KVAC)* car ces dernières requièrent que le vérifieur connaisse la clé secrète de l'entité qui a délivré l'accréditation. Suivant la même approche que Chase, Meiklejohn and Zaverucha [CMZ14], nous nous basons sur les MACs algébriques, et plus précisément sur notre $\mathsf{MAC}^n_{\mathsf{BB}}$ algébrique introduit dans le chapitre 4, afin de concevoir notre système d'accréditations anonymes à clés secrètes. Notre système permet plusieurs présentations d'une même accréditation, et ce de manière totalement intraçable, tout en étant presque aussi efficace que le système U-Prove de Microsoft (dont les accréditations ne satisfont pourtant pas cette caractéristique). Par ailleurs, il peut facilement être converti, si nécessaire, en un système d'accréditations anonymes classique. Ainsi, un utilisateur pourrait prouver la possession de son accréditation à n'importe quelle entité (même si cette dernière ne connaît pas la clé de l'émetteur). Ces résultats ont été publiés dans l'article "Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials" à SAC 2016 [BBDT16].

Un protocole d'authentification et d'identification anonyme pour l'eSIM. Par la suite, dans le chapitre 7, afin de résoudre une problématique qui s'est posée à la GSM Association (GSMA), nous spécifions un protocole d'authentification et d'identification anonyme pour la

nouvelle génération de cartes SIM embarquées connue sous le nom d'embedded SIMs (eSIMs). Notre protocole repose essentiellement sur notre schéma d'attestations anonymes, qui est fondé sur le schéma de signature de Boneh-Boyen, décrit dans le chapitre 4. Il vise à permettre l'authentification et l'identification anonyme de l'eSIM auprès d'une entité tierce, indépendante de l'opérateur, connue sous le nom de *serveur de découverte*. De la sorte, suite à un changement d'opérateur, l'eSIM pourra récupérer à distance son nouveau profil réseau tout en préservant la vie privée de son propriétaire face à un serveur de découverte malhonnête. Un brevet décrivant cette solution a été déposé à l'INPI en novembre 2015 [BCGT15].

Un système de vote électronique résistant à la coercition. Finalement, dans le chapitre 8, nous proposons un système de vote électronique pratique et permettant de rendre inutile toute tentative de coercition à l'encontre d'un votant. Ce système repose sur l'utilisation d'accréditations anonymes afin de permettre à un électeur, dûment inscrit sur les registres électoraux, de voter de manière anonyme. Grâce à l'utilisation de notre schéma de MAC séquentiellement aggrégés détaillé dans le chapitre 4, les accréditations des électeurs peuvent être efficacement mises à jour afin de leur permettre de prendre part, ultérieurement, à d'autres élections. En outre, les accréditations de certains électeurs (par exemple ceux ne pouvant pas prendre part à l'élection car déchus de leur droit de vote) peuvent être révoquées. Ces travaux ont fait l'objet de l'article "*Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant*" [ABBT16] à VOTING'16.

Chapter 1

Introduction

1.1 Context

The concept of Machine to Machine (M2M) refers to all systems involving connected devices which can communicate with each other, or with a remote server, in an autonomous way (*i.e.* with no or little human intervention) so as to gather information, notify users of a given event, and even trigger some actions. Thereby, M2M enables a better management of resources and provide users with greater comfort, convenience as well as peace of mind. To date, there have been around 5 billion M2M devices connected to the wireless networks and, according to Ericsson's [Eri11] as well as Cisco's estimation [Eva11], their number is expected to reach 50 billion by 2020.

Unfortunately, an important hurdle that may jeopardize M2M growth and even hinder the massive roll-out of some M2M applications is security. Indeed, the nature of M2M environments and the sensitivity of exchanged data make M2M applications vulnerable to numerous attacks that may be initiated by a malicious service provider, an external adversary or even a fraudulent user. Furthermore, owing to the pervasiveness of M2M devices, a very large amount of personal data can be collected. If not properly protected, it may entail serious privacy concerns. Indeed, recorded information may disclose personal information such as habits and health status. Addressing these security and privacy issues is a quite challenging task particularly due to the heterogeneity of M2M devices, which are mainly inexpensive and resource constrained, and the stringent time constraints of certain applications (*e.g.* electronic Toll (eToll)).

In this thesis, we mainly focus on M2M applications involving a Secure Element (SE) such as a SIM card, and more specifically on combining two security properties that may seem contradictory at first glance, namely authentication and privacy. Thereby, users can enjoy the benefits of M2M applications while preserving their privacy. Hereinafter, we briefly define these two properties, then we give an overview of our contributions.

Authentication is one of the primary goals of cryptography which enables an entity to check the identity of another one (*i.e.* it is really what it claims to be) before initiating "real" communication. Usually, an authentication protocol involves two parties, but more parties can be involved as well. Besides, authentication can be mutual (*i.e.* both parties have to authenticate each other) or unilateral (*i.e.* only one party needs to be authenticated). Today, authentication is used everywhere. For instance, a user who wants to check his current account balance through an application on his smartphone must first authenticate himself using his login and password.

Privacy is a less common property that has been gaining particular interest in the last decades with the emergence of new technologies that handles users' personal data. More specifically, privacy aims to prevent the disclosure of any irrelevant and unnecessary personal information (*e.g.*

date of birth, habits) so as to comply with data minimization principles. For instance, a service provider (*e.g.* an eToll company) does not have to know the details of the different trips that a user has taken, but rather only the associated total charges.

To date, several schemes have aimed to ensure both the privacy and authentication properties. However, existing proposals did not really succeed in satisfying the security and privacy requirements as well as the need for efficiency (so as to meet the stringent time constraints of some applications), whilst being suitable for resource constrained environments like secure elements (*e.g.* SIM cards).

1.2 Contributions

Efficient cryptographic primitives. In order to overcome the constraints stemming from devices limited computational resources and applications stringent time constraints, more efficient cryptographic primitives had to be introduced. With this purpose in mind, we design in Chapter 4 five new cryptographic primitives that are of independent interest. More specifically, we first propose a partially blind signature scheme which enables multiple presentations of the same signature in an unlinkable way. Then, we build an efficient sequential aggregate Message Authentication Code (MAC) scheme. Next, we focus on algebraic MAC schemes which rely on group operations rather than hash functions or block ciphers. In this respect, we design a new algebraic MAC scheme referred to as $\mathsf{MAC}_{\mathsf{BB}}$ that can be extended to support n messages. Finally, we propose two practical Direct Anonymous Attestation (DAA) schemes where all computations on the platform's side are performed by the Trusted Platform Module (TPM). All proposals are suitable for resource constrained environments such as SIM cards as they require no pairings computations on the user's side.

In subsequent chapters, using the designed primitives as main building blocks, we build four practical privacy-preserving protocols, namely a private eCash system, an efficient coercion-resistant electronic voting scheme, an anonymous authentication and identification protocol for embedded SIM (eSIM) as well as a Keyed-Verification Anonymous Credentials (KVAC) system.

Private eCash system. To provide a *privacy-preserving* digital analogue of hard currency, Chaum introduced in 1982 electronic cash (eCash) [Cha83]. Since then, three main types of eCash systems have been defined: *transferable*, *divisible* and *compact*. As their name implies, transferable eCash enables the transfer of coins between two merchants (*e.g.* to buy goods) or a merchant and a user (*e.g.* to give back change) whereas divisible eCash allows a user to split a coin of a large value into a few coins so as to overcome change issues. As for compact eCash system, it enables the user to withdraw a set of N coins at once, and then spend them one by one. In Chapter 5, we introduce a new type of eCash system referred to as *private eCash* This new type is intended for applications, such as eTicketing, electronic Toll (eToll) as well as Electric Vehicle Charging (EVC), which involve a single entity acting as both merchant and bank. To build our eCash system, we use our partially blind signature scheme introduced in Chapter 4 as the main building block. More precisely, a payment token consists of a partially blind signature on a common information and a value only known to the user. Using this unique and reusable token, a user can settle several expenses of different amounts in an unlinkable way. This work resulted in the paper "*Private eCash in Practice*" [BBD+16], published at FC'16.

Chaum also envisioned a prevalent privacy-preserving cryptographic primitive known as anonymous credentials [Cha83]. The latter allows users to obtain a credential (*i.e.* driving license, student ID card, etc.) from an issuer and then, later, prove the possession of this credential, in

an unlinkable way, without revealing any additional information. Anonymous credentials have several applications including eCash and electronic voting.

Keyed-Verification anonymous credentials. In Chapter 6, we focus on a particular type of anonymous credentials systems referred to as Keyed-Verification Anonymous Credentials (KVAC) since they require the verifier to know the issuer secret key. Following the same approach as Chase, Meiklejohn and Zaverucha [CMZ14], we rely on algebraic Message Authentication Codes (MAC), and more precisely on our $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme introduced in Chapter 4, to build our KVAC system. Our proposal, which provides multi-show unlinkability (*i.e.* one can prove possession of the same credential several times in an unlinkable manner) whilst being almost as efficient as Microsoft's U-Prove, can be easily turned into a public-key anonymous credential system. Thus, a user can prove possession of his credential to any entity (*i.e.* even one that does not hold the issuer secret key). These results have been published in the paper "Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials" [BBDT16] presented at SAC 2016.

Anonymous authentication and identification protocol for embedded SIM. Next, in Chapter 7, we cope with an issue that has arisen at the GSM Association (GSMA) by designing a privacy-preserving authentication and identification protocol for embedded SIM (eSIM). Our protocol, which is based on the BB-based Pre-DAA scheme proposed in Chapter 4, aims to enable the anonymous authentication and identification of an eSIM to a Mobile Network Operator (MNO) independent entity known as Discovery Server (DS). Thereby, following a change of MNO, an eSIM can be remotely provisioned with its new network profile while preserving the privacy of its owner against a malicious DS. This proposal was patented on November, 2015 [BCGT15].

Coercion-resistant remote electronic voting system. Finally, in Chapter 8, we address the coercion-resistance issue of remote electronic voting which stems from the lack of private polling booths. Thus, a voter can be constrained to cast a given vote or to reveal her voting secret. Based on anonymous credentials, we build an efficient coercion-resistant remote electronic voting scheme that is practical for real polls. Furthermore, using our sequential aggregate MAC scheme detailed in Chapter 4, voters credentials can be efficiently updated so as to enable credentials revocation as well multiple elections. These results have been published in the paper "*Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant*" [ABBT16] at VOTING'16.

Chapter 2

M2M Characteristics and Security Issues

Contents

In this chapter, we briefly recall some of the main current Machine to Machine (M2M) applications, then we review the typical M2M communication architecture proposed by the European Telecommunications Standards Institute (ETSI). We also provide an overview on a new generation of SIM cards, known as embedded SIM (eSIM), that has been standardized by the GSM Association (GSMA) and which will play a significant role in the overall growth of M2M. Next, we present the most important security issues that M2M devices and communications have to face. Finally, we identify M2M characteristics that make the design of new security schemes a challenging task.

2.1 M2M Applications

Machine to Machine (M2M) is constantly evolving and covering more and more applications that do not only aim to provide users with more convenience and peace of mind but also save energy and enable a better management of assets, traffic, etc. The applications currently proposed can be classified according to the field of application into five categories, namely Automotive, eHealth, Smart Metering, Smart City and Smart Home as shown in Figure 2.1. As stated in [GSM14b], the automotive sector has attracted a lot of attention which led to the emergence of several automotive applications.

It is worth mentioning that, in all these applications, the communication is bidirectional. In fact, the M2M device does not only monitor some events but also receives information such as updates and controls from the application domain. Nevertheless, the Down Link (DL) traffic from the server to the M2M devices is much lower than the Up Link (UL) traffic and generally consists of short control packets unicasted to an M2M device or multicasted to a group of devices [ETS13].

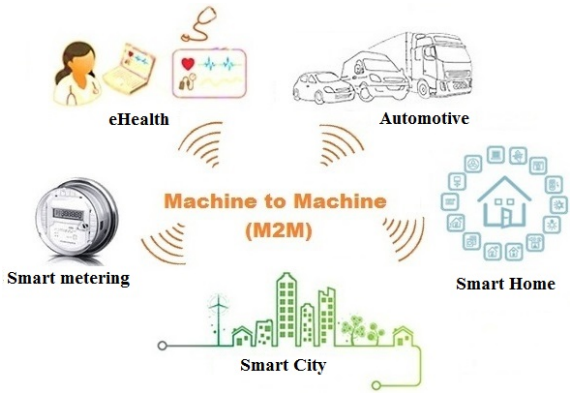

Figure 2.1: M2M Applications

2.1.1 Automotive

This category encompasses all applications involving a car or a transportation system. To operate, each vehicle has to be equipped with a communication module which generally includes a Global Positioning System (GPS) and a SIM (or eSIM) card enabling a bidirectional communication. Here is a list of the main applications.

- *Emergency Call (eCall)* [Eur13, 3GP15] aims to provide rapid assistance to motorists whenever an accident occurs. Via action of in-vehicle sensors, it automatically triggers an emergency voice call and sends some key information about the accident. The European Commission stated that eCall will be mandatory in cars as of April 2018.

- *Electronic Toll (eToll)* eliminates the delays and queue lines at tollbooths since it enables users to pay for the toll fares with no need to stop.

- *Pay-As-You-Drive (PAYD)* is a new kind of customized insurance where the charges are computed based on where, when and how the user drives.

- *Fleet Management* improves efficiency and allows a better management of assets as it enables continuous track of their position and state.

- *Stolen Vehicle Tracking (SVT)*, also known as Stolen Vehicle Recovery, offers vehicle owners peace of mind as it allows them to track their vehicle if it is ever stolen.

2.1.2 eHealth

eHealth applications enable the remote monitoring of people's health and fitness information through the recording of a set of indicators such as blood pressure and heart rate. To do so, a set of sensors must be worn by the user. Due to their limited resources, these sensors forward the collected data to a gateway, generally a smartphone, which handles their transmission to the remote server where healthcare professionals and users can visualize them. Thus, it is similar to Body Sensor Networks (BSNs) except that the communication is bidirectional and may be a many-to/from-one communication. Hereinafter, we briefly describe the most important applications.

- *Remote Patient Monitoring (RPM)* enables the remote management of diseases by periodically collecting and reporting patients' physiological vital signs values (*e.g.* gas, water and electricity).

- *Fitness Information Management* aims to report the fitness records of the user during exercise in order to subsequently provide him with a feedback.

- *Elderly, Children and Herds Tracking* enable the localization of children, elderly people and herds as well as the monitoring of their vital signs and activity level.

2.1.3 Smart Metering

As their name implies, these applications are related to smart meters. They aim to enable a better and more efficient management (and use) of water, gas and electricity. In what follows, we detail the main smart metering applications.

- *Smart Meters* enable both users and resource supplier to remotely read the metering values. Moreover, they allow the remote activation and disabling of supply.

- *Smart Grid (SG)* [FFK+11] enables a two-way exchange of energy and information between the power plant and whatever consumes or produces energy. Thereby, the available energy is properly managed with low losses and high levels of quality of supply.

- *Electric Vehicle Charging (EVC)* allows users to locate the nearest available charging station and book a time slot if needed. So, it can also be considered as an automotive application. Within a Smart Grid, EVC enables a better management of energy as it directs users to certain stations and remotely configures the mode of charging according to the power demand and supply.

2.1.4 Smart City

This category includes "green" applications whose goal is to save energy. To reduce costs, several gateways are deployed all over the relevant area (*e.g.* a city) in order to aggregate and transmit the data collected by the different sensors.

- *Smart Parking* provides drivers with an efficient way to quickly find the nearest available parking space. Thereby, it cuts congestion and avoids waste of time and gas.

- *Smart Waste Management* enables the avoidance of several unnecessary journeys as it sends a notification whenever a container is nearly full.

- *Smart Street Lighting* enables the setting of the lighting level according to the sensed values.

- *Air Pollution Monitoring* allows an efficient and continuous measurement of the air quality.

- *Traffic Information* enables a driver to be aware of the traffic conditions and warning messages related to the area in which he is driving. Deployed cameras could be used to detect red light violations and even compute cars speed. They can also help automatically manage traffic lights so as to improve the traffic flow in the city.

- *Inventory Management* optimizes delivery schedules and allows a more efficient re-stocking. Vending machines and networked printers are good examples showing the benefits of such application.

2.1.5 Smart Home

Smart Home applications, also known as Smart Home, encompasses applications providing customers with more convenience and a higher quality of life by enabling the remote detection or execution of certain tasks.

- *Remote Control of Appliances* enables the remote or automatic turn on/off of a set of devices as well as the control of their status.

- *Intrusion Detection* notifies the user whenever someone comes in (or goes out) the relevant area (*e.g.* an office, a house, etc.) and enables the detection of any risky situation.

- *Leak Detection* aims to detect gas or water leaks as well as fire.

2.2 ETSI M2M Architecture

M2M has attracted the attention of some Standards Development Organizations (SDO). In particular, the ETSI mainly aimed to design a common M2M architecture whose primary goal is to foster the development of network independent applications and ensure the interoperability of M2M services. In this section, we provide an overview of the typical M2M architecture and the corresponding interfaces proposed by the ETSI in the Technical Specifications (TS) 102 690 [ETS13] and 102 921 [ETS14].

2.2.1 High Level Architecture

From a functional point of view, the M2M architecture can be split into three interlinked domains; namely, *device domain*, *network domain* and *application domain*. In what follows, we briefly introduce them.

- **Device domain** consists of the set of devices involved in M2M applications. Some of these devices are equipped with specific sensing technology as it is the case with body sensors while other devices simply record current values. Once the necessary data is gathered, the M2M device forwards it through the network domain towards the remote server. This domain is also referred to as M2M domain.

- **Network domain** consists of a set of heterogeneous access networks enabling the device domain and the application domain to communicate. Among these access networks, we note UTRAN, GERAN, satellite, xDSL, WiMAX, WLAN, etc.

- **Application domain** includes the servers where data collected by the M2M devices or other network nodes are stored. These real-time data is made available to legitimate users through a variety of M2M applications thereby enabling remote monitoring and sending of control instructions to M2M devices.

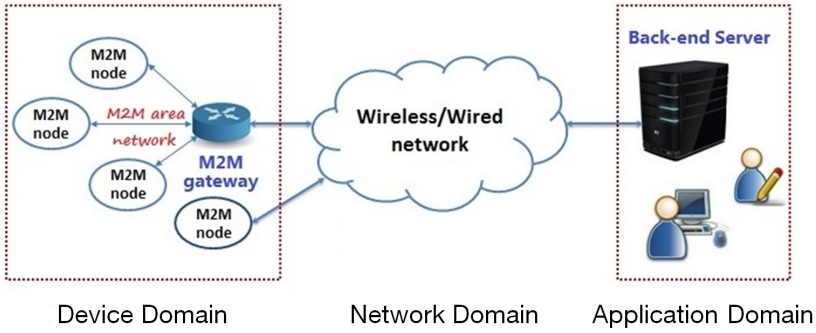

Figure 2.2: Typical M2M Architecture

As shown in Figure 2.2, the communication between the M2M device and the network domain is either direct or via a gateway through the use of an M2M area network. The latter provides connectivity between the M2M gateway and the M2M devices by relying on short range wireless technologies such as ZigBee, Bluetooth Low Energy (BLE), etc. To enable direct communication, the M2M device ought to be equipped with a communication module enabling it to access the network via mobile or fixed line and perform authentication, registration and management procedures by itself. If it is not the case, especially when dealing with constrained devices such as sensors, all these procedures are performed by the M2M gateway on behalf of the device.

2.2.2 Service Capability Layer

In TS 102 690 [ETS13], ETSI provides an M2M architecture with a generic set of capabilities. More precisely, it introduces a Service Capability Layer (SCL) that has been designed to work with REpresentational State Transfer (REST) compliant architecture style. SCL has an important role in the ETSI M2M architecture since it provides functions that are to be shared between the different applications and exposes them in the form of features through a set of open interfaces (*e.g.*, via an API). As it hides the complexity of underlying networks, the SCL simplifies application development thereby fostering the development of network independent new services. It also aims to ensure the interoperability of M2M services and to facilitate the deployment of vertical applications.

Among the defined SCLs, there is a service capability layer called SEC (for security) which handles the M2M service bootstrapping and the key establishment. Generic Communication (GC) is another service capability responsible for securing data exchanges and application delivery. There must exists an SCL in both device and network/application domains. In the device domain, the SCL is either inside the M2M device itself (DSCL for Device SCL) or within the gateway (GSCL for Gateway SCL) as shown in the figure 2.3. The SCL within the network/application domain is referred to as NSCL for Network SCL.

2.2.3 M2M Interfaces

To enable communications between the different M2M entities, four different interfaces have been introduced [ETS13] (see Figure 2.3). They are defined as follows:

- **dIa** is the interface linking the application and the SCL located in the device domain (except for the case 3 where the SCL is in the network domain). As it is shown in Figure 2.3, we may have three scenarios. In the first case, the Device Application (DA) and the Service Capability Layer (DSCL) are within the same device. While in the second, there is no service capability implemented in the device and this latter resides in the gateway (GSCL).

In the third case, which is used when dealing with non-ETSI compliant devices, the dIa connects the DA and the NSCL.

- **mId** is the interface linking two service capabilities, one residing in the network domain (NSCL) and the other in the device or the gateway domain DSCL/GSCL.
- **mIa** is the interface between the NSCL and the application residing in the network domain, referred to as Network Application (NA). The functions supported by the NSCL are exposed to the network application via this interface.
- **mIm** is an inter-domain interface which enables two NSCLs located in two different network domains to communicate.

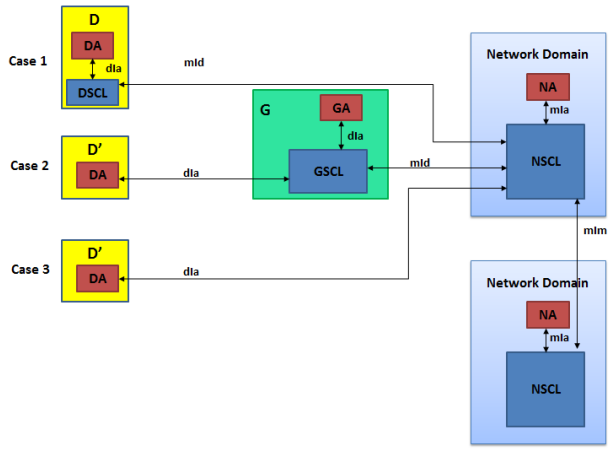

Figure 2.3: ETSI interfaces according to deployment scenario [ETS13]

To illustrate the role of each of these entities, we consider the case of a Smart Home owner who would like to remotely turn on his heater. To do so, through his mobile phone, he accesses the NA and sends the request to the NSCL which, in its turn, will forward it to the corresponding GSCL to finally be received by the DA residing in the heater. Note that all these exchanges are imperceptible to the user.

2.2.4 ETSI Key Hierarchy

Along with the M2M architecture and the corresponding interfaces, the ETSI standardization body also proposed a key hierarchy where each set of provisioned M2M device/gateway holds an M2M root key (K_{mr}). There are three different ways enabling the provisioning of the root key to the M2M device or gateway. The first consists in pre-provisioning it in a secure manner during manufacture or deployment whereas, in the second, the root key is derived from mobile network authentication credentials based on the Authentication and Key Agreement (AKA) protocol using either Generic Bootstrapping Architecture (GBA) [3GP16] or Extensible Authentication Protocol (EAP) [IET04] based procedures. In the last case, the M2M root key is provisioned in an access network independent procedure through the use of either EAP over Protocol for carrying Authentication for Network Access (PANA) [IET08] or Transport Layer Security (TLS) [IET10] over the Transmission Control Protocol (TCP). From the root key, an M2M connection key (K_{mc}) is derived. The K_{mc} key is used to secure the mId interface which links the DSCL/GSCL and the NSCL. At each new M2M service connection procedure, a new K_{mc} key is calculated.

Unfortunately, ETSI architecture has not been widely adopted and it does not seem like it is going to happen any time soon. In fact, the M2M market is still highly fragmented with several vertical solutions designed in a separate and independent way.

The GSM Association (GSMA) also looked into potential M2M issues and more precisely into the problem of provisioning SIM cards, that are soldered in the M2M device, with their new subscription profiles. This led to the design of a new generation of SIM cards, called embedded SIM, that we introduce in the following section.

2.3 Embedded SIM

Some M2M devices, like smart meters for example, are not designed in a way enabling the removal or replacement of the SIM card. In fact, the SIM is either hardly accessible or even soldered into the M2M device at its manufacture. Besides, in use cases such as vending machines where a large number of devices are deployed in different locations, the replacement of all SIM cards following the change of network operator subscription can be quite costly. These two cases raised a new issue: how can one change the subscription associated to a SIM card while keeping the same chip and level of security as classical SIM cards. To tackle this problem, the GSMA designed a new generation of SIM cards known as embedded UICC (eUICC) or embedded SIM (eSIM) along with an architecture for Over-The-Air (OTA) remote SIM provisioning. Thereby, an eSIM can be remotely provisioned with its new network profile at every change of subscription or device ownership. Hereinafter, we give an overview on the eUICC and remote SIM provisioning architectures as well as the registration notification, profile generation and download procedures.

2.3.1 eUICC Architecture

The particular features of an eUICC are essentially its support of a remote provisioning functionality and its capability to store *several Mobile Network Operator (MNO) profiles* as shown in Figure 2.4. Though an eSIM may contain many profiles, only one profile can be enabled at a time and the eSIM behaves as if it contains no other profile. When a profile is active, the device can connect to the MNO defined in this profile. The different profiles may be classified in three main categories:

- *Test profile* which can only be used when the device is in test mode.

- *Provisioning profile* is automatically enabled in the cases where no other profile is enabled. It aims to provide connectivity for the sole purpose of downloading a new operational profile. Once done, it is automatically disabled.

- *Operational profile* is a conventional profile that includes, amongst other, a Network Access Application (NAA) enabling the eUICC registration to the Mobile Network (MN) as well as associated security credentials (secret keys, algorithms, certificates, etc). From a user and device perspective, an *operational* profile can be seen as a profile of a classical SIM card (*i.e.* it has the same structure and functions as a regular SIM card).

Each eUICC profile is installed within a distinct *Security Domain* (SD) [Glo15] known as Issuer Security Domain Profile (ISD-P) and uniquely identified by its Integrated Circuit Card Identifier (ICCID). A profile ICCID is equivalent to the ICCID of a classical UICC [ETS16]. An additional identifier, referred to as *Embedded UICC Identifier (EID)*, is used as the unique physical identifier of the eUICC. The EID, which is not modifiable, plays a significant role in the remote SIM provisioning architecture described below. Note that, in this section, we only provide the information necessary to clearly understand our proposal described in Chapter 7. For more details about the eUICC architecture, we refer the reader to [GSM16a, GSM16c].

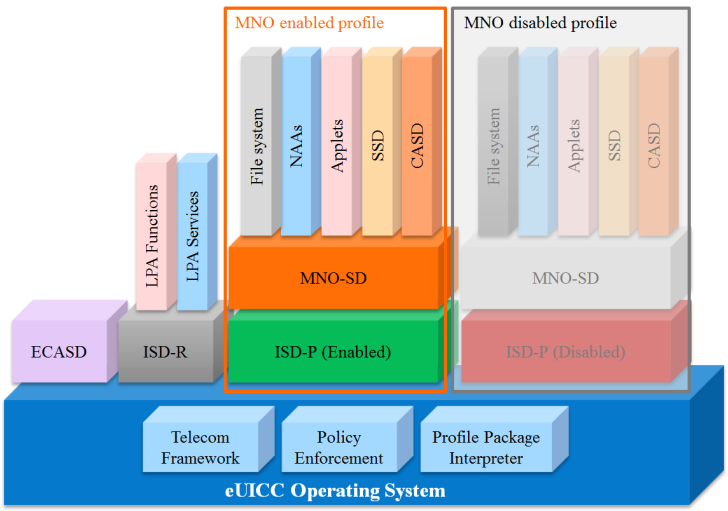

Figure 2.4: High-Level Representation of an eUICC [GSM16c]

2.3.2 GSMA Remote SIM Provisioning Architecture

To enable the remote provisioning of an eSIM with its new operational profile, the GSMA introduced a functional architecture [GSM14a, GSM16b] called *remote SIM provisioning architecture* as well as its corresponding technical specifications [GSM16a, GSM16c]. This architecture involves five main entities, namely the eUICC, the MNO, the eUICC Manufacturer (EUM), the Subscription Manager-Data Preparation+ (SM-DP+) and the Subscription Manager-Discovery Server (SM-DS). In what follows, we briefly introduce each entity and its role within the remote SIM provisioning architecture illustrated in Figure 2.5.

- **eUICC Manufacturer (EUM):** it is the entity responsible for the issuance of eUICCs. It must also provide each manufactured eUICC with a certificate that will subsequently enable it to be authenticated.

- **eUICC:** it is uniquely identified by its eUICC Identifier (EID) and can hold several profiles. It also contains a set of Local Profile Assistant (LPA) Services which provide access to the data and services required by the LPA functions. They consist of a Local User Interface (LUI), a Local Profile Download (LPD) and a Local Discovery Service (LDS). More precisely, the LUI enables the user to perform local profile management whereas the LDS is responsible for the retrieval of notifications sent by the SM-DP+ and stored in the SM-DS. If a profile package is available for a given eUICC, it is the LDP that handles its download from the SM-DP+. As shown in Figure 2.5, depending on the implementation, the LPA functions may be either in the eUICC itself or in the device containing the eUICC. In the sequel, for the sake of clarity, we will use the term LPA to refer to either LPD, LUI or LDS.

- **Mobile Network Operator (MNO):** whenever the user takes out a new subscription or changes the one associated to an eUICC, the MNO orders the SM-DP+ to generate or manage the corresponding profile.

- **Subscription Manager-Data Preparation+ (SM-DP+)**: it is responsible for the creation, management and protection of profiles upon the request of the MNO. It must also ask for the creation of the target ISD-P in the eUICC and handle the secure delivery of the profile through the LPA. Furthermore, when there is a pending action (profile download or management) for a specific eUICC, it ought to transmit a notification along with the associated EID to the SM-DS.

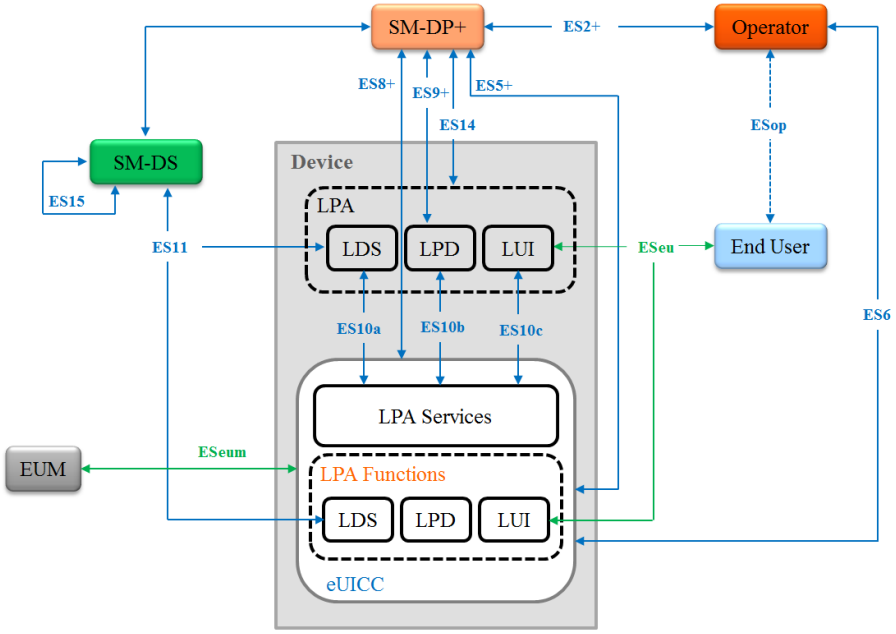

Figure 2.5: Remote SIM Provisioning Architecture [GSM16b]

- **Subscription Manager-Discovery Server (SM-DS)**: it is an MNO-independent third party that mainly aims to respond to LPA requests asking whether there is a pending action for a given eUICC or not. Besides, in the cases where the eUICC is not configured with a default SM-DP+, the SM-DS allows the eUICC (in particular, its LPA) to determine which SM-DP+ holds its new profile. Sometimes, a default SM-DS may not be configured in the eUICC either. If so, the logical location of the default SM-DS is recovered based on the eUICC Identifier (EID).

2.3.3 Procedures Description

In what follows, we briefly review the procedures that are chronologically triggered, whenever the user takes out a new subscription, in order to provide the corresponding eUICC with its new network profile.

1. *Protected Profile Generation Procedure*: it is the first procedure that the MNO triggers further to a new subscription. More specifically, the MNO provides the SM-DP+ with the profile description and asks it to generate and securely store the corresponding profile as shown in Figure 2.6-(1). Once the profile is created, the SM-DP+ informs the MNO, which registers the profile in its operator system.

2. *Notification Registration Procedure*: once the profile is generated, the SM-DP+ transmits a notification registration to the SM-DS. It mainly consists of the EID of the target eUICC, the address of the SM-DP+ and a unique identifier (Notification-ID). Upon receiving this notification, the SM-DS stores it and acknowledge its receipt (see Figure 2.6 -(2)).

3. *Discovery Request Procedure*: if there is no SM-DP+ address configured in the eUICC, the LDS triggers the discovery request procedure by sending a request to the eUICC. Upon its receipt, the eUICC generates an eUICC authorisation (that will enable its authentication to the SM-DS) using its private key. It is transmitted along with the address of the default SM-DS and the eUICC identifier (EID) to the LDS. These latters are forwarded to the

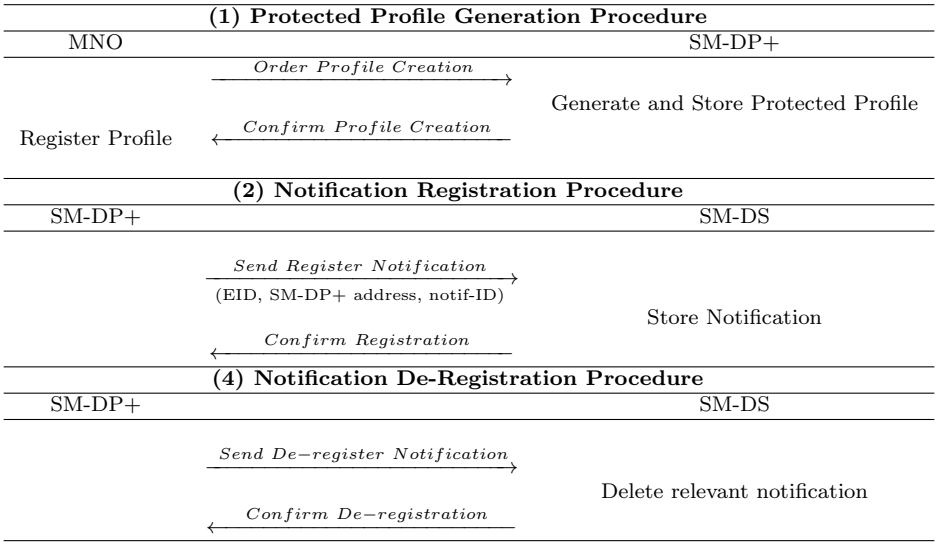

Figure 2.6: Profile Generation and Notification Registration/De-registration Procedures

corresponding SM-DS. If the eUICC is successfully authenticated, the SM-DS responds by providing the address of the SM-DP+ associated to that EID (*i.e.* the SM-DP+ that has generated the eUICC new profile). This procedure is depicted in Figure 2.7.

4. *Profile download*: the eUICC gets in touch with the relevant SM-DP+. After mutually authenticating each other, the eUICC can finally download its new profile. (see Figure 2.7).

5. *Notification De-Registration Procedure*: once the eUICC downloads its new profile, the SM-DP+ triggers this procedure so that the SM-DS can discard notifications that are not useful anymore (see Figure 2.6 -(4)).

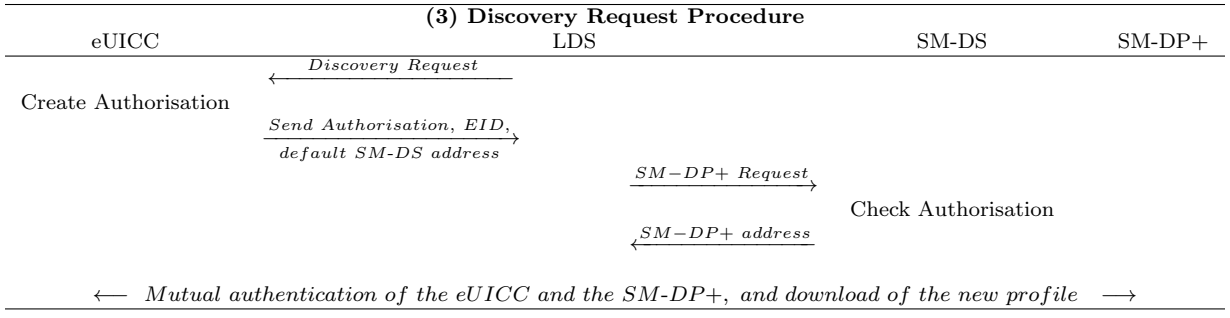

Figure 2.7: Discovery Request and Profile Download Procedures

2.4 M2M Security Issues

Although M2M have not induced new threats, it has however amplified the existing ones since, in the case of M2M applications, these threats may not only lead to financial losses but could also jeopardize human lives (*e.g.* if an attacker alters the information transmitted by the eCall application or a sensor in the case of eHealth applications). Indeed, as they rely on the fusion of heterogeneous networks, M2M communications have to cope with all the security threats of other network-based communications. Besides, similarly to Wireless Sensor Networks (WSNs), M2M devices are generally deployed in reachable locations and expected to operate for extended periods. This makes them particularly vulnerable to several physical attacks conducted against

the device hardware and software [CSS+09]. Furthermore, sensed data, which are forwarded to a remote server through wireless or wired networks, may also be subject to several threats. Last but not least, attacks against the proper functioning of the system can also be performed.

In this section, we present the main threats against M2M communications and devices while classifying them in three categories depending on the targeted entity; namely the M2M device, the proper functioning of the system or the exchanged data. These categories are respectively referred to as physical, logical and data attacks. One can also classify the potential attacks as active or passive according to whether they disrupt the operations of M2M applications or not.

2.4.1 Physical Attacks

This first category includes the different attacks that target the physical layer as well as M2M devices hardware and software. Hereinafter, we describe the most important ones.

- *Side channel attacks* [Law09]: M2M devices are generally deployed in reachable locations where adversaries can easily access them and perform side channel attacks. These attacks could be based on either power consumption, timing information, fault or electromagnetic leaks. Using one (or a combination) of these techniques, the attacker's goal consists in recovering the user's secret key. In fact, if he manages to retrieve the key used to encrypt exchanged information, he would be able to decrypt all the exchanged data.

- *Software modification and malwares*: Software modifications can be performed by an adversary, or even a malicious user, so as to alter the proper operations of the M2M device and service. Malicious users may do so to reduce the amount of charges that they have to pay. Such misbehaviour could be noticed when dealing with smart metering or eToll applications. Note that these attacks may be conducted even without physical access to the M2M device since software updates can be performed Over The Air (OTA). The impact of such threats worsens when it comes to eHealth or automotive applications especially if the adversary succeeds in taking the control of the device.

- *Destruction or theft of the M2M device*: As they are deployed in reachable locations, M2M devices (or the SIM cards included within them) can be easily stolen or destroyed. It is, however, worth mentioning that the stealing of eSIMs is not possible as they are generally welded to the M2M device (see Section 2.3).

2.4.2 Logical Attacks

This category encompasses attacks that may target the proper functioning of the system without making any changes to the device software. In what follows, the most significant threats are briefly described.

- *Impersonation*: An adversary may try to spoof the identity of any of the entities involved in an M2M application (*i.e.* back-end server, M2M device or a gateway and so forth). Such attack may lead to important financial losses and could even endanger human lives (*e.g.* eHealth applications). Indeed, an adversary who succeeds in spoofing a smart meter identity can make its owner pay for the adversary's charges. It is even worse if the adversary manages to impersonate the server as he would be able to remotely control the associated M2M devices. Through identity spoofing, the adversary can also launch other attacks (*e.g.* spread malwares) and cause Denial of Service as explained in what follows.

- *Denial of Service (DoS)* [YGS15]: Most M2M devices are battery powered. Thus, an adversary may cause application failure by repeatedly broadcasting meaningless packets that will prematurely drain the device battery. Such attacks may be conducted, for example,

when dealing with *intrusion detection* applications in order to prevent the user from being notified. Furthermore, DoS attacks may also intend to cause the unavailability of resources. For most applications, this may lead to significant financial losses and it can even result in blackouts when dealing with *smart grid*. Note that all the involved entities may be subject to a DoS attack whether it be the device, the gateway, the underlying infrastructure or even the remote server.

- *Relay attacks*: An adversary may conduct a relay attack to make an entity believe that it is in the vicinity of the sender or receiver. This attack may target the device, gateway or the network domain. Adversaries generally use such an attack in order to get the response corresponding to a given challenge. It may be a challenge sent by a user's smart lock, for example. By providing a correct answer to the challenge, the adversary unlocks the smart lock and can thus access the user's home.

2.4.3 Data Attacks

This category includes the threats that may target the exchanged information. Among the main attacks, we note *privacy attacks*, *data modification and false information injection* as well as *selective forwarding/interception* presented below.

- *Eavesdropping*: It is a passive attack (*i.e.* the effective operations of an M2M application are not disrupted) during which the attacker attempts to learn sensitive information. Usually, such an attack is difficult to detect. Besides, the large amount of data exchanged by M2M applications encourages attackers to eavesdrop communications as collected data could be used for commercial or fraud purposes.

- *Privacy attacks* [JKD12]: Owing to the pervasiveness of M2M devices and applications, malicious parties can invade user's privacy by linking M2M devices or the transiting information to individuals. This may enable the disclosure of personal information such as habits, health condition, religion, and so forth. Indeed, if the device Media Access Control (MAC) address indicates that it is a heart monitoring device, for example, then the adversary will know that its owner suffers from heart problems. Furthermore, some applications such as Pay-As-You-Drive (PAYD) insurance or electronic Toll (eToll), which keep a record of user's localization information, entail serious privacy concerns as they enable the adversary (which could be the service provider) to track users' journeys.

- *Data modification and false information injection*: Gathered information can be compromised during its transmission as well as at rest on a device or an application server. If we consider the case of eHealth (respectively eCall applications), the modification of measured values (localization information) can endanger people's lives. As regards to false data injection attacks [Yu12], they mainly cause financial losses. Further to a fake eCall, for example, the emergency services would be called upon and get to a place without real need.

- *Selective forwarding/interception*: An adversary may intercept and delay or drop some of the transmitted packets. The impact of such a threat depends on the content of the dropped packets. If the dropped information originates from a sensitive application, such as eHealth or eCall, the impact of such a threat can be quite significant.

2.5 Security Requirements

To prevent the threats that M2M communications and devices may face, a set of security properties should be guaranteed. They mainly consists of:

- *Authentication:* it may refer to either entity authentication or data origin authentication. More precisely, entity authentication enables communicating parties to check that the other entity is really who it claims to be whereas data origin authentication, as it name implies, ensures that a message really originates from a given entity.

- *Confidentiality:* it protects the content of gathered as well as exchanged information by preventing them from being read by unauthorized entities such as eavesdroppers.

- *Device and Data Integrity:* it keeps devices as well as, transiting and stored, data from being altered by any illegitimate entity.

- *Availability:* it ensures that authorized entities can always have access to a given information (or application) whenever needed.

- *Non-repudiation:* it makes sure that an entity cannot subsequently falsely deny a given action (*e.g.* sending a packet, triggering a given command, etc).

- *Privacy:* it prevents the disclosure of any sensitive or personal information such as habits and health status.

The different cryptographic primitives aiming to ensure at least one of these properties will be detailed further on (see Section 3.3).

2.6 M2M Characteristics and Key Challenges

Withstanding the security threats listed above is by no means an easy task owing to the different characteristics and challenges of M2M communications and devices. Indeed, the specific nature of M2M networks generally comprising thousands of devices, that are left unattended and expected to operate for several years, makes securing them a tricky task. Among these challenges, we identify the following:

- **Scalability:** A top concern when dealing with M2M communications is scalability. Indeed, the produced data as well as the network traffic will rise as the number of devices increases, thus leading to scalability issues. Let us consider, for example, the case of several devices trying to simultaneously perform network authentication. Since current mobile technologies have not been designed to withstand such a large number of devices, this would probably entail network congestion. Therefore, the design of scalable authentication mechanisms is crucial particularly in the case of real-time applications. Furthermore, some applications (*e.g.* city automation) involve a large number of M2M devices that usually secure their communications using the same key. In such case, a scalable key management scheme that enables the establishment/update of the shared key following membership changes (join/leave of devices) is required. In contrast, in the case of pairwise keys, a large number of keys has to be stored (*i.e.* a distinct key associated to each device). An efficient M2M security scheme ought to achieve a better trade-off between performances, security and number of required keys.

- **Devices heterogeneity:** Devices capabilities in terms of processing and energy vary a lot, ranging from very resource constrained devices such as sensors and actuators to more powerful devices like smartphones. Thus, when designing a security mechanisms, one must take into account the characteristics of the different devices. It would be even better if the designed scheme could leverage the capabilities of more powerful devices.

- **Resource constraints:** Due to their low cost, most M2M devices suffer from resource constraints (energy, storage and computing). These constraints make both energy efficiency and lightweight design fundamental features of any scheme aiming to secure M2M communications.

- **Various types of end-to-end communications:** The end-to-end communication differs from an M2M application to another. Indeed, we may have one-to/from-one, one-to/from-many as well as many-to/from-many communications. Thus, the security mechanisms must be suitable for both peer-to-peer and group communications.

- **Delay constraints/real-time communication:** Some M2M applications such as eHealth or eCall are life-critical and thus sensitive to delay. Therefore, the set-up of a security mechanism must not impact the Quality of Service (QoS) of the corresponding applications.

- **Limited bandwidth:** Given bandwidth scarcity and the expected number of M2M devices, the additional overhead related to, *e.g.*, the key generation and authentication mechanisms should be minimized to the fullest extent possible.

- **Access Control**: Within the same application, we may have two or more stakeholders with different access rights. It is, therefore, essential to limit access to gathered data to only authorized stakeholders (*i.e.* one should not be able to read any data that he is not allowed to).

- **Robustness:** The M2M devices are sometimes deployed in unreachable areas which requires the security mechanisms to be fault tolerant. Moreover, in other cases, M2M devices may be easily accessible by malicious parties. Consequently, tamper resistant hardware is recommended for certain applications.

Table 2.1: M2M Applications Characteristics and Requirements

Application	Type of communication	Delay constraint	Limited energy	Privacy
eHealth	Many-to-One	✓	✓	✓
eCall		✓		
Smart Home			✓	
Monitoring			✓	
Smart Grid		✓		✓
PAYD	One-to-One			✓
eToll		✓		✓
bCall				
Fleet management	One-to-One (for UL) One-to-Many (for DL)			✓
Smart metering		✓	✓	✓
EVC	Many-to-One			✓
Smart City	Many-to-Many		✓	

It is obvious that the characteristics and requirements of M2M applications make the design of a security scheme a challenging task. Unfortunately, it does not seem feasible to set up a solution that is suitable for all the applications due to their different characteristics and requirements as shown in Table 2.1. It is worth emphasizing that the availability of the application and integrity of exchanged data are always required. Confidentiality is also essential except when dealing with some smart city applications such as air pollution monitoring. Besides, almost all devices suffer from limited computational resources.

It is also noteworthy to mention that applications characteristics vary a lot depending on the scenario. Let us, for example, consider smart metering applications. Indeed, water and gas meters are subject to energy constraints while it is not the case of electricity smart meters as they are connected to the mains supply. Similarly, some fleet management applications require

privacy-preserving solutions (*e.g.* when dealing with car rental). Nevertheless, it is not the case of all fleet management applications (*e.g.* public transport vehicles).

2.7 Conclusion

In this chapter, we first presented the most important M2M applications. Then, we reviewed both ETSI M2M architecture as well as the eSIM concept and the associated remote SIM provisioning architecture proposed by the GSMA. We also addressed the challenges and threats that arise when dealing with M2M communications and devices along with M2M characteristics that makes it challenging to design new security schemes. Due to the heterogeneity of devices and the different requirements associated to each application, a single security solution cannot be both optimal and suitable for all applications.

In this thesis, we focus on M2M devices that include a SIM card and more precisely on two security properties, namely authentication and privacy. Before detailing the designed cryptographic primitives and protocols, we introduce, in the following chapter, the main mathematical tools and building blocks required all along this thesis.

Chapter 3

Mathematical Tools and Cryptographic Building Blocks

Contents

In this chapter, we introduce the main mathematical tools and cryptographic building blocks used in this thesis. We start by presenting groups and fields as well as the classical computational assumptions upon which we rely to prove the security of our proposed schemes and protocols. Then, we define the concept of provable security and some of the widely used security models. Finally, we formally present the necessary cryptographic primitives while briefly recalling the cryptographic schemes that we will use to design our new schemes and protocols.

3.1 Preliminaries

3.1.1 Mathematical Tools

3.1.1.1 Group

Definition 3.1 (Group)**.** A *group* $(\mathbb{G}, \cdot)$ consists of a nonempty set $\mathbb{G}$ and a binary operation $\cdot : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ satisfying the following properties.

- *Associativity:* $\forall\ (x, y, z) \in \mathbb{G}^3$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

- *Identity element:* $\exists\ e \in \mathbb{G}$ such that $\forall x \in \mathbb{G}$, $x \cdot e = e \cdot x = x$. e is unique and is called the *identity element* of $\mathbb{G}$, denoted by $1_{\mathbb{G}}$.
- *Inverse:* $\forall\ x \in \mathbb{G}$, $\exists\ y \in \mathbb{G}$ such that $x \cdot y = y \cdot x = e$. y is unique and is called the *inverse* of x, denoted by x^{-1}.

Definition 3.2 (Subgroup)**.** Let $(\mathbb{G}, \cdot)$ be a group and $\mathbb{H}$ a subset of $\mathbb{G}$. $(\mathbb{H}, \cdot)$ is a subgroup of $(\mathbb{G}, \cdot)$ if the following three requirements are satisfied.

- $1_{\mathbb{G}} \in \mathbb{H}$.
- $\forall\ (x, y) \in \mathbb{H}^2$, $x \cdot y \in \mathbb{H}$.
- $\forall\ x \in \mathbb{H}$, $x^{-1} \in \mathbb{H}$.

For simplicity, in the sequel, the group $(\mathbb{G}, \cdot)$ will be denoted by $\mathbb{G}$. Any subgroup of a group is itself a group.

Definition 3.3.

- *Commutativity:* A group $\mathbb{G}$ is *abelian* or *commutative* if the binary operation is commutative i.e. $\forall (x, y) \in \mathbb{G}^2, x \cdot y = y \cdot x$.
- *Subgroup generated by an element:* Let x be an element of the group $\mathbb{G}$, the set $\{x^n, n \in \mathbb{Z}\}$, denoted by $\langle x \rangle$, is called the *subgroup generated* by x
- *Cyclic group:* A group $\mathbb{G}$ is *cyclic* if $\mathbb{G}$ is generated by a single element i.e. $\exists x \in \mathbb{G}$ such that $\mathbb{G} = \langle x \rangle$. x is called *generator* of $\mathbb{G}$.
- *Finite group:* A group $\mathbb{G}$ is *finite* if $\mathbb{G}$, as a set, is finite.
- *Order of a group:* The number of elements of a finite group $\mathbb{G}$, denoted by $|\mathbb{G}|$, is called the *order* of $\mathbb{G}$.
- *Order of an element:* The order of an element x of a group $\mathbb{G}$, denoted by $|x|$, is, when the subgroup $\langle x \rangle$ is finite, the order of this subgroup i.e. the least positive integer n such that $x^n = 1$.

Any group of prime order is cyclic and any element $x \in \mathbb{G}$ other than $1_{\mathbb{G}}$ is a generator of $\mathbb{G}$. Throughout this thesis, we only consider abelian finite groups for which the multiplicative notation (binary operation $\cdot$) is frequently replaced by the additive notation (binary operation $+$) and $1_{\mathbb{G}}$ is replaced by $0_{\mathbb{G}}$.

3.1.1.2 Fields

Definition 3.4 (Ring)**.** A *ring* $(\mathbb{A}, +, \cdot)$ consists of a nonempty set $\mathbb{A}$ and two binary operations $+ : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ and $\cdot : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ satisfying the following properties.

- $(\mathbb{A}, +)$ is an abelian group.
- *Multiplicative associativity:* $\forall\ (x, y, z) \in \mathbb{A}^3$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.
- *Distributive multiplication over addition:* $\forall\ (x, y, z) \in \mathbb{A}^3$, $x \cdot (y + z) = x \cdot y + x \cdot z$ and $(x + y) \cdot z = x \cdot z + y \cdot z$

If $(\mathbb{A}, +, \cdot)$ is also multiplicatively commutative, then it is called a commutative ring. $(\mathbb{A}, +, \cdot)$ is said to be *unitary* if it includes an identity element $1_{\mathbb{A}}$ for the binary operation "$\cdot$" that is different from the identity element $0_{\mathbb{A}}$ of the binary operation "$+$".

Definition 3.5 (Field)**.** A *field* $(\mathbb{K}, +, \cdot)$ consists of a nonempty set $\mathbb{K}$ and two binary operations $+ : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$ and $\cdot : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$ satisfying the following properties.

- $(\mathbb{K}, +, \cdot)$ is a unitary ring.
- *Multiplicative inverse:* Let $\mathbb{K}^*$ denote $\mathbb{K} \backslash \{0_{\mathbb{K}}\}$. $\forall\, x \in \mathbb{K}^*$, $\exists\, y \in \mathbb{K}^*$ such that $x \cdot y = y \cdot x = 1_{\mathbb{K}}$. y is unique and denoted by x^{-1}.

A field $(\mathbb{K}, +, \cdot)$ is finite if $\mathbb{K}$ is a finite set. As a result, $\mathbb{K}^*, \cdot)$ is a group. $|\mathbb{K}|$, the number of elements of $\mathbb{K}$, is the *order* of the field. If $(\mathbb{K}, +, \cdot)$ is commutative as a ring, then it is called a commutative field. A field *characteristic* is, when it exists, the smallest positive integer n such that $\underbrace{1 + 1 + \ldots + 1}_{n \text{ times}} = 0$. If there is no such n, then the field is said to have a characteristic equal to 0. Any finite field $(\mathbb{K}, +, \cdot)$ also called *Galois field*, satisfies the following properties:

- $(\mathbb{K}, +, \cdot)$ is commutative.
- The multiplicative group $(\mathbb{K}^*, \cdot)$ is cyclic.
- The order of $(\mathbb{K}, +, \cdot)$ is a power of a prime. This means that it is of the form p^n where p is the characteristic and n is an integer such that $n \geq 1$. Conversely, for each prime integer p and any positive integer $n \geq 1$, there exists exactly one (up to an isomorphism) finite field $(\mathbb{K}, +, \cdot)$ such that $|\mathbb{K}| = p^n$. Such a field is denoted by $\mathbb{F}_{p^n}$. Thus, if p is prime, $\mathbb{F}_p = \mathbb{Z}_p$. It is however noteworthy to mention that $\mathbb{F}_{p^n} \neq \mathbb{Z}_{p^n}$ when $n > 1$ as $\mathbb{Z}_{p^n}$ is not a field.

3.1.1.3 Elliptic Curves

During the eighties, Miller [Mil86] and Koblitz [Kob87] independently introduced Elliptic Curve Cryptography (ECC) whose security relies on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC has attracted a lot of attention as it provides the same security levels as non-ECC cryptography but with much smaller parameters such as key and modulus sizes. Hence, it enables memory and energy savings as well as faster computations. Since it requires less resources, it is better suited for resource constrained environments such as smart cards.

Definition 3.6 (Elliptic Curve over a finite field)**.** An elliptic curve E over a finite field $\mathbb{F}_{p^n}$ consists of a *point at infinity* $\mathcal{O}$ along with the set of pairs (x, y) verifying the following *Weierstrass* equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \text{ where } a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_{p^n}$$

If the prime number $p \neq 2, 3$, the *Weierstrass* equation can be simplified as follows:

$$y^2 = x^3 + a \cdot x + b \text{ where } (a, b) \in \mathbb{F}_{p^n}^2 \text{ and } 4a^3 + 27b^2 \neq 0.$$

The condition $4a^3 + 27b^2 \neq 0$ ensures that the curve is *non-singular*. In particular, this means that we can compute the tangent at every point, except $\mathcal{O}$, to the curve. Otherwise, the curve is said *singular*.

From an algebraic point of view, an elliptic curve is an abelian group with the point at infinity $\mathcal{O}$ defined as the identity element and the addition operation, denoted $+$, defined as follows.

Definition 3.7 (Addition law)**.** *Let P and Q be two different points on the elliptic curve $E(\mathbb{F}_{p^n})$.*

- *The inverse of $P = (x_P, y_P) \neq \mathcal{O}$ is denoted by $-P$ whose coordinates are $(x_P, -y_P) \in E(\mathbb{F}_{p^n})$. Additionally, $-\mathcal{O} = \mathcal{O}$, $P + \mathcal{O}P$ and $\mathcal{O} + P = P$.*

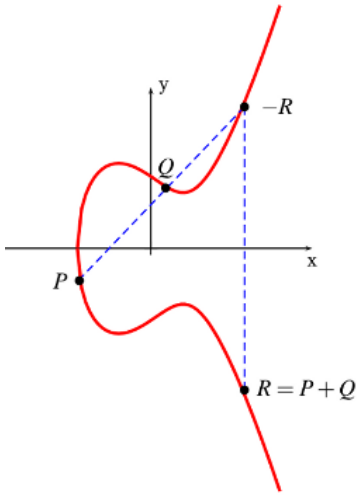

Figure 3.1: Addition of two distinct points P and Q on an elliptic curve

- *If $P \neq -P$, the tangent line to the curve E at point P intersects the curve at a second point $R = (x_R, y_R) \in E(\mathbb{F}_{p^n}) \setminus \mathcal{O}$. $-R$ is equal to the sum $P + P = [2]P$.*
- *If $P \neq -Q$, the line through P and Q intersects the curve E in a third point $R = (x_R, y_R) \in E(\mathbb{F}_{p^n}) \setminus \mathcal{O}$. $-R$ is equal to the sum of P and Q (see Figure 3.1).*

One can easily show that the addition law verifies all the required properties and that $(E(\mathbb{F}_{p^n}), +)$ is an abelian group.
The addition law of the group $(E(\mathbb{F}_{p^n}), +)$ can also be described algebraically. The sum of the points $P = (x_P, y_P) \neq \mathcal{O}$ and $Q = (x_Q, y_Q) \neq \mathcal{O}$ is the point $R = (x_R, y_R)$ with

$$x_R = \lambda^2 - x_P - x_Q \quad \text{and} \quad y_R = \lambda(x_P - x_R) - y_P$$

where λ is defined as:

- $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$ if $P \neq \pm Q$ and $P, Q \neq \mathcal{O}$
- $\lambda = \frac{3x_P^2 + a}{2y_P}$ if $P = Q$ and $P \neq \mathcal{O}$

3.1.1.4 Bilinear Maps

Bilinear maps, also known as pairings, first came to prominence in 1991 with the famous attack of Menezes, Vanstone and Okamoto [MVO91, MOV93], known as MOV reduction, where they were used to solve the elliptic curve discrete logarithm problem on some elliptic curves $E(\mathbb{F}_{p^k})$ having a small so-called "embedding degree" k. Later, in 2000, Joux [Jou00] presented the first positive application (for cryptographic rather than cryptanalytic purposes) of bilinear maps. Indeed, he proposed a one-round 3-party Diffie-Hellman key agreement protocol that relies on the use of bilinear maps. Following the work of Joux, many cryptographers started investigating the possibility of using bilinear maps. This led to the design of several revolutionary and powerful cryptographic schemes such as the Identity Based Encryption scheme of Boneh and Franklin [BF01] and the short signature scheme of Boneh, Lynn and Shacham [BLS01]. Hereinafter, we provide a formal definition of a bilinear map.

Definition 3.8 (Bilinear Map)**.** Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p. We say that the function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if it satisfies the following properties:

- *Bilinearity:* $\forall g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$.
- *Non-degeneracy:* $\forall g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$, if $e(g, \tilde{g}) = 1_{\mathbb{G}_T}$ then $g = 1_{\mathbb{G}_1}$ or $\tilde{g} = 1_{\mathbb{G}_2}$.
- *Computability:* The mapping e is efficiently computable. (We will define "efficiently computable" in Section 3.1.2.2).

A pairing is called *symmetric* if $\mathbb{G}_1 = \mathbb{G}_2$. Otherwise, it is said to be *asymmetric*. Note that $\mathbb{G}_1$ and $\mathbb{G}_2$ are always isomorphic. Two examples of well-known pairings are the Weil [Wei40] and Tate pairings [GF94].

Usually, the groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are chosen as follows. $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of an elliptic curve whereas $\mathbb{G}_T$ is a subgroup of a finite field. A careful choice of the elliptic curve enables an efficient and secure implementation of pairings. Galbraith, Paterson and Smart [GPS08] classified bilinear maps in three types depending on the existence (or not) of computable isomorphism(s) between $\mathbb{G}_1$ and $\mathbb{G}_2$:

- *Type I*: There exist two efficiently computable isomorphisms $\phi_1 : \mathbb{G}_1 \to \mathbb{G}_2$ and $\phi_2 : \mathbb{G}_2 \to \mathbb{G}_1$.
- *Type II*: There exists an efficiently computable isomorphism from $\phi : \mathbb{G}_1 \to \mathbb{G}_2$ but none in the reverse direction.
- *Type III*: There is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ in either direction.

Type III pairings are known to provide better performances than type I and type II pairings. Besides, in type III pairings, several computational assumptions hold while it is not the case with type I or type II pairings.

3.1.2 Basic Functions and Complexity Definitions

In this section, we briefly recall some basic complexity definitions and present some particular functions widely used in cryptography such as hash functions.

3.1.2.1 Turing Machines

In 1936, Alan Turing introduced a machine known today as *Turing machine* [Tur36]. These machines can simulate *any* computer algorithm regardless of its complexity. To date, they have been the most widely used model of computation in both computability and complexity theory. Roughly speaking, a Turing machine consists of:

- A *tape*, that extends infinitely to the right, divided into cells. Each cell stores a symbol belonging to a finite set called the tape alphabet and which contains a blank symbol.
- A *tape head* that can move along the tape (to the right or, if possible, to the left), one cell per move. It can also read the cell it is currently scanning and replace the cell symbol by another one.
- A finite set of *states* which contains three special states, namely a start state, an accept state, and a reject state.
- A finite set of *instructions*. Depending on its current state and the symbol it is reading, the "tape head" writes a new symbol in the cell it is scanning (which may be the same symbol), moves to the left or to the right (or possibly stays at the current cell), and enters a new state.

More formally, a Turing machine is defined as follows.

Definition 3.9 (Turing Machine)**.** A Turing machine is a 7-tuple $T = (\Sigma, \mathcal{Q}, \sigma, \delta, \Delta, q_0, \mathcal{F})$ where

- Σ is a finite and non-empty set of alphabet symbols.
- $\mathcal{Q}$ is a finite and non-empty set of states.
- $\sigma : \mathcal{Q} \times \Sigma \to \Sigma$ is the writing function.
- δ: $\mathcal{Q} \times \Sigma \to \mathcal{Q}$ is the state changing function.
- Δ: $\mathcal{Q} \times \Sigma \to \{L, R\}$ is the transition function where L is a left shift and R is a right shift.
- $q_0 \in \mathcal{Q}$ is the initial state.
- $\mathcal{F} \subset \mathcal{Q}$ is the set of final states.

For a given input x, a Turing machine can have several types of complexity classes. In this thesis, we mainly focus on time complexity, denoted by $TM(x)$, which is commonly estimated through the total count of the number of Turing machine steps from the initial to the final state.

3.1.2.2 Basic Complexity Definitions

An algorithm A is said to be *of polynomial time complexity* if there exists a polynomial $p(.)$ such that, given any input $x \in \{0,1\}^*$, the running time of A is upper bounded by $p(|x|)$ where $|x|$ is the size of the input x. More formally, polynomial time complexity is defined as follows.

Definition 3.10 (Polynomial Time Complexity)**.** Let $T_{\mathcal{M}}(n) = sup\{T_{\mathcal{M}(\mathcal{I})}, |\mathcal{I}| = n\}$ where $\mathcal{M}$ is a Turing Machine. The time complexity of $\mathcal{M}$ is said to be polynomial if

$$\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{N}^*, \text{ such that } \forall n \geq n_0, T_{\mathcal{M}}(n) \leq n^c$$

In the complexity theory framework, if an algorithm A is of polynomial time then it is considered as *efficient* and a problem is said to be *hard* if there is no known polynomial time algorithm that can solve it.

Definition 3.11 (Negligible Function)**.** A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible* if it vanishes faster than the reciprocal of any polynomial. That is,

$$\forall c > 0, \exists k_0 > 0 \text{ such that } \forall k \geq k_0, \epsilon(k) < \frac{1}{k^c}$$

Thereby, we get the following two definitions of negligible and overwhelming probabilities.

Definition 3.12 (Negligible Probability)**.** Let P be a probability that depends on a positive integer k (the security parameter). P is said to be negligible if it is a negligible function of k.

Definition 3.13 (Overwhelming Probability)**.** A probability P is said to be overwhelming if the probability $1 - P$ is negligible.

3.1.2.3 One-way Function

One-way functions are a fundamental tool for cryptography. They are easy to compute in one direction but hard to invert. More formally, a one-way function is defined as follows.

Definition 3.14 (One-way Function)**.** Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. f is said to be a *one-way function* if it satisfies the following two properties:

- *Efficiency:* There exists an efficient algorithm that, given any $x \in \{0,1\}^*$ as input, outputs $f(x)$.

- *Hardness to invert:* There is no efficient algorithm that can invert f. More formally, for any efficient algorithm $\mathcal{A}$ (also known as adversary), it holds that, for all sufficiently large k, the following probability is negligible.

$$\mathtt{Pr}[x \leftarrow \{0,1\}^k; y \leftarrow f(x); x' \leftarrow \mathcal{A}(1^k, y) : f(x') = y]$$

3.1.2.4 Hash Function

A hash function is a special one-way function that maps input strings of arbitrary length $\{0,1\}^*$ to short fixed length output strings $\{0,1\}^k$ where k is a security parameter. A specific kind of hash functions known as *cryptographically secure hash functions* is widely used in cryptography. Compared to ordinary hash functions, these hash functions ensures some additional security properties and can be defined as follows.

Definition 3.15 (Cryptographically secure hash function)**.** A hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$, where k is a security parameter, is said to be *cryptographically secure* if it meets the following three properties:

- *Pre-image resistance (one-way):* Given $y \in \{0,1\}^k$, the probability that an adversary finds a value x such that $\mathcal{H}(x) = y$ is negligible. That is, the following probability is negligible.

$$\mathtt{Pr}[y \leftarrow \{0,1\}^k; x \leftarrow \mathcal{A}(1^k, y) : y = \mathcal{H}(x)]$$

- *Second pre-image resistance:* Given $x \in \{0,1\}^*$, the probability that an adversary finds a value $x' \neq x$ such that $\mathcal{H}(x) = \mathcal{H}(x')$ is negligible. That is,

$$\mathtt{Pr}[x \leftarrow \{0,1\}^*; x' \leftarrow \mathcal{A}(1^k, x) : \mathcal{H}(x') = \mathcal{H}(x) \text{ and } x' \neq x]$$

  is negligible. This property is also known as *weak collision resistance.*

- *Collision resistance:* The probability to find two distinct values x,$x' \in \{0,1\}^*$ such that $\mathcal{H}(x) = \mathcal{H}(x')$ is negligible. That is,

$$\mathtt{Pr}[(x', x) \leftarrow \mathcal{A}(1^k) : \mathcal{H}(x') = \mathcal{H}(x)]$$

  is negligible. This property is also known as *strong collision resistance.*

Several hash functions have already been proposed such as MD5, SHA1, SHA256 and SHA3. However, only few of them are still considered as cryptographically secure. In this thesis, only cryptographically secure hash functions are used.

3.1.2.5 Pseudo-Random Function

A function is called *pseudo-random* if there exists no efficient algorithm that can distinguish between the output of this function and a random value.

Definition 3.16 (Pseudo-random function family)**.** Let m and l be two polynomial functions, k a security parameter and $\mathcal{R}_k$ the set of functions:

$$\{0,1\}^{m(k)} \rightarrow \{0,1\}^{l(k)}$$

The family $\mathcal{F}_k = \{F_s\}_{s\in\{0,\ 1\}^k}$ is defined as the set of functions with an index s ($\mathcal{F}_k \subseteq \mathcal{R}_k$). $\mathcal{F}_k$ is called a pseudo-random function family if it satisfies the following two properties:

- *Efficiency:* $\forall\ s \in \{0,1\}^k$ and $x \in \{0,1\}^{m(k)}$, there exists an efficient algorithm F such that $F(s,x) = F_s(x)$.
- *Pseudo-randomness:* For any adversary $\mathcal{A}$ such that the number of its requests to F is polynomially limited, the following probability (where $R \in \mathcal{R}_k$) is negligible.

$$|\mathtt{Pr}[x \leftarrow \{0,1\}^{m(k)} : \mathcal{A}(F_s(x)) = 1] - \mathtt{Pr}[x \leftarrow \{0,1\}^{m(k)} : \mathcal{A}(R(x)) = 1]|$$

3.1.3 Computational Hardness Assumptions

In this thesis, we rely on a set of classical computationally hard assumptions that are defined in a cyclic group of prime order. These latter will enable us to prove the security of the designed schemes/protocols as subsequently explained in section 3.2. Hereinafter, we recall the computational assumptions that we use. Depending on the context, the multiplicative or additive notation for groups will be used.

Definition 3.17 (Discrete Logarithm (DL) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p and g a generator of $\mathbb{G}$. The Discrete Logarithm assumption states that, given $y \in \mathbb{G}$, it is hard to find the integer $x \in \mathbb{Z}_p$ such that $g^x = y$. The integer x corresponds to the discrete logarithm of y in the base g and is usually denoted by $x = log_g(y)$.

Definition 3.18 (One-More Discrete Logarithm (OMDL) Assumption)**.** Let $\mathbb{G}$ be a multiplicative group of prime order p, g a generator of $\mathbb{G}$, $\mathcal{O}_1$ a challenge oracle that returns a random element $Y \in \mathbb{G}$ and $\mathcal{O}_2$ a discrete logarithm oracle with respect to the base g. The One-More Discrete Logarithm assumption states that, after t queries to $\mathcal{O}_1$ (where t is chosen by the adversary) and **at most** $t-1$ queries to $\mathcal{O}_2$, it is hard to retrieve the discrete logarithms of all received t elements.

Definition 3.19 (Decisional Diffie-Hellman (DDH) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p. The Decisional Diffie-Hellman assumption states that, given a generator $g \in \mathbb{G}$, two elements $g^a, g^b \in \mathbb{G}$ and a candidate $X \in \mathbb{G}$, it is hard to decide whether $X = g^{ab}$ or not.

The DDH assumption can also be defined as follows:

Definition 3.20 (Decisional Diffie-Hellman' (DDH′) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p. The Decisional Diffie-Hellman' assumption states that, given two generators $g, h \in \mathbb{G}$ and two elements $g^a, h^b \in \mathbb{G}$, it is hard to decide whether $a = b$ or not.

Definition 3.21 (Gap Diffie-Hellman (GDH) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p. The Gap Diffie-Hellman assumption states that, given a generator $g \in \mathbb{G}$, two elements $g^a, g^b \in \mathbb{G}$ and with the help of a DDH oracle (which indicates whether a given quadruple $(g, h, g^a, g^b) \in \mathbb{G}_4$ is a valid Diffie-Hellman quadruple or not), it is hard to compute the element $C = g^{ab} \in \mathbb{G}$.

Lysyanskaya, Rivest, Sahai and Wolf put forth in [LRSW00] the LRSW assumption so as to prove the security of their anonymous credential system. Their assumption, defined hereinafter, is proven in the generic group model (introduced in Section 3.2.3.3).

Definition 3.22 (LRSW Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order q and g be a generator of $\mathbb{G}$. The LRSW assumption states that, given $X_0 = g^{x_0}$, $X_1 = g^{x_1}$ and an oracle $\mathcal{O}^{\mathsf{LRSW}}_{X_0,X_1}$ that takes on input a message $m \in \mathbb{Z}_q$ and outputs $A = (a, a^{x_1}, a^{x_0+mx_0x_1})$ such that a is randomly chosen, it is hard to compute $A' = (a', a'^{x_1}, a'^{x_0+mx_0x_1})$ for another value a', if m has not been queried to the oracle $\mathcal{O}^{\mathsf{LRSW}}_{X_0,X_1}$.

To prove the security of their signatures schemes, Pointcheval and Sanders [PS16] introduced two new assumptions which are variants of the classical LRSW assumption in type-3 bilinear groups. In what follows, we recall the first assumption which is proven secure in the generic group model (see Section 3.2.3.3).

Definition 3.23 (Assumption 1)**.** Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ be a bilinear group setting of type 3, with h and $\tilde{h}$ two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, which are of prime order p. The assumption 1 states that, given $(h, X_1, \tilde{h}, \tilde{X}_0, \tilde{X}_1)$ where $X_1 = h^{x_1}$, $\tilde{X}_0 = \tilde{h}^{x_0}$ and $\tilde{X}_1 = \tilde{h}^{x_1}$ such that x_0,$x_1 \in_R \mathbb{Z}_p^*$ and having an unlimited access to an oracle $\mathcal{O}_1$ which, on input $m \in \mathbb{Z}_p$, chooses a random $u \in \mathbb{G}_1$ and outputs the pair $P = (u, u^{x_0+mx_1})$, it is hard to efficiently generate a valid pair P, with $u \neq 1_{\mathbb{G}_1}$, for a new m^* that has not been queried to $\mathcal{O}_1$.

Definition 3.24 (eXternal Diffie-Hellman (XDH) Assumption)**.** Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear map. The eXternal Diffie-Hellman assumption, denoted by XDH, states that the DDH assumption holds in $\mathbb{G}_1$.

To prove the security of their short signature scheme (*cf.* Section 3.3.2.5), Boneh and Boyen [BB04] introduced, in 2004, the q-Strong Diffie-Hellman ($q-$SDH) assumption. Since then, it has been used as an underlying basis to prove the security of several pairing-based protocols. This assumption is defined as follows.

Definition 3.25 (q-Strong Diffie-Hellman ($q-$SDH) Assumption)**.** Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p, generated by g_1 and g_2 respectively. In the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, the q-Strong Diffie-Hellman assumption states that, given $(g_1, g_1^y, g_1^{y^2}, \ldots, g_1^{y^q}, g_2, g_2^y) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ as input, it is hard to output a pair $(x, g_1^{\frac{1}{y+x}}) \in \mathbb{Z}_p \setminus \{-y\} \times \mathbb{G}_1$.

It is noteworthy to mention that, when $\mathbb{G}_1 = \mathbb{G}_2$, the pair (g_2, g_2^x) is redundant as it can be computed by raising (g_1, g_1^x) to a random power.

The $q-$SDH assumption is believed to be hard even in gap-DDH groups, *i.e.* groups in which there is an efficient test to determine, with probability 1, on input (g, h, g^x, h^y) if $x = y \bmod p$ or not. For instance, a cyclic group $\mathbb{G}_1$ equipped with a symmetric bilinear map is a good example of a gap-DDH group since one can verify if the quadruple (g, h, g^x, h^y) is a DH quadruple by checking if $e(g, h^y) = e(g^x, h)$.

It has been proven in [FPV09] that the hardness of the $q-$SDH assumption in gap-DDH groups implies the one of the *gap* $q-$SDH$-$III assumption defined as follows.

Definition 3.26 (Gap q-Strong Diffie-Hellman (*gap* $q-$SDH$-$III) Assumption.)**.** *Let $\mathbb{G}$ be a cyclic group of prime order p. The Gap q-Strong Diffie-Hellman assumption states that, given $(g, h, g^y) \in \mathbb{G}^3$ and q distinct triplets $(r_i, m_i, A_i = (g^{m_i}h)^{\frac{1}{y+r_i}}) \in \mathbb{Z}_p^2 \times \mathbb{G}$ and having access to a* DDH *oracle (that indicates whether a given quadruple $(g, h, g^x, h^y) \in \mathbb{G}^4$ is a DH quadruple or not), it is hard to output a* new *triple $(r, m, A = (g^m h)^{\frac{1}{y+r}})$ where $(r, m) \in \mathbb{Z}_p^2$.*

Definition 3.27 (q-Decisional Diffie-Hellman Inversion ($q-$ DDHI) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p. The q-Decisional Diffie-Hellman assumption states that, given a generator $g \in \mathbb{G}$ and the values $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}) \in \mathbb{G}^{q+1}$, for a random $\alpha \in \mathbb{Z}_q$ and a candidate $X \in \mathbb{G}$, it is hard to decide whether $X = g^{\frac{1}{\alpha}}$ or not.

Definition 3.28 (Decisional Composite Residuosity (DCR) Assumption)**.** The DCR assumption states that it is hard to distinguish $\mathbb{Z}_{n^2}^n$ from $\mathbb{Z}_{n^2}^*$ where $\mathbb{Z}_{n^2}^n = \{z \in \mathbb{Z}_{n^2}^* \text{ such that } \exists y \in \mathbb{Z}_{n^2}^* : z = y^n mod\ n^2\}$, the set of n^{th} *residues*.

3.2 Provable Security

Whenever designing a new cryptographic primitive, it is of utmost importance to prove its security, in a complexity theory sense, by showing that it is hard to break by a well-defined class of attackers. Such a proof provides useful confidence in the protocol security.

Usually, cryptographers follow three steps to prove the security of a scheme:

1. Define the set of security properties that ought to be met to ensure the security of the scheme.

2. Specify the computational hardness assumption(s) to be considered.

3. Reduce the security of the scheme to this (these) hard problem(s) defined through computational hardness assumption(s).

Hereinafter, we define what is meant by *"reduce the security of the scheme"*.

3.2.1 Reductionist Security

The concept of provable security was first introduced by Goldwasser and Micali [GM84] in 1984 to prove the security of an asymmetric encryption scheme. Typically, the security of a cryptographic scheme (or the fact that it ensures a given security property) is formally proven by *reducing* the problem of breaking its security to the problem of breaking an underlying known hard problem such as those defined in section 3.1.3. This simply means that if an adversary $\mathcal{A}$ can break a security property of a scheme, then he can be used as subroutine to solve a hard problem.

Shortly afterwards, in a joint work with Rivest [GMR88], they provided the definition of security for a digital signature scheme. Unfortunately, even though they are polynomial, the reductions are computationally costly and completely unpractical. Thus, improvements were required. It was in this spirit that Bellare and Rogaway introduced, in 1996, the concept of *exact security* [BR96]. This approach aims to explicitly and accurately estimate the computational complexities of adversarial tasks. Following this work, Ohta and Okamoto proposed the reduction technique known as *concrete security* [OO98] which provide more efficient security results. More recently, the concept of *practical security* was introduced by Pointcheval [Poi02].

To sum up, if we consider an adversary $\mathcal{A}$ reaching his goal within time t and a reduction which solves the hard problem within $t' = f(t)$, three security concepts can be defined:

- *Asymptotic security*: f is polynomial (bounded by a t polynomial).

- *Exact security*: f is explicit.

- *Practical security*: f is "small" (*e.g.* linear).

3.2.2 Game-based and Simulation-based Security

There are two prevalent types of security proofs, namely *game-based* and *simulation-based*. The former, as its name implies, involves a game (or experiment) between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, whereas the latter relies on an *ideal* functionality. In what follows, we give a brief overview on both game-based and simulation-based security.

3.2.2.1 Game-based Security

In the game-based approach, which is widely used, a security property is formalized through a game, also called *security experiment*, between an efficient adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. To model the adversary's real capabilities, we provide him access to a set of oracles and his goal is to solve the game with non-negligible probability. The difference between the adversary's probability of wining the game and the probability of wining the game through random guessing is known as the *adversary's advantage*. A security property is satisfied as long as the adversary's advantage of wining the corresponding game is negligible.

To simplify game-based security proofs and make them more easily verifiable, Shoup [**?**] suggested in 2004 a new approach for carrying out reductionist security proofs. His technique, commonly referred to as *game hopping technique*, consists in organizing a proof as a sequence of games. Hereinafter, we briefly describe it.

Shoup's game hopping technique. It is useful in the case of complex cryptographic protocols (*e.g.* group key establishment protocols) and works as follows. First, the real attack game (Game 0) with respect to a given adversary is formally defined. Then, a sequence of games (Game 1,...,Game n) are constructed such that the adversary cannot detect the changes between two consecutive games. The last game (Game n) is constructed so that the probability that the event S_n occurs is negligibly close to the target probability (either $\mathtt{0}$ or $\frac{1}{2}$). Let S_i denote the event that an adversary wins Game i. By construction, $\mathtt{Pr}[S_i]$ is negligibly close to $\mathtt{Pr}[S_{i+1}]$. Thus, the probability that event S_0 occurs is negligibly close to the probability of the event S_n. Thereby, one can conclude and prove the security of the scheme. It is however worth mentioning that, depending on the real attack game, this technique may be sometimes difficult to apply.

3.2.2.2 Simulation-based Security

Simulation-based security, in the sense of secure two/multi-party computation, rely on the real-ideal world paradigm. More precisely, in addition to the real world, one considers an ideal world where an *ideal functionality*, denoted by $\mathcal{F}$, is defined such that all interactions between the different parties and the adversary, in the ideal world, are made via $\mathcal{F}$. In such a setting, a protocol is considered secure if the view of the protocol execution in the real world, denoted $view_{Real}$, is computationally indistinguishable from the view of the functionality execution in the ideal world, denoted by $view_{Ideal}$. By definition, everything goes well in the ideal world and no attack will succeed. Thus, to prove the security of a protocol, one must build a simulator $\mathcal{S}$ that will emulate a probabilistic polynomial time adversary in the ideal world.

Simulation-based proofs present a significant advantage over game-based proofs as they lead to "composable" notions of security. That is, if one uses a set of protocols independently proven secure (through simulation-based proofs) as building blocks of a complex protocol, then the resulting "composed" protocol is secure [Can01].

3.2.3 Security Models

A security proof mainly depends on the underlying security model. A scheme proven secure in a given security model may be insecure in another one. Indeed, the security model specifies the

set of security properties that should be fulfilled to ensure the security of the scheme (*i.e.* what it actually means to be secure).

To get efficient and practical schemes that are proven secure, one must sometimes make some *ideal* assumptions. One way consists in proving that the scheme is secure in an ideal setting (*e.g.* in the random oracle model) rather than in the standard model. It is, however, worth to mention that a security proof in an idealized model does not provide the same level of security guarantees as a proof in the standard model. Hereinafter, we briefly define those two models as well as the generic group model.

3.2.3.1 Random Oracle Model (ROM)

In cryptography, one of the most widely accepted ideal models is the *random oracle model* whose concept was first introduced by Fiat and Shamir [FS86] in 1986. Later, in 1993, it was formalized by Bellare and Rogaway [BR93]. The basic idea of the random oracle model is to assume that hash functions behave like ideal random functions. Thereby, whenever an adversary wants to compute the output of a hash function, he needs to query a random oracle. Indeed, a security proof in the random oracle model does not depend on a specific hash function as this latter is replaced by a random oracle whose output is indistinguishable from the output of a perfectly random function. When queried twice with the same input, the random oracle outputs the same value it previously returned for that input. If a real attacker against a scheme proven secure in the ROM succeeds, then he has necessarily exploited a weakness of the hash function.

3.2.3.2 Standard Model

The random oracle model has been subject to a lot of criticism due to its limitations. Canetti, Goldreich and Halevi showed in [CGH98, CGH04] the impossibility of implementing random oracles. Indeed, some schemes proven secure in the random oracle model turned out to be totally insecure when implemented with *any* hash function. Therefore, cryptographers had to define a new model that does not rely on any idealized assumption so as to prove the security of new schemes. Such a model is referred to as the *Standard Model*. This model allows to achieve much stronger security guarantees as schemes security relies solely on computational assumptions. Unfortunately, this comes at the cost of less efficient schemes.

3.2.3.3 Generic Group Model (GGM)

The generic group model [Nec94, Sho97, Mau05] is another idealized model which assumes that the relevant group has no special structure or feature that can be exploited by an adversary. Indeed, in this model, the adversary must query an oracle to perform any group operation. The generic group model is usually used to define the security level provided by new computational hardness assumptions as well as new cryptosystems.

3.3 Cryptographic primitives

In this section, the main cryptographic primitives required throughout this thesis are formally defined. We first address a symmetric key cryptographic primitive, namely Message Authentication Code (MAC), then move to the asymmetric setting and more precisely digital signature, public key encryption, commitments and proofs of knowledge.

3.3.1 Message Authentication Code

Message Authentication Code, commonly known as MAC, is one of the most basic cryptographic primitives. It is a symmetric technique that provides message authentication as well as message

integrity by simply appending a tag, referred to as MAC, to the corresponding message. This MAC is computed using a secret key sk that is shared between the sender and the receiver. In what follows, we provide a more formal definition of a MAC scheme.

Definition 3.29 (Message Authentication Code)**.** *A Message Authentication Code (MAC) scheme consists of the following four algorithms:*

- $\texttt{Setup}(1^{\texttt{k}})$: *a probabilistic algorithm, on input an integer k, outputs the public parameters pp.*
- $\texttt{KeyGen}(\texttt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs a secret key sk.*
- $\texttt{MAC}(\texttt{pp}, \texttt{sk}, \texttt{m})$: *an algorithm, on input the public parameters pp, a message $m \in \{0,1\}^*$ as well as the secret key sk, outputs a valid tag τ.*
- $\texttt{Verify}(\texttt{pp}, \texttt{sk}, \texttt{m}, \tau)$: *a deterministic algorithm, on input the public parameters pp, the secret key sk, a message m and a tag τ, outputs either* `1` *(*accept*) or* `0` *(*reject*).*

A message authentication code scheme must satisfy the following two properties:

- *Authenticity*: Anyone holding the secret key sk that has been used to generate a MAC can verify its validity.
- *Integrity*: If any changes are made to the original message m, then the MAC becomes invalid (with overwhelming probability).
- *Validity*: If the tag τ is valid with respect to both m and sk, then $\texttt{Verify}(\texttt{pp}, \texttt{sk}, \texttt{m}, \tau)$ should output `1` (with overwhelming probability).

To date, several MAC schemes have already been proposed. Some of them are *deterministic* while others are *probabilistic*. Hereinafter, we define both types as well as the condition under which each one of them is considered secure.

3.3.1.1 Deterministic MAC schemes

When considering a deterministic MAC scheme, for a given message m, there exists a sole valid authentication tag τ. Typically, a deterministic MAC should be *unforgeable under chosen message attack* (UF-CMA). That is, it should be hard for an adversary $\mathcal{A}$, that can make `MAC` queries, to produce a valid tag τ on a message m that has not been queried. Figure 3.2 provides a more formal definition of the UF-CMA experiment. The security experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ mainly consists of the following three steps:

- *Pre-Query*: $\mathcal{C}$ executes the `Setup` and `KeyGen` algorithms to get pp as well as the secret key sk.
- *Query*: During this phase, $\mathcal{A}$ can query the $\mathcal{O}\texttt{MAC}$ oracle to obtain the tag τ associated to a given message m.
- *Output*: Eventually, $\mathcal{A}$ outputs a pair (m', τ'). He wins the game if the τ' is a valid tag on message m' (*i.e.* $\texttt{Verify}(\texttt{pp}, \texttt{sk}, \texttt{m}', \tau') = 1$) and the message m' has not already been queried to $\mathcal{O}\texttt{MAC}$.

The adversary's success probability, which is commonly called *advantage of the adversary* and denoted $\texttt{Adv}_{\mathcal{A}}^{\text{UF-CMA}}(1^k)$, is defined as $\texttt{Adv}_{\mathcal{A}}^{\text{UF-CMA}}(1^k) = \texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{UF-CMA}}(1^k) = 1]$.

$\mathtt{Exp}_{\mathcal{A}}^{\text{UF-CMA}}(1^k)$

1. $pp \leftarrow \mathtt{Setup}(1^k)$
2. $sk \leftarrow \mathtt{KeyGen}(pp)$
3. $(m, \tau) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{MAC}}(pp)$ (i.e. $\mathcal{A}$ holds pp and has access to an $\mathcal{O}\mathtt{MAC}$ oracle)
4. If m has been queried to the $\mathcal{O}\mathtt{MAC}$ oracle then return 0.
5. Return $\mathtt{Verify}(pp, sk, m, \tau)$

Figure 3.2: UF-CMA Security

3.3.1.2 Probabilistic MAC schemes

In the case of a probabilistic MAC scheme, contrary to a deterministic one, several valid tags may correspond to a given message m. When considering such a MAC scheme, the desired security notion is stronger than the one for deterministic MACs as the adversary can also make verification queries. More specifically, it is known as *unforgeable under chosen message and verification attack* (UF-CMVA). That is, it is hard for an adversary, that can make `MAC` as well as `Verify` queries, to provide a valid pair (τ, m) such that the message m has not already been queried to the MAC oracle. The UF-CMVA security experiment is the same as the UF-CMA one except that the third step is replaced by

$$3.\ (m, \tau) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{MAC}, \mathcal{O}\mathtt{Verify}}(pp)$$

As for the adversary advantage, it is defined as $\mathtt{Adv}_{\mathcal{A}}^{\text{UF-CMVA}}(1^k) = \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{\text{UF-CMVA}}(1^k) = 1]$.

A yet stronger security notion for probabilistic MACs, denoted sUF-CMVA, exists. In such variant, the adversary wins even if m has already been queried to the $\mathcal{O}\mathtt{MAC}$ oracle, provided that the oracle did not output the pair (m, τ).

Two probabilistic algebraic MAC schemes, which are constructed using a cyclic group and proven UF-CMVA secure under the DDH assumption, have recently been proposed by Chase, Meiklejohn and Zaverucha [CMZ14]. Hereinafter, we detail the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme upon which we rely to design our private eCash system described in Chapter 5.

3.3.1.3 $\mathsf{MAC}_{\mathsf{GGM}}$

A particular feature of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme is that the issuer and the verifier of the MAC are actually the same entity and thus share a set of keys. Besides, they can *optionally* publish some public parameters associated to their secret keys without jeopardizing the scheme security. In what follows, the $\mathsf{MAC}_{\mathsf{GGM}}$ construction is reviewed by explaining how to produce a MAC on n distinct messages $(m_1, \ldots, m_n)$:

- $\mathtt{Setup}(1^k)$ creates the system public parameters denoted $pp = (\mathbb{G}, q, g, h)$ where $\mathbb{G}$ is a cyclic group of prime order q, a k-bit prime, and g, h are two random generators such that $\log_g h$ is unknown.

- $\mathtt{KeyGen}(pp)$ randomly selects a secret key $sk = \vec{x} = (x_0, x_1, \ldots, x_n) \in_R \mathbb{F}_q^{n+1}$ and a value $\tilde{x}_0 \in_R \mathbb{F}_q$ to build a commitment $C_{x_0} = g^{x_0} h^{\tilde{x}_0}$ to the secret value x_0 (see section 3.3.4). Denoted by *iparams*, $(C_{x_0}, X_1 = h^{x_1}, \ldots, X_n = h^{x_n})$ corresponds to the issuer's public parameters.

- $\texttt{MAC}(pp, sk, \vec{m})$ produces an authentication tag (u, u') on $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{F}_q^n$ where $u \in_R \mathbb{G}\backslash\{1\}$ and $u' = u^{x_0 + x_1 m_1 + \cdots + x_n m_n}$.

- $\texttt{Verify}(pp, sk, \vec{m}, (u, u'))$ checks the validity of the tag with respect to the message $\vec{m}$. The tag is accepted only if $u \neq 1$ and $u' = u^{x_0 + x_1 m_1 + \cdots + x_n m_n}$.

As previously mentioned, the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme has been proven unforgeable under chosen message and verification attack (UF-CMVA), in the generic group model, under the DDH assumption.

3.3.2 Digital Signature

The idea of *digital signatures* was put forth by Diffie and Hellman [DH06] in 1976. Similarly to a MAC, a digital signature aims to ensure the authenticity as well as the integrity of a message. However, unlike a MAC, given a message m and a signature σ, anyone must be able to verify the validity of the signature using the signer's public key. Furthermore, only the entity holding the private key, namely the *signer*, should be able to compute such a digital signature.

Definition 3.30 (Digital signature scheme)**.** *A digital signature scheme consists of the following four algorithms:*

- $\texttt{Setup}(1^{\texttt{k}})$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*

- $\texttt{KeyGen}(\texttt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs a pair of keys (sk, pk) where sk is the private key and pk is the associated public key.*

- $\texttt{Sign}(\texttt{pp}, \texttt{sk}, \texttt{m})$: *an algorithm, on input the public parameters pp, a message $m \in \{0,1\}^*$ as well as the private key sk, outputs a digital signature σ on the message m (such a signature will be said "valid").*

- $\texttt{Verify}(\texttt{pp}, \texttt{pk}, \texttt{m}, \sigma)$: *a deterministic algorithm, on input the public parameters pp, the public key pk, a message m and a signature σ, outputs* `0` *or* `1`*.*

A digital signature scheme must satisfy the following three properties:

- *Validity*: Given a digital signature σ on a message m generated using the private key sk, the output of `Verify` algorithm using the associated public key pk will be equal to `1`.

- *Integrity*: A digital signature protects the integrity of the message. Precisely, if any change is made to the original signed message m, then the output of `Verify` algorithm will be equal to `0` with overwhelming probability.

- *Non-repudiation*: A signer that has signed a message cannot falsely deny it afterwards. Precisely, given a digital signature σ generated using the private sk, then the output of `Verify` algorithm will be equal to `1` with overwhelming probability.

Possible forgeries and attacks. Depending on the attacker's goal, we can distinguish four types of attacks against a digital signature scheme:

- *Total break*: The adversary can recover a user's private key from his public key. Thereby, he is able to produce a valid signature on *any* message of his choice.

- *Universal forgery (UF)*: The adversary can generate a valid signature σ on any message m without knowing the corresponding private key.

- *Selective forgery (SF)*: The adversary, who has no knowledge of the signer's private key, can produce a valid signature σ on a message m that he chose *prior* to the attack.
- *Existential forgery (EF)*: The adversary, who has no knowledge of the signer's private key, can create a valid signature σ on a message m of his choice. The message could be absolutely meaningless.

The attacks can also be classified into three categories according to the type of information the attacker holds and the means at his disposal:

- *No Message Attack (NMA)*: the attacker only knows the public key of the signer.
- *Known Message Attack (KMA)*: the attacker knows the public key of the signer as well as a set of valid message/signature pairs.
- *Adaptive Chosen Message Attack (CMA)*: this attacker is stronger than previous ones as he knows the public key of the signer and can obtain a signature on any message of his choice.

A cryptographic scheme which resists to attack XX with respect to the category YY is said to be XX-YY. Usually, a digital signature scheme is considered secure if it is EUF-CMA *i.e. existentially unforgeable under chosen message attack*. This means that, even an adversary that can query a signature of some messages of his choice should not be able to produce a signature σ on a message m that has not been queried. Figure 3.3 provides a more formal definition of the EUF-CMA experiment. The advantage of the adversary $\mathtt{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^k)$ is defined as $\mathtt{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^k) = \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}(1^k) = 1]$.

$\mathtt{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}(1^k)$

1. $pp \leftarrow \mathtt{Setup}(1^k)$
2. $(sk, pk) \leftarrow \mathtt{KeyGen}(pp, pk)$
3. $(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign}}(pk)$
4. If m has been queried to the $\mathcal{O}\mathtt{Sign}$ oracle then return 0.
5. Return $\mathtt{Verify}(\mathtt{pp}, \mathtt{pk}, \mathtt{m}, \sigma')$

Figure 3.3: EUF-CMA Security

In what follows, we define some variants of basic digital signatures, namely *group signatures*, *blind signatures* and *aggregate signatures*.

3.3.2.1 Group Signature

Introduced by Chaum and van Heijst [CH91] in 1991, a group signature enables any group member to sign a message on behalf of the group in an anonymous manner. Thereby, the identity of the signer is kept secret. Sometimes, an exceptional circumstance may require to revoke the anonymity of a given signature (*i.e.* to recover the identity of the signer). To this end, some systems introduce several designated trusted authorities, commonly called *revocation authorities*. These authorities can also selectively revoke group membership of a given member.

Definition 3.31 (Group signature scheme)**.** *A group signature scheme often involves three stakeholders:*

1. *A **group manager** that handles user's registration to the group.*
2. ***Group members** that must first register with the group manager in order to be able to generate anonymous group signatures on behalf of the group.*
3. *A **revocation authority** that can lift the anonymity of a given signature to recover the signer's identity.*

It consists of the following six algorithms and protocols:

- $\texttt{Setup}(1^{\texttt{k}})$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*
- $\texttt{KeyGen}(\texttt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs the public key of the group gpk, the manager secret key sk_{gm} as well as the key of the revocation authority sk_{ra}.*
- `Join`: *an interactive protocol between a user and the group manager which, on input pp and gpk, outputs to the user a group secret key gsk_u as well as a group membership certificate ζ.*
- $\texttt{Sign}(\texttt{pp}, \texttt{m}, \zeta, \texttt{gsk}_\texttt{u})$: *a probabilistic algorithm, on input a message $m \in \{0,1\}^*$, a membership certificate ζ as well as a user's group secret key gsk_u, outputs a group signature σ on the message m.*
- $\texttt{Verify}(\texttt{pp}, \texttt{m}, \sigma, \texttt{gpk})$: *a deterministic algorithm, on input a group signature σ as well as the group manager public key gpk, outputs* `0` *or* `1`.
- $\texttt{Open}(\texttt{m}, \sigma, \texttt{sk}_\texttt{ra}, \texttt{gpk})$: *a deterministic algorithm, on input a valid group signature, the corresponding message m and the group public key gpk, outputs the identity of the group member who produced the signature σ.*

In addition to the previously defined security properties that any digital signature scheme must satisfy, a group signature scheme must also fulfill the following three security properties [BSZ05]:

- *Anonymity*: Except the revocation authority, no one should be able to lift the anonymity of a signature or link two signatures produced by the same signer. Thereby, this property also captures the intuitive informal notion of unlinkability [AT99].
- *Traceability*: Even a set of colluding group members cannot produce a valid group signature without the `Open` algorithm being able to return the identity of one of the colluding members.
- *Non-frameability*: No one, even a coalition between a set of users and the group manager, should be able to falsely accuse an honest group member of signing a message. This property covers the intuitive informal notions of exculpability and framing. Besides, together with the traceability property, it captures both the unforgeability and coalition-resistance security requirements [AT99].

3.3.2.2 Direct Anonymous Attestation (DAA)

Brickell, Camenisch and Chen [BCC04] introduced in 2004 a particular group signature scheme known as Direct Anonymous Attestation (DAA) that does not support signature opening capabilities, but instead, ensures the linkability of certain signatures. More precisely, a tag is appended to the signature so as to enable the linkability of signatures (*i.e.* attestations) created with the same private key and with respect to the same basename, denoted by `bsn`.

This privacy-preserving authentication protocol was initially intended for a particular hardware module, known as Trusted Platform Module (TPM) (*e.g.* a secure element in a smartphone), to allow its remote authentication to an external party (*e.g.* a service provider) whilst preserving the privacy of its owner. Usually, part of the attestation computation is delegated to the host embedding it (*i.e.* PC or smartphone), which is generally much more powerful but untrustworthy. In [BFG+13], Bernhard *et al.* introduced a special DAA scheme where the TPM performs all the computations and none is delegated to the host. The latter is referred to as Pre-DAA.

In this thesis, we will mainly focus on Pre-DAA schemes. Hereinafter, we first provide a formal definition of Pre-DAA schemes, followed by their security and privacy requirements.

Definition 3.32 (Pre-Direct Anonymous Attestation (Pre-DAA) scheme)**.** *A Pre-Direct Anonymous Attestation (DAA) scheme usually involves two stakeholders:*

1. *An* ***issuer*** $\mathcal{I}$ ***(i.e. group manager)*** *that handles the issuance of TPM's group signing keys.*

2. ***TPMs (i.e group members)*** *that must hold a valid group signing key in order to be able to generate anonymous attestations.*

It consists of the following nine algorithms and protocols:

- $\mathtt{Setup}(1^{\mathtt{k}})$: *a probabilistic algorithm, on input an integer* k*, outputs the system public parameters* pp*.*

- $\mathtt{IKeyGen}(\mathtt{pp})$: *a probabilistic algorithm, on input the public parameters* pp*, outputs the public/private key pair* $(gmsk, gmpk)$ *of the issuer.*

- $\mathtt{UKeyGen}(\mathtt{pp}, \mathtt{i})$: *a probabilistic algorithm, on input the public parameters* pp *and a TPM's identifier* i*, outputs the secret key* sk_i *of the TPM.*

- $\mathtt{Join}(\mathtt{pp}, \mathtt{gmpk}, \mathtt{gmsk}, \mathtt{i}, \mathtt{sk_i})$: *an interactive protocol between a TPM and the issuer which, on input* pp*,* $gmpk$*,* $gmsk$ *as well as the TPM's secret key* sk_i*, outputs to the TPM group signing key* gsk_i*.*

- $\mathtt{GSign}(\mathtt{pp}, \mathtt{m}, \mathtt{gsk_i}, \mathtt{bsn})$: *a probabilistic algorithm, on input a message* $m \in \{0,1\}^*$*, a TPM secret key* sk_i *as well as its group signing key* gsk_i *and a basename* $\mathtt{bsn}$*, outputs an anonymous signature* σ *on the message* m *and with respect to* $\mathtt{bsn}$*.*

- $\mathtt{GVerify}(\mathtt{pp}, \mathtt{m}, \mathtt{bsn}, \sigma, \mathtt{gmpk})$: *a deterministic algorithm, on input a group signature* σ*, a message* m*, a basename* $\mathtt{bsn}$ *as well as the issuer's public key* $gmpk$*, outputs* $\mathtt{0}$ *or* $\mathtt{1}$*.*

- $\mathtt{Identify_T}(\mathcal{T}, \mathtt{sk_i})$: *a deterministic algorithm, on input a transcript* $\mathcal{T}$ *resulting from the execution of the* Join *protocol and a secret key* sk_i*, outputs* $\mathtt{1}$ *if* $\mathcal{T}$ *has been produced using* sk_i*. Otherwise, it returns* $\mathtt{0}$*.*

- $\mathtt{Identify_S}(\sigma, \mathtt{m}, \mathtt{bsn}, \mathtt{sk_i})$: *a deterministic algorithm, on input a signature* σ*, a message* m*, a basename* $\mathtt{bsn}$ *and a secret key* sk_i*, outputs* $\mathtt{1}$ *if the signature* σ *on* m *and for* $\mathtt{bsn}$ *could have been produced using the secret key* sk_i*. Otherwise, it returns* $\mathtt{0}$*.*

- $\mathtt{Link}(\mathtt{gmpk}, \sigma, \mathtt{m}, \sigma', \mathtt{m}', \mathtt{bsn})$: *a deterministic algorithm, on input two signatures* σ *and* σ'*, two messages* m *and* m' *as well a basename* $\mathtt{bsn}$*, outputs* $\mathtt{1}$ *if both* σ *and* σ' *are valid signatures on, respectively,* m *and* m' *with respect to the same basename* $\mathtt{bsn} \neq \perp$ *and were produced by the same TPM (i.e. using the same* sk_i*). Otherwise, it returns* $\mathtt{0}$*.*

Security and Privacy Models. Hereinafter, we recall the different properties that a pre-DAA scheme [BFG+13] must satisfy to be secure whilst preserving user's privacy. For that, a pre-DAA scheme must ensure four security properties, namely *correctness*, *traceability*, *non-frameability* as well as *anonymity*. In what follows, we formally define them through a set of experiments between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$. We distinguish two sets of users: $\mathcal{HU}$, the set of honest users and $\mathcal{CU}$, the set of corrupted users (whose secret key sk_i and group signing key gsk_i are known to the adversary). We also define three lists: $\mathbb{S}$, the list of all queries to the signing oracle; $\mathbb{C}$, the list of queries to the challenge oracle; and $\mathbb{T}$, the list of transcript views associated to the different executions of the Join protocol. To model the capabilities of the adversary $\mathcal{A}$, we give him access to some oracles defined as follows:

- $\mathcal{O}\texttt{Add}(i)$ is an oracle used by $\mathcal{A}$ to create a new honest user identified by i.
- $\mathcal{O}\texttt{AddCorrupt}(i)$ is an oracle used by $\mathcal{A}$ to create a new corrupted user identified by i. His secret key is known to $\mathcal{A}$.
- $\mathcal{O}\texttt{Join}_I(i)$ is an oracle that executes the issuer's side of the Join protocol. This oracle will be used to simulate the execution of the Join protocol between an, honest or corrupted, user u_i and an *honest* issuer.
- $\mathcal{O}\texttt{Join}_U(i)$ is an oracle that executes the user's side of the Join protocol. This oracle will be used by $\mathcal{A}$ acting as a malicious issuer. The adversary provides the oracle with an honest user's identifier i. If the latter accepts, $\mathcal{A}$ gets a transcript view $\mathcal{T}$ of the protocol execution, that is saved in $\mathbb{T}$.
- $\mathcal{O}\texttt{Corrupt}(i)$ is an oracle used by $\mathcal{A}$ to corrupt the user identified by i. Thus, $\mathcal{A}$ gets both the secret key sk_i and the group signing key gsk_i of user i. Concurrently, i is moved from $\mathcal{HU}$ to $\mathcal{CU}$.
- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ is an oracle used by $\mathcal{A}$ to obtain a signature on m and for bsn produced by the honest user i. Concurrently, the triple $(i, m, \texttt{bsn})$ is saved in $\mathbb{S}$.
- $\mathcal{O}\texttt{CH}_b(i_0, i_1, \texttt{bsn}, m)$ is an oracle used by $\mathcal{A}$ to obtain a signature σ on m by i_b.

The four security and privacy requirements of a pre-DAA are defined as follows:

Correctness. This property ensures the proper functioning of the system. A Pre-DAA scheme provides correctness if the following four statements are met: (1) the issued group signing key gsk_i is valid; (2) a valid signature is accepted by the verifier $\mathcal{V}$; (3) a valid signature can be traced to the correct sk_i; and (4) two signatures produced by the same user (*i.e.* same key sk_i) and for the same bsn are linkable. Besides, it is noteworthy to mention that each transcript $\mathcal{T}$ resulting from the execution of the Join protocol must have a unique sk_i associated to it.

Traceability. Roughly speaking, traceability requires the following two statements to be met. Indeed, no adversary should be able to produce: (1) a valid signature which cannot be traced to a secret key that has been committed to during an execution of the Join protocol; or (2) two *unlinkable* signatures for the same basename and under the same secret key. Thus, as shown in Figure 3.4, the traceability experiment consists of two subgames. In the first subgame (*i.e.* steps 3 and 4), the issuer is assumed to be honest whereas, in the second one (*i.e.* steps 5, 6 and 7), the issuer as well as all users are assumed to be corrupted. The advantage $\texttt{Adv}_{\mathcal{A}}^{\text{trace}}(1^k)$ of the adversary is defined as $\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{trace}}(1^k) = 1]$. A pre-DAA scheme satisfies the *traceability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\mathtt{Exp}^{\mathrm{trace}}_{\mathcal{A}}(1^k)$

1. $pp \leftarrow \mathsf{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \mathbb{T} \leftarrow \varnothing$
2. $(gmpk, gmsk) \leftarrow \mathsf{IKeyGen}(pp)$
3. $(\sigma, m, \mathtt{bsn}, sk_1, \ldots, sk_l) \longleftarrow \mathcal{A}_1^{\mathcal{O}}(gmpk)$ where $\mathcal{O} = \{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Join}_I, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{GSign}\}$
4. If the following 3 conditions hold, then return 1
   (a) $\mathtt{GVerify}(gmpk, \sigma, m, \mathtt{bsn}) = 1$.
   (b) $\forall \mathcal{T} \in \mathbb{T}, \exists i \in \{1, \ldots, l\}$ such that $\mathtt{Identify}_{\mathtt{T}}(\mathcal{T}, sk_i) = 1$.
   (c) $\forall i \in \{1, \ldots, l\}, \mathtt{Identify}_{\mathtt{S}}(\sigma, m, \mathtt{bsn}, sk_i) = 0$.
5. $(\sigma_0, m_0, \sigma_1, m_1, \mathtt{bsn}, sk) \longleftarrow \mathcal{A}_2(gmpk, gmsk)$
6. If $\mathtt{bsn} = \perp$, then return 0.
7. If the following 3 conditions hold, then return 1
   (a) $\forall b \in \{0,1\}, \mathtt{GVerify}(gmpk, \sigma_b, m_b, \mathtt{bsn}) = 1$.
   (b) $\forall b \in \{0,1\}, \mathtt{Identify}_{\mathtt{S}}(\sigma_b, m_b, \mathtt{bsn}, sk) = 1$.
   (c) $\mathtt{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathtt{bsn}) = 0$.
8. Return 0

Figure 3.4: Traceability Security Experiment

Anonymity. Roughly speaking, *anonymity* requires that, given two identities i_0 and i_1, no entity can determine which of the two identities produced a specific signature σ. More formally, the anonymity experiment $\mathtt{Exp}^{\mathrm{anon}-b}_{\mathcal{A}}(1^k)$ is detailed in Figure 3.5. The adversary queries the $\mathcal{O}\mathtt{CH}_b$ oracle on $(i_0, i_1, \mathtt{bsn}, m)$ as input and has to guess the identity i_b of the user who generated the returned signature σ. To prevent the adversary from trivially wining the game, once he has queried the $\mathcal{O}\mathtt{CH}_b$ oracle, he has a restricted access to the $\mathcal{O}\mathtt{CH}_b$ and $\mathcal{O}\mathtt{GSign}$ oracles. More precisely, he can no longer call on the $\mathcal{O}\mathtt{GSign}$ and $\mathcal{O}\mathtt{CH}_b$ oracles for the same $(i, \mathtt{bsn})$ pair. The advantage $\mathtt{Adv}^{\mathrm{anon}-b}_{\mathcal{A}}(1^k)$ of the adversary is defined as $\mathtt{Adv}^{\mathrm{anon}-b}_{\mathcal{A}}(1^k) = |\mathtt{Pr}[\mathtt{Exp}^{\mathrm{anon}-b}_{\mathcal{A}}(1^k) = b] - 1/2|$. A pre-DAA scheme is considered *anonymous* if this advantage is negligible for any PPT adversary.

$\mathtt{Exp}^{\mathrm{anon}-b}_{\mathcal{A}}(1^k)$

1. $pp \leftarrow \mathsf{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \mathbb{S} \leftarrow \varnothing, \mathbb{C} \leftarrow \varnothing$
2. $(gmpk, gmsk) \leftarrow \mathsf{IKeyGen}(pp)$
3. $b' \longleftarrow \mathcal{A}^{\mathcal{O}}(gmpk, gmsk)$ where $\mathcal{O} = \{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Join}_U, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{GSign}^*, \mathcal{O}\mathtt{CH}_b{}^*\}$ such that $\mathcal{O}\mathtt{GSign}^*$ and $\mathcal{O}\mathtt{CH}_b{}^*$ denotes a restricted access to these oracles.
4. If $i_0 \in \mathcal{CU}$ or $i_1 \in \mathcal{CU}$, then return $\perp$.
5. If $\exists i, m, \sigma, \mathtt{bsn}$ such that $\mathtt{bsn} \neq \perp$, $(i, \mathtt{bsn}) \in \mathbb{C}$ and $(i, m, \mathtt{bsn}) \in \mathbb{S}$, then return $\perp$.
6. Return b'

Figure 3.5: Anonymity Security Experiment

Non-frameability. Informally, non-frameability requires the following two statements to be satisfied. Indeed, no adversary should be able to output: (1) a signature that can be traced to a given user i who has not produced a signature on the corresponding message/basename pair; or (2) two signatures that are linkable even though they should not. Thus, as shown in Figure 3.6, the non-frameability experiment also consists of two subgames. In the first subgame (*i.e.* steps 3 and 4), the adversary's goal is to frame an honest user whereas, in the second one (*i.e.* steps

5 to 9), he has control over all users as well as the issuer. The advantage $\mathtt{Adv}_{\mathcal{A}}^{\text{non-fram}}(1^k)$ of the adversary is defined as $\mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k) = 1]$. A pre-DAA scheme satisfies the *non-frameability* requirement if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\mathtt{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k)$

1. $pp \leftarrow \mathsf{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \mathbb{S} \leftarrow \varnothing$
2. $(gmpk, gmsk) \leftarrow \mathsf{IKeyGen}(pp)$
3. $(\sigma, i, m, \mathtt{bsn}) \longleftarrow \mathcal{A}_1^{\mathcal{O}}(gmpk, gmsk)$ where $\mathcal{O} = \{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Join}_U, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{GSign}\}$
4. If the following 4 conditions hold, then return 1
   (a) $\mathtt{GVerify}(gmpk, \sigma, m, \mathtt{bsn}) = 1$.
   (b) $i \in \mathcal{HU}$.
   (c) $(i, m, \mathtt{bsn}) \notin \mathbb{S}$.
   (d) $\mathtt{Identify}_{\mathbb{S}}(\sigma, m, \mathtt{bsn}, sk_i) = 1$.
5. $(\sigma_0, m_0, \mathtt{bsn}_0, \sigma_1, m_1, \mathtt{bsn}_1, sk) \longleftarrow \mathcal{A}_2(gmpk, gmsk)$
6. If $\exists b \in \{0,1\}$ such that $\mathtt{GVerify}(gmpk, \sigma_b, m_b, \mathtt{bsn}_b) = 0$, then return 0.
7. If $\forall b \in \{0,1\}$, $\mathtt{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathtt{bsn}_b) = 0$, then return 0.
8. For $b \in \{0,1\}$, if $\mathtt{Identify}_{\mathbb{S}}(\sigma_b, m_b, \mathtt{bsn}_b, sk) = 1$ and $\mathtt{Identify}_{\mathbb{S}}(\sigma_{1-b}, m_{1-b}, \mathtt{bsn}_{1-b}, sk) = 0$, then return 1.
9. If $\mathtt{bsn}_0 \neq \mathtt{bsn}_1$ or $\mathtt{bsn}_0 = \perp$ or $\mathtt{bsn}_1 = \perp$, then return 1.
10. Return 0

Figure 3.6: Non-Frameability Security Experiment

3.3.2.3 Blind Signature

Blind signature, which was also proposed by Chaum [Cha83], is a variant of classical digital signatures that allows a receiver $\mathcal{R}$ to get a signature of a message without revealing any information about it to the signer $\mathcal{S}$.

Definition 3.33 (Blind signature scheme)**.** *A blind signature scheme consists of the following four algorithms:*

- $\mathtt{Setup}(1^k)$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*
- $\mathtt{KeyGen}(\mathtt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs a pair of keys (sk, pk) where sk is the private key of the signer and pk is the associated public key.*
- $\mathtt{Sign}(\mathtt{pp}, \mathtt{sk}, \mathtt{m}')$: *an interactive protocol, on input the public parameters pp, a blinded version m' of a message $m \in \{0,1\}^*$ as well as the private key sk, outputs a digital signature σ on the message m.*
- $\mathtt{Verify}(\mathtt{pp}, \mathtt{pk}, \mathtt{m}, \sigma)$: *a deterministic algorithm, on input the public parameters pp, the public key pk, a message m and a signature σ, outputs either* `0` *or* `1`*.*

A blind signature scheme should satisfy two additional security properties:

- *One-more Unforgeability*: it should be impossible for the receiver $\mathcal{R}$ to obtain $L+1$ signatures after *at most* L signing requests.

- *Unlinkability* (also called *Blindness*): the signed message must remain secret to the signer and the latter should not be able to (1) link two message/signature pairs (i.e. decide whether two signatures were produced by the same signer or not), or (2) link a message/signature pair to a specific execution of the protocol `Sign` (*i.e.* identify for whom a given signature was issued).

Sometimes, in use cases like electronic Cash (eCash) for example, the signer $\mathcal{S}$ may want to add some information to the blind signature such as a date, a validity period or even an amount. To address this issue, Abe and Fujisaki [AF96] proposed in 1996 an extension of blind signature known as *partially blind signature*.

Partially blind signatures. A partially blind signature allows the receiver of the signature $\mathcal{R}$ and the signer $\mathcal{S}$ to agree on a common information `info` to be added to the blind signature of a message m.

Definition 3.34 (Partially blind signature scheme)**.** *More formally, a partially blind signature scheme consists of the following four algorithms:*

- $\texttt{Setup}(1^{\texttt{k}})$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*
- $\texttt{KeyGen}(\texttt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs a pair of keys (sk, pk) where sk is the private key of the signer and pk is the associated public key.*
- $\texttt{Sign}(\texttt{pp}, \texttt{sk}, \texttt{m}', \texttt{info})$: *an interactive protocol, on input the public parameters pp, a blinded version m' of a message $m \in \{0,1\}^*$, a common information `info` that the receiver and the signer agreed on as well as the private key sk, outputs a digital signature σ on both m and `info`.*
- $\texttt{Verify}(\texttt{pp}, \texttt{pk}, \texttt{m}, \texttt{info}, \sigma)$: *a deterministic algorithm, on input the public parameters pp, the public key pk, a message m, a common information `info` and a signature σ, outputs either `0` or `1`.*

The additional security properties that a partially blind signature scheme must meet are the same as those of a blind signature scheme (i.e. one-more unforgeability and unlinkability).

3.3.2.4 Aggregate Signature

In 2003, Boneh, Gentry, Lynn and Shacham [BGLS03] introduced a particular case of digital signature, denoted *aggregate signature*, that additionally supports aggregation. Indeed, an aggregate signature scheme allows to aggregate n signatures on n distinct messages produced by n signers into a single compact signature. Along with the n messages, the resulting signature will convince the verifier that the n messages were, indeed, signed by the n signers.

Definition 3.35 (aggregate signature scheme)**.** *An aggregate signature scheme consists of the following five algorithms:*

- $\texttt{Setup}(1^{\texttt{k}})$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*
- $\texttt{KeyGen}(\texttt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs the users' private/public key pairs $\{(sk_i, pk_i)\}_{i=1}^{i=n}$.*
- $\texttt{Sign}(\texttt{pp}, \texttt{sk}_\texttt{i}, \texttt{m}_\texttt{i})$: *an algorithm, on input the public parameters pp, a message m_i as well as a private key sk_i, outputs a signature σ_i on m_i.*

- $\texttt{Aggregation}(\texttt{pp}, \{\texttt{m}_\texttt{i}, \sigma_\texttt{i}\}_{\texttt{i=1}}^{\texttt{i=n}})$: *an algorithm, on input n distinct messages* $\vec{m} = (m_1, m_2, \ldots, m_n)$ *and the corresponding signatures* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$*, outputs a unique signature* σ *on* $\vec{m}$*.*

- $\texttt{Verify}(\texttt{pp}, \{\texttt{pk}_\texttt{i}, \texttt{m}_\texttt{i}\}_{\texttt{i=1}}^{\texttt{i=n}}, \vec{\texttt{m}}, \sigma)$: *a deterministic algorithm, on input n messages* $(m_1, \ldots, m_n)$*, a signature* σ *and n public keys* $(pk_1, \ldots, pk_n)$*, outputs either* `0` *or* `1`*.*

A specific type of aggregate signatures, where the signatures are sequentially (rather than independently) created, is called *Sequential aggregate signature*. Indeed, as its name implies, the final signature is generated sequentially with each signer signing the aggregate signature in turn.

Definition 3.36 (Sequential aggregate signature scheme)**.** *A sequential aggregate signature scheme consists of the following four algorithms:*

- $\texttt{Setup}(1^\texttt{k})$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*

- $\texttt{KeyGen}(\texttt{pp})$: *a probabilistic algorithm, on input the public parameters pp, outputs the users' private/public key pairs* $\{(sk_i, pk_i)\}_{i=1}^{i=n}$*.*

- $\texttt{AggSign}(\texttt{pp}, \sigma, \texttt{m}_\texttt{n}, \texttt{sk}_\texttt{n})$: *an algorithm, on input the public parameters pp, an aggregate signature* σ *on a set of messages* $(m_1, m_2, \ldots, m_{n-1})$ *as well as the private key* sk_n *and a message* m_n*, outputs a signature* σ' *on* $\vec{m} = (m_1, m_2, \ldots, m_n)$*.*

- $\texttt{Verify}(\texttt{pp}, \{\texttt{pk}_\texttt{i}, \texttt{m}_\texttt{i}\}_{\texttt{i=1}}^{\texttt{i=n}}, \sigma')$: *a deterministic algorithm, on input a signature* σ'*, n messages* $(m_1, \ldots, m_n)$ *and n public keys* $(pk_1, \ldots, pk_n)$*, outputs either* `0` *or* `1`*.*

Security Model. A sequential aggregate signature scheme must be *existentially unforgeable under chosen message attacks* (EUF-CMA). That is, no adversary should be able to forge an aggregate signature, on a set of messages of his choice, by a set of users whose secret keys are not all known to him. More precisely, the corresponding security experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ consists of the following four steps:

- *Setup*: $\mathcal{C}$ initializes a key list `KeyList` as empty. By running the `Setup` and `KeyGen` algorithms, he respectively defines the system public parameters *pp* and the signing/verification key pair (sk^*, pk^*). $\mathcal{A}$ is provided with the verification key pk^*.

- *Join Queries*: $\mathcal{A}$ adaptively requests to append the public keys pk_i to the list `KeyList`.

- *Aggregate signatures Queries*: $\mathcal{A}$ adaptively asks for aggregate signatures on *at most q* messages $(m_1, \ldots, m_q)$ under the challenge public key pk^*. More precisely, for each query, $\mathcal{A}$ provides an aggregate signature σ_i on the block of messages $(m_{i,1}, \ldots, m_{i,r_i})$ under the public keys $(pk_{i,1}, \ldots, pk_{i,r_i})$ all belonging to `KeyList`. In response, $\mathcal{C}$ returns the aggregate signature on $(m_{i,1}, \ldots, m_{i,r_i}, m_i)$ produced using sk^*.

- *Output*: Eventually, $\mathcal{A}$ outputs an aggregate signature σ on the messages $(m_1^*, \ldots, m_r^*)$ under the public keys $(pk_1, \ldots, pk_r)$. He wins the game if the following three conditions are met:

  - $\texttt{Verify}(pp, \{pk_i, m_i^*\}_{i=1}^{i=n}, \sigma) = 1$;
  - For all $pk_j \neq pk^*$, $pk_j \in \texttt{KeyList}$;
  - For some $j^* \in \{1, \ldots, r\}$, $pk^* = pk_{j^*}$ and $m_{j^*}^*$ has not already been queried to the signing oracle (*i.e.* for $i \in \{1, \ldots, q\}$, $m_i \neq m_{j^*}^*$).

The adversary's success probability, which is commonly called *advantage of the adversary* and denoted $\texttt{Adv}_{\mathcal{A}}^{\text{UF-CMA}}(1^k)$, is defined as $\texttt{Adv}_{\mathcal{A}}^{\text{UF-CMA}}(1^k) = \texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{UF-CMA}}(1^k) = 1]$.

3.3.2.5 Used Digital Signature Schemes

To design our protocols, we use three well-known digital signature schemes, namely RSA, Schnorr and Boneh-Boyen signature schemes. In this section, we briefly recall them.

RSA. Rivest, Shamir and Adleman signature scheme [RSA78], which relies on the hash and sign concept, is defined as follows:

- $\texttt{KeyGen}(1^k)$: Selects two distinct k-bit prime integers p and q, then chooses an integer e that is co-prime with $\phi(n) = (p-1)(q-1)$ where ϕ is Euler's totient function. Next, it determines $d = e^{-1} \ mod \ \phi(n)$. The signer's private key is (n, d) and the corresponding public key consists of the pair (n, e).
- $\texttt{Sign}(m, d, n, e)$: To sign a message m, the signer computes $\sigma = m^d \ mod \ n$. It returns σ as the digital signature on m.
- $\texttt{Verify}(m, \sigma, n, e)$: A signature σ is valid only if $m = \sigma^e \ mod \ n$.

Schnorr. The Schnorr signature [Sch90, Sch91] is an efficient discrete logarithm-based signature scheme and works as follows:

- $\texttt{Setup}(1^k, 1^{k'})$: Generate the system public parameters $pp = (p, q, \mathbb{Z}_p, g, \mathcal{H})$ where p and q are two prime integers such that $q|p-1$, $p > 2^{k'}$ and $q > 2^k$, whereas $g \in \mathbb{Z}_p^*$ and $\mathcal{H}$ is a hash function.
- $\texttt{KeyGen}(pp)$: Chooses a random $x \in \mathbb{Z}_q^*$ and computes $y = g^x \ mod \ p$. The signer's private key is x and the corresponding public key is y.
- $\texttt{Sign}(pp, x, m)$: To sign a message $m \in \{0,1\}^*$, the signer first selects a random $a \in_R \mathbb{Z}_q$. Then, he computes $t = g^a \ mod \ p$, $c = \mathcal{H}(m, t)$ and $s = a + cx \ mod \ q$. The signature on the message m is $\sigma = (s, c)$.
- $\texttt{Verify}(pp, \sigma, m, X)$: Given a pair (s, c) and a message m, the verifier computes $t' = g^s y^{-c} \ mod \ p$ and checks whether $c \stackrel{?}{=} \mathcal{H}(m, t')$. If so, the signature is valid.

Boneh-Boyen. Using Boneh and Boyen signature scheme, subsequently denoted BB, one may prove the knowledge of a signature on a message without revealing neither the signature nor the message. The two original BB signature variants, proposed in [BB04], fulfil different security properties but both rely on the use of pairings. There is, however, a variant [CCJT14] where no pairing computations are required. Thereby, BB signatures may also be used when dealing with constrained environments that cannot handle pairing computations such as SIM/UICC cards. In the sequel, we describe the most efficient variant of the two original BB schemes as well as the one without pairing.

Boneh-Boyen signature

- $\texttt{Setup}(1^k)$: Generates the system public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map.
- $\texttt{KeyGen}(pp)$: Chooses two random generators $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$ as well as a random $y \in \mathbb{Z}_p$. The signer's private key is y and the corresponding public key consists of $(g, \tilde{g}, \tilde{Y}, z)$ where $\tilde{Y} = \tilde{g}^y$ and $z = e(g, \tilde{g})$.
- $\texttt{Sign}(pp, m, y)$: Given the private key y and a message $m \in \mathbb{Z}_p$, this algorithm returns the signature $\sigma = g^{\frac{1}{y+m}}$ on the message m.

- `Verify`$(pp, \sigma, m, \tilde{g}, \tilde{Y}, z)$: To verify the validity of a signature σ with respect to a message m, the verifier checks whether $e(\sigma, \tilde{Y}\tilde{g}^m) \stackrel{?}{=} z$. If so, the signature is valid and the algorithm outputs `1`. Otherwise, it returns `0`.

Boneh-Boyen signature without pairing

- `Setup`(1^k): Generates the system public parameters $pp = (\mathbb{G}, p)$ where $\mathbb{G}$ is a cyclic group with prime order p and such that the DDH problem is hard in $\mathbb{G}$.

- `KeyGen`(pp): Chooses two random generators $g_1, g_2 \in \mathbb{G}$ as well as a random $y \in \mathbb{Z}_p$. The signer's private key is y and the corresponding public key is $Y = g_2^y$.

- `Sign`(pp, m, y): The signature σ on a message $m \in \mathbb{Z}_p$ consists of $A = g_1^{\frac{1}{y+m}}$ along with a Zero-Knowledge Proof of Knowledge π (see section 3.3.5) defined as $\pi = \text{PoK}\ \{\alpha : Y = g_2^\alpha \ \wedge A^\alpha = g_1 A^{-m}\}$.

- `Verify`(pp, σ, m, Y): To verify the validity of a signature A on a message m, the verifier checks the validity of the proof π. If so, the signature is valid and the algorithm outputs `1`. Otherwise, it returns `0`.

Remark 1. By definition, $A = g_1^{\frac{1}{y+m}}$. This implies that $A^y = g_1 A^{-m}$. Since the group $\mathbb{G}$ is not equipped with a bilinear map (*i.e.* it is not a gap-DDH group), the signer has to additionally prove that the signature is valid with respect to m. To this end, he provides a Zero-Knowledge Proof of Knowledge π (see Section 3.3.5) that the discrete logarithm of $(g_1 A^{-m})$ in the base A is equal to the discrete logarithm of Y in the base g_2 (*i.e.* $\pi = \text{PoK}\ \{\alpha : Y = g_2^\alpha \ \wedge A^\alpha = g_1 A^{-m}\}$). Such proof of equality of discrete logarithms has been introduced by Chaum and Pedersen in [CP93]. Thus, if π is valid, then so is the signature A.

Theorem 3.1. *The Boneh-Boyen signature scheme without pairings is existentially unforgeable under chosen message attack, in the random oracle model, assuming that the q-Strong Diffie-Hellman ($q - \mathsf{SDH}$) assumption holds. [CCJT14]*

3.3.3 Public Key Encryption

Encryption mainly aims to provide the confidentiality of exchanged information by ensuring that only the sender and the legitimate receiver(s) can know the message content. Until 1976, all encryption schemes were symmetric (*i.e.* the same key is used for encryption and decryption). However, Diffie and Hellman's seminal paper "New directions in cryptography" revolutionized the world of cryptography by introducing the concept of public-key cryptography. Unlike secret key schemes, a public key encryption scheme relies on the use of a public/private key pair. The public key of the receiver, known to everyone, is used to encrypt the message while his private key, which should be kept secret, allows the legitimate receiver to decrypt the received ciphertext to recover the message.

3.3.3.1 Definition and Security Requirements

Definition 3.37 (Public key encryption scheme)**.** *More formally, a public key encryption scheme is defined through the following four algorithms:*

- `Setup`(1^k): *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*

- `KeyGen`(pp): *a probabilistic algorithm, on input the public parameters pp, outputs a pair of keys (sk, pk) where pk is the public key of the user and sk is the associated private key.*

- $\texttt{Enc}(\texttt{pp}, \texttt{m}, \texttt{pk})$: *an algorithm, on input a message* $m \in \{0,1\}^*$ *and a public key* pk*, outputs a ciphertext* c*.*

- $\texttt{Dec}(\texttt{pp}, \texttt{c}, \texttt{sk})$: *a deterministic algorithm, on input a ciphertext* c *as well as a private key* sk*, outputs the corresponding plaintext (i.e. the message* m*).*

A public key encryption scheme must satisfy the following two properties:

- *Validity*: a message encrypted using a public key pk must be properly decrypted using the corresponding private key sk.

- *Confidentiality*: a ciphertext reveals no useful information about the encrypted message unless if one knows the corresponding private key.

Similarly to digital signature schemes, and as it is the case of any cryptographic scheme, the security level ensured by a public key encryption scheme is defined according to the attacker goal's and the means at his disposal.

Possible attacks. If we consider an adversary who aims to recover the plaintext corresponding to a given ciphertext, then we distinguish the following two attacks:

- *Total break*: Given the public key of a user, the adversary manage to recover the corresponding private key. Thereby, he is able to decrypt and retrieve the plaintext associated to *any* encrypted message.

- *Partial break*: The adversary recovers the plaintext corresponding to a given ciphertext without learning the private key.

Sometimes, the adversary is less ambitious and his goal may simply consists in breaking one of the following security properties:

- *One-Wayness (OW)*: an attacker, that does not possess the private key, cannot recover the whole plaintext corresponding to a given ciphertext.

- *Indistinguishability (IND)*: an attacker is unable to distinguish the ciphertexts of two different messages. In other words, given two messages of the same length and a ciphertext c, he cannot determine which message has been encrypted. This security property is also known as *semantic security.*

- *Non-Malleability (NM)*: given the ciphertext c of an unknown message m, an attacker cannot produce a ciphertext c' of a message m' such that the two plaintexts m and m' are related to each other.

The attacks can also be classified into three categories according to the type of information the attacker holds and the means at his disposal:

- *Chosen Plaintext Attack (CPA)*: the adversary can obtain the ciphertext of any plaintext of his choice. This is possible since the encryption key is publicly known.

- *Chosen Ciphertext Attack (CCA1)*: the adversary can recover the plaintexts corresponding to some chosen ciphertexts without knowing the associated private key. This is made possible by providing him access to a decryption oracle that he can only query until the receipt of the challenge.

- *Adaptive Chosen Ciphertext Attack (CCA2)*: the adversary has access to a decryption oracle even after the receipt of the challenge. Thereby, he can retrieve the plaintext of any ciphertext (except the challenge, obviously) and adapt his queries throughout his attack.

3.3.3.2 Threshold Encryption Schemes

In a threshold cryptosystem [DF89], the public key and the corresponding private key are cooperatively generated by n parties. Then, the private key is shared among them. To decrypt a ciphertext, at least t (which corresponds to the threshold) out of n parties must collaborate. Indeed, t out of n shares of the private key are necessary to recover the whole key.

3.3.3.3 Used Public Key Encryption Schemes

Hereinafter, we review the main public key encryption schemes that we used to design our privacy-preserving protocols, namely ElGamal and Paillier encryption schemes.

ElGamal. The ElGamal public key encryption scheme [EG85] is defined as follows:

- $\texttt{Setup}(1^k)$: generates the system public parameters $pp = (\mathbb{G}, p, g)$ where $\mathbb{G}$ is a cyclic group with prime order p and g is a random generator of $\mathbb{G}$.
- $\texttt{KeyGen}(\texttt{pp})$: randomly chooses the private key $sk \in \mathbb{Z}_p^*$ and computes the corresponding public key $pk = g^{sk} \ mod \ p \in \mathbb{G}$.
- $\texttt{Enc}(pp, pk, m)$: randomly selects a random $r \in \mathbb{Z}_p^*$. The encryption of a message $m \in G$ consists of the pair (c_1, c_2) where $c_1 = g^r \ mod \ p$ and $c_2 = m \cdot pk^r$.
- $\texttt{Dec}(pp, sk, c_1, c_2)$: using the private key sk, it decrypts the ciphertext (c_1, c_2) by computing $m = \frac{c_2}{c_1^{sk}}$.

The ElGamal cryptosystem is semantically secure under the DDH assumption defined in Section 3.1.3. Besides, it is multiplicatively homomorphic. Indeed, given two ciphertexts $C_1 = (c_1, c_2)$ and $C_2 = (c_1', c_2')$ of two messages m_1 and m_2, one can efficiently compute the ciphertext of the product $m_1 m_2$ of the two original messages as $C' = (c_1'' = c_1 c_1', c_2'' = c_2 c_2')$.

Modified ElGamal. This variant of ElGamal encryption scheme was introduced by Jarecki and Lysyanskaya [JL00] and is defined as follows:

- $\texttt{Setup}(1^k)$: generates the system public parameters $pp = (\mathbb{G}, p)$ where $\mathbb{G}$ is a cyclic group with prime order p.
- $\texttt{KeyGen}(\texttt{pp})$: randomly chooses $y_1, y_2 \in_R \mathbb{Z}_p^*$ and $g_1, g_2 \in_R \mathbb{G}$. The private key sk is defined as $sk = (y_1, y_2)$ and the corresponding public key $pk = (g_1, g_2, h = g_1^{y_1} g_2^{y_2})$.
- $\texttt{Enc}(pp, pk, m)$: randomly selects a random $s \in \mathbb{Z}_p^*$. The encryption of a message $m \in G$ consists of the triplet (c_1, c_2, c_3) where $c_1 = g_1^s \ mod \ p$, $c_2 = g_2^s \ mod \ p$ and $c_3 = m \cdot h^s$.
- $\texttt{Dec}(pp, sk, c_1, c_2, c_3)$: using the private key sk, it decrypts the ciphertext (c_1, c_2, c_3) by computing $m = \frac{c_3}{c_1^{y_1} c_2^{y_2}}$.

The modified ElGamal scheme is also *semantically secure* under the DDH assumption. Hereinafter, we provide a proof sketch which is taken almost verbatim from [JCJ05].

Proof (sketch). Let us assume that there exists a PPT algorithm $\mathcal{A}$ which can break the semantic security of the Modified ElGamal encryption scheme. Then, there exists an algorithm $\mathcal{B}$ that breaks the DDH assumption. To prove this claim, we construct $\mathcal{B}$ as follows. $\mathcal{B}$ receives on input from its challenger $\mathcal{C}$ a quadruple (g_1, g_2, h_1, h_2) of the DDH problem and has to determine whether it is a DH quadruple or not. So, $\mathcal{B}$ constructs the public key for the Modified ElGamal

scheme as follows. It chooses x_1 and x_2 at random, sets $h = g_1^{x_1} g_2^{x_2}$. Then, it provides $\mathcal{A}$ with (g_1, g_2, h) as the challenge parameters of the Modified ElGamal scheme.

Given two messages m_0 and m_1 (that $\mathcal{A}$ has output and wants to be challenged on), $\mathcal{B}$ proceeds as follows. First, it flips a random bit b. Then, it computes the ciphertext of m_b as $(h_1^k, h_2^k, m h_1^{k x_1} h_2^{k x_2})$ where k is chosen at random.

If the given quadruple is a DH one, then the ciphertext has the right distribution. Indeed, for some k', $h_1^k = g_1^{k'}$, $h_2^k = g_2^{k'}$ and $(h_1^{x_1} h_2^{x_2})^k = h^{k'}$. Otherwise (if it is not a DH quadruple), then one can easily check that $\mathcal{A}$ gets no information at all about the encrypted message (as, to decrypt the ciphertext, $\mathcal{A}$ must know the secrets x_1 and x_2 which are information theoretically hidden by h).

□

Paillier. The Paillier asymmetric encryption scheme [Pai99] works as follows:

- `KeyGen`: randomly chooses two different prime numbers a and b such that $a, b > 2$, $|a| = |b|$ and $gcd(ab, (a-1)(b-1)) = 1$. Then, it computes $n = ab$, $\lambda = lcm(a-1, b-1)$, $\mu = \lambda^{-1} \ mod \ n$ and $g_P = n + 1$. The private key is $sk = (a, b)$ and the corresponding public key is $pk = (n, g_P)$.

- $\texttt{Enc}(pk, m)$: selects a random $r \in \mathbb{Z}_n^*$. The encryption c of a message $m \in \mathbb{Z}_n$ is computed as $c = g_P^m r^n \ mod \ n^2$.

- $\texttt{Dec}(sk, c)$: decrypts the ciphertext c to recover m by computing $m = \frac{(c^\lambda \ mod \ n^2) - 1}{n} \ \mu \ mod \ n$.

The Paillier cryptosystem is semantically secure against chosen plaintext attacks (IND-CPA) under the Decisional Composite Residuosity (DCR) assumption.

3.3.4 Commitment

Commitments are quite useful in cryptography as they allow an entity to commit to a message m without revealing it. Once the commitment generated, the committing party must not be able to modify the committed message (*i.e.* claim that she has committed to a different message). Besides, she may sometimes need to open the commitment so as to ensure that she has really committed to a given message. To this end, the committing party must provide some additional information. Hereinafter, we provide a more formal definition.

3.3.4.1 Definition and Security Requirements

Definition 3.38 (Commitment)**.** *More formally, a commitment scheme consists of the following three algorithms:*

- $\texttt{Setup}(1^k)$: *a probabilistic algorithm, on input an integer k, outputs the system public parameters pp.*

- $\texttt{Commit}(\texttt{pp}, \texttt{m})$: *a probabilistic algorithm, on input a message m and the public parameters pp, outputs a commitment C to a message m as well as some opening information R required for the verification of the validity of the commitment.*

- $\texttt{Open}(\texttt{pp}, \texttt{m}, \texttt{C}, \texttt{R})$: *a deterministic algorithm, on input a message m, a commitment C and the opening information R, outputs either* `0` *or* `1` *depending on the validity of C with respect to m and R.*

A commitment scheme should meet the following two security requirements:

- *Hiding*: a commitment should not reveal any information about the committed message m. More formally, given two messages m_1 and m_2 and a commitment C to one of them, an adversary should not be able to tell which message has been committed to.

- *Binding*: a commitment C should be bound to the original message m so that the committing entity is unable to modify m afterwards. More formally, an adversary should not be able to output a commitment C along with two distinct pairs (m, r) and (m', r') such that $\texttt{Open}(\texttt{pp}, \texttt{m}, \texttt{C}, \texttt{r}) = \texttt{Open}(\texttt{pp}, \texttt{m}', \texttt{C}, \texttt{r}') = 1$.

It is worth mentioning that a commitment scheme cannot be both *perfectly hiding* and *perfectly binding*. In what follows, we recall one of the widely used commitment schemes, namely the Pedersen's commitment [Ped92].

3.3.4.2 Pedersen's Commitment

The Pedersen's commitment scheme works as follows:

- $\texttt{Setup}(1^k)$: takes as input an integer k and outputs the system public parameters $pp = (\mathbb{G}, p, g, h)$ where $\mathbb{G}$ is a cyclic group of prime order p and g and h are two random generators of $\mathbb{G}$.

- $\texttt{Commit}(pp, m)$: given a message $m \in \mathbb{Z}_p$, it randomly selects a value $r \in \mathbb{Z}_p$. Then, it computes $C = g^m h^r$ as the commitment to m. The algorithm returns (C, r).

- $\texttt{Open}(pp, m, C, r)$: given C, m and r as input, this algorithm checks whether $C \stackrel{?}{=} g^m h^r$. If so, it outputs `1` and `0` otherwise.

The Pedersen's commitment is *perfectly hiding* but only *computationally binding* under the DL assumption.

Indeed, given a commitment $C_m = g^m h^r$, every possible m' is equally likely to be the value committed in C_m. Indeed, if we consider m, m' and a value r, $\exists r'$ such that $g^m h^r = g^{m'} h^{r'}$. Indeed, if we denote by a the discrete logarithm of h in the base g (*i.e.* $h = g^a$, with $a \neq 0$), then $r' = \frac{m - m' + ra}{a}$. Consequently, C_m leaks no information about m which proves that the Pedersen's commitment is perfectly hiding.

Let us now explain why the *binding* property only holds under the DL assumption. To do so, we consider a committing party which is able to retrieve a, the discrete logarithm of h in the base g (*i.e.* $h = g^a$). Thus, it can select another message $m' \in \mathbb{Z}_p^*$ and compute the corresponding r' such that $m + ra = m' + r'a$ (*i.e.* $g^m h^r = g^{m'} h^{r'}$). Thereby, the Pedersen's commitment C_m can be opened to two distinct pairs (m, r) and (m', r') which makes it not binding anymore.

3.3.5 Proofs of Knowledge

The notion of *Zero-Knowledge interactive proofs* was put forward by Goldwasser, Micali and Rackoff in 1985 [GMR85]. Such proofs enable a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a given statement is valid without revealing anything else. Shortly after that, Feige, Fiat and Shamir [FFS] introduced the concept of *Zero Knowledge Proof of Knowledge*, denoted ZKPK, where the prover also convinces the verifier that he knows a secret satisfying a given statement without revealing anything else about it. Since then, ZKPKs have been widely used to design various protocols such as Schnorr's identification protocol described hereinafter. But first, we formally define zero-knowledge proofs of knowledge and recall the different properties that they must satisfy.

3.3.5.1 Zero-Knowledge Proof of Knowledge

Definition 3.39 (Zero Knowledge Proof of Knowledge)**.** *A Zero Knowledge Proof of Knowledge (ZKPK), is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, which meets the following three requirements:*

- Completeness*: a valid prover, i.e. knowing the secret s, should be accepted by an honest verifier with overwhelming probability.*

- Soundness*: a malicious prover, i.e. that does not know any secret verifying the statement $\mathcal{R}$, should be rejected by an honest verifier with overwhelming probability. This means that, using any prover accepted with overwhelming probability, it should be possible to build a Turing machine, called* knowledge extractor*, that is able to retrieve a valid secret s'.*

- Zero Knowledge*: the verifier knows nothing about the secret s except that it satisfies the statement $\mathcal{R}$ and that it is known to the prover. In other words, $\forall\ \mathcal{V}$, one should be able to build a simulator $\mathcal{S}$ whose output is indistinguishable from the transcript of the interactions between an honest prover $\mathcal{P}$ and $\mathcal{V}$.*

In the sequel, we will use the usual notation PoK$\{\alpha, \beta :$ *statement about* $\alpha, \beta\}$ introduced by Camenisch and Stadler [CS97] and where Greek letters correspond to the secrets known to the prover $\mathcal{P}$. Besides, we will only consider proofs that fulfill the zero-knowledge requirement with respect to an honest verifier. Such zero-knowledge proofs are called *honest-verifier* ZKPKs and, as shown below, they can be easily turned into non-interactive ZKPKs. It is also noteworthy to mention that there exists a generic technique that transforms *honest-verifier* ZK proofs into ZK proofs with respect to any verifier [DGOW95].

An honest-verifier ZKPK consists of three main rounds, namely *commitment*, *challenge* and *response*. Hereinafter, we provide more details about these three rounds through the review of Schnorr's identification protocol.

Schnorr's protocol. Schnorr's identification protocol, proposed in [Sch90], allows a prover to convince a verifier of his knowledge of the discrete logarithm x, in the base g, of a public value y (*i.e.* it enables the generation of the proof PoK$\{\alpha : y = g^{\alpha}\}$). The protocol works as follows.

- `Setup`(1^k): Generate the system public parameters $pp = (p, q, \mathbb{Z}_p, g)$ where p and q are two prime integers such that $q|p-1$ and $q > 2^k$, whereas $g \in \mathbb{Z}_p^*$.

- `KeyGen`(pp): Selects $x \in_R \mathbb{Z}_q^*$ as the prover's private key. The associated public key is $y = g^x \ mod\ p$.

- `Interactions`$(pp, \mathcal{P}(x), \mathcal{V}(y))$: The prover $\mathcal{P}$, who holds the secret x, and the verifier $\mathcal{V}$ engage in the following three round protocol:

  1. *Commitment*: $\mathcal{P}$ randomly chooses $a \in_R \mathbb{Z}_q$, then produces the commitment $t = g^a \ mod\ p$. The witness t is sent to $\mathcal{V}$.
  2. *Challenge*: once $\mathcal{V}$ receives t, it chooses a random challenge $c \in \mathbb{Z}_p$ and transmits it to $\mathcal{P}$.
  3. *Response*: upon the receipt of the challenge c, $\mathcal{P}$ computes $s = a + cx \ mod\ q$ and provides it to $\mathcal{V}$.

The verifier is convinced that $\mathcal{P}$ really knows the secret x only if $t = g^s y^{-c} \ mod\ p$.

OR Proofs of knowledge. An *OR* zero knowledge proof of knowledge enables the prover to convince the verifier that he knows a secret satisfying one statement among several ones without revealing which statement is met. If we consider the case of a prover knowing the discrete logarithm of one of two values in the base g, then the proof is denoted $\text{PoK}\{\alpha : y = g^\alpha \vee y' = g^\alpha\}$. The interactions between the prover and the verifier are as follows:

- *Commitment:* $\mathcal{P}$ randomly chooses $a \in_R \mathbb{Z}_q$, then produces the commitment $t = g^a \ mod \ p$ corresponding to the statement that is valid. For the other statement, the commitment is computed as $t' = g^{s'} y'^{-c'} \ mod \ p$ where c' and s' are selected at random. Both t and t' are sent to $\mathcal{V}$.

- *Challenge:* Once $\mathcal{V}$ receives t and t', it chooses a random challenge $c \in \mathbb{Z}_q$ and forwards it to $\mathcal{P}$.

- *Response:* $\mathcal{P}$ computes $C = c \oplus c'$ and $s = a + Cx \ mod \ q$ then provide $\mathcal{V}$ with (C, c', s, s').

The verifier is convinced that $\mathcal{P}$ really knows the discrete logarithm of y or y' only if $c = C \oplus c'$, $t = g^s y^{-C} \ mod \ p$ and $t' = g^{s'} y'^{-c'} \ mod \ p$.

Designated Verifier Proof (DVP). Introduced by Jakobsson, Sako and Impagliazzo [JSI96], a Designated Verifier Proof (DVP) is a particular ZKPK that only convinces the intended verifier and is completely useless for anyone else. A DVP mainly consists of a proof that either *"I know the private key associated to the designated verifier public key"* ***or*** *the given statement $\mathcal{R}$ is true.* Thus, upon the receipt of the proof, the designated verifier will be convinced that the statement $\mathcal{R}$ is valid whereas anyone else has no means to know if it is the case since the intended verifier can always prove himself to be the designated verifier. All he has to do is prove the knowledge of the private key associated to the designated verifier public key.

3.3.5.2 Non-Interactive Zero-Knowledge Proof of Knowledge

In 1986, Fiat and Shamir [FS87, PS00] introduced a technique enabling to convert a three round honest verifier ZKPK into a non-interactive ZKPK denoted NIZKPK. The main idea consists in letting the prover compute the challenge c instead of having the verifier select it at random. More precisely, $\mathcal{P}$ computes it as the output of a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$ taking as input the commitment t and the involved public values. Hereinafter, we convert Schnorr's interactive protocol into a non-interactive one using Fiat-Shamir heuristic. The `Setup` and `KeyGen` algorithms remain the same whereas the `Interactions` protocol is updated as follows:

- *Commitment*: $\mathcal{P}$ randomly chooses $a \in_R \mathbb{Z}_q$, then produces the commitment $t = g^a \ mod \ p$. The witness t is sent to $\mathcal{V}$.

- *Challenge*: $\mathcal{P}$ computes the challenge $c = \mathcal{H}(t, g, y)$.

- *Response*: Given the challenge c, $\mathcal{P}$ computes $s = a + cx \mod q$ and provides the tuple (t, c, s) to $\mathcal{V}$. Upon their receipt, the verifier first computes $t' = g^s y^{-c} \ mod \ p$. $\mathcal{V}$ is convinced that $\mathcal{P}$ really knows the secret x only if $c = \mathcal{H}(t', g, y)$.

The obtained NIZKPK may also be turned into a signature scheme commonly called *signature of knowledge* and denoted SoK. For that, the prover has just to add the corresponding message m to the input of the hash function (*i.e.* replace the challenge by $c = \mathcal{H}(m, t)$ and compute everything else as described above). Actually, this is how a Schnorr's signature as detailed in Section 3.3.2.5 is created.

3.4 Conclusion

In this chapter, we first recalled the required mathematical tools as well as the computational hardness assumptions that we rely on in this thesis. Then, we gave an overview on the concept of provable security. Finally, we formally defined the main cryptographic primitives needed in this thesis and reviewed the constructions of the used schemes such as the recent algebraic MAC scheme due to Chase *et al.* [CMZ14].

In the next chapter, we introduce new cryptographic schemes that either provide an additional feature that was not supported by previous proposals, or simply improve their efficiency. Afterwards, in Chapters 5-6, using our designed schemes as building blocks, we propose four privacy-preserving protocols that are suitable for resource constrained environments such as SIM cards while achieving a good trade-off between efficiency and the required security and privacy requirements.

Chapter 4

Design of Efficient Cryptographic Primitives

Contents

To cope with the various constraints stemming from devices with limited computational resources and applications stringent time constraints, more efficient cryptographic primitives need to be introduced. In this respect, we propose in this chapter five new efficient cryptographic primitives, that are of independent interest, and formally prove that they meet the expected security requirements. More precisely, they consist of a partially blind signature scheme [BBD+16], an algebraic MAC scheme [BBDT16], a sequential aggregate Message Authentication Code (MAC) scheme [ABBT16], and two pre-Direct Anonymous Attestations (Pre-DAA) schemes. The introduced primitives are used as the main building blocks of our practical privacy-preserving protocols detailed in subsequent chapters.

4.1 A Perfectly Unlinkable Partially Blind Signature Scheme

As previously mentioned in Section 3.3.2.3, a partially blind signature scheme allows the receiver of the signature $\mathcal{R}$ and the signer $\mathcal{S}$ to agree on a common information `info` to be added to the blind signature of a message m. Thereby, $\mathcal{S}$ can keep some control over the signed message by adding an expiration date or a validity period for example. In what follows, we detail our new partially blind signature scheme which enables multiple presentations of the same signature in an unlinkable way.

4.1.1 Scheme Description

Our partially blind signature scheme, which is based on the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme due to Chase *et al.* [CMZ14] (reviewed in Section 3.3.1.3), works as follows:

$\mathsf{Setup}(1^k)$ creates the system public parameters denoted by $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, h, \tilde{h}, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three cyclic group of prime order q, a k-bit prime, and g, h are two random generators of $\mathbb{G}_1$ such that $\log_g h$ is unknown, $\tilde{h}$ is a random generator of $\mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a Type-3 bilinear map. In the sequel, all computations on exponents are computed modulo q.

$\mathsf{KeyGen}(pp)$ randomly selects $\vec{x} = (x_0, x_1, \ldots, x_n) \in_R \mathbb{Z}_q^{n+1}$ as the signer's private key, denoted by sk, and picks a value $\tilde{x}_0 \in_R \mathbb{Z}_q$ to build a commitment $C_{x_0} = g^{x_0} h^{\tilde{x}_0}$ to the secret x_0. Denoted by $iparams$, $(C_{x_0}, X_1 = h^{x_1}, \ldots, X_n = h^{x_n}, \tilde{X}_0 = \tilde{h}^{x_0}, \ldots, \tilde{X}_n = \tilde{h}^{x_n})$ corresponds to the signer's public parameters.

$\mathsf{Sign}(\mathcal{R}(\vec{m}), \mathcal{S}(sk), info, pp, iparams)$ is an interactive protocol between a receiver $\mathcal{R}$ who wants to get a signature on a block of messages $\vec{m} = (m_1, \ldots, m_n)$ and a signer $\mathcal{S}$ who holds sk and acts as the signer. The partially blind signature on m and $info$ is produced as follows:

1. Blind: $\mathcal{R}$ sends the common value `info`, a commitment $C_{\vec{m}} = h^r X_1^{m_1} \ldots X_n^{m_n}$ to the block of messages $\vec{m} = (m_1, m_2, \ldots, m_n)$ where $r \in_R \mathbb{Z}_q^*$ as well as a ZKPK π_1 ensuring that the commitment is well-formed. The proof π_1 is defined as $\pi_1 = PoK\{\alpha_1, \alpha_2, \ldots, \alpha_n, \beta : C_{\vec{m}} = h^\beta X_1^{\alpha_1} \ldots X_n^{\alpha_n}\}$.
2. CSign: Before signing the commitment $C_{\vec{m}}$, $\mathcal{S}$ first checks the validity of π_1. If so, he computes $u'' = u^{x_0}(C_{\vec{m}}(X_n)^{\texttt{info}})^b$ where $b \in_R \mathbb{Z}_q^*$ and $u = h^b$. Then, he provides $\mathcal{R}$ with the partially blind signature $((u, u''), \texttt{info})$ as well as a ZKPK π_2 proving that $u'' = u^{x_0 + x_1 m_1 + \ldots + x_{n-1} m_{n-1} + x_n(m_n + \texttt{info})} h^{br}$. π_2 is defined as $\pi_2 = PoK\{\alpha, \beta, \gamma : u'' = u^\alpha (C_{\vec{m}}(X_n)^{\texttt{info}})^\beta \wedge C_{x_0} = g^\alpha h^\gamma \wedge u = h^\beta\}$.
3. Retrieval: Upon their receipt, $\mathcal{R}$ first checks the validity of π_2. If so, it unblinds (u, u'') to retrieve the signature $\sigma = (u, u') = (u, \frac{u''}{u^r}) = (u, u^{x_0 + x_1 m_1 + \ldots + x_{n-1} m_{n-1} + x_n(m_n + \texttt{info})})$ on both $\vec{m}$ and `info`.

Receiver $\mathcal{R}$		**Signer $\mathcal{S}$**
Private input: $m_1, \ldots, m_n$		**Private input:** $x_0, x_1, \ldots, x_n$
Choose $r, a_1, a_2, \ldots, a_n, b_1 \xleftarrow{R} \mathbb{Z}_q^*$ **Compute** $C_{\vec{m}} \leftarrow h^r X_1^{m_1} \ldots X_n^{m_n}$ $t_1 \leftarrow h^{b_1} X_1^{a_1} \ldots X_n^{a_n}$		
Compute $c = \mathcal{H}(Ch, t_1)$	$\xleftarrow{\quad Ch \quad}$	**Choose** $Ch \in_R \mathbb{Z}_q^*$
$\forall i \in \{1, \ldots, n\}, s_i \leftarrow a_i + c m_i \mod q$ and $s \leftarrow b_1 + cr \mod q$	$\xrightarrow{\quad C_{\vec{m}},\ \pi_1 = (c, s_1, \ldots, s_n, s) \quad}$	**Compute** $t_1' = h^s X_1^{s_1} \ldots X_n^{s_n} (C_{\vec{m}})^{-c}$ **Check** $\mathcal{H}(Ch, t_1') \stackrel{?}{=} c$ **and Choose** $b \xleftarrow{R} \mathbb{Z}_q^*$ **Compute** $u \leftarrow h^b$ $u'' \leftarrow u^{x_0}(C_{\vec{m}}(X_n)^{\texttt{info}})^b$ **Choose** $a_1, a_2, a_3 \xleftarrow{R} Z_q^*$ **Compute** $t_1 \leftarrow u^{a_1}(C_{\vec{m}} X_n^{info})^{a_2}$ $t_2 = g^{a_1} h^{a_3}$, $t_3 \leftarrow h^{a_2}$ and $c_1 \leftarrow \mathcal{H}(t_1, t_2, t_3)$
Compute $t_1' \leftarrow u^{s_1}(C_{\vec{m}} X_n^{info})^{s_2} u''^{-c_1}$ $t_2' \leftarrow g^{s_1} h^{s_3} (C_{x_0})^{-c_1}$ and $t_3' \leftarrow h^{s_2} u^{-c_1}$ **Check** $\mathcal{H}(t_1', t_2', t_3') \stackrel{?}{=} c_1$ **Compute** $u' \leftarrow \frac{u''}{u^r}$	$\xleftarrow{\quad (u, u''), \pi_2 = (c_1, s_1, s_2, s_3) \quad}$	**Compute** $s_1 \leftarrow a_1 + c_1 x_0 \ mod\ q$ $s_2 \leftarrow a_2 + c_1 b \ mod\ q$ $s_3 \leftarrow a_3 + c_1 \tilde{x}_0 \ mod\ q$

Figure 4.1: Issuance of a partially blind signature

4. Randomization: To show the obtained signature in an anonymous way, the receiver just has to randomize it by computing $\sigma^l = (u^l, (u')^l)$ where $l \in_R \mathbb{Z}_q^*$.

Verify$(pp, iparams, \vec{m}, \texttt{info}, \sigma)$ checks the validity of the digital signature $\sigma = (u, u')$ with respect to a message $\vec{m}$ and a common information info. This algorithm works differently depending on whether the verifier holds $(x_1, \ldots, x_n)$ or not. If so, the signature is valid only if $u \neq 1$ and $u' = u^{x_0 + x_1 m_1 + \ldots + x_{n-1} m_{n-1} + x_n (m_n + \texttt{info})}$. Otherwise, the verifier relies on pairing computations so as to check the validity of σ. More precisely, the signature is valid only if $u \neq 1$ and $e(u, \tilde{X}_0) \cdot e(u, \tilde{X}_1)^{m_1} \cdot e(u, \tilde{X}_2)^{m_2} \cdots e(u, \tilde{X}_n)^{m_n + \texttt{info}} = e(u', \tilde{h})$.

4.1.2 Security Analysis

As mentioned in section 3.3.2.3, a partially blind signature scheme should provide *one-more unforgeability* and *unlinkability*. Hereinafter, we prove that our partially blind signature scheme fulfills these two properties. It is even *perfectly* unlinkable.

Theorem 4.1. *Our partially blind signature scheme is* one-more unforgeable*, in the random oracle model, under the assumption* 1 *(see Section 3.23).*

Proof (sketch). In the sequel, $\mathcal{O}$PBS will refer to an oracle that, on input a block of messages $\vec{m} = (m_1, m_2, \ldots, m_n)$ hidden using a Pedersen's commitment, outputs a partially blind signature $\sigma = (u, u'')$ on $\vec{m}$ and info.

Let $\mathcal{A}$ be an adversary against the one-more unforgeability of our partially blind signature PBS scheme. Using $\mathcal{A}$ as a subroutine, we construct a reduction $\mathcal{B}$ against the assumption 1. The adversary $\mathcal{A}$ succeeds only if, after L signature queries to the $\mathcal{O}$PBS oracle, he manages to output, with non negligible probability, $L+1$ valid $(\vec{m}, \texttt{info}, \sigma)$ tuples on distinct messages and such that $\forall i, j \in \{1, \ldots, L+1\}, j \neq i,\ \vec{m}_i + \overrightarrow{\texttt{info}}_i \neq \vec{m}_j + \overrightarrow{\texttt{info}}_j$ where $\overrightarrow{\texttt{info}}_i = (\underbrace{0, 0, \ldots, 0}_{(n-1)\ times}, \texttt{info}_i)$.

Let $\mathcal{C}$ denote the challenger of the game. $\mathcal{B}$ receives on input from $\mathcal{C}$ the parameters $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ as well as $(h, X_1 = h^{x_1}, \ldots, X_n = h^{x_n}, \tilde{h}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}, \ldots, \tilde{X}_n = \tilde{h}^{x_n})$ and has access to the oracle $\mathcal{O}_1$ which, on input $\vec{m} \in \mathbb{Z}_q^n$, outputs a pair $P = (u, u^{x_0 + x_1 m_1 + \ldots + x_n m_n})$ where $u \in_R \mathbb{G}_1$. $\mathcal{B}$ can choose g and C_{x_0} at random. Indeed, $\forall x_0 \in \mathbb{Z}_q$, $\exists \tilde{x}_0 \in \mathbb{Z}_q$ such that $C_{x_0} = g^{x_0} h^{\tilde{x}_0}$. Then, it provides $\mathcal{A}$ with $(pp,\ h,\ g,\ \tilde{h},\ C_{x_0},\ X_1, X_2, \ldots, X_n, \tilde{X}_0, \tilde{X}_1, \ldots,\ \tilde{X}_n)$ and, through queries to $\mathcal{O}_1$, it perfectly simulates $\mathcal{A}$'s requests to the $\mathcal{O}$PBS oracle. More precisely, $\mathcal{B}$ uses the replay technique as well as the soundness property of π_1 to retrieve the values of r and $\vec{m} = (m_1, m_2, \ldots, m_n)$. Then, it calls on the oracle $\mathcal{O}_1$ with $\vec{m} = (m_1, m_2, \ldots, m_n + \texttt{info})$ as input to get a valid pair $(u, u' = u^{x_0 + m_1 x_1 + \ldots + (m_n + \texttt{info}) x_n})$. Thus, $\mathcal{B}$ can compute $(u, u'' = u'u^r)$, a partially blind signature on $\vec{m}$ and info. As for π_2, $\mathcal{B}$ can perfectly simulate it in the random oracle model. Therefore, $\mathcal{B}$ provides $\mathcal{A}$ with (u, u'') as well as π_2.

Eventually, after L calls to $\mathcal{O}$PBS, $\mathcal{A}$ returns with non negligible probability $L+1$ valid $(\vec{m}, \texttt{info}, \sigma)$ tuples such that $\forall i, j \in \{1, \ldots, L+1\}, j \neq i,\ \vec{m}_i + \overrightarrow{\texttt{info}}_i \neq \vec{m}_j + \overrightarrow{\texttt{info}}_j$ where $\overrightarrow{\texttt{info}}_i = (\underbrace{0, 0, \ldots, 0}_{(n-1)\ times}, \texttt{info}_i)$. Therefore, $\mathcal{B}$ gets $L+1$ valid $(\vec{m}', \sigma)$ pairs on $L+1$ distinct messages where $\vec{m}' = (m_1, m_2, \ldots, m_n + info)$; hence breaking the assumption 1. □

Theorem 4.2. *Our partially blind signature scheme is* perfectly unlinkable.

Proof (sketch). The view of the signer upon an execution of the protocol Sign consists of a Pedersen's commitment $C_{\vec{m}} = h^r X_1^{m_1} \ldots\ X_n^{m_n}$, a proof π_1 and the computed pair $(u = h^b, u'' = u^{x_0 + m_1 x_1 + \ldots + (m_n + info) x_n} h^{br})$.

The proof π_1 is a classical variant of Schnorr's proof of knowledge of a discrete logarithm (*c.f.* Section 3.3.5.1) which is an honest verifier ZKPK. Therefore, it does not reveal any information on r and the m_i's. Besides, the Pedersen's commitment is perfectly hiding (*c.f.* Section 3.3.4.2). Therefore, neither $C_{\vec{m}}$ nor π_1 leaks any information about $\vec{m}$. Consequently, only the pair (u, u'') may help the adversary $\mathcal{A}$ link a valid message/signature pair to a specific execution of the Sign protocol.

Let us consider a randomized version $(v = \tilde{u}^l, v' = (\tilde{u}')^l)$ of a valid signature $(\tilde{u}, \tilde{u}')$ on the set of messages $(m'_1, m'_2, \ldots, m'_n + \texttt{info})$ distinct from $(m_1, m_2, \ldots, m_n + \texttt{info})$. For any such pair (v, v'), there exists l', r' such that $v = u^{l'}$ and $C_{\vec{m}} = h^{r'} X_1^{m'_1} \ldots X_n^{m'_n}$. Let $u' = \frac{u''}{u^{r'}}$, then $v' = (u')^{l'}$. Therefore, (v, v') could also have been obtained during the signing session that led to the transcript $(C_{\vec{m}}, \pi_1, u, u'')$. Consequently, our partially blind signature is perfectly unlinkable. □

4.2 An Improved Algebraic MAC Scheme

In this section, we design a new algebraic MAC scheme which relies on a *pairing-free* variant of the Boneh-Boyen's signature scheme [CCJT14]. Unlike Chase *et al.* algebraic MAC schemes [CMZ14], our proposal only requires a *sole* secret key regardless of the number of messages to be signed. Furthermore, MACs generated using our scheme may be publicly verifiable. In what follows, we detail our construction which can be applied to both a single message as well as a block of messages.

4.2.1 Scheme Description

4.2.1.1 $\mathsf{MAC}_{\mathsf{BB}}$

Our algebraic MAC scheme for a single message m, referred to as $\mathsf{MAC}_{\mathsf{BB}}$, is defined through the following four algorithms:

$\texttt{Setup}(1^k)$ creates the system public parameters $pp = (\mathbb{G}, p, h, g_0, g_1, g)$ where $\mathbb{G}$ is a cyclic group of prime order p, a k-bit prime, and h, g_0, g_1, g are four random generators of $\mathbb{G}$.

$\texttt{KeyGen}(pp)$ selects a random value $y \in_R \mathbb{Z}_p$ as the issuer's private key and *optionally* computes the corresponding public key $Y = g_0^y$.

$\texttt{MAC}(m, y)$ picks two random values $r, s \in_R \mathbb{Z}_p$ and computes $A = (g_1^m g^s h)^{\frac{1}{y+r}}$. The MAC on a message m consists of the triple (A, r, s).

$\texttt{Verify}(m, A, r, s, y)$ checks the validity of the MAC (A, r, s) with respect to the message m. The MAC is valid only if $(g_1^m g^s h)^{\frac{1}{y+r}} = A$.

4.2.1.2 $\mathsf{MAC}^n_{\mathsf{BB}}$

Our algebraic MAC scheme can be generalized to support a block of n messages $(m_1, \ldots, m_n)$. This extension is referred to as $\mathsf{MAC}^n_{\mathsf{BB}}$ and works as follows:

$\texttt{Setup}(1^{\texttt{k}})$ creates the system public parameters $pp = (\mathbb{G}, p, g_1, g_2, \ldots, g_n, h, g_0, g)$ where $\mathbb{G}$ is a cyclic group of prime order p, a k-bit prime, and $h, g, g_0, g_1, \ldots, g_n$ are random generators of $\mathbb{G}$.

$\texttt{KeyGen}(pp)$ selects a random value $y \in_R \mathbb{Z}_p$ as the issuer's private key and *optionally* computes the corresponding public key $Y = g_0^y$.

$\texttt{MAC}(\vec{m}, y)$ takes as input a block of n messages $\vec{m} = (m_1, \ldots, m_n)$ and picks $r, s \in_R \mathbb{Z}_p$ so as to compute $A = (g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}}$. The MAC on $\vec{m}$ consists of the triple (A, r, s).

$\texttt{Verify}(\vec{m}, A, r, s, y)$ checks the validity of the MAC with respect to the block of messages $\vec{m}$. The MAC is valid only if $(g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}} = A$.

One particular feature of our algebraic MAC scheme is that anyone can verify the validity of a given MAC by himself (*i.e.* without neither knowing the private key y nor querying the `Verify` algorithm). Indeed, a MAC on $\vec{m} = (m_1, \ldots, m_n)$ consists of the triple (A, r, s) such that $A = (g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}}$. This implies that $A^{y+r} = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h$ and hence, $B = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h \cdot A^{-r} = A^y$. Therefore, if the issuer of the MAC also provides a ZKPK defined as $\pi = \text{PoK}\{\gamma : B = A^\gamma \wedge Y = g_0^\gamma\}$, then its receiver will be convinced that the MAC is valid.

Furthermore, unlike both algebraic MAC schemes due to Chase *et al.* [CMZ14], the issuer does not have to hold as many private keys as messages but rather a sole private key regardless of the number of messages.

4.2.2 Security Analysis

4.2.2.1 Security of $\mathsf{MAC_{BB}}$

Theorem 4.3. *Our* $\mathsf{MAC_{BB}}$ *scheme is* sUF-CMVA[1] *secure under the gap* $q-\mathsf{SDH}-\mathsf{III}$ *assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary who breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme with non-negligible probability. Using $\mathcal{A}$, we construct a reduction $\mathcal{B}$ against the $q-\mathsf{SDH}$ assumption in gap-DDH groups (which implies the *gap* $q-\mathsf{SDH}-\mathsf{III}$ assumption). $\mathcal{A}$ can ask for tags on any message of his choice and receives the corresponding tags (A_i, r_i, s_i) for $i \in \{1, \ldots, q\}$ where q denotes the number of requests to the $\mathcal{O}\texttt{MAC}$ oracle. Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) for the message m. We distinguish the following two types of forgeries:

- **Type-1 Forger:** an adversary that outputs a valid tag (A, r, s) on m such that $(A, r) \neq (A_i, r_i)$ for all $i \in \{1, \ldots, q\}$.
- **Type-2 Forger:** an adversary that outputs a valid tag (A, r, s) on m such that $(A, r) = (A_j, r_j)$ for some $j \in \{1, \ldots, q\}$ and $(m, s) \neq (m_j, s_j)$.

We show that, regardless of their type, both adversaries can be used to break the gap $q-\mathsf{SDH}$ assumption. However, the reduction works differently for each type of forger. Consequently, $\mathcal{B}$ initially chooses a random bit $c_{mode} \in \{1, 2\}$ which indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$**:** $\mathcal{B}$ receives on input from its $q-\mathsf{SDH}$ challenger, denoted by $\mathcal{C}$, the public parameters (g_0, g_1, h) and the public key $Y = g_0^y$ as well as q random, and distinct, triplets (A_i, r_i, m_i) such that $A_i = (g_1^{m_i} h)^{\frac{1}{r_i+y}}$ for $i \in \{1, \ldots, q\}$. As it is against the *gap* $q-\mathsf{SDH}-\mathsf{III}$ assumption, $\mathcal{B}$ has access to a DDH oracle, denoted by $\mathcal{O}\texttt{DDH}$, that decides whether a given quadruple (g, h, g^x, h^y) is a valid Diffie-Hellman quadruple (*i.e.* whether $x \stackrel{?}{=} y \mod p$) or not. $\mathcal{B}$ also randomly chooses $v \in_R \mathbb{Z}_p$ and computes $g = g_1^v$. Thereby, it can provide $\mathcal{A}$ with the public parameters (g_0, g_1, h, g, Y) and answer his requests as follows:

[1] Recall that sUF-CMVA refers to *strong* UF-CMVA where the adversary wins even if he outputs a **new** valid pair (m, τ) on a message m that has already been queried to the $\mathcal{O}\texttt{MAC}$ oracle.

- $\mathcal{O}\texttt{MAC}$ requests: given m as input, $\mathcal{B}$ first picks one of the received triplets (A_i, r_i, m_i) that has not already been used. Then, it computes s_i such that $m+vs_i = m_i$. Finally, it provides $\mathcal{A}$ with the triplet (A_i, r_i, s_i) which is a valid MAC on m (*i.e.* $A_i^{y+r_i} = g_1^m g^{s_i} h$). Thereby, the simulation of this oracle is perfect.

- $\mathcal{O}\texttt{Verify}$ requests: given a quadruple (A, r, s, m), $\mathcal{B}$ first verifies that $A \neq 1$ and computes $B = A^{-r} g_1^m g^s h$. Then, it provides the quadruple (g_0, A, Y, B) as input to the $\mathcal{O}\texttt{DDH}$ oracle so as to know if it is valid or not. $\mathcal{B}$ forwards the oracle's answer to $\mathcal{A}$, thus perfectly simulating $\mathcal{O}\texttt{Verify}$.

Eventually, after q queries to $\mathcal{O}\texttt{MAC}$ and q_v queries to $\mathcal{O}\texttt{Verify}$, $\mathcal{A}$ outputs his forgery (A, r, s) on m such that it breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme. Using these values, $\mathcal{B}$ computes $\tilde{m} = m + sv$ and outputs its forgery $(A, r, \tilde{m})$, thus breaking the $q - \mathsf{SDH}$ assumption with the same advantage as $\mathcal{A}$.

If $c_{mode} = 2$: A Type-2 adversary $\mathcal{A}$ is rather used, as a subroutine, to construct a reduction $\mathcal{B}$ against the DL problem. In such case, $\mathcal{B}$ receives on input from its DL challenger, denoted by $\mathcal{C}$, the challenge $(g_1, H = g_1^v)$. Its goal is to find the value v. For this purpose, it first randomly chooses $(y, g_0, h) \in_R \mathbb{Z}_p \times \mathbb{G}^2$ and computes $Y = g_0^y$. $\mathcal{B}$ also sets g as $g = H$. Thereby, it can provide $\mathcal{A}$ with the public parameters (g_1, g_0, h, g, Y) and answer his requests as follows:

- $\mathcal{O}\texttt{MAC}$ requests: as it holds y, $\mathcal{B}$ can generate a valid MAC (A, r, s) on any queried message m. To do so, it simply computes $A = (g_1^m g^s h)^{\frac{1}{y+r}}$ where $r, s \in \mathbb{Z}_p^*$.

- $\mathcal{O}\texttt{Verify}$ requests: given a quadruple (A, r, s, m), $\mathcal{B}$ computes $\tilde{A} = (g_1^m g^s h)^{\frac{1}{y+r}}$. To check its validity, and answer $\mathcal{A}$'s query, $\mathcal{B}$ verifies whether $\tilde{A} \stackrel{?}{=} A$.

Eventually, after q queries to $\mathcal{O}\texttt{MAC}$ and q_v queries to $\mathcal{O}\texttt{Verify}$, $\mathcal{A}$ outputs his forgery (A, r, s) on m such that he breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme. By assumption, (A, r) is equal to one of the (A_j, r_j) output by the $\mathcal{O}\texttt{MAC}$ oracle following $\mathcal{A}$'s request for some $j \in \{1, \ldots, q\}$. Since $(A, r) = (A_j, r_j)$, then $A_j^{y+r_j} = g_1^{m_j} g^{s_j} h = A^{y+r} = g_1^m g^s h$ and so, $g_1^{m_j} g^{s_j} = g_1^m g^s$. We therefore necessarily have $s_j \neq s$, otherwise this would imply that $m = m_j$ (contradicting the fact that we have supposed $(m, s) \neq (m_j, s_j)$). Thereby, $g = g_1^{(\frac{m-m_j}{s_j-s})}$. Using the values (m, m_j, s, s_j), $\mathcal{B}$ can recover v, hence breaking the DL problem. If $\mathcal{B}$ can break the DL problem, then it can break the $q - \mathsf{SDH}$ problem (by finding the discrete logarithm y of g^y in the base g).

$\mathcal{B}$ can guess which type of forgery a particular adversary $\mathcal{A}$ will output with probability $1/2$. So, $\mathcal{B}$ can break the gap $q - \mathsf{SDH}$ problem with probability $\varepsilon/2$ where ε is the probability that $\mathcal{A}$ breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme. Therefore, under the gap $q - \mathsf{SDH}$ assumption, our $\mathsf{MAC_{BB}}$ scheme is sUF-CMVA secure.

□

4.2.2.2 Security of $\mathsf{MAC}^n_{\mathsf{BB}}$

Theorem 4.4. *Our $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme is sUF-CMVA secure under the assumption that $\mathsf{MAC_{BB}}$ is sUF-CMVA.*

Proof (sketch). Let $\mathcal{A}$ be an adversary who breaks the unforgeability of our $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme with non-negligible probability. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ against the unforgeability of $\mathsf{MAC_{BB}}$. $\mathcal{A}$ can ask for tags on blocks of messages $\vec{m_1} = (m_1^1, \ldots, m_n^1)$, $\vec{m_2} = (m_1^2, \ldots, m_n^2), \ldots,$ $\vec{m_q} = (m_1^q, \ldots, m_n^q)$ and receives the corresponding tags (A_i, r_i, s_i) for $i \in \{1, \ldots, q\}$. Eventually,

$\mathcal{A}$ outputs his forgery (A, r, s) for the block of messages $\vec{m} = (m_1, \ldots, m_n)$. We differentiate two types of forgers:

- **Type-1 Forger:** an adversary that outputs a forgery where $(A, r, s) \neq (A_i, r_i, s_i)$ for $i \in \{1, \ldots, q\}$.
- **Type-2 Forger:** an adversary that outputs a forgery where $(A, r, s) = (A_i, r_i, s_i)$ for some $i \in \{1, \ldots, q\}$ and $(m'_1, \ldots, m'_n) \neq (m^i_1, \ldots, m^i_n)$.

We show that any of these two forgers can be used to forge $\mathsf{MAC_{BB}}$ tags. The reduction works differently for each type of forger. Therefore, $\mathcal{B}$ initially chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will emulate.

If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its $\mathsf{MAC_{BB}}$ challenger, denoted by $\mathcal{C}$, the public parameters (g_0, g_1, g, h) as well as the public key $Y = g_0^y$. Then, $\mathcal{B}$ constructs the public parameters for $\mathcal{A}$ as follows: for $i \in \{2, \ldots, n\}$, $\mathcal{B}$ chooses $\alpha_i \in_R \mathbb{Z}_p^*$ and computes $g_i = g_1^{\alpha_i}$. The parameters g_0, g_1, g, h and Y are the same as those sent by $\mathcal{C}$. $\mathcal{B}$ can answer $\mathcal{A}$'s requests as follows:

- $\mathcal{O}$`Verify` requests: when $\mathcal{A}$ sends a verify request to $\mathcal{B}$ on (A, r, s) and a block of messages $(m_1, \ldots, m_n)$, $\mathcal{B}$ computes $M = m_1 + \alpha_2 m_2 + \ldots + \alpha_n m_n$. Then, it queries its $\mathsf{MAC_{BB}}$ `Verify` oracle on (A, r, s, M) and forwards the oracle's answer to $\mathcal{A}$.
- $\mathcal{O}$`MAC` requests: when $\mathcal{A}$ sends a tag request to $\mathcal{B}$ on the block of messages $(m_1, \ldots, m_n)$, $\mathcal{B}$ queries the $\mathsf{MAC_{BB}}$ oracle on $M = m_1 + \alpha_2 m_2 + \ldots + \alpha_n m_n$. Thus, $\mathcal{B}$ obtains the tag (A_i, r_i, s_i). It sends back (A_i, r_i, s_i) to $\mathcal{A}$ which is a valid $\mathsf{MAC^n_{BB}}$ tag on $(m_1, \ldots, m_n)$.

Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) on the block of messages $(m_1, \ldots, m_n)$. Using these values, $\mathcal{B}$ directly outputs its $\mathsf{MAC_{BB}}$ forgery (A, r, s) on $M' = m_1 + \alpha_2 m_2 + \ldots + \alpha_n m_n$. Therefore, $\mathcal{B}$ breaks the unforgeability of $\mathsf{MAC_{BB}}$ with the same advantage as $\mathcal{A}$.

If $c_{mode} = 2$: In this case, $\mathcal{A}$ is rather used as a subroutine to construct a reduction $\mathcal{B}$ against the DL problem. $\mathcal{B}$ receives on input from its DL challenger, denoted by $\mathcal{C}$, the challenge $(g, H = g^v)$. The goal of $\mathcal{B}$ consists in finding the value of v. For that purpose, it first randomly chooses $(y, g_0, h) \in_R \mathbb{Z}_p \times \mathbb{G}^2$ and computes $Y = g_0^y$. Then, it chooses $I \in \{1, \ldots, n\}$ and $(n-1)$ random values $\alpha_i \in \mathbb{Z}_p^*$. It computes, for $i \neq I, g_i = g^{\alpha_i}$ and defines $g_I = H$. $\mathcal{B}$ can answer $\mathcal{A}$'s requests as follows:

- $\mathcal{O}$`Verify` requests: when $\mathcal{A}$ sends a verify request to $\mathcal{B}$ on (A, r, s) and a block of messages $\vec{m} = (m_1, \ldots, m_n)$, $\mathcal{B}$ computes $\tilde{A} = (g_1^{m_1} \ldots g_n^{m_n} g^s \cdot h)^{\frac{1}{y+r}}$. It can thus check the validity of the quadruple $(A, r, s, \vec{m})$ by verifying whether $\tilde{A} \stackrel{?}{=} A$.
- $\mathcal{O}$`MAC` requests: as it holds y, $\mathcal{B}$ can generate a valid MAC (A, r, s) on any queried block of messages $(m_1, \ldots, m_n)$. To this end, it chooses $r, s \in_R \mathbb{Z}_p^*$ and computes $A = (g_1^{m_1} \ldots g_n^{m_n} g^s \cdot h)^{\frac{1}{y+r}}$.

Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) on a block of messages $\vec{m} = (m_1, \ldots, m_n)$. By assumption, (A, r, s) is equal to one of $\mathcal{A}$'s requests, let us say (A_i, r_i, s_i), but it is a forgery on a new block of messages. Therefore, $(m^i_1, \ldots, m^i_n) \neq (m_1, \ldots, m_n)$ (one can easily show that there is at least one difference between the two blocks of messages). So, with probability $\frac{1}{n}, m^i_I \neq m_I$. Thus, since $(A, r, s) = (A_i, r_i, s_i)$, we have $g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} = g_1^{m^i_1} g_2^{m^i_2} \ldots g_n^{m^i_n}$. Hence, the discrete logarithm v of $H = g_I$ in the base g is equal to: $v = \sum_{j \neq I}^{n} \alpha_j \frac{(m_j - m^i_j)}{m^i_I - m_I}$. Therefore, $\mathcal{B}$ can find v

with probability $\frac{\varepsilon}{n}$, where ε is the probability that $\mathcal{A}$ breaks the unforgeability of $\mathsf{MAC}^n_{\mathsf{BB}}$. If $\mathcal{B}$ can break the discrete logarithm DL problem then, it can break the $\mathsf{MAC}_{\mathsf{BB}}$ scheme (by finding the discrete logarithm of Y in the base g_0).

We can guess which of the two forgers a particular adversary $\mathcal{A}$ is with probability 1/2. So, assuming the most pessimistic scenario (case 2), $\mathcal{B}$ can break the unforgeability of $\mathsf{MAC}_{\mathsf{BB}}$ with probability $\varepsilon/2n$.

□

4.3 A New Sequential Aggregate MAC Scheme

As mentioned in Section 3.3.2.4, a sequential aggregate signature scheme is a particular type of aggregate signature schemes where the final signature is created sequentially with each signer signing the aggregate signature in turn. The *secret key* analogue of these signature schemes is referred to as sequential aggregate Message Authentication Code (MAC) schemes [KL08].

Hereinafter, based on the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme due to Chase *et al.* [CMZ14], we design an efficient sequential aggregate MAC scheme which supports n users with n different messages.

4.3.1 Scheme Description

Our sequential aggregate MAC scheme is defined through the following four algorithms: Setup, KeyGen, AggMAC and Verify. For the sake of clarity, we detail it in the case of two users denoted by $\mathcal{U}_1$ and $\mathcal{U}_2$.

$\mathsf{Setup}(1^k)$ creates the system public parameters denoted by $pp = (\mathbb{G}, q, g, h, Y_0)$ where $\mathbb{G}$ is a cyclic group of prime order q, a k-bit prime, g and h are two random generators of $\mathbb{G}$ such that $\log_g h$ is unknown, and $Y_0 = g^{x_0}$.

$\mathsf{KeyGen}(pp)$ generates the secret key $sk_1 = x_1 \in_R \mathbb{Z}_q^*$ of the first user $\mathcal{U}_1$, and $sk_2 = x_2 \in_R \mathbb{Z}_q^*$ of the second user $\mathcal{U}_2$. The corresponding public parameters, denoted *params*, are respectively $X_1 = h^{x_1}$ and $X_2 = h^{x_2}$.

$\mathsf{AggMAC}(pp, \mathcal{U}_1(sk_1, m_1), \mathcal{U}_2(sk_2, m_2))$ produces an aggregate MAC on both messages m_1 and m_2 sequentially by users $\mathcal{U}_1$ and $\mathcal{U}_2$. First, $\mathcal{U}_1$ randomly selects $t_1 \in \mathbb{Z}_q^*$ and generates the MAC $\sigma_1 = (u, u')$ on the message m_1 where $u = g^{t_1}$ and $u' = (g^{m_1 x_1} Y_0)^{t_1} = u^{x_0 + m_1 x_1}$. Then, he provides $\mathcal{U}_2$ with σ_1 as well as a ZKPK π_1 that u' was correctly computed. π_1 is defined as $\pi_1 = \text{PoK}\{\alpha, \beta : u = g^\alpha \wedge u' = (g^{m_1\beta} Y_0)^\alpha \wedge X_1 = h^\beta\}$.

1^{st} **user** $\mathcal{U}_1$		2^{nd} **user** $\mathcal{U}_2$
Public input: $\mathbb{G}, q, g, h, Y, X_1, X_2$		
Private input: x_1		**Private input:** x_2
Choose $t_1 \xleftarrow{R} \mathbb{Z}_q^*$ **Compute** $u \leftarrow g^{t_1}$ and $u' \leftarrow (g^{m_1 x_1} Y_0)^{t_1}$ **Build** $\pi_1 = PoK\{\alpha, \beta : u = g^\alpha \wedge X_1 = h^\beta \wedge u' = (g^{m_1\beta} Y_0)^\alpha\}$ **Set** $\sigma_1 \leftarrow (u, u')$	$\xrightarrow{\sigma_1, \pi_1}$	**Choose** $t_2 \xleftarrow{R} \mathbb{Z}_q^*$ **Compute** $w \leftarrow u^{t_2}$ and $w' \leftarrow (u' u^{m_2 x_2})^{t_2}$ **Build** $\pi_2 = \text{PoK}\{\alpha, \beta : w = u^\alpha \wedge w' = (u' u^{m_2\beta})^\alpha \wedge X_2 = h^\beta\}$ **Set** $\sigma_2 \leftarrow (w, w')$

Figure 4.2: Our sequential aggregate MAC scheme

Upon their receipt, $\mathcal{U}_2$ first verifies the validity of π_1. If so, he computes the sequential aggregate MAC $\sigma_2 = (w, w') = (u^{t_2}, (u'u^{m_2 x_2})^{t_2})$ on both m_1 and m_2 where $t_2 \in_R \mathbb{Z}_q^*$. $\mathcal{U}_2$ also builds a ZKPK π_2 defined as $\pi_2 = \text{PoK}\{\alpha, \beta : w = u^\alpha \wedge w' = (u'u^{m_2\beta})^\alpha \wedge X_2 = h^\beta\}$.

$\mathsf{Verify}(pp, \sigma_2, m_1, m_2, sk_1, sk_2)$ checks whether $\sigma_2 = (w, w')$ is a valid aggregate of the MAC of $\mathcal{U}_1$ on m_1 and $\mathcal{U}_2$ on m_2. More precisely, it verifies if $u \neq 1$ and $w' \stackrel{?}{=} w^{x_0 + m_1 x_1 + m_2 x_2}$.

4.3.2 Security Analysis

Theorem 4.5. *Our sequential aggregate MAC scheme is* existentially unforgeable under chosen message and verification attacks *(EUF-CMVA), in the random oracle model, under the assumption that* $\mathsf{MAC}_{\mathsf{GGM}}$ *is UF-CMVA secure.*

Proof (sketch). In the sequel, we consider the setting proposed by Lu *et al.* [LOS+06] where, in order to add a new public parameter pk (associated to a secret key sk) to `KeyList`, the users must first prove the knowledge of the associated secret key sk. Thereby, the reduction will be able to answer all the adversary's MAC queries. Consequently, during *Join queries* (see Section 3.3.2.4), whenever $\mathcal{A}$ requests to add a public parameter pk to `KeyList`, he must additionally prove the knowledge of the associated secret key sk.

Let $\mathcal{A}$ be an adversary against the EUF-CMVA security of our sequential aggregate MAC scheme. Using $\mathcal{A}$ as a subroutine, we construct a reduction $\mathcal{B}$ against the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme[2]. $\mathcal{B}$ receives on input from its challenger, denoted by $\mathcal{C}$, the public parameters of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme $(pp, C_{x_0} = g^{x_0}h^x, X_1 = h^{x_1})$ and has access to two oracles: $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathcal{O}\mathtt{Verify}$. $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ provides it with a valid $\mathsf{MAC}_{\mathsf{GGM}}$ on any message of its choice whereas $\mathcal{O}\mathtt{Verify}$ allows it to check the validity of any (message, MAC) pair. So, $\mathcal{B}$ queries $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ on the message $m = 0$[3] and receives $\sigma_0 = (u, u' = u^{x_0})$, a MAC on 0. Then, it sets $g = u$, $Y_0 = u'$ and $sk^* = x_1$ (which is unknown to it) and provides $\mathcal{A}$ with $pp = (\mathbb{G}, q, g, h, Y_0)$ as well as X_1, the public parameter associated to sk^*. Next, $\mathcal{B}$ initializes the key list `KeyList` as empty, and through queries to the $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathcal{O}\mathtt{Verify}$ oracles, it perfectly simulates $\mathcal{A}$'s requests as follows:

- $\mathcal{O}\mathtt{Join}$ requests: whenever $\mathcal{A}$ asks $\mathcal{B}$ to add a public parameter pk_i to the key list `KeyList`, $\mathcal{B}$ rewinds $\mathcal{A}$ so as to extract the associated secret key sk_i. Next, it appends pk_i to `KeyList` and saves the pair (pk_i, sk_i) ($\mathcal{B}$ will subsequently require it to simulate aggregate MACs).

- $\mathcal{O}\mathtt{AggMAC}$ requests: to answer $\mathcal{A}$'s query for an aggregation of a message m_i to the aggregate MAC $\sigma_i = (u_i, u_i')$ on $M_i = (m_{i,1}, \ldots, m_{i,l})$ under the secret keys $SK_i = (sk_{i,1}, \ldots, sk_{i,l})$, $\mathcal{B}$ proceeds as follows. First, $\mathcal{B}$ verifies the validity of σ_i (it aborts if it is not valid). To do so, it computes $u_i'' = u_i' u_i^{-\sum_{j=1}^{j=l} sk_{i,j} m_{i,j}}$, then queries the $\mathcal{O}\mathtt{Verify}$ oracle on input (u_i, u_i'') and the message $m = 0$. If so (*i.e.* $\mathcal{O}\mathtt{Verify}$ returns `1`), $\mathcal{B}$ queries the $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ oracle on m_i as input. Thereby, it obtains $\sigma = (u, u')$. Since all the public parameters $PK_i = (pk_{i,1}, \ldots, pk_{i,l})$ belong to `KeyList`, then $\mathcal{B}$ knows all the associated secrets $SK_i = (sk_{i,1}, \ldots, sk_{i,l})$. Consequently, $\mathcal{B}$ selects a random $t \in \mathbb{Z}_q$ and computes $\sigma' = (u^t, (u'u^{\sum_{j=1}^{j=l} sk_{i,j} m_{i,j}})^t)$ that it returns to $\mathcal{A}$. The latter is a valid aggregate MAC on the block of messages $(m_{i,1}, \ldots, m_{i,l}, m_i)$.

- $\mathcal{O}\mathtt{Verify}$ requests: to verify whether $\sigma_i = (u_i, u_i')$ is a valid sequential aggregate MAC on $M_i = (m_{i,1}, \ldots, m_{i,l})$ under the set of keys $SK_i = (sk_{i,1}, \ldots, sk_{i,l})$ corresponding to the public parameters $PK_i = (pk_{i,1}, \ldots, pk_{i,l})$, $\mathcal{B}$ proceeds as follows. It first computes

[2] Hereinafter, we will consider the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme in the case of a single message.

[3] Even though this request has not been initiated by $\mathcal{A}$, $m = 0$ is considered as one of the messages that $\mathcal{A}$ has queried to $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$. Otherwise, $\mathcal{A}$ would trivially win.

$u''_i = u'_i u_i^{-\sum_{j=1, sk_{i,j} \neq sk^*}^{j=l} sk_{i,j}\ m_{i,j}}$. Then, it calls on the $\mathcal{O}\texttt{Verify}$ oracle on input (u_i, u''_i). The answer is forwarded to $\mathcal{A}$, hence perfectly simulating this oracle.

Eventually, $\mathcal{A}$ outputs with non negligible probability a sequential aggregate MAC $\sigma^* = (\sigma_1^*, \sigma_2^*)$ on the block of messages $M^* = (m_1^*, \ldots, m_l^*)$ under $SK^* = (sk_1, \ldots, sk_l)$ such that he breaks the EUF-CMVA security of our sequential aggregate MAC scheme. $\mathcal{A}$'s forgery is successful only if the following three conditions hold: (1) $\mathsf{Verify}(\sigma^*, M^*, SK^*) = 1$; (2) for all $sk_j \neq sk^*$, there exists an associated $pk_j \in \texttt{KeyList}$; and (3) for some $j^* \in \{1, \ldots, l\}$, $sk_j = sk^*$ and $m_{j^*}^*$ has not been queried to the $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ oracle.

The condition (1) means that σ^* is a valid sequential aggregate MAC, whereas (2) implies that $\mathcal{B}$ knows all the secret keys sk_i for $i \in \{1, \ldots, l\}$ except sk^*. Thereby, $\mathcal{B}$ can recover a valid MAC $\sigma' = (\sigma_1^*, \sigma_2^* {\sigma_1^*}^{-\sum_{sk_j \neq sk^*} sk_j\ m_j^*}) = (\sigma_1^*, (\sigma_1^*)^{x_0 + m_{j^*}^* x_1})$ on a message m_j^* that has never been queried to the $\mathcal{O}\mathsf{MAC}_{\mathsf{GGM}}$ oracle. Hence, $\mathcal{B}$ can break the UF-CMVA security of $\mathsf{MAC}_{\mathsf{GGM}}$.

Therefore, our sequential aggregate MAC scheme is EUF-CMVA secure, in the random oracle model, under the assumption that $\mathsf{MAC}_{\mathsf{GGM}}$ is UF-CMVA. □

4.4 Practical Pre-Direct Anonymous Attestation Schemes

As previously mentioned in Section 3.3.2.2, Direct Anonymous Attestation (DAA) is a privacy-preserving authentication protocol initially introduced so as to allow the anonymous authentication of Trusted Platform Modules (TPMs) (*e.g.* a secure element in a smartphone), to an external party (*e.g.* a service provider). Thereby, the TPM can be remotely authenticated whilst preserving the privacy of its owner.

This cryptographic protocol, and some of its extensions such as Intel's Enhanced Privacy ID (EPID) [BL07, BL11], have been widely deployed in millions of chips. Usually, part of the attestation computation is delegated to the host embedding it (*i.e.* PC or smartphone), which is generally much more powerful. However, in use cases related to M2M and IoT, the host may be as resource constrained as the TPM. Furthermore, any malware residing in the host may enable the tracking of the TPM owner.

In [BFG+13], Bernhard *et al.* proposed a DAA scheme, referred to as Pre-DAA, where all the computations on the user's side are performed by the TPM. However, similarly to [BL10], their proposal requires pairing computations on the user's side. Thus, it is inappropriate for SIM cards as they do not support such heavy computations.

In this subsection, we propose two Pre-DAA schemes where all computations on the platform side are carried out by the TPM. Both schemes are suitable for resource constrained devices like SIM cards as they require no pairing computations on the platform side. The first scheme is built upon $\mathsf{MAC}_{\mathsf{GGM}}$ [CMZ14] whereas the second is based on a pairing-free variant of the Boneh Boyen's signature scheme [CCJT14]. In what follows, we detail both proposals which involve three main entities: an issuer $\mathcal{I}$, a user $\mathcal{U}$ (more precisely, his platform $\mathcal{P}$) and some verifiers $\mathcal{V}$, and formally prove their security.

4.4.1 Schemes Description

4.4.1.1 $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA Scheme

Our first Pre-DAA scheme, which is based on $\mathsf{MAC}_{\mathsf{GGM}}$, works as follows:

$\mathsf{Setup}(1^k)$ creates the system public parameters denoted by $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, \tilde{h}, \mathcal{H}, e)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of order p, a k-bit prime, g, h are two random generators of $\mathbb{G}_1$ whereas $\tilde{h}$ is a random generator of $\mathbb{G}_2$, $\mathcal{H}$ is hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}_1$

(that will be considered as a random oracle in the security proof) and $e : \mathbb{G}_1 \text{ x } \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a Type-3 bilinear map.

$\mathsf{IKeyGen}(pp)$ randomly chooses $x_0, x_1 \in_R \mathbb{Z}_p^*$. Then, it sets the issuer's private key $gmsk$ as $gmsk = (x_0, x_1)$. The associated public key is $gmpk = (C_{x_0} = g^{x_0}h^{\tilde{x}_0}, X_1 = h^{x_1}, \tilde{X_0} = \tilde{h}^{x_0}, \tilde{X_1} = \tilde{h}^{x_1})$ where $\tilde{x_0} \in_R \mathbb{Z}_p^*$.

$\mathsf{UKeyGen}(pp, i)$ outputs the private/public endorsement key pair (esk_i, epk_i) of user i which will be subsequently used to authenticate him. It also selects a random values $s_1 \in_R \mathbb{Z}_p^*$ as the secret key of user i, denoted sk_i.

$\mathsf{Join}(i, esk_i, gmsk)$ allows a new user i to join the group and get his group signing key gsk_i. To this end, he computes $C_{s_1} = X_1^{s_1}$ a commitment to his secret s_1. The user also builds a ZKPK π_1 defined as $\pi_1 = \text{PoK}\{\alpha : C_{s_1} = X_1^{\alpha}\}$. Once done, he provides the issuer with C_{s_1} as well as the proof π_1. Upon their receipt, the issuer first verifies the validity of the proof π_1. Then, he picks $b \in_R \mathbb{Z}_p^*$ and generates a pair (u, u') associated to s_1 where $u = h^b$ and $u' = u^{x_0}C_{s_1}^b = u^{x_0+s_1x_1}$. Finally, the user is provided with (u, u') along with a ZKPK π_2 defined as $\pi_2 = \text{PoK}\{\alpha, \beta, \gamma : u = h^{\alpha} \wedge u' = u^{\beta}C_{s_1}^{\alpha} \wedge C_{x_0} = g^{\beta}h^{\gamma}\}$. If the proof π_2 is valid, the user sets his group signing key as $gsk_i = (u, u')$.

$\mathsf{GSign}(gsk_i, sk_i, m, \mathtt{bsn})$ enables the group member i, holding sk_i and gsk_i, to generate an anonymous signature σ on a message $m \in \{0,1\}^*$ and for the basename $\mathtt{bsn}$. The signature is computed as follows. First, the user randomly selects $l \in_R \mathbb{Z}_p^*$ and computes $gsk_i^l = (w, w')$, a randomized version of his group signing key, where $w = u^l$ and $w' = (u')^l$. Then, he computes $c_1 = w^{s_1}$ and $T = \mathcal{H}(\mathtt{bsn})^{s_1}$. A signature on m and for $\mathtt{bsn}$ consists of $\sigma = (w, w', c_1, T, \pi_3)$ where $\pi_3 = \text{PoK}\{\alpha : c_1 = w^{\alpha} \wedge T = \mathcal{H}(\mathtt{bsn})^{\alpha}\}$.

$\mathsf{GVerify}(gmpk, m, \mathtt{bsn}, \sigma)$ checks the validity of a signature σ with respect to a message m and a basename $\mathtt{bsn}$. The signature is valid only if $w \neq 1$, $e(w, \tilde{X_0}) \cdot e(c_1, \tilde{X_1}) = e(w', \tilde{h})$ and the verification of the proof π_3 succeeds. If so, the algorithm returns $\mathtt{1}$. Otherwise, it outputs $\mathtt{0}$.

$\mathsf{Identify}_{\mathsf{T}}(\mathcal{T}, sk_i)$ returns $\mathtt{1}$ if the transcript $\mathcal{T}$ resulting from the execution of the Join protocol has been produced with the secret key $sk_i = s_1$ (*i.e.* if C_{s_1} is a commitment to s_1). Otherwise, it returns $\mathtt{0}$.

$\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathtt{bsn}, sk_i)$ returns $\mathtt{1}$ if the signature σ on the message m could have been produced using the secret key $sk_i = s_1$ (*i.e.* if $T = \mathcal{H}(\mathtt{bsn})^{s_1}$). Otherwise, it returns $\mathtt{0}$.

$\mathsf{Link}(gmpk, \sigma, m, \sigma', m', \mathtt{bsn})$ returns $\mathtt{1}$ if both σ and σ' are valid signatures on, respectively, m and m' with respect to the same basename $\mathtt{bsn} \neq \perp$ and were produced by the same user (*i.e.* if $T = T'$). Otherwise, it returns $\mathtt{0}$.

Possible extension. Our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA scheme can be extended to support more than one secret (or a secret and some attributes). Hereinafter, we detail the extension in the case of two secrets.

$\mathsf{Setup}(1^k)$ creates the system public parameters denoted by $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, \tilde{h}, \mathcal{H}, e)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of order p, a k-bit prime, g, h are two random generators of $\mathbb{G}_1$ whereas $\tilde{h}$ is a random generator of $\mathbb{G}_2$, $\mathcal{H}$ is hash function $\mathcal{H} : \{0,1\}^* \rightarrow G_1$ (that will be considered as a random oracle in the security proof) and $e : \mathbb{G}_1 \text{ x } \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a Type-3 bilinear map.

$\mathsf{IKeyGen}(pp)$ randomly chooses $x_0, x_1, x_2 \in_R \mathbb{Z}_p^*$. Then, it sets the issuer's private key $gmsk$ as $gmsk = (x_0, x_1, x_2)$. The associated public key is $gmpk = (C_{x_0} = g^{x_0}h^{\tilde{x}_0}, X_1 = h^{x_1}, X_2 = h^{x_2}, \tilde{X_0} = \tilde{h}^{x_0}, \tilde{X_1} = \tilde{h}^{x_1}, \tilde{X_2} = \tilde{h}^{x_2})$ where $\tilde{x_0} \in_R \mathbb{Z}_p^*$.

$\mathsf{UKeyGen}(pp, i)$ outputs the private/public endorsement key pair (esk_i, epk_i) of user i which will be subsequently used to authenticate him. It also selects two random values $s_1, s_2 \in_R \mathbb{Z}_p^*$. The secret key of user i is defined as $sk_i = (s_1, s_2)$.

$\mathsf{Join}(i, esk_i, gmsk)$ allows a new user i to join the group and get his group signing key gsk_i. To this end, he selects $r \in_R \mathbb{Z}_p^*$ then computes $C_{s_1} = X_1^{s_1} h^r$, $C_{s_2} = X_2^{s_2}$ and $C = h^{s_1}$, three commitments to his secrets s_1 and s_2. The user also builds a ZKPK π_1 defined as $\pi_1 = \text{PoK}\{\alpha, \beta, \gamma : C_{s_1} = X_1^{\alpha} h^{\beta} \wedge C_{s_2} = X_2^{\gamma} \wedge C = h^{\alpha}\}$. Once done, he provides the issuer with C_{s_1}, C_{s_2} and C as well as the proof π_1 and a signature $S = Sign_{esk_i}(C_{s_1}, C_{s_2}, C)$ on them computed using esk_i. Upon their receipt, the issuer first verifies the validity of the signature and the proof π_1. Then, he picks $b \in_R \mathbb{Z}_p^*$ and generates a pair (u, u'') associated to both s_1 and s_2 where $u = h^b$ and $u'' = u^{x_0}(C_{s_1} C_{s_2})^b$. Finally, the user is provided with (u, u'') along with a ZKPK π_2 defined as $\pi_2 = \text{PoK}\{\alpha, \beta, \gamma : u = h^{\alpha} \wedge u'' = u^{\beta}(C_{s_1} C_{s_2})^{\alpha} \wedge C_{x_0} = g^{\beta} h^{\gamma}\}$. If the proof π_2 is valid, the user sets his group signing key as $gsk_i = (u, u' = \frac{u''}{u^r}) = (u, u^{x_0 + s_1 x_1 + s_2 x_2})$.

$\mathsf{GSign}(gsk_i, sk_i, m, \mathtt{bsn})$ enables the group member i, holding sk_i and gsk_i, to generate an anonymous signature σ on a message $m \in \{0,1\}^*$ and for the basename $\mathtt{bsn}$. The signature is computed as follows. First, the user randomly selects $r_1 \in_R \mathbb{Z}_p^*$ and computes $gsk_i^{r_1} = (w, w')$, a randomized version of his group signing key, where $w = u^{r_1}$ and $w' = (u')^{r_1}$. Then, he randomly selects $z_1, z_2, z_3 \in_R \mathbb{Z}_p^*$ and computes $c_1 = w^{s_1} h^{z_1}, c_2 = w^{s_2} h^{z_2}, c' = w' g^{z_3}, V = g^{-z_3} X_1^{z_1} X_2^{z_2}$ and $T = \mathcal{H}(\mathtt{bsn})^{s_1}$. A signature on m and for $\mathtt{bsn}$ consists of $\sigma = (w, c_1, c_2, c', V, T, \pi_3)$ where $\pi_3 = \text{PoK}\{\alpha, \beta, \gamma, \lambda, \phi : c_1 = w^{\alpha} h^{\gamma} \wedge c_2 = w^{\beta} h^{\lambda} \wedge V = g^{-\phi} X_1^{\gamma} X_2^{\lambda} \wedge T = \mathcal{H}(\mathtt{bsn})^{\alpha}\}$.

$\mathsf{GVerify}(gmpk, m, \mathtt{bsn}, \sigma)$ checks the validity of a signature σ with respect to a message m and a basename $\mathtt{bsn}$. The signature is valid only if $w \neq 1$, $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \cdot e(c_2, \tilde{X}_2) = e(c'V, \tilde{h})$ and the verification of the proof π_3 succeeds. If so, the algorithm returns $\mathtt{1}$. Otherwise, it outputs $\mathtt{0}$.

$\mathsf{Identify}_{\mathsf{T}}(\mathcal{T}, sk_i)$ returns $\mathtt{1}$ if the transcript $\mathcal{T}$ resulting from the execution of the Join protocol has been produced with the secret key $sk_i = (s_1, s_2)$ (*i.e.* if C_{s_1}, C_{s_2} and C are commitments to s_1 and s_2). Otherwise, it returns $\mathtt{0}$.

$\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathtt{bsn}, sk_i)$ returns $\mathtt{1}$ if the signature σ on the message m could have been produced using the secret key $sk_i = (s_1, s_2)$ (*i.e.* if $T = \mathcal{H}(\mathtt{bsn})^{s_1}$). Otherwise, it returns $\mathtt{0}$.

$\mathsf{Link}(gmpk, \sigma, m, \sigma', m', \mathtt{bsn})$ returns $\mathtt{1}$ if both σ and σ' are valid signatures on, respectively, m and m' with respect to the same basename $\mathtt{bsn} \neq \perp$ and were produced by the same user (*i.e.* if $T = T'$). Otherwise, it returns $\mathtt{0}$.

4.4.1.2 BB-based Pre-DAA Scheme

Our second Pre-DAA scheme, which is based on a pairing-free variant of the Boneh Boyen's signature scheme [CCJT14], is a direct variant of our KVAC system [BBDT16] subsequently detailed in Chapter 6. It is noteworthy to mention that shortly after we designed this pre-DAA scheme, Camenisch *et al.* proposed a similar scheme [CDL16a].

Our BB-based Pre-DAA scheme works as follows:

$\mathsf{Setup}(1^k)$ creates the system public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_0, g_1, g_2, h, f, \tilde{g}_0, \mathcal{H}, e)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of order p, a k-bit prime, (h, g_0, g_1, g_2, f) are five random generators of $\mathbb{G}_1$ whereas $\tilde{g}_0$ is a random generator of $\mathbb{G}_2$, $\mathcal{H}$ is hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}_1$ (that will be considered as a random oracle in the security proof) and $e : \mathbb{G}_1 \text{ x } \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a Type-3 bilinear map.

$\mathsf{IKeyGen}(pp)$ randomly chooses $y \in_R \mathbb{Z}_p^*$ as the issuer's private key and computes the associated public keys $Y = g_0^y$ and $W = \tilde{g}_0^y$.

$\mathsf{UKeyGen}(pp, i)$ outputs the private/public endorsement key pair (esk_i, epk_i) of user i which will be subsequently used to authenticate him. It also picks two random values $s_1, s_2 \in_R \mathbb{Z}_p^*$. The secret key of user i is defined as $sk_i = (s_1, s_2)$.

$\mathsf{Join}(i, esk_i, gmsk)$ allows a new user i to join the group and obtain his group signing key gsk_i. To this end, he computes $C_{s_1} = g_1^{s_1}$ and $C_{s_2} = g_2^{s_2}$, two commitments to his secrets s_1 and s_2 respectively. The user also builds a ZKPK π_1 defined as $\pi_1 = \text{PoK}\{\alpha, \beta : C_{s_1} = g_1^{\alpha} \wedge C_{s_2} = g_2^{\beta}\}$. Once done, he provides the issuer with C_{s_1} and C_{s_2} as well as the proof π_1 and a signature $S = Sign_{esk_i}(C_{s_1}, C_{s_2})$ on them produced using the private key esk_i. Upon their receipt, the issuer first verifies the validity of the signature and the proof π_1. Then, he picks $r, s' \in_R \mathbb{Z}_p^*$ and computes $A = (C_{s_1} C_{s_2} g_2^{s'} h)^{\frac{1}{y+r}}$. He also builds a ZKPK π_2 to ensure that A is well-formed. π_2 is defined as $\pi_2 = \text{PoK}\{\gamma : B = A^{\gamma} \wedge Y = g_0^{\gamma}\}$ where $B = C_{s_1} C_{s_2} g_2^{s'} h \cdot A^{-r} = A^y$. Finally, the user is provided with (A, r, s') along with the proof π_2. If π_2 is valid, the user sets his group signing key as $gsk_i = (A, r, s_u)$ where $s_u = s_2 + s'$ is only known to $\mathcal{U}$.

$\mathsf{GSign}(gsk_i, sk_i, m, \mathtt{bsn})$ enables the group member i, holding sk_i and gsk_i, to anonymously sign a message $m \in \{0,1\}^*$ and with respect to the basename $\mathtt{bsn}$. The signature σ is computed as follows. First, the user randomly selects $l, t \in_R \mathbb{Z}_p^*$ and computes $B_0 = A^l$, a randomized version of his group signing key. Then, he computes $C = g_1^{ls_1} g_2^{ls_u} h^l B_0^{-r} = A^{l^y} = B_0^y$, $E = C^{\frac{1}{l}} f^t$ as well as $T = \mathcal{H}(\mathtt{bsn})^{s_1}$. An anonymous signature on m and for $\mathtt{bsn}$ consists of $\sigma = (B_0, C, E, T, \pi_3)$ where $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \delta_2, \theta, \gamma : E = C^{\alpha} f^{\beta} \wedge E \cdot h^{-1} = g_1^{\delta_1} g_2^{\delta_2} \cdot B_0^{\lambda} \cdot f^{\beta} \wedge C = E^{\theta} f^{\gamma} \wedge T = \mathcal{H}(\mathtt{bsn})^{\delta_1}\}$.

$\mathsf{GVerify}(gmpk, m, \mathtt{bsn}, \sigma)$ checks the validity of a signature σ with respect to a message m and a basename $\mathtt{bsn}$. The signature is valid only if $e(C, \tilde{g}_0) = e(B_0, W)$ and the verification of the proof π_3 succeeds. If so, the algorithm returns $\mathtt{1}$. Otherwise, it outputs $\mathtt{0}$.

$\mathsf{Identify}_{\mathsf{T}}(\mathcal{T}, sk_i)$ returns $\mathtt{1}$ if the transcript $\mathcal{T}$ resulting from the execution of the Join protocol has been produced with the secret key $sk_i = (s_1, s_2)$ (*i.e.* if C_{s_1} and C_{s_2} are commitments to respectively s_1 and s_2). Otherwise, it returns $\mathtt{0}$.

$\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathtt{bsn}, sk_i)$ returns $\mathtt{1}$ if the signature σ on the message m could have been produced using the secret key $sk_i = (s_1, s_2)$ (*i.e.* if $T = \mathcal{H}(\mathtt{bsn})^{s_1}$). Otherwise, it returns $\mathtt{0}$.

$\mathsf{Link}(gmpk, \sigma, m, \sigma', m', \mathtt{bsn})$ returns $\mathtt{1}$ if both σ and σ' are valid signatures on, respectively, m and m' with respect to the same basename $\mathtt{bsn} \neq \perp$ and were produced by the same user (*i.e.* if $T = T'$). Otherwise, it returns $\mathtt{0}$.

4.4.2 Efficiency Comparison

In Table 4.1, we compare the efficiency of our two pre-DAA schemes with that of the most efficient DAA schemes. More precisely, we provide the total estimated cost of generating a signature as well as the computational cost of verifying it, since they are the most time critical phases.

We use the following notation: $k\mathbb{G}_i$ and $k\mathbb{G}_i^j$ respectively denote k exponentiations in the group $\mathbb{G}_i$ and k j-multi-exponentiations in the group $\mathbb{G}_i$ whereas kP refers to k pairing computations. When attributes are supported, n, r and u are used to denote the total number of attributes, the number of revealed attributes and the number of unrevealed ones respectively.

As shown in Table 4.1, LRSW-based DAA schemes are more efficient than $q-\mathsf{SDH}$-based schemes. More specifically, our $\mathsf{MAC}_{\mathsf{GGM}}$-based scheme have the most efficient signing operation. Furthermore, aside from [CDL16a] which was introduced just after our BB-based pre-DAA

scheme, all others $q-$SDH-based DAA schemes require the platform to perform either pairing computations or computations in $\mathbb{G}_2$ (or $\mathbb{G}_T$). Thus, unlike our BB-based pre-DAA scheme, they are not suitable for SIM cards (since the latter cannot handle such computations). It is also worth mentioning that, as stated in [CDL16b], the ZKPKs associated to the schemes [BL10, Che10, CU15] are flawed (they do not really prove the possession of a valid membership credential).

LRSW based DAA schemes		
Schemes	Sign	Verify
CPS10 [CPS10]	$7\mathbb{G}_1$	$2\mathbb{G}_1^2$, 4P
CU15-1 [CU15]	7+n+u $\mathbb{G}_1$	$2\mathbb{G}_1$, $2\mathbb{G}_1^n$, $2\mathbb{G}_1^r$, $2\mathbb{G}_1^u$, 6P
CDL16-1 [CDL16c]	$9\mathbb{G}_1$	$2\mathbb{G}_1^2$, 4P
Our $\mathsf{MAC}_{\mathsf{GGM}}$-based scheme	$\mathbf{6}\mathbb{G}_1$	$\mathbf{2}\mathbb{G}_1^2$**, 3P**
q-SDH based DAA schemes		
Schemes	Sign	Verify
CF08 [CF08]	$3\mathbb{G}_1$, $2\mathbb{G}_T$, $2\mathbb{G}_1^2$, 1P	$1\mathbb{G}_1^2$, $2\mathbb{G}_1^3$, $1\mathbb{G}_T^5$, 3P
Ch10 [Che10]	$3\mathbb{G}_1$, $1\mathbb{G}_T$, $1\mathbb{G}_T^3$	$1\mathbb{G}_1^2$, $1\mathbb{G}_2^2$, $1\mathbb{G}_T^4$, 1P
BL10 [BL10]	$3\mathbb{G}_1$, $1\mathbb{G}_1^2$, $1\mathbb{G}_T$, 1P	$1\mathbb{G}_1^2$, $1\mathbb{G}_2^2$, $1\mathbb{G}_T^4$, 1P
CU15-2 [CU15]	$5\mathbb{G}_1$, $1\mathbb{G}_1^{2+u}$, 2P	$1\mathbb{G}_1^2$, $1\mathbb{G}_1^{4+n}$, 2P
CDL16-2 [CDL16b]	$4\mathbb{G}_1$, $1\mathbb{G}_1^{2+u}$, $1\mathbb{G}_T$, 1P	$1\mathbb{G}_1^2$, $1\mathbb{G}_2^2$, $1\mathbb{G}_T^{4+n}$, 1P
CDL16[4] [CDL16a]	$5\mathbb{G}_1$, $2\mathbb{G}_1^2$, $1\mathbb{G}_1^{2+u}$	$1\mathbb{G}_1^2$, $1\mathbb{G}_1^3$, $1\ G_1^{5+n}$, 2P
Our BB-based scheme	$\mathbf{3}\mathbb{G}_1$**,** $\mathbf{4}\mathbb{G}_1^2$**,** $\mathbf{1}\mathbb{G}_1^{3+u}$	$\mathbf{1}\mathbb{G}_1^2$**,** $\mathbf{2}\mathbb{G}_1^3$**,** $\mathbf{1}\mathbb{G}_1^{4+u}$**,** $\mathbf{1}\mathbb{G}_1^{r+1}$**, 2P**

Table 4.1: DAA schemes efficiency comparison

4.4.3 Security Analysis

In this section, we formally prove that our two pre-DAA schemes provide the expected security properties (*i.e. correctness*, *traceability*, *non-frameability* and *anonymity*) in the random oracle model. As usual, *correctness* follows by inspection.

4.4.3.1 Security of our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA Scheme

Theorem 4.6. *Our $\mathsf{MAC}_{\mathsf{GGM}}$-based pre-DAA scheme is* traceable*, in the ROM, under the assumption* 1 *introduced in [PS16] (see Section 3.23).*

Proof (sketch). Let $\mathcal{A}$ be an adversary who breaks the traceability requirement of our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA scheme with non-negligible probability. As mentioned in section 3.3.2.2, we distinguish the following two types of forgers:

- **Type-1 Forger:** An adversary that manages to output a signature σ that cannot be traced to a secret key previously queried to the $\mathcal{O}\mathtt{Join}_I$ oracle.
- **Type-2 Forger:** An adversary that outputs two signatures σ_0 and σ_1 under the same secret sk_i and for the same basename `bsn`, and yet are unlinkable.

In what follows, we show that a Type-1 forger can be used as a subroutine to construct an algorithm $\mathcal{B}$ against the assumption 1, whereas Type-2 forgery cannot happen. Initially, $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its challenger, denoted by $\mathcal{C}$, the public parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, h, \tilde{h})$ as well as $X_1 = h^{x_1}$, $X_2 = h^{x_2}$, $\tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}$ and $\tilde{X}_2 = \tilde{h}^{x_2}$. As it is against the assumption 1, $\mathcal{B}$ has access to the oracle $\mathcal{O}_1$ which, on input $m_1, m_2 \in \mathbb{Z}_p$, outputs the pair $P = (t, t^{x_0+m_1x_1+m_2x_2})$ where $t \in_R \mathbb{G}_1$. $\mathcal{B}$ chooses the generator g and the value of C_{x_0} at random. Thereby, it can provide $\mathcal{A}$ with the public parameters pp of the scheme and answers his (*i.e.* $\mathcal{A}$'s) requests as follows:

- $\mathcal{O}\texttt{Add}(i)$ requests: $\mathcal{B}$ creates a new user i and picks two random values $s_1 \in_R \mathbb{Z}_p^*$ as his secret key sk_i. $\mathcal{B}$ also generates user's i public/private endorsement key pair denoted by (esk_i, epk_i).
- $\mathcal{O}\texttt{AddCorrupt}(i)$ requests: $\mathcal{B}$ does nothing.
- $\mathcal{O}\texttt{Join}_I(i)$ requests: Upon the receipt of C_{s_1}, C_{s_2}, C, the proof π_1 and the signature $Sign$, $\mathcal{B}$ first checks the validity of both $Sign$ and π_1. If so, it uses the soundness property of π_1 to recover s_1, s_2 and r. Then, it queries the $\mathcal{O}_1$ oracle on s_1 and s_2 to obtain the pair $(u, u' = u^{x_0+s_1x_1+s_2x_2})$. As it knows r, $\mathcal{B}$ can compute $u'' = u'u^r$. Besides, in the ROM, π_2 can be perfectly simulated using standard techniques. Thereby, $\mathcal{B}$ can answer $\mathcal{A}$'s request by providing him with the pair (u, u'') as well as the proof π_2.
- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{B}$ provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.
- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: As it holds both sk_i and gsk_i, $\mathcal{B}$ can generate a signature on m and for bsn, that it returns to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs with non-negligible probability a valid triple $(\sigma, m, \texttt{bsn})$ such that the signature σ was produced using an sk_i that is not associated to any of the calls to the $\mathcal{O}\texttt{Join}_I$ oracle. Using the soundness of π_3, $\mathcal{B}$ extracts s_1 and s_2 as well as $w' = c'g^{-z_3}$. Thus, $\mathcal{B}$ obtains a valid pair $(w, w' = w^{x_0+s_1x_1+s_2x_2})$ on the pair (s_1, s_2) that has never been queried to the $\mathcal{O}_1$ oracle. Consequently, using $\mathcal{A}$, $\mathcal{B}$ can break the assumption 1.

If $c_{mode} = 2$: $\mathcal{A}$ eventually outputs $(\sigma_0, m_0, \sigma_1, m_1, \texttt{bsn}, sk)$ such that σ_0 and σ_1 are valid signatures on respectively m_0 and m_1, and for the same basename bsn. Besides, the two signatures were produced using sk (as, by definition, both $\mathsf{Identify}_{\mathsf{S}}(\sigma_0, m_0, \texttt{bsn}, sk)$ and $\mathsf{Identify}_{\mathsf{S}}(\sigma_1, m_1, \texttt{bsn}, sk)$ output 1). Owing to the completeness of the proof π_3, the tags T_0 and T_1 associated to respectively σ_0 and σ_1 are necessarily defined as $T_0 = \mathcal{H}(\texttt{bsn})^{s_1}$ and $T_1 = \mathcal{H}(\texttt{bsn})^{s_1}$. Therefore, we have $T_0 = T_1$. By definition, $\mathcal{A}$ wins if the output of $\mathsf{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \texttt{bsn})$ is 0 (*i.e.* $T_0 \neq T_1$). Thus, such forgery can never occur.

Therefore, if $\mathcal{A}$ can break the traceability property of our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA scheme, then $\mathcal{B}$ can break assumption 1 with the same probability. Thus, under the assumption 1, our Pre-DAA scheme provides the traceability requirement in the ROM.

□

Theorem 4.7. *Our* $\mathsf{MAC}_{\mathsf{GGM}}$*-based pre-DAA scheme is* non-frameable*, in the ROM, under the one-more discrete logarithm (*OMDL*) assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary against the non-frameability requirement of our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA scheme. As previously mentioned in section 3.3.2.2, we distinguish the following two types of forgeries:

- **Type-1 Forger:** An adversary that manages to output a signature σ on m and for bsn that is traced to a specific user that has never produced such a signature.
- **Type-2 Forger:** An adversary that outputs two signatures σ_0 and σ_1 that are linkable even though they should not (*i.e.* they were either produced with different sk_i or with respect to two distinct basenames).

In what follows, we show that a Type-1 forger can be used to construct a reduction $\mathcal{B}$ against the OMDL assumption whilst a Type-2 forgery cannot happen. Initially, $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its challenger $\mathcal{C}$ a random instance $(h^{u_1}, h^{u_2}, \ldots, h^{u_n})$ of the OMDL problem where h is a random generator of $\mathbb{G}_1$. As it is against the one-more DL assumption, $\mathcal{B}$ has access to a DL oracle. The adversary $\mathcal{A}$ picks three random values $x_0, x_1, x_2 \in_R \mathbb{Z}_p^*$ as the issuer's private key and publishes the associated public key $gmpk = (C_{x_0} = g^{x_0}h^{\tilde{x}_0}, X_1 = h^{x_1}, X_2 = h^{x_2}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}, \tilde{X}_2 = \tilde{h}^{x_2})$ where $\tilde{x_0} \in_R \mathbb{Z}_p^*$. $\mathcal{B}$ answers $\mathcal{A}$'s requests as follows:

- $\mathcal{O}\texttt{Add}(i)$ requests: $\mathcal{B}$ creates a new user u_i and randomly selects $s_2^i \in_R \mathbb{Z}_p^*$. Using its input of the OMDL problem, it sets s_1^i such as $C_i = h^{s_1^i} = h^{u_i}$. The user's secret key is defined as $sk_i = (s_1^i, s_2^i)$ where s_1^i is unknown to both $\mathcal{B}$ and $\mathcal{A}$. $\mathcal{B}$ also generates u_i's public/private endorsement key pair denoted by (esk_i, epk_i).

- $\mathcal{O}\texttt{AddCorrupt}$ requests: $\mathcal{B}$ does nothing.

- $\mathcal{O}\texttt{Join}_U(i)$ requests: $\mathcal{B}$ computes $C_{s_2^i} = X_2^{s_2^i}$ and sets $C_i = h^{u_i}$. The value of $C_{s_1^i}$ can be randomly chosen. Indeed, $\forall s_1^i \in \mathbb{Z}_p$, $\exists r \in \mathbb{Z}_p$ such that $C_{s_1^i} = h^r X_1^{s_1^i}$. Since $\mathcal{B}$ holds esk_i, then it can compute the signature S. As for π_1, $\mathcal{B}$ can perfectly simulate it in the random oracle model. If the protocol does not abort, $\mathcal{B}$ obtains a valid group signing key $gsk_i = (u, u')$ associated to $sk_i = (s_1^i, s_2^i)$, where s_1^i is unknown to both $\mathcal{A}$ and $\mathcal{B}$, as well as the proof π_2. Using the soundness property of π_2, $\mathcal{B}$ retrieves the value of b (required to simulate the $\mathcal{O}\texttt{GSign}$ oracle).

- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{B}$ calls on the DL oracle with h^{u_i} as input. Thereby, it recovers $u_i = s_1^i$. Thus, it can provide $\mathcal{A}$ with the secret key $sk_i = (s_1^i, s_2^i)$ along with the group signing key gsk_i of user i.

- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: As it holds s_2^i and gsk_i, $\mathcal{B}$ can compute most of the required values except c_1, T and the proof π_3. Using b, which was extracted during the run of Join_U from π_2, $\mathcal{B}$ sets $c_1 = C_i^{br_1}h^{z_1} = w^{s_1^i}h^{z_1}$ where $z_1 \in_R \mathbb{Z}_p^*$. To compute T, $\mathcal{B}$ proceeds as follows: it randomly selects $l \in_R \mathbb{Z}_p^*$ and sets $H = \mathcal{H}(\texttt{bsn}) = h^l$. Thus, $T = \mathcal{H}(\texttt{bsn})^{u_i} = C_i^l$. During subsequent calls to $\mathcal{O}\texttt{GSign}$ on input the same i and bsn, $\mathcal{B}$ returns the same value T. As for π_3, it can be easily simulated in the ROM. Hence, $\mathcal{B}$ can perfectly simulate the GSign algorithm.

Eventually, after d calls to the $\mathcal{O}\texttt{Corrupt}$ oracle, $\mathcal{A}$ outputs with non negligible probability a valid signature σ on m and for bsn such that $\mathsf{Identify}_\mathsf{S}(\sigma, m, \texttt{bsn}, sk_i)$ outputs 1 whereas the user holding sk_i has never produced a signature on that m and for bsn. By definition of the experiment, we know that the corresponding user is *honest*. Thus, the value u_i associated to the user's *unknown* secret s_1^i is still in the input of the OMDL problem. Using the soundness property of π_3, $\mathcal{B}$ retrieves the associated secrets s_1^i and s_2^i where $s_1^i = u_i$. Thus, $\mathcal{B}$ recovers u_i, the discrete logarithm of the challenge h^{u_i}. By outputting u_i along with the d secrets $\{u_j\}_{j=1}^{j=d}$ that it has obtained by querying the DL oracle, $\mathcal{B}$ breaks the OMDL assumption.

If $c_{mode} = 2$: $\mathcal{A}$ eventually outputs two valid signatures σ_0 and σ_1, on respectively m_0 and m_1, that are linkable even though they should not. That is, they were either generated using two different keys (sk_0 and sk_1) or/and are for different basenames ($\texttt{bsn}_0$ and $\texttt{bsn}_1$).

Let $sk_b = (s_1^b, s_2^b)$ denote the key used to generate σ_b and $sk = (\tilde{s}_1, \tilde{s}_2)$ the key output by $\mathcal{A}$. If $\mathcal{A}$'s attack against the non-frameability property is successful, then this implies that condition 6 of the non-frameability experiment is true. In particular, owing to the completeness of π_3, we have

$$T_0 = \mathcal{H}(\texttt{bsn}_0)^{s_1^0} \quad \text{and} \quad T_1 = \mathcal{H}(\texttt{bsn}_1)^{s_1^1} \tag{4.1}$$

The condition 7 should also be true. Thus, $\exists b \in \{0,1\}$ such that $\texttt{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \texttt{bsn}_b) = 1$. Without loss of generality, let us assume that $b = 0$. From the validity of this assertion, we can deduce that

$$T_0 = \mathcal{H}(\texttt{bsn}_0)^{s_1^0} = T_1 = \mathcal{H}(\texttt{bsn}_0)^{s_1^1} \tag{4.2}$$

This implies that

$$s_1^0 = s_1^1 \ (mod\ p) \tag{4.3}$$

For $\mathcal{A}$'s forgery to be successful, either condition 8 or 9 should be true. Suppose that assertion 8 is true. This implies that

$$T_0 = \mathcal{H}(\texttt{bsn}_0)^{\tilde{s}_1} \tag{4.4}$$

and

$$T_1 \neq \mathcal{H}(\texttt{bsn}_1)^{\tilde{s}_1} \tag{4.5}$$

This is impossible. Indeed, (4.2) and (4.4) imply that $s_1^0 = \tilde{s}_1$. Thus, from (4.3), we have $s_1^0 = \tilde{s}_1 = s_1^1$. From (4.1), we also know that $T_1 = \mathcal{H}(\texttt{bsn}_1)^{s_1^1} = \mathcal{H}(\texttt{bsn})^{\tilde{s_1}}$, which contradicts (4.5).

Let us now assume that condition 9 is true, *i.e.* that $\texttt{bsn}_0 \neq \texttt{bsn}_1$. (4.1) and (4.2) imply that $T_1 = \mathcal{H}(\texttt{bsn}_1)^{s_1^1} = \mathcal{H}(\texttt{bsn}_0)^{s_1^1}$, hence $\mathcal{H}(\texttt{bsn}_0) = \mathcal{H}(\texttt{bsn}_1)$ where $\texttt{bsn}_0 \neq \texttt{bsn}_1$. This would imply that $\mathcal{A}$ broke the *second pre-image resistance* property of the hash function $\mathcal{H}$ which is unfeasible in the ROM.

Therefore, a Type-2 forgery can never occur.

Consequently, under the OMDL assumption, our $\mathsf{MAC}_{\mathsf{GGM}}$-based pre-DAA scheme ensures the non-frameability requirement.

□

Theorem 4.8. *Our* $\mathsf{MAC}_{\mathsf{GGM}}$*-based pre-DAA scheme satisfies the* anonymity *requirement, in the ROM, under the* XDH *assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary against the anonymity requirement of our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA scheme. Using Shoup's game hopping technique, where proofs are organized as sequences of games, we prove that our Pre-DAA scheme ensures the anonymity requirement under the XDH assumption. Hereinafter, we provide a high level description of the initial game (Game 0) along with a brief description of the transition between Game 0 and Game 1.

Game 0: It corresponds to the real attack game with respect to an efficient adversary $\mathcal{A}$.

The Challenger $\mathcal{C}$ randomly chooses the system public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathcal{H}, e)$ as well as the issuer's private key $gmsk = (x_0, x_1, x_2)$ and computes the associated public key $gmpk = (C_{x_0} = g^{x_0} h^{\tilde{x}_0}, X_1 = h^{x_1}, X_2 = h^{x_2}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}, \tilde{X}_2 = \tilde{h}^{x_2})$ where $h \in_R \mathbb{G}_1$ and $\tilde{x}_0 \in_R \mathbb{Z}_q$. Then, it provides $\mathcal{A}$ with $gmsk = (x_0, x_1, x_2)$ and answers his requests as follows:

- $\mathcal{O}\texttt{Add}(i)$ requests: $\mathcal{C}$ creates a new user i and picks a random value $s_1^i \in_R \mathbb{Z}_p^*$ as his secret key sk_i. $\mathcal{C}$ also generates user's i public/private endorsement key pair denoted by (esk_i, epk_i).

- $\mathcal{O}\texttt{AddCorrupt}(i)$ requests: $\mathcal{C}$ does nothing.

- $\mathcal{O}\texttt{Join}_U(i)$ requests: $\mathcal{C}$ computes $C_{s_1^i}, C_{s_2^i}$ and C_i, then builds π_1 to obtain the group signing key gsk_i.

- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{C}$ provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.

- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: $\mathcal{C}$ uses sk_i and gsk_i to generate a signature σ on m and for bsn, that it returns to $\mathcal{A}$.

Eventually, $\mathcal{A}$ chooses $(i_0, i_1, \texttt{bsn}, m)$ that he provides as input to $\mathcal{O}\texttt{CH}_b$. The oracle outputs a signature σ produced by user b. $\mathcal{A}$'s goal is to guess the value of b. Even after receiving his challenge, $\mathcal{A}$ can still query the $\mathcal{O}\texttt{Add}$, $\mathcal{O}\texttt{AddCorrupt}$, $\mathcal{O}\texttt{Join}_U$, $\mathcal{O}\texttt{Corrupt}$ and $\mathcal{O}\texttt{GSign}$ oracles but with some restrictions. More precisely, he cannot call the $\mathcal{O}\texttt{Corrupt}$ oracle on i_0 or i_1. Besides, he is not allowed to query the $\mathcal{O}\texttt{GSign}$ oracle with either $(i_0, m, \texttt{bsn})$ or $(i_1, m, \texttt{bsn})$ as input. Otherwise, he would trivially win the game.

So, $\mathcal{A}$ eventually outputs his guess b'. Let S_0 be the event that $b = b'$ in Game 0 (*i.e.* the event that $\mathcal{A}$ wins Game 0) and S_1 denote the event that $b = b'$ in Game 1. We have $\texttt{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2| = |\texttt{Pr}[S_0] - 1/2|$.

We construct the Game 1 as follows:

Game 1: It is the same game as Game 0 except that we replace $T = \mathcal{H}(\texttt{bsn})^{s_1^b}$ by a random value and simulate the proof π_3. Such a proof can be easily simulated in the ROM using classical techniques. Under the XDH assumption, $\mathcal{A}$ cannot detect this change. In fact, one can easily construct a XDH distinguisher $\mathcal{D}$ with an advantage of solving the XDH problem, denoted by $\texttt{Adv}_{\mathsf{XDH}}$, satisfying $|\texttt{Pr}[S_0] - \texttt{Pr}[S_1]| \leq \texttt{Adv}_{\mathsf{XDH}}(1^k)$.

In Game 1, the challenger provides no information (in a strong information theoretic sense) about the bit b to the adversary $\mathcal{A}$. This is due to the *perfectly hiding* property of the Pedersen's commitment (see Section 3.3.4.2). Let us consider a valid signature (w, c_1, c_2, c', V). For any group signing key $gsk_i = (\tilde{u}, \tilde{u}')$, there exists $\tilde{l}, \tilde{z}_1, \tilde{z}_2, \tilde{z}_3$ such that $w = \tilde{u}^{\tilde{l}}$, $w' = (\tilde{u}')^{\tilde{l}}$, $c_1 = w^{\tilde{s}_1} h^{\tilde{z}_1}$, $c_2 = w^{\tilde{s}_2} h^{\tilde{z}_2}$, $c' = w' g^{\tilde{z}_3}$ and $V = g^{-\tilde{z}_3} X_1^{\tilde{z}_1} X_2^{\tilde{z}_2}$. Besides, the only value involving the user's secret key and computed in a non-perfectly hiding way (*i.e.* $T = \mathcal{H}(\texttt{bsn})^{s_1^b}$) was replaced by a random value. Therefore, $\texttt{Pr}[S_1] = 1/2$. Thus, we have

$$\texttt{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2| = |\texttt{Pr}[S_0] - \texttt{Pr}[S_1]| \leq \texttt{Adv}_{\mathsf{XDH}}(1^k)$$

Consequently, under the XDH assumption, the adversary's advantage is negligible. Therefore, we conclude that our $\mathsf{MAC}_{\mathsf{GGM}}$-based Pre-DAA scheme satisfies the anonymity requirement under the XDH assumption. □

4.4.3.2 Security of our BB-based Pre-DAA Scheme

Theorem 4.9. *Our BB-based pre-DAA scheme is* traceable*, in the ROM, under the* $q - \mathsf{SDH}$ *assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary who breaks the traceability requirement of our BB-based Pre-DAA scheme with non-negligible probability. As previously, we distinguish the following two types of forgers:

- **Type-1 Forger:** An adversary that manages to output a signature σ that cannot be traced to a secret key previously queried to the $\mathcal{O}\texttt{Join}_I$ oracle.
- **Type-2 Forger:** An adversary that outputs two signatures σ_0 and σ_1 under the same secret sk_i and for the same basename bsn, and yet are unlinkable.

In what follows, we show that a Type-1 forger can be used as a subroutine to construct an algorithm $\mathcal{B}$ against the $q - \mathsf{SDH}$ assumption, whereas Type-2 forgery cannot happen. Initially, $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$: Let $\mathcal{A}$ be a Type-1 Forger. Using $\mathcal{A}$, we construct a reduction $\mathcal{B}$ against the $q-\mathsf{SDH}$ assumption. $\mathcal{A}$ can ask for a group signing key on any secret of his choice and receives in response one of the tuples (A_i, r_i, s'_i) for $i \in \{1, \ldots, q\}$, where q denotes the number of requests to the $\mathcal{O}\texttt{Join}_I$ oracle. Eventually, $\mathcal{A}$ outputs his forgery $(\sigma, m, \texttt{bsn})$ that cannot be linked to any of the secret keys previously queried to the $\mathcal{O}\texttt{Join}_I$ oracle. We distinguish the following two types of forgeries:

- **Type-1.1 Forger:** an adversary that outputs a valid signature $(\sigma, m, \texttt{bsn})$ on m such that the associated group signing key $gsk = (A, r, s_u)$ satisfies $(A, r) \neq (A_i, r_i)$ for all $i \in \{1, \ldots, q\}$.
- **Type-1.2 Forger:** an adversary that outputs a valid signature $(\sigma, m, \texttt{bsn})$ on m such that the associated group signing key $gsk = (A, r, s_u)$ satisfies $(A, r) = (A_j, r_j)$ for some $j \in \{1, \ldots, q\}$ whereas $sk \neq sk_j$.

Hereinafter, we show that both types of adversaries can be used to break the $q-\mathsf{SDH}$ assumption. However, the reduction works differently for each type of forger. Consequently, $\mathcal{B}$ initially chooses a second random bit $c'_{mode} \in \{1, 2\}$ which indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c'_{mode} = 1$: $\mathcal{B}$ receives on input from its $q-\mathsf{SDH}$ challenger, denoted by $\mathcal{C}$, the public parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, p, g_0, \tilde{g}_0, h)$, the public keys $Y = g_0^y$ and $W = \tilde{g}_0^y$ as well as q random, and distinct, triplets (A_i, r_i, m_i) such that $A_i = (g_0^{m_i} h)^{\frac{1}{r_i + y}}$ for $i \in \{1, \ldots, q\}$. $\mathcal{B}$ randomly chooses $u, v \in_R \mathbb{Z}_p$, then computes $g_1 = g_0^v$ and $g_2 = g_0^u$ whereas f is chosen at random. Thereby, $\mathcal{B}$ can provide $\mathcal{A}$ with the scheme public parameters pp and answer $\mathcal{A}$'s requests as follows:

- $\mathcal{O}\texttt{Add}(i)$ requests: $\mathcal{B}$ creates a new user i and picks two random values $s_1, s_2 \in_R \mathbb{Z}_p^*$ as his secret key sk_i. $\mathcal{B}$ also generates the public/private endorsement key pair of user i denoted by (esk_i, epk_i).
- $\mathcal{O}\texttt{AddCorrupt}(i)$ requests: $\mathcal{B}$ does nothing.
- $\mathcal{O}\texttt{Join}_I()$ requests: Upon the receipt of C_{s_1}, C_{s_2}, the proof π_1 and the signature $Sign$, $\mathcal{B}$ first checks the validity of both $Sign$ and π_1. If so, it uses the soundness property of π_1 to recover s_1 and s_2. Then, $\mathcal{B}$ picks one of the received triplets (A_i, r_i, m_i) that has not already been used and computes s'_i such that $vs_1 + u(s_2 + s'_i) = m_i$. Finally, it provides $\mathcal{A}$ with the triple (A_i, r_i, s'_i) and simulates the proof π_2. Thus, the user's gets the group signing key $gsk_i = (A_i, r_i, s_u)$ where $s_u = s_2 + s'_i$.
- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{B}$ provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.
- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: As it holds both sk_i and gsk_i, $\mathcal{B}$ can generate a signature on m and for $\texttt{bsn}$, that it returns to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs with non-negligible probability a valid triple $(\sigma, m, \texttt{bsn})$ such that the signature σ was produced using an sk_i that has never been involved in any of the calls to the $\mathcal{O}\texttt{Join}_I$ oracle. Using the soundness of π_3, $\mathcal{B}$ extracts s_1, s_u, r and l. Thereby, $\mathcal{B}$ retrieves the corresponding group signing key (A, r, s_u). By assumption, $(A, r) \neq (A_i, r_i)$ for all $i \in \{1, \ldots, q\}$. Therefore, $\mathcal{B}$ outputs $(A, r, \tilde{m})$ where $\tilde{m} = vs_1 + us_u$ as his forgery, hence breaking the $q-\mathsf{SDH}$ assumption with the same advantage as $\mathcal{A}$.

If $c'_{mode} = 2$: A Type-1.2 adversary $\mathcal{A}$ is rather used, as a subroutine, to construct a reduction $\mathcal{B}$ against the DL problem. In such case, $\mathcal{B}$ receives on input from its DL challenger, denoted by $\mathcal{C}$, the challenge $(g_2, H = g_2^v)$. Its goal is to find the value v. For this purpose, it first randomly chooses $(y, h, g_0, f, \tilde{g}_0) \in_R \mathbb{Z}_p \times \mathbb{G}_1^3 \times \mathbb{G}_2$ and computes $Y = g_0^y$ and $W = \tilde{g}_0^y$. $\mathcal{B}$ also sets g_1 as $g_1 = H$. Thereby, it can provide $\mathcal{A}$ with the public parameters $(h, g_0, g_1, g_2, f, \tilde{g}_0, Y, W)$ and answer his requests as follows:

- $\mathcal{O}\texttt{Add}(i)$ requests: $\mathcal{B}$ creates a new user i and picks two random values $s_1, s_2 \in_R \mathbb{Z}_p^*$ as his secret key sk_i. $\mathcal{B}$ also generates the public/private endorsement key pair of user i denoted by (esk_i, epk_i).

- $\mathcal{O}\texttt{AddCorrupt}(i)$ requests: $\mathcal{B}$ does nothing.

- $\mathcal{O}\texttt{Join}_I()$ requests: Upon the receipt of C_{s_1}, C_{s_2}, the proof π_1 and the signature $Sign$, $\mathcal{B}$ first checks the validity of both $Sign$ and π_1. If so, it uses the soundness property of π_1 to recover s_1 and s_2. Then, as it holds y, $\mathcal{B}$ can generate a valid group signing key $gsk_i = (A_i, r_i, s_u^i = s_2 + s'_i)$ where $s'_i, r_i \in_R \mathbb{Z}_p^*$ and $A_i = (g_1^{s_1} g_2^{s_2+s'_i} h)^{\frac{1}{y+r_i}}$. Finally, it builds π_2 and provides it to $\mathcal{A}$ along with gsk_i.

- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{B}$ provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.

- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: As it holds both sk_i and gsk_i, $\mathcal{B}$ can generate a signature on m and for bsn, that it returns to $\mathcal{A}$.

Eventually, after q queries to $\mathcal{O}\texttt{Join}$, $\mathcal{A}$ outputs his forgery $(\sigma, m, \texttt{bsn})$ such that it breaks the traceability requirement of our BB-based Pre-DAA scheme. By assumption, the associated group signing key $gsk = (A, r, s_u)$ satisfies $(A, r) = (A_j, r_j)$ for some $j \in \{1, \ldots, q\}$. Since $(A, r) = (A_j, r_j)$, then $A_j^{y+r_j} = g_1^{s_1^j} g_2^{s_2^j+s'_j} h = A^{y+r} = g_1^{s_1} g_2^{s_2+s'} h$ and so, $g_1^{s_1^j} g_2^{s_2^j+s'_j} = g_1^{s_1} g_2^{s_2+s'}$. We have $s_1^j \neq s_1$ (otherwise the signature output by $\mathcal{A}$ would be traced to the user j). Therefore, $g_1 = g_2^{(\frac{s_2^j+s'_j-s_2-s'}{s_1-s_1^j})}$. Using the values $(s_1, s_1^j, s_2, s_2^j, s', s'_j)$, $\mathcal{B}$ can recover v which is equal to $(\frac{s_2^j+s'_j-s_2-s'}{s_1-s_1^j})$, hence breaking the DL problem. If $\mathcal{B}$ can break the DL problem, then it can break the $q-$SDH problem (by finding the discrete logarithm y of $Y = g_0^y$ in the base g_0).

If $c_{mode} = 2$: $\mathcal{A}$ eventually outputs $(\sigma_0, m_0, \sigma_1, m_1, \texttt{bsn}, sk)$ such that σ_0 and σ_1 are valid signatures on respectively m_0 and m_1, and for the same basename bsn. Besides, the two signatures were produced using $sk = (s_1, s_2)$ (as, by definition, both $\mathsf{Identify}_{\mathsf{S}}(\sigma_0, m_0, \texttt{bsn}, sk)$ and $\mathsf{Identify}_{\mathsf{S}}(\sigma_1, m_1, \texttt{bsn}, sk)$ output 1). Owing to the completeness of the proof π_3, the tags T_0 and T_1 associated to respectively σ_0 and σ_1 are necessarily defined as $T_0 = \mathcal{H}(\texttt{bsn})^{s_1}$ and $T_1 = \mathcal{H}(\texttt{bsn})^{s_1}$. Therefore, we have $T_0 = T_1$. By assumption, $\mathcal{A}$ wins if the output of $\mathsf{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \texttt{bsn})$ is 0 (*i.e.* $T_0 \neq T_1$). Thus, such forgery can never occur.

Therefore, if $\mathcal{A}$ can break the traceability property of our BB-based Pre-DAA scheme, then $\mathcal{B}$ can break the $q-$SDH assumption. Thus, under the $q-$SDH assumption, our Pre-DAA scheme provides the traceability requirement in the ROM.

□

Theorem 4.10. *Our BB-based pre-DAA scheme is* non-frameable*, in the ROM, under the one-more discrete logarithm (*OMDL*) assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary against the non-frameability requirement of our BB-based Pre-DAA scheme. As previously mentioned in Section 3.3.2.2, we distinguish the following two types of forgeries:

- **Type-1 Forger:** An adversary that manages to output a signature σ on m and for $\texttt{bsn}$ that is traced to a specific user that has never produced such a signature.
- **Type-2 Forger:** An adversary that outputs two signatures σ_0 and σ_1 that are linkable even though they should not (*i.e.* they were either produced with different sk_i or for two distinct basenames).

In what follows, we show that a Type-1 forger can be used to construct a reduction $\mathcal{B}$ against the OMDL assumption whilst a Type-2 forgery cannot happen. Initially, $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its challenger $\mathcal{C}$ a random instance $(g^{u_1}, g^{u_2}, \ldots, g^{u_n})$ of the OMDL problem where g is a random generator of $\mathbb{G}_1$. As it is against the one-more DL assumption, $\mathcal{B}$ has access to a DL oracle. The adversary $\mathcal{A}$ picks a random value $y \in_R \mathbb{Z}_p^*$ as the issuer's private key and publishes the associated public keys $Y = g_0^y$ and $W = \tilde{g}_0^y$ where g_0 and $\tilde{g}_0$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $\mathcal{B}$ sets $g_1 = g$ whilst the remaining generators are chosen at random. $\mathcal{B}$ answers $\mathcal{A}$'s requests as follows:

- $\mathcal{O}\texttt{Add}(i)$ requests: $\mathcal{B}$ first randomly chooses $s_2^i \in_R \mathbb{Z}_p^*$. Then, using its input of the OMDL problem, it sets s_1^i such as $C_{s_1^i} = g_1^{s_1^i} = g^{u_i}$. The user's secret key is defined as $sk_i = (s_1^i, s_2^i)$ where s_1^i is unknown to $\mathcal{B}$. $\mathcal{B}$ also generates the public/private endorsement key pair of user i denoted by (esk_i, epk_i).
- $\mathcal{O}\texttt{AddCorrupt}(i)$ requests: $\mathcal{B}$ does nothing.
- $\mathcal{O}\texttt{Join}_U(i)$ requests: $\mathcal{B}$ sets $C_{s_1^i} = g^{u_i}$ and computes $C_{s_2^i} = g_2^{s_2^i}$. Then, it computes S using esk_i. As for π_1, $\mathcal{B}$ can perfectly simulate it in the random oracle model. If the protocol does not abort, $\mathcal{B}$ obtains the triplet (A, r, s') which corresponds to the membership credential $(A, r, s_u = s_2^i + s')$ on s_1^i which is unknown to both $\mathcal{A}$ and $\mathcal{B}$.
- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{B}$ calls on the DL oracle with g^{u_i} as input. Thereby, it can recover $u_i = s_1^i$ and provide $\mathcal{A}$ with the secret key $sk_i = (s_1^i, s_2^i)$ along with the group signing key gsk_i of user i.
- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: As it holds s_u and A, $\mathcal{B}$ can directly compute $B_0 = A^l$ where $l \in_R \mathbb{Z}_p^*$. As for C, it is computed as $C = (g^{u_i} g_2^{s_u} h)^l B_0^{-r}$. Using C, $\mathcal{B}$ computes $E = C^{\frac{1}{l}} f^t$ where $t \in_R \mathbb{Z}_p^*$. To compute T, $\mathcal{B}$ chooses $b \in \mathbb{Z}_p^*$ and sets $\mathcal{H}(\texttt{bsn}) = g^b$. Consequently, $T = \mathcal{H}(\texttt{bsn})^{s_1^i} = (g^{u_i})^b$. For subsequent requests with the same i and $\texttt{bsn}$, $\mathcal{B}$ uses the same tag T. As regards to the proof π_3, it can be easily simulated in the ROM using classical techniques. Hence, $\mathcal{B}$ can perfectly simulate the $\texttt{GSign}$ algorithm.

Eventually, after d calls to the $\mathcal{O}\texttt{Corrupt}$ oracle, $\mathcal{A}$ outputs with non negligible probability a valid signature σ on m and for $\texttt{bsn}$, such that $\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \texttt{bsn}, sk_i)$ outputs $\texttt{1}$ whilst the user holding $sk_i = (s_1^i, s_2^i)$ has never produced a signature on that m and for $\texttt{bsn}$. By definition of the experiment, we know that the corresponding user is *honest*. Thus, the u_i associated to the user's *unknown* secret s_1^i is still in the input of the OMDL problem. Using the soundness property of π_3, $\mathcal{B}$ retrieves the associated secrets s_1^i and s_2^i where $s_1^i = u_i$. Thereby, $\mathcal{B}$ outputs u_i along with the d secrets $\{u_j\}_{j=1}^{j=d}$ that it has previously obtained by querying the DL oracle, hence breaking the OMDL assumption.

If $c_{mode} = 2$: $\mathcal{A}$ eventually outputs two valid signatures σ_0 and σ_1, on respectively m_0 and m_1, that are linkable even though they should not. That is, they were either generated using two different keys (sk_0 and sk_1) or/and are for different basenames ($\mathtt{bsn}_0$ and $\mathtt{bsn}_1$).

Let $sk_b = (s_1^b, s_2^b)$ denote the key used to generate σ_b and $sk = (\tilde{s}_1, \tilde{s}_2)$ the key output by $\mathcal{A}$. If $\mathcal{A}$'s attack against the non-frameability property is successful, then this implies that condition 6 of the non-frameability experiment is true. In particular, owing to the completeness of π_3, we have

$$T_0 = \mathcal{H}(\mathtt{bsn}_0)^{s_1^0} \quad \text{and} \quad T_1 = \mathcal{H}(\mathtt{bsn}_1)^{s_1^1} \tag{4.6}$$

The condition 7 should also be true. Thus, $\exists b \in \{0,1\}$ such that $\mathtt{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathtt{bsn}_b) = 1$. Without loss of generality, let us assume that $b = 0$. From the validity of this assertion, we can deduce that

$$T_0 = \mathcal{H}(\mathtt{bsn}_0)^{s_1^0} = T_1 = \mathcal{H}(\mathtt{bsn}_0)^{s_1^1} \tag{4.7}$$

This implies that

$$s_1^0 = s_1^1 \ (mod\ p) \tag{4.8}$$

For $\mathcal{A}$'s forgery to be successful, either condition 8 or 9 should be true. Suppose that assertion 8 is true. This implies that

$$T_0 = \mathcal{H}(\mathtt{bsn}_0)^{\tilde{s}_1} \tag{4.9}$$

and

$$T_1 \neq \mathcal{H}(\mathtt{bsn}_1)^{\tilde{s}_1} \tag{4.10}$$

This is impossible. Indeed, (4.7) and (4.9) imply that $s_1^0 = \tilde{s}_1$. Thus, from (4.8), we have $s_1^0 = \tilde{s}_1 = s_1^1$. From (4.6), we also know that $T_1 = \mathcal{H}(\mathtt{bsn}_1)^{s_1^1} = \mathcal{H}(\mathtt{bsn})^{\tilde{s_1}}$, which contradicts (4.10).

Let us now assume that condition 9 is true, *i.e.* that $\mathtt{bsn}_0 \neq \mathtt{bsn}_1$. (4.6) and (4.7) imply that $T_1 = \mathcal{H}(\mathtt{bsn}_1)^{s_1^1} = \mathcal{H}(\mathtt{bsn}_0)^{s_1^1}$, hence $\mathcal{H}(\mathtt{bsn}_0) = \mathcal{H}(\mathtt{bsn}_1)$ where $\mathtt{bsn}_0 \neq \mathtt{bsn}_1$. This would imply that $\mathcal{A}$ broke the *second pre-image resistance* property of the hash function $\mathcal{H}$ which is unfeasible in the ROM.

Therefore, a Type-2 forgery can never occur. Thus, under the OMDL assumption, our BB-based pre-DAA scheme ensures the non-frameability requirement.

□

Theorem 4.11. *Our BB-based pre-DAA scheme satisfies the* anonymity *requirement, in the ROM, under the* XDH *assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary against the anonymity requirement of our BB-based Pre-DAA scheme. Using Shoup's game hopping technique, where proofs are organized as sequences of games, we prove that our Pre-DAA scheme ensures the anonymity requirement under the XDH assumption. Hereinafter, we provide a high level description of the initial game (Game 0) along with a brief description of the transition between Game 0 and Game 1.

Game 0: It corresponds to the real attack game with respect to an efficient adversary $\mathcal{A}$.

The Challenger $\mathcal{C}$ randomly chooses the system public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathcal{H}, e)$ as well as the issuer's private key y and computes the associated public keys $Y = g_0^y$ and $W = \tilde{g}_0^y$. Then, it provides $\mathcal{A}$ with $gmsk = y$ and answers his requests as follows:

- $\mathcal{O}\mathtt{Add}(i)$ requests: $\mathcal{C}$ creates a new user i and picks two random values $s_1^i, s_2^i \in_R \mathbb{Z}_p^*$ as his secret key sk_i. $\mathcal{C}$ also generates user's i public/private endorsement key pair denoted by (esk_i, epk_i).

- $\mathcal{O}\mathtt{AddCorrupt}(i)$ requests: $\mathcal{C}$ does nothing.

- $\mathcal{O}\texttt{Join}_U(i)$ requests: $\mathcal{C}$ computes $C_{s_1^i}$ and $C_{s_2^i}$, then builds π_1 to obtain the group signing key gsk_i.

- $\mathcal{O}\texttt{Corrupt}(i)$ requests: $\mathcal{C}$ provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.

- $\mathcal{O}\texttt{GSign}(i, m, \texttt{bsn})$ requests: $\mathcal{C}$ uses sk_i and gsk_i to generate a signature σ on m and for $\texttt{bsn}$, that it returns to $\mathcal{A}$.

Eventually, $\mathcal{A}$ chooses $(i_0, i_1, \texttt{bsn}, m)$ that he provides as input to the oracle $\mathcal{O}\texttt{CH}_b$. The oracle outputs a signature σ produced by user b. $\mathcal{A}$'s goal is to guess the value of b. Even after receiving his challenge, $\mathcal{A}$ can still query the $\mathcal{O}\texttt{Add}$, $\mathcal{O}\texttt{AddCorrupt}$, $\mathcal{O}\texttt{Join}_U$, $\mathcal{O}\texttt{Corrupt}$ and $\mathcal{O}\texttt{GSign}$ oracles but with some restrictions. More precisely, he cannot call the $\mathcal{O}\texttt{Corrupt}$ oracle on i_0 or i_1. Besides, he is not allowed to query the $\mathcal{O}\texttt{GSign}$ oracle with either $(i_0, m, \texttt{bsn})$ or $(i_1, m, \texttt{bsn})$ as input. Otherwise, he would trivially win the game.

So, $\mathcal{A}$ eventually outputs his guess b'. Let S_0 be the event that $b = b'$ in Game 0 (*i.e.* the event that $\mathcal{A}$ wins Game 0) and S_1 denote the event that $b = b'$ in Game 1. We have $\texttt{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2| = |\texttt{Pr}[S_0] - 1/2|$.

We construct the Game 1 as follows:

Game 1: It is the same game as Game 0 except that we replace $T = \mathcal{H}(\texttt{bsn})^{s_1^b}$ by a random value and simulate the proof π_3. Such a proof can be easily simulated in the ROM using classical techniques. Under the XDH assumption, $\mathcal{A}$ cannot detect this change. In fact, one can easily construct a XDH distinguisher $\mathcal{D}$ with an advantage of solving the XDH problem, denoted by $\texttt{Adv}_{\mathsf{XDH}}$, satisfying $|\texttt{Pr}[S_0] - \texttt{Pr}[S_1]| \leq \texttt{Adv}_{\mathsf{XDH}}(1^k)$.

In Game 1, the challenger provides no information (in a strong information theoretic sense) about the bit b to the adversary $\mathcal{A}$. This is due to the zero-knowledge property of π_3 and the fact that, for all $B_0 = A^l$, there exist A', l' such that $B_0 = A'^{l'}$. Moreover, C is simply equal to B_0^y, so it does not reveal any additional information. Therefore, $\texttt{Pr}[S_1] = 1/2$. Thus, we have

$$\texttt{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2| = |\texttt{Pr}[S_0] - \texttt{Pr}[S_1]| \leq \texttt{Adv}_{\mathsf{XDH}}(1^k)$$

Consequently, under the XDH assumption, the adversary's advantage is negligible. Thus, we conclude that our BB-based Pre-DAA scheme satisfies the anonymity requirement under the XDH assumption. □

4.5 Conclusion

In this chapter, we introduced five new cryptographic primitives that are of independent interest and formally proved their security. In subsequent chapters, we show how they can be used to build privacy-preserving protocols. More precisely, we rely on our partially blind signature scheme to design a practical private eCash system intended for applications such as electronic Toll (eToll) and Electric Vehicle Charging (EVC). Then, based on our *pairing-free* BB-based algebraic MAC scheme, we construct a practical Keyed-Verification Anonymous Credentials (KVAC) system suitable for resource constrained environments such as SIM cards. Our KVAC system can be easily turned into an efficient public key anonymous credentials system. As regards to our Pre-DAA schemes, they are used as the main building block of an anonymous authentication and identification protocol for eSIMs. Finally, our sequential aggregate MAC scheme is used to build an efficient coercion-resistant remote electronic voting scheme that enables multiple elections and credentials revocation.

Chapter 5

A Practical Private eCash System

Contents

In this chapter, we propose a private eCash system intended for applications that involve a single entity acting as both merchant and bank such as electronic Toll (eToll), electronic Ticketing (eTicketing) and EVC (Electric Vehicle Charging). Our proposal relies on our partially blind signature scheme introduced in Section 4.1 so as to allow multiple presentations of the same signature (payment token) in an unlinkable way. Thereby, users can settle expenses of different amounts using a *single* reusable payment token. Our eCash system also enables identity as well as token revocations under exceptional circumstances. To show its efficiency even in constrained environment, we implemented it on a GlobalPlatform [Glo15] compliant SIM card. In particular, a transaction can be performed in just 205 *ms*. We also formally proved the security of our proposal in the random oracle model. The results [BBD+16] presented in this chapter have been published in the paper "Private eCash in Practice" as part of the proceedings of the conference FC'16.

5.1 Electronic Cash (eCash)

Electronic Cash (eCash), introduced by Chaum [Cha83], is the digital analogue of hard currency where users withdraw electronic coins from a bank and subsequently spend them to merchants. Later, each merchant deposits the received coins to the bank.

Using eCash, user's anonymity is protected both from the bank and merchants. More precisely, eCash prevents the following pairs of events from being linked: a withdrawal and a spending or two spendings by the same user. However, owing to its digital nature, eCash can be easily duplicated and hence, fraudulently spent more than once. Thus, eCash protocols must be designed in a way that enables the detection of "double-spending" and the identification of defrauders.

To be as attractive and user friendly as possible, the user's side of an eCash system is sometimes implemented on a mobile device or even a smart card. Therefore, protocols have to comply with both the limited resources of such environments as well as the stringent delay constraint arising from transactions requirements.

In this chapter, we propose a privacy-preserving electronic payment system for applications such as electronic Toll (eToll) and Electric Vehicle Charging (EVC) which significantly invade user's privacy. Indeed, transactions records may disclose user's location at a given time and reveal personal information such as work, health status or habits. To design our protocol, we leverage a common feature to all these applications: they involve a single entity acting as both bank and merchant. We refer to eCash systems suitable for these particular use cases as *Private eCash*.

5.2 Related Work

Recently, several schemes have addressed the privacy issue in the case of eToll, EVC and public transport. Unfortunately, they have not completely succeeded in finding a good tradeoff between the security as well as privacy requirements, flexibility and performance. In the sequel, we only focus on schemes related to eCash although other approaches exist in the literature [PBB09, BRT+10, MMCS11].

Public transport users' privacy was tackled in [RHBP13], [MDND15] and [ALT+15]. In the first two proposals, users pay for their trips through the use of payment tokens that are worth the highest possible fare. As fares have different values, a refund process is set up to guarantee the accurate charging of users. However, to ensure user's anonymity with respect to the transport company, the scheme of Rupp *et al.* [RHBP13] entails heavy verifications (*i.e.* pairing computations) that constrained devices such as SIM cards cannot handle. The proposal of Milutinovic *et al.* [MDND15] is also computationally expensive and less efficient than [RHBP13] as refunds are separately collected on distinct refund tokens. Arfaoui *et al.* [ALT+15] protocol meets the stringent delay requirement and is efficient even when implemented in constrained environment. It is also the only one allowing anonymity revocation under exceptional circumstances. Nevertheless, their scheme does not enable flexible prices which limits its use.

As regards to EVC user's privacy, it was addressed by Au *et al.* [ALF+14]. However, their scheme requires costly ZKPK and is not suitable for time-sensitive applications.

Finally, Day *et al.* [DHKG11] proposed two privacy-preserving payment systems for eToll. The first one only provides partial privacy as it relies on spot checks that record some of the user's spatio-temporal information in order to enable the detection and identification of defrauders. Furthermore, through an exhaustive search on tokens, it is possible to trace all users' trips. In contrast, the second proposal provides full anonymity and enables double spending detection. Unfortunately, to be efficient, users have to hold a large number of tokens where each one can only be used at a specific time. Besides, similarly to eCoupons [LMY15], both proposals do not allow flexible prices, which we believe to be an important issue.

5.3 Private eCash System Framework: The eToll use case

5.3.1 Stakeholders

In the case of the eToll application, our private eCash system involves three main entities: a user $\mathcal{U}$, a toll company $\mathcal{TC}$ and a revocation authority $\mathcal{RA}$. The toll company manages a set of tollbooths with which it shares a set of keys $\vec{x} = (x_0, x_1, x_2)$. Besides, it provides each registered user $\mathcal{U}$ with a badge that will perform all computations on his behalf. In the sequel, excepted otherwise mentioned, $\mathcal{TC}$ will refer to both the toll company and tollbooths. The term *user* will interchangeably refer to a user and his badge. For security reasons, the role of $\mathcal{RA}$ is distributed among several authorities which must collaborate in order to revoke the anonymity of a user or a payment token.

5.3.2 Trust Assumptions and Performance Requirements

Trust assumptions. None of the entities involved in such systems can be fully trusted since each one of them has some incentives to cheat. Only the user's badge is subject to the limited trust assumption that all the computations it performs are correct. Nevertheless, any attempt to cheat by tampering it must be detected.

Performance requirement. To be effective and suitable for most use cases, the eCash must meet a stringent delay constraint. In fact, a transaction must be performed in **at most** 300 ms [GSM12]. This requirement must be fulfilled whilst taking into account the limited computational capabilities of SIM cards which, we recall, cannot handle heavy computations and, in particular, pairing computations.

5.3.3 System Framework

To model an eToll pricing system, six different phases are required: (1) A *Setup* phase enables the initialization of the public parameters and the generation of required keys. (2) The *Registration* phase, as its name implies, allows a user to register to the eToll system. (3) The *Token Issuance* phase provides legitimate users with a unique reusable payment token. (4) During *Access Control*, a user uses his payment token to be granted access at tollbooths. (5) The *Toll Computation* phase allows the computation of the user's bill based on his current token value. Finally, (6) the *Revocation* phase, which may be triggered under exceptional circumstances, enables user's anonymity and token revocations. These six phases can be modelled through the following set of algorithms and protocols.

1. The *Setup* phase, which aims to initialize the system parameters as well as the required keys, consists of the following two algorithms:
   - `PubParam`(1^k): On input a security parameter k, this probabilistic algorithm outputs the system public parameters pp.
   - `KeyGen`(pp): On input the system public parameters pp, this probabilistic algorithm outputs the private/public key pairs of the different entities: the toll company is provided with (sk_{tc}, pk_{tc}) and the revocation authority gets (sk_{ra}, pk_{ra}). The toll company is also provided with a set of private keys $\vec{x} = (x_0, x_1, x_2)$ that are shared with the tollbooths.
2. *Registration*
   - `UKeyGen`(pp, ID_u): On input a user's identifier ID_u as well as the public parameters pp, this probabilistic algorithm outputs a private/public key pair (sk_u, pk_u) associated to ID_u.

- $\texttt{Register}(\mathcal{U}(ID_u, pk_u), \mathcal{TC}(\texttt{DB}_{\texttt{REG}}))$: This is an interactive protocol between the toll company $\mathcal{TC}$ and a user $\mathcal{U}$. At the end of this protocol, the toll company registers the new user and saves his identifier as well as his public key within the database of registered users denoted $\texttt{DB}_{\texttt{REG}}$.

3. *Token Issuance*

   $\texttt{TokenIssue}(\mathcal{U}(sk_u, pk_{ra}), \mathcal{TC}(\vec{x}, \texttt{DB}_{\texttt{REG}}), N_{\texttt{max}})$: This is an interactive protocol between the toll company $\mathcal{TC}$ and a user $\mathcal{U}$. At the end of this protocol, the user receives a payment token T, worth M, that will enable him to take at most $N_{\texttt{max}}$ trips. Concurrently, the transcript view of the protocol execution ($\texttt{transc}_{\texttt{TI}}$) is saved in the database $\texttt{DB}_{\texttt{REG}}$.

4. *Access Control*

   $\texttt{AccessControl}(\mathcal{U}(T, pk_{ra}), \mathcal{TC}(sk_{tc}, \vec{x}, \texttt{DB}_{\texttt{AC}}), v)$: This is an interactive protocol between a tollbooth $\mathcal{TC}$ and a user $\mathcal{U}$ who provides his payment token T to be granted access at a tollbooth. If the token is valid and the toll fare $v < M$, the tollbooth updates T to add the refund associated to the current transaction (*i.e.* $M - v$). Otherwise, it outputs $\perp$ and aborts. Concurrently, the transcript view $\texttt{transc}_{\texttt{AC}}$ of the protocol execution is saved in a database denoted by $\texttt{DB}_{\texttt{AC}}$.

5. *Toll Computation*

   $\texttt{TollCompute}(\mathcal{U}(T), \mathcal{TC}(\vec{x}, \texttt{DB}_{\texttt{REG}}))$: This algorithm, which is executed by the toll company, takes as input the current token held by the user. According to its value, it outputs the overall toll charges the user $\mathcal{U}$ owes to the toll company.

6. *Revocation*: Two different revocations, namely user's anonymity and token revocations, may be exceptionally triggered. Both are performed by the revocation authority $\mathcal{RA}$ upon a court order or a legitimate request of the toll company $\mathcal{TC}$.

   - $\texttt{IDRetrieve}(\texttt{transc}_{\texttt{AC}}, sk_{ra}, \texttt{DB}_{\texttt{REG}})$: This deterministic algorithm takes as input the private key of the revocation authority, an anonymous access control transcript view $\texttt{transc}_{\texttt{AC}}$ and the registration database $\texttt{DB}_{\texttt{REG}}$. It outputs the identifier ID_u of the user who performed this transaction.
   - $\texttt{TokenRevoc}(ID_u, \texttt{transc}_{\texttt{TI}}, sk_{ra})$: This deterministic algorithm takes as input a user's identifier ID_u, the private key sk_{ra} of the revocation authority as well as a transcript view of an execution of $\texttt{TokenIssue}$. It outputs all the valid tags T_i that can be generated by the user whose identifier is ID_u.

5.4 Security and Privacy Models

In this section, we define the different properties that a private eCash system must satisfy to be both secure and privacy-friendly, namely *correctness*, *unforgeability*, *non-frameability* and *unlinkability*. They are formalized through a set of experiments that involve games between a challenger $\mathcal{C}$ and a Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$. We distinguish two sets of users, namely $\mathcal{HU}$ and $\mathcal{CU}$. The former corresponds to the set of registered honest users whose private keys are unknown to $\mathcal{A}$ while $\mathcal{CU}$ denotes the set of corrupted users for whom $\mathcal{A}$ knows both the public and private keys. To model the capabilities of $\mathcal{A}$, we give him access to several oracles defined as follows:

- $\mathcal{O}\texttt{Register}(ID_u)$ is an oracle used by $\mathcal{A}$ to create a new honest user identified by ID_u and generate his pair of keys. The challenger runs the $\texttt{UKeyGen}$ algorithm and provides $\mathcal{A}$ with the public key pk_u while keeping sk_u secret. The identity ID_u and the associated public key pk_u will be added to the set $\mathcal{HU}$.

- $\mathcal{O}\texttt{RegisterCorrupt}(ID_u)$ is an oracle that enables $\mathcal{A}$ to create a new corrupted user with a pre-defined pair of keys (sk_u, pk_u). The user identified by ID_u is added to the set $\mathcal{CU}$. Only his public key pk_u is known to the challenger.

- $\mathcal{O}\texttt{Corrupt}(ID_u)$ is an oracle used by $\mathcal{A}$ to corrupt the honest user identified by ID_u. Upon the execution of this oracle, the challenger provides $\mathcal{A}$ with the corresponding secret key sk_u. Furthermore, the user identified by ID_u is moved from $\mathcal{HU}$ to $\mathcal{CU}$.

- $\mathcal{O}\texttt{TokenIssue}_U(ID_u)$ is an oracle that executes the user's side of the `TokenIssue` protocol. This oracle will be used by $\mathcal{A}$ acting as a malicious $\mathcal{TC}$. The adversary provides the oracle with an honest user's identifier ID_u. If the corresponding user accepts the protocol, the adversary gets a transcript view $\texttt{transc}_{\texttt{TI}}$ of the protocol execution.

- $\mathcal{O}\texttt{TokenIssue}_{TC}()$ is an oracle that executes the $\mathcal{TC}$'s side of the `TokenIssue` protocol. This oracle will be used to simulate the protocol execution between a corrupted or honest user and an honest $\mathcal{TC}$. Even though the adversary always sees the transcript view $\texttt{transc}_{\texttt{TI}}$ of the protocol execution, the issued token is known to $\mathcal{A}$ only if the simulated user is corrupted.

- $\mathcal{O}\texttt{AccessControl}_U(ID_u, v)$ is an oracle which executes the user's side of the `AccessControl` protocol. It will be used by $\mathcal{A}$ acting as a malicious $\mathcal{TC}$. If the user accepts the protocol, the adversary gets a transcript view $\texttt{transc}_{\texttt{AC}}$ of the `AccessControl` protocol execution with an associated toll fare equal to v. Concurrently, $\texttt{transc}_{\texttt{AC}}$ is saved in both $\texttt{DB}_{\texttt{u}}$ and $\texttt{DB}_{\texttt{AC}}$.

- $\mathcal{O}\texttt{AccessControl}_{TC}()$ is an oracle that executes the $\mathcal{TC}$'s side of the `AccessControl` protocol. It simulates the protocol execution between an honest or corrupted user and an honest $\mathcal{TC}$. The transcript view $\texttt{transc}_{\texttt{AC}}$ is saved in $\texttt{DB}_{\texttt{AC}}$.

- $\mathcal{O}\texttt{IDRetrieve}(\texttt{transc}_{\texttt{AC}})$ is an oracle that executes the $\mathcal{RA}$'s side of the `IDRetrieve` protocol. The adversary $\mathcal{A}$, who acts as $\mathcal{TC}$, provides a valid anonymous transcript view $\texttt{transc}_{\texttt{AC}}$ and gets, in return, the identity ID_u of the user who performed this transaction.

- $\mathcal{O}\texttt{TokenRevoc}(ID_u, \texttt{transc}_{\texttt{TI}})$ is an oracle that executes the $\mathcal{RA}$'s side of the protocol `TokenRevoc`. The adversary $\mathcal{A}$, who acts as $\mathcal{TC}$, provides the identifier ID_u of a user $\mathcal{U}$ as well as a transcript view of the execution of the `TokenIssue` protocol by this user. This oracle will return all the tags T_i that can be generated by this user.

- $\mathcal{O}\texttt{TollCompute}(ID_u)$ is an oracle that returns the total amount that the user identified by ID_u owes to $\mathcal{TC}$.

5.4.1 Security Model

To be secure, a private eCash system should ensure three properties: *correctness*, *unforgeability* and *non-frameability*. In what follows, we provide both informal as well as formal definitions of these properties.

Correctness. As usual, *correctness*, which ensures the proper functioning of the system, is primordial. In our case, correctness is satisfied if the following four statements are met: (1) the user's registration went well; (2) the provided tokens are valid; (3) the final bill corresponds exactly to the fees associated to the users' trips; (4) the revocation authority can always identify the user who initiated a given `AccessControl` transcript view.

Unforgeability. Informally, unforgeability requires that no entity, even a set of colluding malicious users, can (1) perform the `AccessControl` protocol without the revocation authority being able to retrieve the identity of the corresponding user (*i.e.* such that `IDRetrieve` outputs $\perp$), (2) cheat in order to pay less toll charges than what it has to. More formally, the unforgeability experiment $\texttt{Exp}_{\mathcal{A}}^{\text{unforg}}(1^k)$ is defined in Figure 5.1. The advantage $\texttt{Adv}_{\mathcal{A}}^{\text{unforg}}(1^k)$ of the adversary is defined as $\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{unforg}}(1^k) = 1]$. A private eCash system satisfies the *unforgeability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\texttt{Exp}_{\mathcal{A}}^{\text{unforg}}(1^k)$

1. $(pp, (sk_{tc}, pk_{tc}), \overrightarrow{x}, (sk_{ra}, pk_{ra})) \leftarrow \texttt{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \texttt{DB}_{\texttt{AC}} \leftarrow \varnothing$
2. $\{T_k\}_{k=1}^{k=n_o} \longleftarrow \mathcal{A}^{\mathcal{O}}()$ where $\mathcal{O} = \{\mathcal{O}\texttt{Register}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{RegisterCorrupt}, \mathcal{O}\texttt{TokenIssue}_{TC}, \mathcal{O}\texttt{AccessControl}_{TC}, \mathcal{O}\texttt{IDRetrieve}, \mathcal{O}\texttt{TokenRevoc}, \mathcal{O}\texttt{TollCompute}\}$ and $\{T_k\}$ is a set of n_0 tokens.
3. If one of the provided tokens T_k is not valid, then it is discarded.
4. If $\exists \texttt{transc}'_{\texttt{AC}} \in \texttt{DB}_{\texttt{AC}}$ such that $\texttt{IDRetrieve}(\texttt{transc}'_{\texttt{AC}}, \texttt{DB}_{\texttt{REG}}, sk_{ra}) = \perp$, then return 1.
5. If $\sum_{k=1}^{n_v} \texttt{TollCompute}(T_k) < \sum_{i=1}^{\#\texttt{DB}_{\texttt{AC}}} Charge(\texttt{transc}_{\texttt{AC}}^{\texttt{i}})$, where $Charge(\texttt{transc}_{\texttt{AC}}^{\texttt{i}})$ corresponds to the toll charge associated to the transcript view $\texttt{transc}_{\texttt{AC}}^{\texttt{i}}$, $\#\texttt{DB}_{\texttt{AC}}$ denotes the number of transcripts in $\texttt{DB}_{\texttt{AC}}$ and n_v is the number of valid tokens, then return 1.
6. Return 0

Figure 5.1: Unforgeability Security Experiment

Non-Frameability. Informally, non-frameability requires that no entity, including the toll company and even with the help of malicious users, can produce a valid transcript that falsely accuse an honest user of performing a given `AccessControl`. More formally, the non-frameability experiment $\texttt{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k)$ is defined in Figure 5.2. The advantage $\texttt{Adv}_{\mathcal{A}}^{\text{non-fram}}(1^k)$ of the adversary is defined as $\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k) = 1]$. An eToll system satisfies the *non-frameability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\texttt{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k)$

1. $(pp, (sk_{tc}, pk_{tc}), \overrightarrow{x}, (sk_{ra}, pk_{ra})) \leftarrow \texttt{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \texttt{DB}_u \leftarrow \varnothing$
2. $\texttt{Transc}_{\texttt{AC}_\texttt{i}} \longleftarrow \mathcal{A}^{\mathcal{O}}(sk_{tc}, \overrightarrow{x}, sk_{ra}, \texttt{DB}_{\texttt{AC}}, \texttt{DB}_{\texttt{REG}})$ where $\mathcal{O} = \{\mathcal{O}\texttt{Register}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{RegisterCorrupt}, \mathcal{O}\texttt{TokenIssue}_U, \mathcal{O}\texttt{AccessControl}_U\}$
3. If $\texttt{Transc}_{\texttt{AC}_\texttt{i}}$ is not valid or $\texttt{IDRetrieve}(\texttt{transc}_{\texttt{AC}_\texttt{i}}, \texttt{DB}_{\texttt{REG}}, sk_{ra}) = \perp$ then return 0.
4. If $\texttt{IDRetrieve}(\texttt{transc}_{\texttt{AC}_\texttt{i}}, \texttt{DB}_{\texttt{REG}}, sk_{ra}) = ID_u \in \mathcal{HU}$ and $\texttt{transc}_{\texttt{AC}_\texttt{i}} \notin \texttt{DB}_u$ then return 1.
5. Return 0

Figure 5.2: Non-Frameability Security Experiment

5.4.2 Privacy Model

To be privacy-friendly, a private eCash system must additionally ensure *unlinkability*. Hereinafter, we define this property first in an informal, then, in a formal manner.

Unlinkability. Roughly speaking, the *unlinkability* property requires that, aside from the revocation authority, no entity (even with the help of malicious users) can (1) link an `AccessControl`

transcript to a specific execution of the `TokenIssue` protocol; (2) link two `AccessControl` transcripts performed by the same user, not necessarily identified; (3) link an `AccessControl` transcript to a specific execution of the `TollCompute` protocol. Note that the *unlinkability* property implies the *anonymity* one (*i.e.* an unlinkable system is necessarily anonymous). Informally, the anonymity property ensures that, apart from the revocation authority, no entity (even with the help of malicious users) should be able to retrieve the identity of the user who performed a given `AccessControl`.

$\mathtt{Exp}_{\mathcal{A}}^{\text{unlink}-b}(1^k)$

1. $(pp, (sk_{tc}, pk_{tc}), \vec{x}, (sk_{ra}, pk_{ra})) \leftarrow \mathtt{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing$
2. $(u_0, u_1, v_0, v_1) \leftarrow \mathcal{A}^{\mathcal{O}}(sk_{tc}, \vec{x}, \mathtt{DB_{AC}}, \mathtt{DB_{REG}})$ where $\mathcal{O} = \{\mathcal{O}\mathtt{Register}, \mathcal{O}\mathtt{RegisterCorrupt}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{TokenIssue}_U, \mathcal{O}\mathtt{AccessControl}_U, \mathcal{O}\mathtt{IDRetrieve}, \mathcal{O}\mathtt{TokenRevoc}, \mathcal{O}\mathtt{TollCompute}\}$ and v_0, v_1 are two toll fares.
3. If u_0 or $u_1 \in \mathcal{CU}$ then output $\perp$.
4. For $i \in \{0, 1\}$, $(T_i, \mathtt{transc}_{\mathtt{TI}_i}) \leftarrow \mathtt{TokenIssue}(\mathcal{C}(sk_{u_i}, pk_{ra}), \mathcal{A}(\vec{x}, \mathtt{DB_{REG}}), N_{\max})$
5. $\mathtt{transc_b} \leftarrow \mathtt{AccessControl}(\mathcal{C}(T_1, pk_{ra}), \mathcal{A}(sk_{tc}, \vec{x}, \mathtt{DB_{AC}}), v_b)$ where $b \in \{0, 1\}$ and $\mathtt{transc}_{1-\mathtt{b}} \leftarrow \mathtt{AccessControl}(\mathcal{C}(T_0, pk_{ra}), \mathcal{A}(sk_{tc}, \vec{x}, \mathtt{DB_{AC}}), v_{1-b})$
6. $b' \leftarrow A^{\mathcal{O}'}(sk_{tc}, \vec{x}, \mathtt{DB_{AC}}, \mathtt{DB_{REG}}, \mathtt{transc}_{\mathtt{TI_b}}, \mathtt{transc}_{\mathtt{TI}_{1-\mathtt{b}}}, \mathtt{transc_b}, \mathtt{transc}_{1-\mathtt{b}})$ where $\mathcal{O}' = \{\mathcal{O}\mathtt{Register}, \mathcal{O}\mathtt{RegisterCorrupt}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{TokenIssue}_U, \mathcal{O}\mathtt{AccessControl}_U, \mathcal{O}\mathtt{IDRetrieve}, \mathcal{O}\mathtt{TokenRevoc}, \mathcal{O}\mathtt{TollCompute}^*\}$ such that $\mathcal{O}\mathtt{TollCompute}^*$ denotes a restricted access to $\mathcal{O}\mathtt{TollCompute}$ to the cases specified above.
7. If $\mathcal{O}\mathtt{Corrupt}(ID_{u_0})$ or $\mathcal{O}\mathtt{Corrupt}(ID_{u_1})$ then output $\perp$.
8. If $\mathcal{O}\mathtt{IDRetrieve}(\mathtt{transc_b})$ or $\mathcal{O}\mathtt{IDRetrieve}(\mathtt{transc}_{1-\mathtt{b}})$ then output $\perp$.
9. If $\mathcal{O}\mathtt{TokenRevoc}(ID_{u_0}, \mathtt{transc}_{\mathtt{TI}_0})$ or $\mathcal{O}\mathtt{TokenRevoc}(ID_{u_1}, \mathtt{transc}_{\mathtt{TI}_1})$ then output $\perp$.
10. Return b'

Figure 5.3: Unlinkability Security Experiment

To achieve his goal, an adversary $\mathcal{A}$ may corrupt all users except two users: u_0 and u_1. He can also collude with the toll company and get its private keys (*i.e.* sk_{tc} and $\vec{x}$). Besides, $\mathcal{A}$ can query the `IDRetrieve` protocol on any transcript of his choice excluding both challenge transcripts (*i.e.* $\mathtt{transc_b}$ and $\mathtt{transc}_{1-\mathtt{b}}$) where $b \in \{0, 1\}$ and he can call the `TokenRevoc` protocol on any identifier, apart from ID_{u_0} and ID_{u_1}. $\mathcal{A}$ has also access to the `TollCompute` protocol. However, to ensure that statements (2) and (3) hold, $\mathcal{A}$'s access to the `TollCompute` protocol is restricted to some particular cases. More precisely, he can only query it if the following three conditions are met:

- **(i)** the toll fares associated to both challenge transcripts are equal (*i.e.* $v_0 = v_1$);
- **(ii)** during the challenge phase, the overall toll charges corresponding to all calls to the oracle $\mathcal{O}\mathtt{AccessControl}$ are the same for u_0 and u_1;
- **(iii)** during the challenge phase (including the challenge transcript), u_0 used exactly the same set of generators g_i as u_1 (*i.e.* for example, if $\{g_1, g_2, g_5, g_9\}$ were used by u_0, then $\mathcal{A}$ cannot query $\mathcal{O}\mathtt{TollCompute}$ unless u_1 has only used $\{g_1, g_2, g_5, g_9\}$).

These restrictions may seem excessive at first glance, but a closer look reveals that they are rather fair and reflect the actual capabilities of a real adversary. Indeed, in a real deployment, the group of users that have taken the same number of trips while using the same set of g_i(s) will be quite large. Thus, a given user will remain anonymous among a large group of users. Furthermore, as shown in [RBHP15], an aggregate amount should not enable the retrieval of the different trips taken by the user.

More formally, our unlinkability experiment $\texttt{Exp}_{\mathcal{A}}^{\text{unlink}-b}(1^k)$ is detailed in Figure 5.3. The advantage $\texttt{Adv}_{\mathcal{A}}^{\text{unlink}-b}(1^k)$ of the adversary is defined as $\texttt{Adv}_{\mathcal{A}}^{\text{unlink}-b}(1^k) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{unlink}-b}(1^k) = b] - 1/2|$. An eToll system is considered *unlinkable* if this advantage is negligible for any PPT adversary $\mathcal{A}$.

We emphasize that the *unlinkability* property defined above is slightly weaker than the classical *full-unlinkability*[1] property as we aimed to achieve a better trade-off between privacy and efficiency. It is, however, worth mentioning that *full-unlinkability* can be achieved by simply adding some randomness through the use of a pseudo-random function as in [ALT+15]. Such change slightly deteriorates the system performances since the modified scheme will additionally require a set membership proof. Nevertheless, it still remains efficient enough for the targeted applications.

5.5 Our Private eCash System

5.5.1 Overview

To benefit from the eToll service, a user must first register to obtain a badge that will perform all computations on his behalf. At the beginning of each billing period, registered users receive a unique reusable token generated using our partially blind signature scheme detailed in Section 4.1. The obtained token is worth the highest possible fare and can be reused at most $N_{\texttt{max}}$ times. To be granted access at tollbooths while preserving his anonymity, the user provides a randomized version of his token. Concurrently, as toll fares are generally different, the user's token is updated in a blinded way, using our partially blind signature scheme, to add the refund amount associated to the transaction. At the end of the billing period, users are charged according to their token value. Such a post-payment approach prevents them from refilling a prepaid account with a large amount of money. Nevertheless, if a user does not return his token, he will pay the maximal allowed amount which corresponds to $N_{\texttt{max}}$ times the highest fare.

5.5.2 Description of the Protocols

As previously mentioned, our private eCash system consists of six main phases, namely *setup*, *Registration*, *Token Issuance*, *Access Control*, *Toll Computation* and *Revocation*. Below, we explain these phases and detail the main protocols in Figures 5.4, 5.5 and 5.6.

5.5.2.1 Setup

Let $pp = (\mathbb{G}, g, h, q, g_R, N_{\texttt{max}}, \{g_i\}_{i=1}^{N_{\texttt{max}}})$ denote the system public parameters where $\mathbb{G}$ is a cyclic group of prime order q and $(g, h, g_R, \{g_i\}_{i=1}^{N_{\texttt{max}}})$ a set of random generators. $N_{\texttt{max}}$ indicates the allowed number of reuses of a token and could be set according to user's needs. Each user $\mathcal{U}$ is also provided with a pair of keys (sk_u, pk_u) that identifies him and which are subsequently used to generate the Schnorr signature.

The toll company shares the secret key $\vec{x} := (x_0, x_1, x_2)$ with tollbooths, that are denoted by $\mathcal{TC}$ as well. The associated public parameters are $C_{x_0} := g^{x_0} h^{\tilde{x}_0}$, a commitment to x_0 where $\tilde{x}_0 \in_R \mathbb{Z}_q^*$, and $X_1 := h^{x_1}$, $X_2 := h^{x_2}$. $\mathcal{TC}$ is also provided with a pair of keys (sk_{tc}, pk_{tc}) used to sign transaction data.

The revocation authorities jointly generate two pairs of keys: (sk_{ra}, pk_{ra}) of the threshold ElGamal encryption scheme and (sk_{rp}, pk_{rp}) of the threshold Paillier cryptosystem. Paillier's encryption scheme is used as an *extractable commitment* scheme (see [HT07]) to satisfy the unforgeability requirement, even in a concurrent setting, where an adversary is allowed to interact with $\mathcal{TC}$ in an arbitrarily interleaving (concurrent) manner. Let g_E and g_P be two generators of

[1] That is, no restrictions are made on $\mathcal{A}$'s access to the `TollCompute` protocol.

$\mathbb{G}$. ElGamal keys are defined as $sk_{ra} := x_T \in \mathbb{Z}_q^*$ and $pk_{ra} := (g_E, X_T := g_E^{x_T})$. Paillier pair of keys consists of $sk_{rp} := (a, b)$ and $pk_{rp} := (g_P, n := ab)$ where a and b are two different random primes such that $|a| = |b|$ and $gcd(ab, (a-1)(b-1)) = 1$. The private keys are shared among $\mathcal{RA}$s [FPS01] and at least t of them should cooperate to identify the user or revoke a token.

5.5.2.2 Registration

To use the service, $\mathcal{U}$ must provide $\mathcal{TC}$ with his public key pk_u and a ZKPK proving the knowledge of the secret key sk_u. If the proof is valid, $\mathcal{U}$ receives a personal badge $\mathcal{B}_u$ including a SIM card. It allows $\mathcal{U}$ to anonymously use the service. Moreover, pk_u is saved in a dedicated database denoted by $\mathtt{DB}_{\mathtt{REG}}$.

5.5.2.3 Token Issuance

The *token issuance* phase occurs at the beginning of each billing period upon a signed request (using Schnorr's signature scheme) of a registered user. During this phase, $\mathcal{TC}$ provides $\mathcal{U}$ with a permission token $T := (u, u' := u^{x_0 + x_1 s_u + x_2 m})$. It is a partially blind signature on the unknown message $s_u = s + s'$ and the common information m corresponding to the refund amount, initially set to 0. In fact, s_u is a secret value only known by $\mathcal{U}$: it involves a secret $s \in_R \mathbb{Z}_q^*$ chosen by $\mathcal{U}$ and hidden from $\mathcal{TC}$ and $s' \in_R \mathbb{Z}_q^*$ chosen by $\mathcal{TC}$ and provided to $\mathcal{U}$. The token is worth the highest possible toll fare M and can be reused $N_{\mathtt{max}}$ times. Two ZKPKs π_1 and π_2 ensure that exchanged values are well-formed (see Figure 4.2). Concurrently, (g^s, s', D) is saved in $\mathtt{DB}_{\mathtt{REG}}$ where D is a Paillier encryption of s used so as to enable the revocation of a payment token.

User $\mathcal{U}$		**Toll Company $\mathcal{TC}$**
Public Input: pp, X_1, X_2 and C_{x_0} **Private Input:** sk_u		**Public Input:** pp, pk_u **and** M **Private Input:** $\mathtt{DB}_{\mathtt{REG}}$, x_0, x_1 **and** x_2
Choose $r, r_1, s \overset{R}{\leftarrow} \mathbb{Z}_q^*$ **Compute** $C \leftarrow h^r X_1^s$, $W = g^s$, $D = g_P^s r_1^n$		
Build $\pi_1 = \mathrm{PoK}\{\alpha, \beta, \gamma : C = h^\alpha X_1^\beta$ $\wedge\ W = g^\beta \wedge\ D = g_P^\beta \gamma^n\}$	$\xrightarrow{C, W, D, \pi_1}$	**Check** π_1 **Choose** $s', b \overset{R}{\leftarrow} \mathbb{Z}_q^*$ **Compute** $u \leftarrow h^b$ $u'' \leftarrow u^{x_0} (C X_1^{s'})^b$
Check π_2 **Compute** $u' \leftarrow \frac{u''}{u^r}$ and $T \leftarrow (u, u')$ $m \leftarrow 0, F \leftarrow \{g_i\}_{i=1}^{N_{\mathtt{max}}}$	$\xleftarrow{g^{s'}, (u, u''), \pi_2}$	**Build** $\pi_2 = \mathrm{PoK}\{\alpha, \beta, \gamma : u = h^\alpha$ $\wedge\ u'' = u^\beta (C X_1^{s'})^\alpha \wedge C_{x_0} = g^\beta h^\gamma\}$
Compute $S_1 = Sign(W, g^{s'}, D)$ $s_u \leftarrow s + s'$	$\xrightarrow{S_1}$ $\xleftarrow{s'}$	**Check** S_1 **Save** (W, D, s', S_1) in $\mathtt{DB}_{\mathtt{REG}}$

Figure 5.4: The Token Issuance Protocol

5.5.2.4 Access Control

To be granted access at tollbooths, $\mathcal{U}$ provides a randomized (and blinded) version of his token T and an ElGamal encryption E of g^{s_u} along with a ZKPK π_3 proving that these values are well-formed. Upon their receipt, $\mathcal{TC}$ checks if the token is valid and has not been overspent (*i.e.* used more than $N_{\mathtt{max}}$ times). To this end, whenever reaching a tollbooth, $\mathcal{B}_u$ randomly chooses a g_i among the set F of unused ones. The selected g_i is then removed from F to prevent over-spending of a token. If checks succeed, $\mathcal{U}$ receives an updated token T with a new m aggregating all the refunds collected so far. This new token is computed using our partially blind signature scheme with a common value equal to the current refund amount. Due to delay constraint, the associated ZKPK π_4 cannot be instantly verified. Thus, $\mathcal{U}$ is also provided with S, an RSA signature with a short public verification exponent, of all the received values. S can be quickly verified upon receipt

while π_4 is rather checked during the idle time of the SIM card. Concurrently, $(E, T_i := g_i^{s_u})$ is saved in the database of transactions $\mathtt{DB_{AC}}$. Note that, to provide *full unlinkability*, one may add randomness in T_i using a pseudo-random function as in [ALT+15].

User $\mathcal{U}$		**Toll Company $\mathcal{TC}$**
Public Input: pp, X_1 and X_2 **Private Input:** (u, u'), s_u, m and F		**Public Input:** pp **Private Input:** x_0, x_1 , x_2 **and** $\mathtt{DB_{AC}}$
Choose $l, r, z_1, z_2, t, d \xleftarrow{R} \mathbb{Z}_q^*$ and $g_i \xleftarrow{R} F$ **Compute** $w \leftarrow u^l, w' \leftarrow (u')^l$, $c' \leftarrow w' g^r$ $c_1 \leftarrow w^{s_u} h^{z_1}$, $c_2 \leftarrow w^m h^{z_2}$ $V \leftarrow g^{-r} X_1^{z_1} X_2^{z_2}$, $T_i \leftarrow g_i^{s_u}$ $A \leftarrow h^t X_1^{s_u} X_2^m$, $F \leftarrow F \backslash \{g_i\}$ $E \leftarrow (e_1 = g_E^d, e_2 = g^{s_u} X_T^d)$		
Build $\pi_3 = \text{PoK}\{\alpha, \beta, \gamma, \delta, \sigma, \mu, \eta : T_i = g_i^\sigma$ $\wedge\ c_2 = w^\mu h^\gamma \wedge V = g^{-\alpha} X_1^\beta X_2^\gamma \wedge e_1 = g_E^\eta$ $\wedge c_1 = w^\sigma h^\beta \wedge A = h^\delta X_1^\sigma X_2^\mu \wedge e_2 = g^\sigma X_T^\eta\}$	$\xrightarrow[T_i, A, g_i, \pi_3, E]{w, c', c_1, c_2, V}$	**Check if** $V \stackrel{?}{=} \frac{w^{x_0} c_1^{x_1} c_2^{x_2}}{c'}$ **Check** π_3 and $T_i \notin \mathtt{DB_{AC}}$ **Choose** $d \xleftarrow{R} \mathbb{Z}_q^*$; **Compute** $y \leftarrow h^d$ **Compute** $y'' \leftarrow y^{x_0} (A X_2^{m'})^d$ **Build** $\pi_4 = \text{PoK}\{\alpha, \beta, \gamma : y = h^\alpha$ $\wedge\ y'' = y^\beta (A X_2^{m'})^\alpha \wedge C_{x_0} = g^\beta h^\gamma\}$
Check π_4 and S **Compute** $y' \leftarrow \frac{y''}{y^t}$, $m \leftarrow m + m'$ and $T \leftarrow (y, y')$	$\xleftarrow{m', y, y'', \pi_4, S}$	**Compute** $S = Sign_{RSA}(m', y, y'', \pi_4)$ **Save** (E, T_i) in $\mathtt{DB_{AC}}$

Figure 5.5: The Access Control Protocol

5.5.2.5 Toll Computation

At the end of the billing period, $\mathcal{U}$ shows his randomized token T and a tag $T_g := g_R^{s_u}$ to be charged for all his trips. The tag ensures that $\mathcal{U}$ has not already asked for a refund during that period. Based on the refund amount m, $\mathcal{TC}$ computes the user's charges and saves T_g in $\mathtt{DB_{REG}}$. More precisely, the different exchanges between $\mathcal{U}$ and $\mathcal{TC}$ are depicted in Figure 5.6.

User $\mathcal{U}$		**Toll Company $\mathcal{TC}$**
Public Input: pp and X_1 **Private Input:** $T := (u, u')$, m, s_u		**Public Input:** pp **Private Input:** x_0, x_1 , x_2 **and** $\mathtt{DB_{REG}}$
Choose $l, r, z_1 \xleftarrow{R} \mathbb{Z}_q^*$ **Compute** $R \leftarrow u^l$, $R' \leftarrow (u')^l$ $c' \leftarrow R' g^r$, $c_1 \leftarrow R^{s_u} h^{z_1}$ $V \leftarrow g^{-r} X_1^{z_1}$, $T_g \leftarrow g_R^{s_u}$		
Build $\pi_5 = \text{PoK}[\alpha, \beta, \gamma : T_g = g_R^\alpha$ $\wedge\ c_1 = R^\alpha h^\beta \wedge V = g^{-\gamma} X_1^\beta]$	$\xrightarrow[V,\ T_g, \pi_5]{m, R, c', c_1}$	**Check if** $V \stackrel{?}{=} \frac{R^{x_0 + m x_2} c_1^{x_1}}{c'}$ **Check** π_5 and $T_g \notin \mathtt{DB_{REG}}$ **Save** T_g in $\mathtt{DB_{REG}}$

Figure 5.6: The Toll Computation Protocol

If a user's token has been used less than $N_{\mathtt{max}}$ times, a specific process is triggered so as to emulate their use with no associated charges. Thereby, $\mathcal{U}$ will only pay for the trips he took. Let us consider the case where $N_{\mathtt{max}} = 30$ whereas the user only took 28 trips (g_8 and g_{12} have not been used for example). In such a case, the user will perform a "special" access control where he shows a randomized version of his token and provides the set of tags that have not been used: T_8 and T_{12} in our example. It is noteworthy to mention that this process, which is detailed in Figure 5.7, must be run prior to the execution of the Toll Computation protocol.

5.5.2.6 Revocation

Two different revocations may be triggered under exceptional circumstances: the revocation of user's anonymity or tokens. In the former, the goal is to identify the user who performed a given

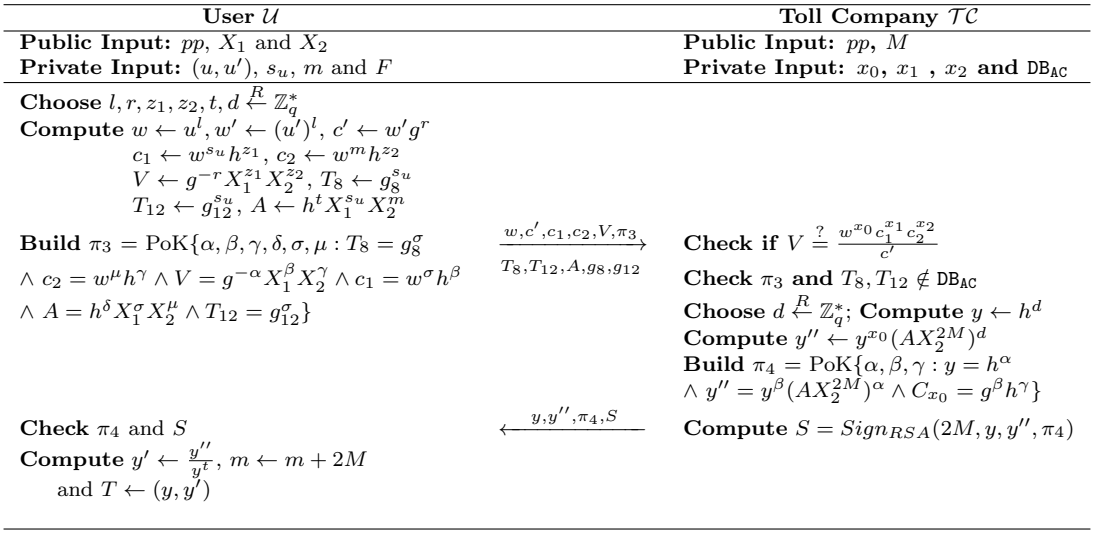

Figure 5.7: The Emulation of the Access Control Protocol

access control (*e.g.* for national security reasons). To do so, $\mathcal{TC}$ sends to $\mathcal{RA}$s the ElGamal encryption E of g^{s_u}. At least t of them should collaborate to recover g^{s_u}. Using the information stored in $\mathtt{DB_{REG}}$, $\mathcal{TC}$ identifies the corresponding user. In the latter, the aim is to revoke a token following, for example, the loss or theft of the user's badge. To this end, $\mathcal{RA}$s are provided with the Paillier encryption D of the secret s that they jointly decrypt. Thereby, $\mathcal{TC}$ can compute $s_u = s + s'$ and thus blacklists all the values $\{T_i := g_i^{s_u}\}_{i=1}^{N_{\max}}$.

5.6 Security Analysis

In this section, we formally prove that our private eCash system satisfies the required security and privacy properties defined in section 5.4. To do so, we will extensively use the following Forking Lemma which applies to signature schemes derived from three-move identification schemes through the use of the Fiat-Shamir technique [FS86] in the ROM.

Forking Lemma. The Forking lemma, which was introduced by PointCheval and Stern [PS00], states that if an adversary $\mathcal{A}$ can produce, with non negligible probability, a valid signature $(m, \sigma_1, h, \sigma_2)$, then $\mathcal{A}$ is able to output, with non negligible probability, two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$.

In our proofs, we use the Forking Lemma to prove that an adversary $\mathcal{A}$ is unable to produce a valid access control transcript $\mathtt{Transc_{AC_i}}$ or a valid Schnorr signature S_1 unless he knows all the underlying secrets, namely (s_u, m, w, w') and sk_u respectively.

5.6.1 Unforgeability

Theorem 5.1 (Unforgeability)**.** *Our eCash system is* unforgeable*, in the random oracle model, under the assumption that* $\mathsf{MAC_{GGM}}$ *is UF-CMVA.*

Proof (sketch). Let $\mathcal{A}$ be an adversary that breaks the unforgeability requirement of our eCash system with non negligible probability. Using $\mathcal{A}$ as a subroutine, we will construct a reduction $\mathcal{B}$ against the UF-CMVA security of the $\mathsf{MAC_{GGM}}$ scheme.
$\mathcal{B}$ receives on input from its challenger, denoted $\mathcal{C}$, the public parameters of the UF-CMVA challenge $pp = (\mathbb{G}, g, h, C_{x_0}, X_1, X_2)$ and has access to the $\mathcal{O}\mathtt{MAC}$ and $\mathcal{O}\mathtt{Verify}$ oracles. As for the remaining generators, they can be randomly selected. $\mathcal{C}$ chooses the private key $\vec{x} = (x_0, x_1, x_2)$

of the toll company whereas $\mathcal{B}$ selects the key pair (sk_{tc}, pk_{tc}). Regarding the key pairs of Paillier and ElGamal encryption schemes (*i.e* (sk_{ra}, pk_{ra}) and (sk_{rp}, pk_{rp})), they can be chosen either by $\mathcal{B}$ or $\mathcal{A}$. $\mathcal{B}$ provides $\mathcal{A}$ with (X_1, X_2, C_{x_0}) and answers its requests as follows:

- $\mathcal{O}\texttt{Register}(ID_{u_i})$: $\mathcal{B}$ randomly chooses the user's secret key $sk_{u_i} \in \mathbb{Z}_q^*$ and computes the associated public key $pk_{u_i} = g^{sk_{u_i}}$ that it transmits to $\mathcal{A}$.
- $\mathcal{O}\texttt{RegisterCorrupt}(ID_{u_i})$: $\mathcal{B}$ does nothing.
- $\mathcal{O}\texttt{Corrupt}(ID_{u_i})$: $\mathcal{B}$ provides $\mathcal{A}$ with the associated private key sk_{u_i}.
- $\mathcal{O}\texttt{TokenIssue}_{TC}()$: Using the replay technique and the soundness property of π_1, $\mathcal{B}$ recovers the secret values s and r. Then, it randomly selects $s' \in_R \mathbb{Z}_q^*$ and calls on the $\mathcal{O}\texttt{MAC}$ oracle on the message $s_u = s + s'$ to obtain the pair (u, u'). Thereby, it can provide $\mathcal{A}$ with the pair $(u, u'' = u'u^r)$ and the corresponding s'. As for the proof π_2, $\mathcal{B}$ can perfectly simulate it in the random oracle model.
- $\mathcal{O}\texttt{AccessControl}_{TC}()$: Using the replay technique and the soundness property of π_3, $\mathcal{B}$ recovers the secrets s_u, m, t, z_1, z_2 and r associated to the values provided by the adversary. It can thus compute the corresponding token (w, w'). Through the call to the $\mathcal{O}\texttt{Verify}$ oracle, $\mathcal{B}$ verifies the validity of the token (w, w'). If (w, w') is a valid MAC on (s_u, m) for an s_u that has not been queried to the $\mathcal{O}\texttt{MAC}$ oracle, then abort (as (w, w', s_u, m) is a valid forgery of $\mathsf{MAC}_{\mathsf{GGM}}$). Otherwise, provided that (w, w') is valid, $\mathcal{B}$ calls on the $\mathcal{O}\texttt{MAC}$ oracle on $(s_u, m + m')$ to get the pair $(y, y' = y^{x_0 + s_u x_1 + (m+m')x_2})$. Thus, it can provide $\mathcal{A}$ with the pair $(y, y'' = y'y^t)$. Regarding the proof π_4, $\mathcal{B}$ can perfectly simulate it in the random oracle model. Since it holds the private key sk_{tc}, $\mathcal{B}$ can produce the RSA signature S. Consequently, $\mathcal{B}$ can perfectly simulate the `AccessControl` protocol for $\mathcal{A}$.
- $\mathcal{O}\texttt{TollCompute}(ID_{u_i})$: Using the replay technique and the soundness property of π_5, $\mathcal{B}$ recovers the secret value r and hence, the token $T = (R, R')$ associated to the values provided by $\mathcal{A}$. Through the call to the $\mathcal{O}\texttt{Verify}$ oracle with T as input, $\mathcal{B}$ verifies the validity of the provided token. If so, it returns the overall toll charges associated to it.

The adversaries against the unforgeability requirement of our eCash system can be classified in two different categories according to the way they proceed to achieve their goals:

- **Type 1:** An adversary that manages to perform an `AccessControl` such that the corresponding transcript view $\texttt{transc}'_{\texttt{AC}}$ cannot be linked to a registered user (*i.e.* $\mathcal{RA}$ is unable to revoke user's anonymity).
- **Type 2:** An adversary that figures out a way to pay less charges than what he has to.

Hereinafter, we show that both forgers can be used to break the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$. As the reduction works differently for the two types of forgers, $\mathcal{B}$ initially chooses $f_{type} \in \{1, 2\}$ at random. The latter corresponds to its guess of the type of forgery $\mathcal{A}$ will output.

If f_{type}=1: Eventually, the adversary $\mathcal{A}$ succeeds with non negligible probability in performing an `AccessControl` such that the associated transcript view $\texttt{transc}'_{\texttt{AC}}$ cannot be linked to a registered user. That is, having $\texttt{transc}'_{\texttt{AC}}$ as input, the protocol `IDRetrieve` returns $\perp$ (*i.e.* an unknown identifier). Two cases are possible:

- *Case 1:* The plaintext $g^{s'_u}$ encrypted in E' is associated to a token generated during a call to the oracle $\mathcal{O}\texttt{TokenIssue}_{TC}$ but no corresponding signature S_1 has been provided. This means that the adversary did not receive the value s', where $s'_u = s + s'$, from the

$\mathcal{O}\texttt{TokenIssue}_{TC}$ oracle. We know from the Forking Lemma and the soundness property of π_3 that $\mathcal{A}$ necessarily knows s'_u as well as s (as he chose it). Hence, he can recover $s' = s'_u - s$. Since the oracle $\mathcal{O}\texttt{TokenIssue}_{TC}$ did not reveal s' but only provided $g^{s'}$, then the adversary $\mathcal{A}$ could be used to extract Discrete Logarithms. Consequently, under the DL assumption, this case can only happen with negligible probability.

- *Case 2:* The plaintext $g^{s'_u}$ encrypted in E' does not correspond to a token that has been issued during an $\mathcal{O}\texttt{TokenIssue}$ request. Thus, s'_u has not been queried to the $\mathcal{O}\texttt{MAC}$ oracle either. Using the Forking Lemma and the soundness property of π_3, $\mathcal{B}$ can extract with non negligible probability the secrets associated to $\texttt{transc}_{\texttt{AC}_\texttt{i}}$, namely (s'_u, m, w, w'). Thus, $\mathcal{B}$ returns to its challenger the pair (w, w'), which corresponds to a valid MAC on (s'_u, m), that has not been obtained following a call to $\mathcal{O}\texttt{MAC}$. Thereby, $\mathcal{B}$ breaks the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme.

If f_{type}=2: Eventually, after n calls to the $\mathcal{O}\texttt{TokenIssue}$ oracle, $\mathcal{A}$ outputs, with non negligible probability, n_v valid tokens $\{T_k\}_{k=1}^{k=n_v}$ such that

$$\sum_{k=1}^{n_v} \texttt{TollCompute}(T_k) < \sum_{j=1}^{\#\texttt{DB}_\texttt{AC}} Charge(\texttt{transc}_\texttt{AC}^\texttt{j})$$

where $Charge(\texttt{transc}_\texttt{AC}^\texttt{j})$ corresponds to the toll charge associated to the transcript view $\texttt{transc}_\texttt{AC}^\texttt{j}$. The following three cases may enable this type of forgery:

- *Case 1:* The adversary figures out a way to use at least one of the issued tokens more than $N_{\texttt{max}}$ times, with non negligible probability. Let $(s_1, s_2, \ldots, s_n)$ denote the n secrets submitted to the $\mathcal{O}\texttt{TokenIssue}$ oracle. Typically, there should be $n \times N_{\texttt{max}}$ distinct pairs (s_{u_i}, g_j) where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, N_{\texttt{max}}\}$. Let L denote the set of legitimate pairs (s_{u_i}, g_j). Thus, there is at least a transcript $\texttt{transc}'_\texttt{AC} \in \texttt{DB}_\texttt{AC}$ whose associated pair $(s_{u'}, g')$ does not belong to L. Since g' necessarily belong to $\{g_j\}_{j=1}^{j=N_{\texttt{max}}}$ (otherwise $\texttt{transc}'_\texttt{AC}$ would not be considered valid), then $s_{u'} \notin \{s_1, s_2, \ldots, s_n\}$. Consequently, $s_{u'}$ has not been queried to the $\mathcal{O}\texttt{TokenIssue}$ oracle. This implies that no call to the $\mathcal{O}\texttt{MAC}$ oracle has had $s_{u'}$ as input neither. In such case, $\mathcal{B}$ picks a random transcript $\texttt{transc}_\texttt{AC}^\texttt{k}$ among the database $\texttt{DB}_\texttt{AC}$. The probability that it selects $\texttt{transc}'_\texttt{AC}$ is $\frac{1}{\#\texttt{DB}_\texttt{AC}}$ where $\#\texttt{DB}_\texttt{AC}$ denotes the number of transcripts in $\texttt{DB}_\texttt{AC}$. Using the Forking Lemma and the soundness property, $\mathcal{B}$ can extract with non negligible probability the secrets associated to $\texttt{transc}'_\texttt{AC}$, namely $(s_{u'}, m, w, w')$. Thus, $\mathcal{B}$ returns to its challenger the pair (w, w'), which corresponds to a valid MAC on $(s_{u'}, m)$, that has not been obtained following a call to $\mathcal{O}\texttt{MAC}$. Thereby, $\mathcal{B}$ breaks the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme with non negligible probability.

- *Case 2:* At least one of the transcript views $\texttt{transc}'_\texttt{AC}$ corresponding to the adversary's calls to $\mathcal{O}\texttt{AccessControl}$ did not involve a token issued following a call to $\mathcal{O}\texttt{TokenIssue}$. This case is similar to a type-1 case-2 adversary which can be used to break the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme.

- *Case 3:* At least one of the n_v valid tokens, $T^* = (u^*, u'^*)$, provided by $\mathcal{A}$ was not issued following a call to the $\mathcal{O}\texttt{AccessControl}$ or $\mathcal{O}\texttt{TokenIssue}$ protocols (*i.e.* the adversary has succeeded in forging a valid token). Let s_{u^*} and m^* be respectively the secret and the refund amount associated to T^*. This case implies that no call to the $\mathcal{O}\texttt{MAC}$ oracle has been made by $\mathcal{B}$ with the pair (s_{u^*}, m^*) as input. Thus, $\mathcal{B}$ picks a random token T_k among the set of valid tokens provided by $\mathcal{A}$. The probability that it selects T^* is $\frac{1}{n_v}$. Using the Forking Lemma and the soundness property, $\mathcal{B}$ can extract with non negligible probability the secrets associated to T^*, namely (s_{u^*}, m^*, u, u'). Thus, $\mathcal{B}$ returns to its challenger the pair (u, u'), which corresponds to a valid MAC on (s_{u^*}, m^*), that has not been issued during

a call to $\mathcal{O}\texttt{MAC}$. Thereby, $\mathcal{B}$ breaks the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme with non negligible probability.

□

5.6.2 Non-frameability

Theorem 5.2 (Non-frameability)**.** *Our eCash system provides the* non-frameability *property, in the random oracle model, under the discrete logarithm* DL *assumption.*

Proof (sketch). Let $\mathcal{A}$ be an adversary against the non-frameability requirement of our eCash system. Using $\mathcal{A}$, we construct a reduction $\mathcal{B}$ against the discrete logarithm (DL) assumption. Depending on the way they proceed to break the non-frameability property of our system, we can distinguish the following two types of adversaries:

- **Type 1:** An adversary that is able to recover the secret s_u associated to a token that was issued following a call to $\mathcal{O}\texttt{TokenIssue}_U$.
- **Type 2:** An adversary that succeeds in generating a valid Schnorr signature S_1 on behalf of an honest user. Thereby, he can impersonate him and obtain as many valid tokens as he wants to.

The reduction works differently for each kind of adversary. Therefore, $\mathcal{B}$ initially chooses a value $\mathcal{A}_{type} \in \{1, 2\}$ at random. The latter corresponds to its guess of the way $\mathcal{A}$ will break the non-frameability of our system (i.e. the type of adversary).

If $\mathcal{A}_{type}$=1: This means that the adversary is able to recover the secret s_u associated to a token that was issued following a call to $\mathcal{O}\texttt{TokenIssue}_U$. In such case, we assume that $\mathcal{B}$ (which acts as $\mathcal{A}$'s challenger) receives on input from its challenger $\mathcal{C}$ the pair (g, g^s) where g is a random generator of $\mathbb{G}$. The adversary $\mathcal{A}$ chooses the private keys $\vec{x} = (x_0, x_1, x_2)$ of the toll company as well as ElGamal pair of keys (sk_{ra}, pk_{ra}) whereas $\mathcal{B}$ selects the Paillier pair of keys (sk_{rp}, pk_{rp}). $\mathcal{B}$ also picks $(N_{\texttt{max}} + 1)$ random values $(\alpha, \beta_1, \ldots \beta_{N_{\texttt{max}}}) \in_R \mathbb{Z}_q^*$ and sets $h = g^\alpha$ and $g_i = g^{\beta_i}, \forall i \in \{1, \ldots, N_{\texttt{max}}\}$. The other generators are randomly chosen. $\mathcal{B}$ constructs the public parameters pp for $\mathcal{A}$ and answers its requests as follows:

- $\mathcal{O}\texttt{Register}(ID_{u_i})$: $\mathcal{B}$ randomly selects the user's secret key sk_{u_i} and computes the associated public key pk_{u_i}.
- $\mathcal{O}\texttt{RegisterCorrupt}(ID_{u_i})$: $\mathcal{B}$ does nothing.
- $\mathcal{O}\texttt{Corrupt}(ID_{u_i})$: $\mathcal{B}$ provides $\mathcal{A}$ with the corresponding private key sk_{u_i}.
- $\mathcal{O}\texttt{TokenIssue}_U(ID_{u_i})$: $\mathcal{B}$ first selects two random values δ_i and λ_i then, using its input g^s of the DL problem, it sets $W = g^{\delta_i s + \lambda_i} = g^{s_i}$. The values $C \in_R \mathbb{Z}_q^*$ and $D \in_R \mathbb{Z}_{n^2}$ can be randomly chosen. Indeed, $\forall s_i, \exists\ r, r_1$ such that $C = h^r X_1^{s_i}$ and $D = g_p^{s_i} r_1^n$. As for π_1 and S_1, $\mathcal{B}$ can perfectly simulate them in the random oracle model. If the protocol does not abort, $\mathcal{B}$ obtains a valid token such that the associated s_{u_i} is defined as $s_{u_i} = s_i + s'_i$ where s'_i is randomly chosen by $\mathcal{A}$ (and also known to $\mathcal{B}$) while $s_i = \delta_i s + \lambda_i$ is unknown to both $\mathcal{B}$ and $\mathcal{A}$.
- $\mathcal{O}\texttt{AccessControl}_U(ID_{u_i}, v)$: As it holds (u, u') and knows m, $\mathcal{B}$ can easily simulate the different values involved in the execution of the `AccessControl` protocol except $c_1 = w^{s_{u_i}} h^{z_1}$, $T_i = g_i^{s_{u_i}}$ and $A = h^t X_1^{s_{u_i}} X_2^m$. More precisely, it randomly chooses $l, z_1, z_2, r, d \in_R \mathbb{Z}_q$ and computes $c' = (u')^l g^r$, $V = g^{-r} X_1^{z_1} X_2^{z_2}$ as well as $c_2 = w^m h^{z_2}$, $e_1 = g_E^d$ and $e_2 = g^{s_i} g^{s'_i} X_T^d$.

As regards to A, it is randomly generated since for all (s_{u_i}, m), there exists t such that $A = h^t X_1^{s_{u_i}} X_2^m$. As for the other values, they are computed as $c_1 = ((g^{s_i})^{\alpha} h^{s'_i})^{bl} h^{z_1} = w^{s_{u_i}} h^{z_1}$ and $T_i = (g^{s_i} g^{s'_i})^{\beta_i} = g_i^{s_{u_i}}$ where b is recovered from π_2 using the replay technique and the soundness property. Regarding the PoK π_3, $\mathcal{B}$ can perfectly simulate it using standard techniques.

Eventually, $\mathcal{A}$ outputs with non negligible probability a valid transcript $\texttt{transc}'_{\texttt{AC}_\texttt{i}}$ of an execution of the `AccessControl` protocol such that, on input $\texttt{transc}'_{\texttt{AC}_\texttt{i}}$, the `IDRetrieve` protocol outputs an identifier $ID_{u_i} \in \mathcal{HU}$ along with a Schnorr signature S_1 proving that a token issued for the user ID_{u_i} was involved in the generation of $\texttt{transc}'_{\texttt{AC}_\texttt{i}}$. As defined in the experiment, $\texttt{transc}'_{\texttt{AC}_\texttt{i}}$ must not have produced following a call to $\mathcal{O}\texttt{AccessControl}_U(ID_{u_i})$.

It follows from the Forking Lemma that if $\mathcal{A}$ is able to produce such a valid transcript $\texttt{Transc}'_{\texttt{AC}}$, then it can produce two valid transcripts $\texttt{Transc}'_{\texttt{AC}_1}$ and $\texttt{Transc}'_{\texttt{AC}_2}$. Thereby, using the replay technique and the soundness property of π_3, $\mathcal{B}$ can extract the associated secret values (s_{u_i}, m, u, u'). Given s_{u_i} and s'_i, $\mathcal{B}$ retrieves $s_i = s_{u_i} - s'_i$. Since it knows the values of δ_i and λ_i, it can recover s the discrete logarithm of the received challenge g^s, hence breaking the DL assumption.

If $\mathcal{A}_{type}$=2: We are dealing with an adversary that succeeds in generating a valid Schnorr signature S_1 on behalf of an honest user. Thus, he can impersonate him and obtain as many valid tokens as he wants to. To get a better reduction, we use the one-more discrete logarithm OMDL assumption. It is, however, worth mentioning that the non-frameability property of our eCash system can also be proven under the DL assumption. We assume that the challenger $\mathcal{C}$ provides $\mathcal{B}$ with a random instance $(g^{sk_1}, g^{sk_2}, \ldots, g^{sk_n})$ of the OMDL problem where g is a random generator of $\mathbb{G}$. As it is against the one-more DL assumption, $\mathcal{B}$ has access to a DL oracle and, similarly to the previous case, it acts as $\mathcal{A}$'s challenger in the non-frameability experiment. $\mathcal{A}$ chooses the private keys $\vec{x} = (x_0, x_1, x_2)$ of the toll company while ElGamal and Paillier pair of keys can be chosen either by $\mathcal{B}$ or $\mathcal{A}$. $\mathcal{B}$ constructs the public parameters pp for $\mathcal{A}$ and answers its requests as follows:

- $\mathcal{O}\texttt{Register}(ID_{u_i})$: $\mathcal{B}$ answers $\mathcal{A}$'s request using his input of the one-more discrete logarithm problem. More precisely, it sets the user public key as $pk_{u_i} = g^{sk_i}$.

- $\mathcal{O}\texttt{RegisterCorrupt}(ID_{u_i})$: $\mathcal{B}$ does nothing.

- $\mathcal{O}\texttt{Corrupt}(ID_{u_j})$: $\mathcal{B}$ calls on the DL oracle with pk_{u_j} as input. Thereby, it can provide $\mathcal{A}$ with the corresponding private key sk_j.

- $\mathcal{O}\texttt{TokenIssue}_U(ID_{u_i})$: $\mathcal{B}$ randomly selects $s \in_R \mathbb{Z}_q^*$ and computes the required values involved in the execution of the `TokenIssue` protocol. As regards to the proof π_1 and the Schnorr signature S_1, $\mathcal{B}$ can perfectly simulate them in the random oracle model.

- $\mathcal{O}\texttt{AccessControl}_U(ID_{u_i}, v)$: $\mathcal{B}$ holds the token (u, u') issued for ID_{u_i} as well as the associated secrets s_{u_i} and m. Thus, it can perfectly simulate the different values involved in the execution of the `AccessControl` protocol.

Eventually, after d calls to the $\mathcal{O}\texttt{Corrupt}$ oracle, the adversary $\mathcal{A}$ outputs a valid transcript view $\texttt{Transc}'_{\texttt{AC}}$ of the execution of the `AccessControl` protocol such that it breaks the non-frameability requirement of our eCash system. We know by the definition of the experiment that the corresponding user is honest. Consequently, the oracle $\mathcal{O}\texttt{Corrupt}$ has not been queried with this user's identifier as input. This implies that the associated public key $pk_{u_i} = g^{sk_i}$ is still in the input of the OMDL problem. As previously mentioned, this type of adversary succeeds if he manages to produce a valid Schnorr signature S_1, enabling him to get a valid payment token.

It follows from the Forking Lemma that if $\mathcal{A}$ is sufficiently powerful to produce a Schnorr signature S_1 then, it can produce two Schnorr signatures with non-negligible probability. Using the replay technique, $\mathcal{B}$ is able to extract the private key sk_i used to generate the Schnorr signature. Thereby, $\mathcal{B}$ outputs sk_i along with the d secret keys $\{sk_{u_j}\}_{j=1}^{j=d}$ that it has received by querying the DL oracle. Hence, $\mathcal{B}$ breaks the one-more discrete logarithm problem.

□

5.6.3 Unlinkability

Theorem 5.3 (Unlinkability)**.** *Our private eCash system is* unlinkable*, in the random oracle model, under the Decisional Composite Residuosity* DCR*, the Decisional Diffie Hellman* DDH *and the q-Decisional Diffie-Hellman Inversion* $q-$DDHI *assumptions.*

Proof (sketch). In the sequel, we prove the *unlinkability* of our eCash system under a slightly weaker model than the one presented in 5.4.2. More precisely, we consider the same experiment except that, during the challenge phase, the adversary $\mathcal{A}$ is not provided access to the $\mathcal{O}\texttt{IDRetrieve}$ oracle. The corresponding requirement is denoted *Unlinkability'*. It should be stressed, however, that in the case of a real deployment, the access to the revocation functionalities is carefully controlled. Consequently, *Unlinkability'* is rather a reasonable requirement.

Nevertheless, it is worth mentioning that our eCash system would satisfy the original *unlinkability* requirement provided that we replace the ElGamal encryption scheme, which is only IND-CPA secure, by an IND-CCA2 secure encryption scheme. As it is well-known, the double encryption of the same plaintext under an IND-CPA encryption scheme and the use of a simulation sound proof of equality of plaintexts lead to an IND-CCA2 encryption scheme [NY90]. Thereby, if we use the double ElGamal encryption scheme which was proven IND-CCA2 [FP01] instead of the classical one, our system would fulfil the original *unlinkability* requirement.

Furthermore, we consider the case where the different generators $\{g_i\}_{i=1}^{i=N_{\texttt{max}}}$ are computed as the output of a pseudo-random function introduced by Dodis and Yampolskiy [DY05]. In particular, for a user holding a token associated to a secret s_u, they are defined as $g_i = g^{\frac{1}{s_u+i+1}}$, $\forall i \in \{1, \ldots, N_{\texttt{max}}\}$. With such an assumption, an adversary can have access to the $\mathcal{O}\texttt{TollCompute}$ oracle regardless of the set of generators used (during the challenge phase) when querying the $\mathcal{O}\texttt{AccessControl}_U$ oracle on u_0 and u_1. Thus, $\mathcal{A}$'s access to the $\mathcal{O}\texttt{TollCompute}$ oracle during the challenge phase is henceforth restricted to the cases where $v_b = v_{1-b}$, the number of calls to $\mathcal{O}\texttt{AccessControl}_U$ on u_0 and u_1 are equal, and the overall toll charges associated to all these calls are the same for u_0 and u_1.

Let $\mathcal{A}$ be an adversary against the unlinkability requirement of our eCash system. $\mathcal{A}$ is said to be a Type-(i, j) distinguisher where $i \leq N_{\texttt{max}} - 1$ and $j \leq N_{\texttt{max}} - 1$ if, during the challenge phase, it makes at most i calls to $\mathcal{O}\texttt{AccessControl}_U$ on u_0 and j calls to $\mathcal{O}\texttt{AccessControl}_U$ on u_1. It is clear that a Type-(i, j) distinguisher is, in particular, a Type-$(N_{\texttt{max}} - 1, N_{\texttt{max}} - 1)$ distinguisher $\forall i \leq N_{\texttt{max}} - 1$ and $\forall j \leq N_{\texttt{max}} - 1$. Thus, without loss of generality, we may focus solely on Type-$(N_{\texttt{max}} - 1, N_{\texttt{max}} - 1)$ adversaries.

So, in the sequel, we assume that $\mathcal{A}$ is a Type-$(N_{\texttt{max}} - 1, N_{\texttt{max}} - 1)$ adversary. Using Shoup's game hopping technique [Sho04], where proofs are organized as sequences of games, we prove that our eCash system satisfies the *Unlinkability'* requirement. Hereinafter, we provide a high level description of the initial game (Game 0) along with brief descriptions of the transitions between successive games.

Game 0: It corresponds to the real attack game with respect to an efficient adversary $\mathcal{A}$.

The Challenger $\mathcal{C}$ randomly chooses the system public parameters $pp = (\mathbb{G}, g, h, q, g_R, N_{\texttt{max}})$, the keys $\vec{x} = (x_0, x_1, x_2)$ and sk_{tc} of the toll company as well as both revocation authorities pair

of keys (*i.e.* (sk_{ra}, pk_{ra}) and (sk_{rp}, pk_{rp})). $\mathcal{C}$ provides $\mathcal{A}$ with both $\vec{x} = (x_0, x_1, x_2)$ and sk_{tc}, and answers $\mathcal{A}$'s requests as follows:

- $\mathcal{O}\texttt{Register}(ID_{u_i})$: $\mathcal{C}$ randomly chooses $sk_{u_i} \in \mathbb{Z}_q^*$ and computes the public key $pk_{u_i} = g^{sk_{u_i}}$ that it forwards to $\mathcal{A}$.
- $\mathcal{O}\texttt{RegisterCorrupt}(ID_{u_i})$: $\mathcal{C}$ does nothing.
- $\mathcal{O}\texttt{Corrupt}(ID_{u_i})$: $\mathcal{C}$ provides $\mathcal{A}$ with the secret key sk_{u_i}.
- $\mathcal{O}\texttt{TokenIssue}_U(ID_{u_i})$: $\mathcal{C}$ chooses a random value $s_i \in \mathbb{Z}_q^*$ and proceeds normally to obtain a token on $s_{u_i} = s_i + s_i'$. To generate the Schnorr signature S_1, it uses the secret key sk_{u_i}.
- $\mathcal{O}\texttt{AccessControl}_U(ID_{u_i}, v)$: $\mathcal{C}$ uses the token issued during $\mathcal{O}\texttt{TokenIssue}_U(ID_{u_i})$ and the associated secrets s_{u_i} and m, as well as a generator g_i (that has not already been used) to perform an access control whose corresponding toll fare is v.
- $\mathcal{O}\texttt{TollCompute}(ID_{u_i})$: $\mathcal{C}$ returns the overall toll charges that ID_{u_i} owes to $\mathcal{TC}$.

The adversary $\mathcal{A}$ selects two honest users u_0 and u_1 as well as two toll fares v_0 and v_1 that he returns to the challenger $\mathcal{C}$. The latter first runs the `TokenIssue` protocol with u_0 and u_1 as input and provides the corresponding transcript views, $\texttt{transc}_{\texttt{TI}_0}$ and $\texttt{transc}_{\texttt{TI}_1}$ to $\mathcal{A}$. Then, $\mathcal{C}$ picks a random bit $b \in \{0,1\}$ and runs the `AccessControl` protocol with respectively (u_1, v_b) and (u_0, v_{1-b}) as input. The corresponding transcripts $\texttt{transc}_\texttt{b}$ and $\texttt{transc}_{\texttt{1-b}}$ are also sent to $\mathcal{A}$ whose goal is to guess the value of b (*i.e.* figure out which user performed the `AccessControl` that costs v_0 (or v_1) with a probability non negligibly better than $\frac{1}{2}$). After receiving the challenge (*i.e.* during the challenge phase), $\mathcal{A}$ can still query the $\mathcal{O}\texttt{Register}, \mathcal{O}\texttt{RegisterCorrupt}, \mathcal{O}\texttt{Corrupt}$, $\mathcal{O}\texttt{TokenIssue}_U, \mathcal{O}\texttt{AccessControl}_U, \mathcal{O}\texttt{TokenRevoc}$ and $\mathcal{O}\texttt{TollCompute}$ oracles but with some restrictions. More precisely, $\mathcal{A}$ can neither query the $\mathcal{O}\texttt{Corrupt}$ oracle on ID_{u_0} or ID_{u_1}, nor the $\mathcal{O}\texttt{TokenRevoc}$ oracle on $(ID_{u_0}, \texttt{transc}_{\texttt{TI}_0})$ or $(ID_{u_1}, \texttt{transc}_{\texttt{TI}_1})$. Moreover, $\mathcal{A}$'s access to the $\mathcal{O}\texttt{TollCompute}$ oracle on ID_{u_0} and ID_{u_1} is restricted to the cases where $v_b = v_{1-b}$, the number of calls to $\mathcal{O}\texttt{AccessControl}_U$ on u_0 and u_1 are equal, and the overall toll charges associated to all these calls are the same for u_0 and u_1. Otherwise, the adversary could easily win the game.

Eventually, $\mathcal{A}$ outputs a bit b'. Let S_0 denote the event that $b = b'$ in Game 0 (*i.e.* the event that $\mathcal{A}$ wins Game 0) and S_i denote the event that $b = b'$ in Game i. We have $\texttt{Adv}_{\mathcal{A}}^{\text{unlink}-b}(1^k) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{unlink}-b}(1^k) = b] - 1/2| = |\texttt{Pr}[S_0] - 1/2|$.

We construct the following sequence of games while preventing the adversary from detecting the changes between two successive games:

- ◇ **Game(0,b)**: It is the same game as **Game 0** except that, when running the `AccessControl` protocol with u_b, we replace the ElGamal ciphertext E_b by an encryption of a random value and simulate the proof π_3^b. Such a proof can be easily simulated in the random oracle model through the use of classical techniques. Under the DDH assumption, the adversary cannot detect this change. In fact, one can easily construct a DDH distinguisher $\mathcal{D}$ with an advantage of solving the DDH problem, $\texttt{Adv}_{\texttt{DDH}}$, satisfying $|\texttt{Pr}[S_0] - \texttt{Pr}[S_{(0,b)}]| \leq \texttt{Adv}_{\texttt{DDH}}(1^k)$.
- ◇ **Game(0,1-b)**: It is the same game as **Game (0,b)** except that, when executing the `AccessControl` protocol with u_{1-b}, we replace the ElGamal ciphertext E_{1-b} by an encryption of a random value. Then, we simulate the proof π_3^{1-b}. Such a proof can be easily simulated in the random oracle model through the use of standard techniques. Under the DDH assumption, the adversary cannot detect this change. In fact, one can easily construct a DDH distinguisher $\mathcal{D}$ with an advantage of solving the DDH problem, $\texttt{Adv}_{\texttt{DDH}}$, satisfying $|\texttt{Pr}[S_{(0,b)}] - \texttt{Pr}[S_{(0,1-b)}]| \leq \texttt{Adv}_{\texttt{DDH}}(1^k)$.

- ◇ **Game(0,1-b,b)**: It is the same game as **Game (0,1-b)** except that, when executing the `AccessControl` protocol with u_b, we replace the value of the tag T_i^b by a random value. Then, we simulate the proof π_3^b. Such a proof can be easily simulated in the random oracle model through the use of classical techniques. Under the $q-\mathsf{DDHI}$ assumption, the adversary cannot detect such change. In fact, one can easily construct a $q-\mathsf{DDHI}$ distinguisher $\mathcal{D}$ with an advantage of solving the $q-\mathsf{DDHI}$ problem, denoted by $\mathtt{Adv}_{q-\mathsf{DDHI}}$, that satisfies $|\mathtt{Pr}[S_{(0,1-b)}] - \mathtt{Pr}[S_{(0,1-b,b)}]| \leq \mathtt{Adv}_{q-\mathsf{DDHI}}(1^k)$.

- ◇ **Game(0,1-b,1-b)**: It is the same game as **Game (0,1-b,b)** except that, when executing the `AccessControl` protocol with u_{1-b}, we replace the tag T_i^{1-b} by a random value. Then, we simulate the proof π_3^{1-b}. Such a proof can be easily simulated in the random oracle model through the use of standard techniques. Under the $q-\mathsf{DDHI}$ assumption, the adversary cannot detect this change. In fact, one can easily construct a $q-\mathsf{DDHI}$ distinguisher $\mathcal{D}$ with an advantage of solving the $q-\mathsf{DDHI}$ problem, denoted by $\mathtt{Adv}_{q-\mathsf{DDHI}}$, satisfying $|\mathtt{Pr}[S_{(0,1-b,b)}] - \mathtt{Pr}[S_{(0,1-b,1-b)}]| \leq \mathtt{Adv}_{q-\mathsf{DDHI}}(1^k)$.

Let **Game 1** be the same game as **Game(0,1-b,1-b)**. It results from the above that

$$|\mathtt{Pr}[S_0] - \mathtt{Pr}[S_1]| \leq 2 \times (\mathtt{Adv}_{DDH}(1^k) + \mathtt{Adv}_{q-\mathsf{DDHI}}(1^k))$$

We proceed similarly for each of the adversary's calls to the $\mathcal{O}\mathtt{AccessControl}_U$ protocol on ID_{u_0} and ID_{u_1}. Thereby, for $k = 1$ to $(N_{\mathtt{max}} - 1)$, we construct the following sequence of games:

- ◇ **Game(k,b)**: It is the same as **Game k = Game (k-1,1-b,1-b)** except that, when executing the `AccessControl` protocol with u_b, we replace the ElGamal ciphertext E_b^k by an encryption of a random value and simulate the proof $\pi_3^{b_k}$.

- ◇ **Game(k,1-b)**: It is the same game as **Game (k,b)** except that, when executing the `AccessControl` protocol with u_{1-b}, we replace the ElGamal ciphertext E_{1-b}^k by an encryption of a random value and simulate the proof $\pi_3^{(1-b)_k}$.

- ◇ **Game(k,1-b,b)**: It is the same game as **Game (k,1-b)** except that, when executing the `AccessControl` protocol with u_b, we replace the value of $T_i^{b_k}$ by a random value and simulate the proof $\pi_3^{b_k}$.

- ◇ **Game(k,1-b,1-b)**: It is the same game as **Game (k,1-b,b)** except that, when executing the `AccessControl` protocol with u_{1-b}, we replace $T_i^{(1-b)_k}$ by a random value and simulate the proof $\pi_3^{(1-b)_k}$.

Let **Game k+1** be the same game as **Game(k,1-b,1-b)**. It results from the above that

$$|\mathtt{Pr}[S_{k+1}] - \mathtt{Pr}[S_k]| \leq 2 \times (\mathtt{Adv}_{DDH}(1^k) + \mathtt{Adv}_{q-\mathsf{DDHI}}(1^k))$$

Following the same reasoning, **Game** $N_{\mathtt{max}} - 1$ is the same game as **Game(**$N_{\mathtt{max}} - 2$**,1-b,1-b)**. However, in the latter game, the challenger $\mathcal{C}$ provides no information (in a strong information theoretic sense) about the bit b to the adversary $\mathcal{A}$. This is due to the fact that all values involving the user's secret s_{u_i} and that are not computed in a perfectly hiding way (*i.e.* ElGamal ciphertext and the tag T_i) have been replaced by random ones. Thus, $\mathtt{Pr}[S_{N_{\mathtt{max}}-1}] = 1/2$.
Therefore, we can set an upper bound for the adversary's advantage $\mathtt{Adv}_{\mathcal{A}}^{\text{unlink}-b}(1^k)$. In fact,

$$\mathtt{Adv}_{\mathcal{A}}^{\text{unlink}-b}(1^k) = |\mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{\text{unlink}-b}(1^k) = b] - 1/2| = |\mathtt{Pr}[S_0] - \mathtt{Pr}[S_{N_{\mathtt{max}}-1}]|.$$

It results from the above that:

$$|\mathtt{Pr}[S_0] - \mathtt{Pr}[S_{N_{\mathtt{max}}-1}]| \leq \textstyle\sum_{j=0}^{N_{\mathtt{max}}-1} |\mathtt{Pr}[S_j] - \mathtt{Pr}[S_{j+1}]| \leq 2N_{\mathtt{max}} \times (\mathtt{Adv}_{DDH}(1^k) + \mathtt{Adv}_{q-\mathsf{DDHI}}(1^k))$$

Thus, we can conclude that under the DDH, $q-$DDHI and DCR assumptions, the advantage of a Type-$(N_{\max}-1, N_{\max}-1)$ adversary is negligible. Consequently, so is the advantage of any Type-(i,j) adversary where $i \leq N_{\max}-1$ and $j \leq N_{\max}-1$.

Thereby, our private eCash system satisfies the *unlinkability'* requirement, in the random oracle model, under the DDH, $q-$DDHI and DCR assumptions. □

5.7 Performance Assessment

To show that our private eCash system is not only secure and privacy-friendly but also efficient, we implemented the user's side of the *Access Control*, which is the most time critical phase, on an NFC-enabled SIM Card. In what follows, we provide more details about our testbed environment and the obtained timing results.

5.7.1 Testbed Environment

The user's side of the *Access Control* protocol was implemented on a Javacard 2.2.2 SIM card, GlobalPlatform 2.2 compliant, which is embedded in a Samsung galaxy S3 NFC smartphone. The only particularity of the used SIM card, compared to the Javacard specifications, is some additional API provided by the card manufacturer enabling modular and elliptic curve operations. Even though the used SIM card is more powerful than most SIM cards, as it requires a cryptoprocessor to be able to handle asymmetric cryptography, it is worth emphasizing that such powerful SIM cards with cryptoprocessors are already widely deployed by some mobile phone carriers, such as Orange in France, to provide NFC-based services.

The toll company ($\mathcal{TC}$) side of the *Access Control* protocol was run on a PC (Quad-Core Intel Xeon CPU @3.70GHz). Since our private eCash system is not only intended for eToll but also for EVC and public transport, communications between the SIM card in the smartphone and the PC (acting as $\mathcal{TC}$) was done in NFC using a standard PC/SC reader (an Omnikey 5321).

The implementation uses a 256-bit Barreto-Naehrig elliptic curve [BN06] which is of embedded degree $k = 12$, and hence is not prone to the MOV attack [MOV93]. To have the fastest possible verification on card, $\mathcal{TC}$ uses a 2048 bits RSA signature with a short public verification exponent.

5.7.2 Implementation Benchmarks

	Battery-On	**Battery-Off**
(1) Card pre-computation	(1672-1688) 1678	
(2) Get data from card	(66-68) 67	85
(3) $\mathcal{TC}$ computations	(9-34) 22	
(4) Send data to card	(96-115) 102	(175-184) 182
(5) Card verification	(501-522) 511	
Total On-line part	(186-224) **205** *ms*	(298-322) **315** *ms*

Table 5.1: Timings ((min-max) average) in ms of the *Access Control* Protocol.

Table 5.1 above gives timing results of the implementation of the *Access Control* protocol. "Battery-Off" denotes a powered-off mobile phone either by the user or because its battery is flat. In such a case, as stated by NFC standards, NFC-access to the SIM card is still possible, but with slightly degraded performances. During *Access Control*, the off-line computations (steps 1 and 5) are launched from the smartphone (battery-on) whereas on-line computations concern steps 2, 3 and 4. The latter can potentially be done with a battery-off smartphone. On average,

the on-line part of the *Access Control* protocol is very fast, even with a powered-off phone. In fact, data exchange is the most time-consuming task.

5.8 Conclusion

In this chapter, using our partially blind signature scheme introduced in Section 4.1 as the main building block, we designed a private eCash system that only requires users to hold a unique reusable token while preserving their privacy. Through its refund process, our proposal supports flexible prices as well as post-payments. Furthermore, we have formally proven that it meets the required security properties. Finally, we showed its efficiency even when implemented on constrained environments such as SIM cards. More precisely, a transaction can be performed in 205 ms when the smartphone is switched on, and 315 ms otherwise.

In the next chapter, we rely on our algebraic $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme introduced in Section 4.2 so as to build an efficient Keyed-Verification Anonymous Credentials (KVAC) systems that can be easily turned into a traditional public-key anonymous credential system.

Chapter 6

A Practical Keyed-Verification Anonymous Credentials System

Contents

In this chapter, based on our algebraic MAC scheme $\mathsf{MAC}^n_{\mathsf{BB}}$ introduced in Section 4.2, we propose a practical Keyed-Verification Anonymous Credentials (KVAC) system that provides multi-show unlinkability of credentials and whose presentation proof is of complexity $O(1)$ in the number of group elements. Through slight modifications (solely on the verifier side), our KVAC system can be easily turned into a public-key credentials system so that anyone can verify credentials validity. Our proposal is almost as efficient as Microsoft's U-Prove anonymous credentials (which does not ensure multi-show unlinkability) and many times faster than IBM's Idemix while being suitable for resource constrained environments such as SIM cards. To show its efficiency and suitability for real-world use cases, we implemented it on a standard NFC SIM

card. In particular, the presentation of a credential with 3 attributes can be performed in only 88 *ms*. This work has resulted in the publication "Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials" [BBDT16] at SAC 2016.

6.1 Anonymous Credentials

Introduced by Chaum [Cha85], anonymous credentials systems allow users to obtain a credential from an issuer and then, later, prove possession of this credential, in an unlinkable way, without revealing any additional information. This primitive has attracted a lot of interest as it complies with data minimization principles that consist in preventing the disclosure of irrelevant and unnecessary information. Typically, an anonymous credentials system is expected to enable users to reveal a subset of the attributes associated to their credentials while keeping the remaining ones hidden. For instance, a service provider only needs to know that a user is legitimate (*i.e.* he is authorized to access the service) without yet being able to collect personal information such as address, date of birth, etc.

Potential applications of anonymous credentials are numerous, including e-cash [HZB+13], public transport and electronic toll (for authentication purposes). In such applications, the system efficiency is an important requirement especially as it is usually deployed on constrained environments like smart cards.

Furthermore, it is desirable that an anonymous credentials system provides multi-show unlinkability. That is, one can prove possession of the same credential several times in an unlinkable manner. However, when it is intended for eCash applications, credentials should be one-show to prevent double spending of coins.

In this chapter, we aim to design an anonymous credentials system that provides multi-show unlinkability while being both efficient and suitable for resource constrained environments like SIM cards (that cannot handle pairing computations). To this end, we rely on our algebraic MAC scheme introduced in 4.2 to construct a practical Keyed-Verification Anonymous Credentials (KVAC) system whose presentation proof is of complexity $O(1)$ in the number of group elements. Next, we explain how to turn our KVAC system into a public key credential system through the use of pairings *solely* on the verifier side. Then, we provide efficiency and complexity evaluations as well as timings results of the implementation of our proposal on an NFC SIM card. Finally, we formally prove the security of our KVAC system under classical assumptions.

6.2 Related Work

One of the most prevalent anonymous credentials systems is Microsoft's U-Prove [Paq13, PZ13] which is based on a blind signature scheme due to Brands [Bra94]. It is quite efficient, as it works in prime-order groups, and supports the selective disclosure of attributes. Nevertheless, U-Prove does not provide multi-show unlinkability unless the user uses a different credential at each proof of possession. Besides, to date, its security has not been formally proven.

A slightly less efficient anonymous attribute-based credentials system has been proposed by Baldimsti *et al.* [BL13]. Their proposal, which relies on an extension of Abe's blind signature scheme [Abe01], is proven secure in the Random Oracle Model (ROM) under the DDH assumption. Recently, Fuchsbauer *et al.* [FHS15] introduced another anonymous credentials system that is proven secure in the standard model. However, similarly to U-Prove, both systems are one-show (*i.e.* credential presentations are linkable if a credential is used more than once).

IBM's Identity Mixer, commonly known as Idemix [IBM10], is built on Camenisch-Lysyanskaya (CL) signature scheme [CL01, CL03]. Unlike previously reviewed credentials systems, Idemix credentials provide multi-show unlinkability but at the cost of a less efficient proof of possession.

Indeed, the used CL signatures are based on the Strong RSA assumption [BP97]. This implies large RSA parameters which make Idemix unsuitable for constrained devices. Despite this, Vullers *et al.* focused in [VA13] on the implementation of Idemix on MULTOS smart cards. Using a 1024-bit modulus, their implementation enables the presentation of a credential with three attributes, one of which is undisclosed, in 1 second. Moreover, Piedra *et al.* [dlPHV14] addressed smart cards limited Random Access Memory (RAM) issues by proposing a RAM-efficient implementation of Idemix. Thereby, smart cards can support Idemix credentials with more than 5 attributes. Unfortunately, even with these implementation improvements, timing results far exceed the time constraints of some use cases, which limits the use of Idemix in practice.

Camenisch and Lysyanskaya introduced in [CL04] an efficient signature scheme defined in bilinear groups and used it to construct an anonymous credentials system. Shortly afterwards, Akagi *et al.* [AMO08] provided a more effective Boneh Boyen-based anonymous credentials system. Recently, Camenisch *et al.* [CDHK15] proposed a Universally Composable (UC) secure anonymous credentials system that provides multi-show unlinkability and whose presentation proof is of constant size. Nevertheless, these three proposals require the prover to compute pairings and/or perform computations in $\mathbb{G}_2$. Thus, they cannot be implemented on SIM cards as the latter cannot handle such heavy computations.

Recently, Chase *et al.* [CMZ14] have opted for the use of symmetric key primitives, instead of digital signatures, so as to achieve better performances. More precisely, they used algebraic Message Authentication Codes (MACs), that relies on group operations rather than block ciphers or hash functions, as the main building block of their credentials system. Their two proposals, denoted $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$, assume that the issuer of the credential and the verifier share a secret key. In such a setting, the anonymous credentials system is referred to as *Keyed-Verification Anonymous Credentials* (KVAC). Unfortunately, their presentation proofs, for n unrevealed attributes, are of complexity $O(n)$ in the number of group elements. Moreover, when credential blind issuance is required, their KVAC systems do not provide perfect anonymity as they rely on ElGamal encryption to hide attributes.

As pointed out in [CMZ14], one can switch between the use of public-key and keyed-verification anonymous credentials which are more efficient. For that, whenever interacting with a new entity, the user proves the possession of a publicly verifiable credential (such as a driving license anonymous credential issued by a government on a set of attributes) and gets back a keyed-verification credential on the same attributes without disclosing them. Thus, during subsequent interactions with that entity, the user will use the keyed-verification credential for better efficiency.

6.3 Keyed-Verification Anonymous Credentials Systems

In this section, we first define Keyed-Verification Anonymous Credentials (KVAC) systems as well as their requirements. Next, we detail our new KVAC system that is built upon our $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme.

6.3.1 Overview on KVAC systems

A keyed-verification anonymous credentials system is defined through the following algorithms which involve three entities: a user $\mathcal{U}$, an issuer $\mathcal{I}$ and a verifier $\mathcal{V}$.

$\mathtt{Setup}(1^k)$ creates the system public parameters pp, given a security parameter k.

$\mathtt{CredKeyGen}(pp)$ generates the issuer's private key sk, which is shared with $\mathcal{V}$, and computes the corresponding public key pk.

`BlindIssue`$(\mathcal{U}(\vec{m}, s), \mathcal{I}(sk))$ is an interactive protocol between a user $\mathcal{U}$ who wants to get an anonymous credential on a set of attributes $\vec{m} = (m_1, \ldots, m_n)$ and a secret value s, without revealing them, and the issuer $\mathcal{I}$ who holds the private key sk. If the protocol does not abort, the user gets a credential σ.

`Show`$(\mathcal{U}(s, \sigma, \vec{m}, \phi), \mathcal{V}(sk, \phi))$ is an interactive protocol between $\mathcal{U}$, who wants to prove that he holds a valid credential on attributes $\vec{m}$ satisfying a given set of statements ϕ, and $\mathcal{V}$, holding the private key sk, whose goal is to check that it is actually true.

6.3.2 Security Model

In addition to the usual *correctness* property, a KVAC system should satisfy four security properties, namely *unforgeability*, *anonymity*, *blind issuance* and *key-parameter consistency*. Roughly speaking, they are defined as follows:

- *Unforgeability*: it should be infeasible for an adversary to generate a valid ZKPK that convinces a verifier that he holds a credential satisfying a given statement, or a set of statements, when it is not actually true;

- *Anonymity*: the presentation proof produced during the protocol `Show` reveals nothing else aside from the statement ϕ being proven;

- *Blind issuance*: `BlindIssue` is a secure two-party protocol for generating credentials on the user's attributes;

- *Key-parameter consistency*: an adversary should not be able to find two secret keys that correspond to the same issuer's public key.

Hereinafter, we provide more formal definitions of these security properties. It is worth mentioning that the security model described below is taken almost verbatim from [CMZ14]. For the sake of simplicity, the security model defines the *correctness*, *unforgeability* and *anonymity* properties in a particular setting where the attributes associated to issued credentials are known to the issuer. This requires an additional secure two party *blind issuing* protocol that allows a user to obtain credentials, on undisclosed attributes, *identical* to those provided by an issuer which knows all attributes.

In the sequel, `ShowVerify` refers to the part of the credential presentation phase (*i.e.* `Show` protocol) that is run on the issuer side. We also require two additional algorithms described as follows:

`Issue`$(sk, \vec{m})$ generates a credential on known attributes $\vec{m} = (m_1, \ldots, m_n)$ using the secret key sk. It is run directly if the issuer is trusted to behave honestly and all attributes are revealed. Otherwise, `BlindIssue` should be used to get credentials with some hidden attributes.

`CredVerify`$(sk, \vec{m}, \sigma)$ verifies, using the secret key sk, that the credential σ is valid with respect to $\vec{m}$. As it reveals both the credential and its attributes, it is never run (for user's privacy protection purposes). Instead, it is used to define the set of valid credentials for attributes $(m_1, \ldots, m_n)$ under sk.

6.3.2.1 Correctness

Let Φ be the set of statements supported by a credentials system and $\mathcal{U}$ be the universe of attribute sets. A KVAC system (`CredKeyGen`, `Issue`, `CredVerify`, `Show`, `ShowVerify`) satisfies *correctness*,

for Φ and $\mathcal{U}$, if for all $(m_1, \ldots, m_n) \in \mathcal{U}$, for all sufficiently large k,

$$Pr[pp \xleftarrow{R} \texttt{Setup}(1^k); (sk, pk) \xleftarrow{R} \texttt{CredKeyGen}(pp);$$
$$\sigma \xleftarrow{R} \texttt{Issue}(sk, (m_1, \ldots, m_n)) : \texttt{CredVerify}(sk, (m_1, \ldots, m_n), \sigma) = 0] = 0$$

and, for all $\phi \in \Phi$, $(m_1, \ldots, m_n) \in \mathcal{U}$ such that $\phi(m_1, \ldots, m_n) = 1$,

$$Pr[pp \xleftarrow{R} \texttt{Setup}(1^k); (sk, pk) \xleftarrow{R} \texttt{CredKeyGen}(pp); \sigma \xleftarrow{R} \texttt{Issue}(sk, (m_1, \ldots, m_n))$$
$$\texttt{Show}(pk, \sigma, (m_1, \ldots, m_n), \phi) \leftrightarrow \texttt{ShowVerify}(sk, \phi) \rightarrow b : b = 0] = 0$$

6.3.2.2 Unforgeability

A credential presentation protocol (*i.e.* `Show`) for KVAC defined through $(\texttt{CredKeyGen}, \texttt{Issue})$ satisfies *unforgeability* if for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A}$, there exists a negligible function ν such that, for all k:

$$Pr[pp \xleftarrow{R} \texttt{Setup}(1^k); (pk, sk) \xleftarrow{R} \texttt{CredKeyGen}(pp);$$
$$(state, \phi) \xleftarrow{R} \mathcal{A}(pp, pk)^{\mathcal{O}\texttt{Issue}(sk, \cdot), \mathcal{O}\texttt{ShowVerify}(sk, \cdot)};$$
$$\mathcal{A}(state) \leftrightarrow \texttt{ShowVerify}(sk, \phi) \rightarrow b \text{ such that}$$
$$b = 1 \wedge (\forall (m_1, \ldots, m_n) \in Q, \phi(m_1, \ldots, m_n) = 0)] = \nu(k)$$

where Q is the list of all attribute sets $(m_1, \ldots, m_n)$ queried to the $\mathcal{O}\texttt{Issue}(sk, \cdot)$ oracle, and all executions of $\mathcal{O}\texttt{ShowVerify}$ are sequential.

6.3.2.3 Anonymity

A credential presentation protocol (*i.e.* `Show`) for KVAC defined through $(\texttt{CredKeyGen}, \texttt{Issue})$ satisfies *anonymity* if for all PPT adversaries $\mathcal{A}$, there exists an efficient algorithm `SimShow`, and a negligible function ν such that for all k, for all $\phi \in \Phi$ and $(m_1, \ldots, m_n) \in \mathcal{U}$ such that $\phi(m_1, \ldots, m_n) = 1$, and for all $pp \xleftarrow{R} \texttt{Setup}(1^k)$ and all $(pk, sk) \xleftarrow{R} \texttt{CredKeyGen}(pp)$, for all σ such that $\texttt{CredVerify}(sk, (m_1, \ldots, m_n), \sigma) = 1$:

$$\{\texttt{Show}(pk, \sigma, (m_1, \ldots, m_n), \phi) \leftrightarrow \mathcal{A} \rightarrow state\} \approx \{\texttt{SimShow}(pk, sk, \phi)\}$$

i.e., $\mathcal{A}$'s view can be simulated by `SimShow` given only ϕ and a valid secret key corresponding to pk.

It is noteworthy that the *anonymity* property defined above ensures user's anonymity, unless information revealed in ϕ enable to identify him.

6.3.2.4 Blind Issuance

We consider a setting where the user wishes to obtain credentials for attributes $(m_1, \ldots, m_n)$ from an issuer that only knows some subset S of those attributes. Then, we consider a function f defined as $f((S, pp, pk), (sk, r), (m_1, \ldots, m_n))$ on shared input (S, pp, pk), issuer input (sk, r) and user input $(m_1, \ldots, m_n)$. f returns:

- $\perp$ to the issuer and "pp error" to the user if (sk, pk) does not belong to the range of $\texttt{CredKeyGen}(pp)$;
- $\perp$ to the issuer and "attributes error" to the user if S does not agree with $(m_1, \ldots, m_n)$;
- $\sigma \xleftarrow{R} \texttt{Issue}(sk, (m_1, \ldots, m_n); r)$ if neither of these errors occurs where $\texttt{Issue}(sk, (m_1, \ldots, m_n); r)$ indicates the execution of $\texttt{Issue}(sk, (m_1, \ldots, m_n))$ with randomness r.

An issuance protocol `BlindIssue` ensures *blind issuance* for `Issue` if it is a secure two-party computation [Gol04] (against malicious adversaries) for the above function.

6.3.2.5 Key-parameter consistency

The key generation algorithm `CredKeyGen` satisfies *key-parameter consistency* if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$, given $pp \xleftarrow{R} \texttt{Setup}(1^k)$ can produce (pk, sk_1, sk_2) such that (pk, sk_1) and (pk, sk_2) are both in the range of $\texttt{CredKeyGen}(pp)$, is negligible (where the probability is over the choice of pp and the random coins of $\mathcal{A}$).

Note that, *correctness*, *blind issuance* and *key-parameter consistency* ensure that if the user receives a credential through `BlindIssue`, then the resulting credential will be accepted by `CredVerify` for the one secret key, corresponding to pk, that the issuer knows. Then, anonymity guarantees that `Show` will not allow the issuer to learn any information beyond ϕ.

6.3.2.6 Secure keyed-verification credential system

We say that $(\texttt{CredKeyGen}, \texttt{CredVerify}, \texttt{Issue}, \texttt{BlindIssue}, \texttt{Show}, \texttt{ShowVerify})$ is a *secure keyed-verification credential system* if these algorithms satisfy the correctness, unforgeability, anonymity, blind issuance and key-parameter consistency properties defined above.

6.4 A Keyed-Verification Anonymous Credentials System based on $\mathsf{MAC}^n_{\mathsf{BB}}$

Based on the designed $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme, we construct a KVAC system involving a user $\mathcal{U}$, an issuer $\mathcal{I}$ and a verifier $\mathcal{V}$. Our KVAC system consists of the following four phases. The two main phases (`BlindIssue` and `Show`) are depicted in Figure 6.1.

6.4.1 Setup

Generate the public parameters $pp = (\mathbb{G}, p, g_1, g_2, \ldots, g_n, g, h, g_0, f)$ where $\mathbb{G}$ is a cyclic group of prime order p, a k-bit prime, and $(h, g, g_0, \{g_i\}_{i=1}^n, f)$ are random generators of $\mathbb{G}$ where DDH is hard. For $i \in \{1, \ldots, n\}$, g_i is associated with a specific type of attributes (*e.g.* age, gender, etc.). This allows us to differentiate attributes and avoid any ambiguity. Note that, in the sequel, all computations on exponents are computed modulo p (*i.e.* mod p).

6.4.2 Key Generation

Choose a random value $y \in_R \mathbb{Z}_p$ as the issuer's private key and compute the corresponding public key $Y = g_0^y$. Each user $\mathcal{U}$ is also provided with a private key sk_u and the associated public key pk_u which may be used to authenticate the user during the issuance of his credentials.

6.4.3 Blind Issuance

To issue a credential on the attributes $(m_1, \ldots, m_n)$, the issuer and the user (who has already been authenticated) engage in the following protocol. First, the user $\mathcal{U}$ builds a commitment $C_m = g_1^{m_1} \ldots g_n^{m_n} g^s$ on his attributes, where $s \in_R \mathbb{Z}_p^*$. Then, he sends it to the issuer $\mathcal{I}$ along with a ZKPK π_1 defined as $\pi_1 = \text{PoK}\{\alpha_1, \ldots, \alpha_{n+1} : C_m = g_1^{\alpha_1} g_2^{\alpha_2} \ldots g_n^{\alpha_n} g^{\alpha_{n+1}}\}$. If the proof is valid, $\mathcal{I}$ randomly picks $r, s' \in_R \mathbb{Z}_p$ and computes $A = (C_m \cdot g^{s'} \cdot h)^{\frac{1}{y+r}}$ which corresponds to a $\mathsf{MAC}^n_{\mathsf{BB}}$ on $(m_1, \ldots, m_n)$. He may also build a ZKPK π_2 ensuring that the credential is well-formed. Such a proof is defined as $\pi_2 = \text{PoK}\{\gamma : B = A^\gamma \wedge Y = g_0^\gamma\}$ where $B = C_m \cdot g^{s'} \cdot h \cdot A^{-r} = A^y$. Then, he provides $\mathcal{U}$ with the triple (A, r, s') along with the proof π_2. Upon receiving them, $\mathcal{U}$ first verifies the validity of π_2, then computes $\tilde{C}_m = C_m \; g^{s'} h$ as well as $s_u = s + s'$, which is a secret value only known to $\mathcal{U}$. Finally, he sets his anonymous credential σ as $\sigma = (A, r, s_u, \tilde{C}_m)$.

Note that in case where $\mathcal{U}$ does not mind revealing his attributes (or a subset of them), he just sends them without using any commitment (respectively, only commits to the attributes that he does not want to reveal).

Public Input: pp, pk_u, Y		
(1) Issuance of a credential (`BlindIssue`)		
User $\mathcal{U}$		**Issuer $\mathcal{I}$**
Private Input: $sk_u \in_R \mathbb{Z}_p^*$ $\overrightarrow{m} = (m_1, \ldots, m_n)$		**Private Input:** $y \in_R \mathbb{Z}_p^*$
Choose $s \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $C_m \leftarrow g_1^{m_1} \cdots g_n^{m_n} g^s$ **Build**		
$\pi_1 = \text{PoK}\{\alpha_1, \cdots, \alpha_{n+1} : C_m = g^{\alpha_{n+1}} \prod_{i=1}^n g_i^{\alpha_i}\}$	$\xrightarrow{C_m, \pi_1}$	**Check** π_1 **and Choose** $r, s' \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $A \leftarrow (C_m \cdot g^{s'} \cdot h)^{\frac{1}{y+r}}$
Check π_2 **Compute** $\tilde{C}_m \leftarrow C_m \cdot g^{s'} \cdot h$ and $s_u \leftarrow s + s'$ $\sigma \leftarrow (A, r, s_u, \tilde{C}_m)$	$\xleftarrow{A, r, s', \pi_2}$	**Build** $\pi_2 = \text{PoK}\{\gamma : Y = g_0^\gamma \wedge A^\gamma = C_m \cdot g^{s'} h \cdot A^{-r}\}$
(2) Proving Knowledge of a Credential (`Show`)		
User $\mathcal{U}$		**Verifier $\mathcal{V}$**
Private Input: $(A, r, s_u, \tilde{C}_m)$, $\overrightarrow{m}$		**Private Input:** y
Choose $l, t \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $B_0 \leftarrow A^l$, $E \leftarrow C^{\frac{1}{l}} \cdot f^t$ $C \leftarrow \tilde{C}_m^l \cdot B_0^{-r}$		
Build $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1}, \gamma, \theta :$ $E \cdot h^{-1} = g_1^{\delta_1} g_2^{\delta_2} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^\lambda f^\beta \wedge E = C^\alpha f^\beta$ $\wedge\ C = E^\theta f^\gamma\}$	$\xrightarrow{B_0, C, E, \pi_3}$	**Compute** $C' \leftarrow B_0^y$ **Check if** $C' \stackrel{?}{=} C$ **Check** π_3

Figure 6.1: Our Keyed-Verification Anonymous Credentials system

6.4.4 Credential Presentation (*i.e.* `Show`)

To anonymously prove that he holds a credential on the attributes $(m_1, \ldots, m_n)$, the user engages in an interactive protocol with the verifier $\mathcal{V}$. First, he randomly selects $l, t \in_R \mathbb{Z}_p^*$ and computes $B_0 = A^l$, a randomized version of his credential. He also computes $C = \tilde{C}_m^l B_0^{-r}$ as well as $E = C^{\frac{1}{l}} f^t$.

Note that by definition, $A^{y+r} = C_m\ g^{s'} h = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s_u} h$. Thus, we have $(A^l)^{y+r} = g_1^{lm_1} g_2^{lm_2} \ldots g_n^{lm_n} g^{ls_u} h^l$. Hence, C is simply equal to $A^{ly} = B_0^y$.

$\mathcal{U}$ also builds a ZKPK π_3 to prove that he really holds a valid credential (*i.e.* he knows the associated attributes/secrets and the value committed in E is different from zero). π_3 is defined as $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \ldots, \delta_{n+1}, \gamma, \theta : E = C^\alpha f^\beta \wedge E \cdot h^{-1} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot B_0^\lambda \cdot f^\beta \wedge C = E^\theta f^\gamma\}$. Once the required values have been computed, $\mathcal{U}$ provides $\mathcal{V}$ with B_0, C and E along with π_3[1].

Upon their receipt, $\mathcal{V}$ first computes $C' = B_0^y$, then verifies that $C = C'$. If so, he checks that π_3 is valid. $\mathcal{V}$ is convinced that $\mathcal{U}$ really holds a valid credential on attributes $(m_1, \ldots, m_n)$ if, and only if, both checks succeed.

Before detailing how our KVAC system can be easily turned into a traditional anonymous credential system, we prove in what follows that, when $C = B_0^y$, π_3 is a ZKPK of a $\mathsf{MAC}^n_{\mathsf{BB}}$ (A, r, s_u) on a block of messages $(m_1, \ldots, m_n)$.

[1] π_3 is detailed in section 6.4.4.1

6.4.4.1 Proof of possession of a credential

We describe an instantiation of our presentation protocol using non-interactive Schnorr-like proofs. As in [CMZ14], our protocol does not include proof of any additional predicates ϕ, but outputs a commitment H on the attributes which may be used as input to further proof protocols.

Hereinafter, we detail the ZKPK $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \ldots, \delta_{n+1}, \gamma, \theta : E = C^{\alpha} f^{\beta} \wedge H = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} \wedge C = E^{\theta} f^{\gamma}\}$ where $E = C^{1/l} f^{t}$, $H = E \cdot h^{-1} = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s_u} B_0^{-r/l} f^{t}$ and $C = E^{l} f^{-tl}$.

Prover		Verifier
Private Input: $\vec{m} = (m_1, \ldots, m_n)$, l, t s_u and r		
Choose $a_1, a_2, \ldots, a_{n+6} \xleftarrow{R} \mathbb{Z}_q^*$		
Compute $t_1 \leftarrow C^{a_1} f^{a_2}$ $t_2 \leftarrow g_1^{a_3} g_2^{a_4} \ldots g_n^{a_{n+2}} g^{a_{n+3}} B_0^{a_{n+4}} f^{a_2}$ $t_3 \leftarrow E^{a_{n+5}} f^{a_{n+6}}$		
Compute $c = \mathcal{H}(Ch, t_1, t_2, t_3)$	$\xleftarrow{Ch}$	**Choose** $Ch \in_R \mathbb{Z}_p^*$
Compute $R_1 \leftarrow a_1 + c/l, R_2 \leftarrow a_2 + ct$ for $i \in \{1, \ldots, n\}, R_{i+2} \leftarrow a_{i+2} + cm_i$ $R_{n+3} \leftarrow a_{n+3} + cs_u$, $R_{n+4} \leftarrow a_{n+4} - \frac{cr}{l}$ $R_{n+5} \leftarrow a_{n+5} + cl$, $R_{n+6} \leftarrow a_{n+6} - ctl$	$\xrightarrow{c, R_1, \ldots, R_{n+6}}$	**Compute** $t'_1 = C^{R_1} f^{R_2} E^{-c}$ $t'_2 = g_1^{R_3} \ldots g_n^{R_{n+2}} g^{R_{n+3}} B_0^{R_{n+4}} f^{R_2} H^{-c}$ $t'_3 = E^{R_{n+5}} f^{R_{n+6}} C^{-c}$ **Check if** $c = \mathcal{H}(Ch, t'_1, t'_2, t'_3)$

Proof. The *completeness* of the protocol follows by inspection. The *soundness* follows from the extraction property of the underlying proof of knowledge[2]. In particular, the extraction property implies that for any prover $\mathcal{P}^*$ that convinces $\mathcal{V}$ with probability ε, there exists an extractor which interacts with $\mathcal{P}^*$ and outputs $(\alpha, \beta, \lambda, \delta_1, \ldots, \delta_{n+1}, \gamma, \theta)$ with probability $poly(\varepsilon)$. Moreover, if we assume that the extractor inputs consists of two transcripts *i.e.* $(\mathbb{G}, g, h, f, B_0, C, E, c, \tilde{c}, R_1, \ldots, R_{n+6}, \tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_{n+6})$, the witness can be obtained by computing $\alpha = \frac{R_1 - \tilde{R}_1}{c - \tilde{c}}$; $\beta = \frac{R_2 - \tilde{R}_2}{c - \tilde{c}}$; $\delta_i = \frac{R_{i+2} - \tilde{R}_{i+2}}{c - \tilde{c}}, \forall i \in \{1, \ldots, n\}$; $\delta_{n+1} = \frac{R_{n+3} - \tilde{R}_{n+3}}{c - \tilde{c}}$; $\lambda = \frac{R_{n+4} - \tilde{R}_{n+4}}{c - \tilde{c}}$; $\theta = \frac{R_{n+5} - \tilde{R}_{n+5}}{c - \tilde{c}}$, $\gamma = \frac{R_{n+6} - \tilde{R}_{n+6}}{c - \tilde{c}}$; (all the computations are done mod p). The extractor succeeds when $(c - \tilde{c})$ in invertible in $\mathbb{Z}_p$. We know that $H = E \cdot h^{-1} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta}$ so $E = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} h$. We also know that $E = C^{\alpha} f^{\beta}$ so $C^{\alpha} f^{\beta} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} h$ and then

$$C^{\alpha} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} h. \tag{6.1}$$

Since $C = B_0^{y}$, we have $B_0^{\alpha y} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} h$ and

$$B_0^{\alpha y - \lambda} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} h. \tag{6.2}$$

If $\alpha \neq 0$, (8.9) implies that

$$(B_0^{\alpha})^{y - \frac{\lambda}{\alpha}} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} h. \tag{6.3}$$

Let $A = B_0^{\alpha}$, $r = -\frac{\lambda}{\alpha}$, $s_u = \delta_{n+1}$ and $m_i = \delta_i$ for $i \in \{1, \ldots, n\}$.

If $\alpha \neq 0$, (6.3) implies that the prover knows a valid $\mathsf{MAC}^n_{\mathsf{BB}}$, (A, r, s_u) on a block of messages $(m_1, \ldots, m_n)$. Note that $y - \frac{\lambda}{\alpha} \neq 0$, otherwise this would imply that the prover knows y which would be equal to $\frac{\lambda}{\alpha}$.

[2]For concurrent security, we could use the Dåmgard protocol [Dam00] which converts any Σ protocol into a three-round interactive ZKPK secure under concurrent composition.

Let us now prove that $\alpha \neq 0$. We know that

$$C = E^{\theta} f^{\gamma} = (C^{\alpha} f^{\beta})^{\theta} f^{\gamma} = C^{\alpha\theta} f^{\beta\theta+\gamma} \Longrightarrow 1 = C^{\alpha\theta-1} f^{\beta\theta+\gamma} \tag{6.4}$$

• If the prover does not know the discrete logarithm of C in the base f, this implies that it only knows one representation $(0, 0)$ of 1 in the base (C, f) [Bra93]. Therefore, $\alpha\theta = 1$ which implies that $\alpha \neq 0$.

• Suppose now that the prover knows the discrete logarithm χ of C in the base f (*i.e.* $C = f^{\chi}$) and that $\alpha = 0$. Since $C = B_0^y$, we have $B_0^y = f^{\chi}$ and then $B_0 = f^{\frac{\chi}{y}}$ (since $Y = g_0^y \neq 1$, this implies that $y \neq 0 \mod p$). From (8.8) and since α is supposed to be equal to 0, we have that $h = g_1^{-\delta_1} g_2^{-\delta_2} \dots g_n^{-\delta_n} g^{-\delta_{n+1}} f^{-\lambda\frac{\chi}{y}}$.

So, the issuer could use the prover as a subroutine to compute a representation of h in the base $(g_1, g_2, \dots, g, f)$. As $(g_1, g_2, \dots, g, f)$ are random generators of $\mathbb{G}$, this is impossible under the DL assumption [Bra93]. Therefore, this means that either $\mathcal{P}^*$ does not know the discrete logarithm of C in the base f or $\alpha \neq 0$. Both cases imply that $\alpha \neq 0$. We therefore conclude that $\alpha \neq 0$ and so the prover knows a valid $\mathsf{MAC}^n_{\mathsf{BB}}$ (A, r, s_u) on a block of messages $(m_1, \dots, m_n)$.

Finally, to prove (honest-verifier) *zero-knowledge*, we construct a simulator `Sim` that will simulate all interactions with any (honest verifier) $\mathcal{V}^*$.

1. `Sim` randomly chooses $l' \in_R \mathbb{Z}_p^*$ and a random generator $E \in_R \mathbb{G}$ and then computes $B_0 = g^{l'}$ and $C = Y^{l'}$.
2. `Sim` randomly chooses $c, R_1, \dots, R_{n+6} \in_R \mathbb{Z}_p^*$ and computes $t_1 = C^{R_1} f^{R_2} E^{-c}$, $t_2 = g_1^{R_3} \dots g_n^{R_{n+2}} g^{R_{n+3}} B_0^{R_{n+4}} f^{R_2} H^{-c}$ and $t_3 = E^{R_{n+5}} f^{R_{n+6}} C^{-c}$.
3. `Sim` outputs $S = \{B_0, C, E, c, R_1, R_2, \dots, R_{n+6}\}$.

Since $\mathbb{G}$ is a prime-order group, then the blinding is perfect in the first step. Indeed, there exists $x \in \mathbb{Z}_p$ such that for a valid $\mathsf{MAC}^n_{\mathsf{BB}}$ (A, r, s_u) on $(m_1, \dots, m_n)$: $A = g_0^x$.

For a random value $l \in \mathbb{Z}_p^*$, we therefore have $B_0 = A^l = g_0^{lx} = g_0^{l'}$ for $l' = lx$. This also implies that $C = A^{ly} = Y^{l'}$. Moreover, there exists t such that $E = C^{\frac{1}{l}} f^t$. Therefore S and $\mathcal{V}^*$'s view of the protocol are statistically indistinguishable.

□

6.5 From Keyed-Verification to Public Key Anonymous Credentials

In this section, we explain how to turn our KVAC system into a public key anonymous credentials system. Thereby, a user would be able to prove possession of a credential to any entity (*i.e.* even one that does not know the issuer's private key). For that, our system should be defined in bilinear groups (see Section 3.1.1.4).

In such a case, the system public parameters are defined as $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, \dots, g_n, g, h, g_0, f, \tilde{g}_0)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of prime order p, e is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, $(h, g, g_0, \{g_i\}_{i=1}^n, f)$ are random generators of $\mathbb{G}_1$ and $\tilde{g}_0$ is a random generator of $\mathbb{G}_2$. The other phases are updated as follows.

Key Generation. The issuer publishes a second public key $W = \tilde{g}_0^y$ associated with his private key y.

Blind Issuance. This phase does not require any change.

Credential Presentation. As the verifier $\mathcal{V}$ does not hold the private key y, some changes are required on his side. More precisely, he must compute two pairings $e(C, \tilde{g}_0)$ and $e(B_0, W)$. $\mathcal{V}$ is convinced that the user really holds a valid credential on $(m_1, \ldots, m_n)$ only if $e(C, \tilde{g}_0) = e(B_0, W)$ and π_3 is valid.

6.6 Efficiency Comparison and Performance Assessment

We first compare the efficiency of our KVAC system to that of the main existing anonymous credentials schemes (*i.e.* U-Prove, Idemix, Bilinear CL, $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$) both in terms of credential size and computational cost related to the creation of a presentation proof since it is the most time-critical phase. Next, we focus on the complexity, in the number of group elements, of KVAC systems presentation proofs. Finally, we provide timing results of the implementation of our *Credential presentation* protocol on a standard NFC SIM card.

6.6.1 Presentation proof computational cost

We compare in Table 6.1 the estimated cost of creating a presentation proof in terms of total number of multi-exponentiations. We use the same notation as [CMZ14] where l-exp denotes the computation of the product of l powers and $l - exp(b_1, \ldots, b_l)$ corresponds to the computation of the product of l powers with exponents of $b_1, \ldots, b_l$ bits (for Idemix). The number of multi-exponentiations depends on three parameters: n, r and c which respectively denote the number of attributes in a credential, the number of revealed attributes and the number of attributes kept secret.

Table 6.1 shows that our KVAC system is competitive with U-Prove (which does not provide multi-show unlinkability) and $\mathsf{MAC}_{\mathsf{GGM}}$ (which requires the verifier to know the issuer's private key and thus does not allow public verifiability). When most of the attributes are not disclosed, our proposal outperforms $\mathsf{MAC}_{\mathsf{GGM}}$.

Table 6.1: Comparison of credential sizes (for s *unlinkable* shows) and presentation proof generation cost.

Schemes	Credential size (in bits)	Number of exponentiations
U-Prove	$1024s$	$2c$ 2-exp and 1 $(n-r+1)$-exp
Idemix	5369	1 1-exp(2048), c 2-exp(256,2046), c 2-exp(592,2385) and 1 $(n-r+2)$-exp (456,3060,592,...,592)
Bilinear CL	$512n$+768	$(3+n)$ 1-exp, $2c$ 2-exp and $3+n$ pairings
$\mathsf{MAC}_{\mathsf{GGM}}$	512	3 1-exp, $2(n-r)$ 2-exp and 1 $(n-r+1)$-exp
$\mathsf{MAC}_{\mathsf{DDH}}$	1024	6 1-exp, $2(n-r+1)$ 2-exp and 2 $(n-r+1)$-exp
$\mathsf{MAC}^n_{\mathsf{BB}}$	**1024**	**1 1-exp, 4 2-exp and 1** $(n-r+3)$**-exp**

It is worth mentioning that all schemes use a 256-bit elliptic curve group, except Idemix which uses a 2048-bit modulus.

6.6.2 Complexity in the number of group elements

As it only requires a multi-commitment to all undisclosed attributes, our presentation proof is of complexity $O(1)$ in the number of group elements. This makes our KVAC system more efficient than Chase *et al.* systems (*i.e.* $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$ [CMZ14]) whose presentation proof is of complexity $O(c)$. Indeed, both of their proposals presentation proof needs c commitments (one for each unreavealed attribute).

6.6.3 Implementation results

To show the efficiency of our anonymous credentials system, we implemented the user's side of the *Credential presentation* phase (*i.e.* `Show` protocol), which is the most time critical phase, on an NFC-enabled SIM Card. In what follows, we provide more details about our testbed environment and the obtained timing results.

6.6.3.1 Testbed Environment

Similarly to our eCash system, the user's side of the `Show` protocol was implemented on a Javacard 2.2.2 SIM card, GlobalPlatform 2.2 compliant, which is embedded in a Samsung galaxy S3 NFC smartphone. As mentioned in Chapter 5, the only particularity of our card (compared to the javacard specifications) is some additional API provided by the card manufacturer enabling operations in modular and elliptic curve arithmetic. To be able to handle asymmetric cryptography on elliptic curves, the used card is equipped with a cryptoprocessor. This makes it more powerful than most cards. It is, however, worth to emphasize that such SIM cards are already widely deployed by some phone carriers to provide NFC based services.

As for the verifier side of the `Show` protocol, it was run on a PC (Quad-Core Intel Xeon CPU @3.70GHz) and communication between the SIM card, in the smartphone, and the PC (acting as the verifier $\mathcal{V}$) was done in NFC using a standard PC/SC reader (an Omnikey 5321).

6.6.3.2 Implementation Benchmarks

Table 6.2 gives timing results of the implementation of our `Show` protocol when a 256-bit prime "pairing friendly" Barreto-Naehrig elliptic curve is used.

In our implementation, the `Show` protocol is split into two parts: an *off-line* part that can be run in advance by the card (during which all the values necessary for an execution of the `Show` protocol in the worst case scenario, i.e. no revealed attributes, are computed) and an *on-line* part that needs to be performed on-line as it depends on the verifier's challenge. Indeed, in our implementation, the proof π_3 is made non-interactive: the verifier sends to the prover a challenge Ch which is included in the computation of the hash value c. Timings are given for $n = 3$, $r = 2$ and $c = 1$.

Table 6.2: Timings ((min-max) average) in ms of the implementation of the protocol `Show`

Off-line part (card) Battery-On: (1352-1392) 1378 *ms*			
On-line part			
Presentation proof (card)		*Proof verification (PC)*	
Battery-On	**Battery-Off**	y **known**	y **unknown**
(81-86) 83 *ms*	(123-124) 123.4 *ms*	(3-14) 5 *ms*	(5-17) 10 *ms*
Total On-line part			
Battery-On		**Battery-Off**	
y **known**	y **unknown**	y **known**	y **unknown**
(84-100) 88 *ms*	(86-103) 93 *ms*	(126-137) 128 *ms*	(128-141) 133 *ms*

The *presentation proof* by the card actually refers to the total time, from the applet selection to the proof reception, including the sending of the challenge by the verifier, but excluding the proof verification. "Battery-Off" denotes a powered-off phone either by the user, or because its battery is flat. In such a situation, as stated by NFC standards, NFC-access to the SIM card is still possible, but with degraded performances. Off-line computations are assumed to be automatically launched by the smartphone (battery-On) after a presentation proof, in anticipation for the next one. It is noteworthy that all computations are entirely done by the card: the smartphone is only used to trigger the `Show` protocol and to power the card. On-line computations refer to computations of R_i values and the hash c involved in the proof π_3 (see Section 6.4.4.1), and

can be potentially carried out even by a battery-Off phone. On average, the On-line part of the presentation proof is very fast even when the phone is powered-off. Actually, data exchange is the most time-consuming task.

6.7 Security Analysis

Theorem 6.1. *Our KVAC system is* unforgeable *under the assumption that* $\mathsf{MAC}^n_{\mathsf{BB}}$ *is sUF-CMVA,* perfectly anonymous *and ensures* blind issuance *as well as* key-parameter consistency *in the Random Oracle Model.*

6.7.1 Correctness

As previously explained, we need to show two properties. The first follows directly from the correctness of our $\mathsf{MAC}^n_{\mathsf{BB}}$. For the second, we prove it in what follows. The $\texttt{Issue}(sk, (m_1, \ldots, m_n))$ algorithm generates credentials of the form $(A, r, s_u, \tilde{C}_m)$ such that $A^{y+r} = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s_u} \cdot h$. Then, if `Show` is executed honestly on both sides, the proof π_3 is accepted owing to the completeness of the proof system. Also, during the `Show` protocol, the following values are computed: $B_0 = A^l$ where $l \in_R \mathbb{Z}_p$ and,

$$\begin{aligned} C &= \tilde{C}_m^l \cdot B_0^{-r} \\ &= g_1^{lm_1} g_2^{lm_2} \ldots g_n^{lm_n} g^{ls_u} \cdot h^l \cdot B_0^{-r} \\ &= (g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s_u} \cdot h)^l \cdot B_0^{-r} \\ &= (A^{y+r})^l \cdot B_0^{-r} = (A^l)^y (A^l)^r B_0^{-r} \\ &= B_0^y B_0^r B_0^{-r} = B_0^y \end{aligned}$$

Thus, the verifier's check on C (*i.e.* $C \stackrel{?}{=} B_0^y$) will succeed and it will accept.

6.7.2 Unforgeability

Here, we prove unforgeability when $\mathcal{A}$ is given credentials generated by the `BlindIssue` protocol. We have shown (see Theorem 4.4) that $\mathsf{MAC}^n_{\mathsf{BB}}$ is unforgeable under the gap $q-\mathsf{SDH}$ assumption.

Suppose there exists an adversary $\mathcal{A}$ who can break the unforgeability property of our anonymous credentials system. We will show that $\mathcal{A}$ can be used to construct an algorithm $\mathcal{B}$ that breaks the unforgeability of $\mathsf{MAC}^n_{\mathsf{BB}}$. $\mathcal{B}$ receives $pp = (\mathbb{G}, p, g_1, \ldots, g_n, g, h, g_0)$ from its $\mathsf{MAC}^n_{\mathsf{BB}}$ challenger along with Y, the issuer's public key. It sends pp and Y to $\mathcal{A}$ and answers his requests as follows:

- When $\mathcal{A}$ queries the $\mathcal{O}\texttt{BlindIssue}$ oracle: $\mathcal{A}$ sends C_m and gives a proof π_1. If π_1 is invalid, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ runs the proof of knowledge extractor to extract $\{m_i\}_{i=1}^n$ and s. $\mathcal{B}$ then queries its $\mathsf{MAC}^n_{\mathsf{BB}}$ oracle on $\{m_i\}_{i=1}^n$ which returns a tag (A, r, s_u) to $\mathcal{B}$. Finally, $\mathcal{B}$ simulates the corresponding proof[3] π_2 and forwards the tag $(A, r, s_u - s)$ along with π_2 to $\mathcal{A}$.

- When $\mathcal{A}$ queries the $\mathcal{O}\texttt{ShowVerify}$ oracle: $\mathcal{A}$ sends B_0, C, E along with a proof π_3. If the proof π_3 is invalid, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ runs the proof of knowledge extractor to extract $\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1}, \gamma$ and θ. If $\alpha = 0$, $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ computes $A = B_0^{\alpha}$, $r = -\frac{\lambda}{\alpha}$ and $s = \delta_{n+1}$. Finally, it queries its `Verify` oracle with $((\delta_1, \delta_2, \ldots, \delta_n), (A, r, s))$ and returns the result to $\mathcal{A}$.

[3]Such a simulated proof can be easily done in the ROM, using standard techniques.

In the final `Show` protocol, $\mathcal{B}$ again extracts $\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1}, \gamma, \theta$ and outputs $((\delta_1, \delta_2, \ldots, \delta_n), (B_0^\alpha, -\frac{\lambda}{\alpha}, \delta_{n+1}))$ as its forgery.

First, note that $\mathcal{B}$'s response to $\mathcal{O}$`BlindIssue` queries are identical to the ones of the honest $\mathcal{O}$`BlindIssue` algorithm. Then, we argue that its response to `ShowVerify` queries are also, with overwhelming probability, identical to the output of a real `ShowVerify` algorithm. To see this, note that the proof of knowledge property guarantees that the extractor succeeds in producing a valid witness with all but negligible probability. Furthermore, if the extractor gives valid $(\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1})$, we have from $E = C^\alpha f^\beta = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \cdot B_0^\lambda \cdot f^\beta$ that

$$C^\alpha = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \cdot B_0^\lambda \Longrightarrow C^\alpha B_0^{-\lambda} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h$$

If the $\mathsf{MAC}^n_{\mathsf{BB}}$ `Verify` oracle outputs 1 on input $((\delta_1, \delta_2, \ldots, \delta_n), (B_0^\alpha, -\frac{\lambda}{\alpha}, \delta_{n+1}))$, this implies that

$$\begin{aligned}
& (B_0^\alpha)^{y - \frac{\lambda}{\alpha}} && = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \\
\Leftrightarrow\ & (B_0^\alpha)^y \cdot B_0^{-\lambda} && = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \\
\Leftrightarrow\ & (B_0^\alpha)^y \cdot B_0^{-\lambda} && = C^\alpha B_0^{-\lambda} \\
\Leftrightarrow\ & (B_0^\alpha)^y && = C^\alpha \\
\Leftrightarrow\ & B_0^y && = C
\end{aligned}$$

Note that α is necessarily different from 0, otherwise $B_0^\alpha = 1$ and would have been rejected by the $\mathsf{MAC}^n_{\mathsf{BB}}$ `Verify` oracle.

Thus, the honest verifier algorithm accepts, if and only if, $(B_0^\alpha, -\frac{\lambda}{\alpha}, \delta_{n+1})$ would be accepted by the $\mathsf{MAC}^n_{\mathsf{BB}}$ `Verify` algorithm for message $(\delta_1, \ldots, \delta_n)$. Similarly, we can argue that $\mathcal{B}$ can extract a valid MAC from the final `Show` protocol whenever $\alpha \neq 0$ and `ShowVerify` would have output 1. Thus, if $\mathcal{A}$ can cause `ShowVerify` to accept for some statement ϕ that is not satisfied by any of the attributes sets queried to $\mathcal{O}$`BlindIssue`, then $\mathcal{B}$ can extract a new message $(\delta_1, \ldots, \delta_n)$ and a valid tag for that message.

6.7.3 Anonymity

Suppose the user is trying to prove that he has a credential for attributes satisfying some statement ϕ. We want to show that there exists an algorithm `SimShow` that, for the adversary $\mathcal{A}$, is indistinguishable from `Show` but that only takes as input the statement ϕ and the secret key sk.

Let $\phi \in \Phi$ and $(m_1, \ldots, m_n) \in \mathcal{U}$ be such that $\phi(m_1, \ldots, m_n) = 1$. Let pp be the system public parameters, Y the issuer's public key and σ be such that `CredVerify`$(sk, \sigma, (m_1, \ldots, m_n)) = 1$. So, σ consists of a quadruple $(A, r, s_u, \tilde{C}_m) \in \mathbb{G} \times \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}$ satisfying $A^{y+r} = g_1^{m_1} \ldots g_n^{m_n} g^{s_u} h$.

`SimShow`(sk, ϕ) behaves as follows: it chooses random values $l' \in_R \mathbb{Z}_p^*$ as well as a random generator $E \in_R \mathbb{G}$. It then computes $B_0 = g_0^{l'}$ and $C = Y^{l'}$. It runs $\mathcal{A}$ with the values (B_0, C, E) as the first message, simulates the proof of knowledge π_3 and outputs whatever $\mathcal{A}$ outputs at the end of the proof.

Let us first show that the values B_0, C and E are distributed identically to those produced by `Show`. Note that since $A \neq 1$, there exists $x \in \mathbb{Z}_p$ such that $A = g_0^x$. For a random value $l \in_R \mathbb{Z}_p$, $B_0 = A^l = g_0^{lx} = g_0^{l'}$ for $l' = lx$. Therefore, we also have $C = A^{ly} = Y^{l'}$. Moreover, there exists t such that $E = C^{1/l} f^t$. Then, the values computed by `SimShow` are identical to those that the normal `Show` protocol would have produced. Owing to the zero-knowledge property of the proof of knowledge, we conclude that the resulting view is indistinguishable from that produced by the adversary interacting with `Show`.

6.7.4 Blind Issuance

Following [CMZ14], we first consider the setting where all of the attributes are known to the issuer. If we consider the case where the user is corrupt, then our simulator `Sim` on shared input

(S, pp, pk) receives the user's list of attributes $(m_1, m_2, \ldots, m_n)$ and forward it to the functionality. The functionality returns "attribute error" if $S \neq (m_1, \ldots, m_n)$ and otherwise it returns σ. If the error does not occur, `Sim` then sends σ and runs the π_2 proof of knowledge simulator to simulate the proof of correctness for σ. By zero-knowledge, this is indistinguishable from the real world.

Next, we consider the case where the issuer is corrupt. In this case, our simulator `Sim` receives $\sigma = (A, r, s')$ from the issuer and runs the verifier for the proof system π_2. If the proof is accepted, it runs the corresponding proof of knowledge extractor to extract $sk = y$ (as well as r and s'). It sends (y, r, s') to the ideal functionality. By the proof of knowledge properties, the credential sent in the real world is $\sigma = ((g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s+s'} h)^{\frac{1}{y+r}}, r, s')$ which is exactly what would be produced by the ideal functionality on input (y, r, s') described above.

Then, we consider this time the `BlindIssue` protocol which keeps attributes hidden. In this setting, we define an ideal functionality $\mathcal{F}$ that takes (y, r, s') as the issuer's input and $(m_1, \ldots, m_n, s, l)$ as the user's input. $\mathcal{F}$ returns a randomized credential $\sigma_l = (A^l, s', r)$ to the user whilst nothing is returned to the issuer.

Consider the case where the user is corrupt. Then, our simulator `Sim` on shared input (S, pp, pk) receives a commitment C_m and runs the verification of the proof of knowledge π_1. If the proof is valid, it then uses the proof of knowledge extractor of π_1 to extract $\{m_i\}_{i=1}^n$ and s, then sends them along with a random l and the set S to the functionality. The functionality returns $\sigma_l = (A^l, r, s')$. So, `Sim` sends σ and runs the π_2 proof of knowledge simulator to simulate the proof of correctness for σ. By the zero-knowledge property, this is indistinguishable from the real world.

Next, we consider the case where the issuer is corrupt. In this case, our simulator `Sim` on shared input (S, pp, pk) receives $\sigma = (A, r, s')$, C_m and π_2 from the issuer and runs the verifier for the proof system π_2. If the proof is valid, it runs the corresponding proof of knowledge extractor to extract $sk = y$, as well as r and s'. Then, it sends (y, r, s') to the ideal functionality. By the proof of knowledge property, a randomized version of the credential sent in the real world is $\sigma_{l'} = (((C_m \cdot g^{s'} h)^{\frac{1}{y+r}})^{l'}, r, s')$. The latter is indistinguishable from what would have been produced by the ideal functionality on input (y, r, s').

6.7.5 Key-parameter consistency

Our KVAC scheme trivially satisfies this requirement. Indeed, our system public parameters pp consists of a cyclic group $\mathbb{G}$ of prime order p along with $(n + 4)$ generators $g_0, g_1, \ldots, g_n, g, h, f$. The issuer's public key is a value $Y \in \mathbb{G}$ and the associated private key is the discrete logarithm of Y in the base g_0. As this discrete logarithm is unique, an adversary cannot find two secret keys corresponding to a same public key Y.

6.8 Conclusion

In this chapter, based on our algebraic $\mathsf{MAC}^n_{\mathsf{BB}}$ scheme, we designed a keyed-verification anonymous credentials (KVAC) system whose presentation proof is efficient both in terms of presentation cost and complexity (in the number of group elements). Our KVAC system provides multi-show unlinkability and, unlike Chase *et al.* KVAC systems, requires the issuer to hold a single private key regardless of the number of attributes. Through slight modifications (solely on the verifier side), our KVAC system can be easily turned into a quite efficient public key anonymous credentials system. Thereby, it can also be used even if the verifier does not hold the issuer's private key.

In the next chapter, based on our practical DAA scheme introduced in Section 4.4, we design a privacy-preserving authentication protocol for embedded SIM (eSIM) so as to cope with a real issue that has arisen at the Groupe Speciale Mobile Association (GSMA).

Chapter 7

An Anonymous Authentication and Identification Protocol for eSIM

Contents

To cope with a real issue that has arisen at the GSM Association (GSMA), we propose in this chapter a privacy-preserving protocol enabling the anonymous authentication and identification of an eSIM to a Mobile Network Operator (MNO) independent third party known as Subscription Manager-Discovery Server (SM-DS). Thereby, following a switch of MNO, an eSIM can be remotely provisioned with its new network profile whilst protecting the privacy of its owner against a malicious discovery server. Our protocol relies on our BB-based Pre-DAA scheme (detailed in Section 4.4) which is suitable for SIM cards as it requires no pairing computations on the platform side. A variant of this proposal was patented on November 23, 2015 [BCGT15].

7.1 Embedded SIM (eSIM)

As mentioned in Chapter 2, some M2M devices (*e.g.* smart meters) are not designed in a way enabling the removal or replacement of the SIM card as the latter is either hardly accessible or welded in the device at its manufacture. Thus, there is no way to switch from one network operator to another. Furthermore, if one decides to change the subscription associated to several devices deployed in different locations, then the replacement of all SIM cards can be quite costly.

To cope with this issue and provide more flexibility, the GSMA introduced a new generation of SIM cards referred to as embedded SIM (eSIM), or eUICC, which enables the change of the subscription associated to a SIM card while keeping the same chip and level of security as classical

SIM cards. The GSMA also proposed an architecture for Over-The-Air remote SIM provisioning with their new network profiles (see Section 2.3.2 for details). This architecture involves a Mobile Network Operator (MNO) *independent* third party known as Subscription Manager-Discovery Server (SM-DS) which mainly aims to provide an eSIM with the address of a server, called Subscription Manager-Data Preparation+ (SM-DP+), that has generated its new profile [GSM16b]. Thereby, the eSIM can get in touch with it to download its new network profile. Indeed, the role of the SM-DS consist in routing the eSIM, or the device embedding it, to the right SM-DP+ so as to download its new profile and install it. To do so, the SM-DS maintains a correspondence table according to the EIDs.

However, if no privacy-preserving mechanisms are set up, the SM-DS would be able to trace the eSIM subscription changes all along its life cycle, hence entailing some privacy concerns. To prevent this, we propose in this chapter a privacy-preserving protocol that enables the discovery server (*i.e.* SM-DS) to anonymously authenticate and identify the relevant eSIM. Thereby, it can provide the eSIM with the address of the corresponding SM-DP+ without knowing with which eSIM it is currently interacting (*i.e.* without knowing the unique identifier, denoted by EID, of the eSIM).

7.2 System Framework

As detailed in Section 2.3.2, the remote SIM provisioning architecture involves four main stakeholders, namely an embedded SIM (eSIM) (also known as eUICC) which is uniquely identified by its eUICC identifier referred to as EID, a mobile network operator (MNO), a Subscription Manager-Data Preparation (SM-DP+), and an MNO independent third party known as Subscription Manager-Discovery Server (SM-DS). It is noteworthy to mention that the eUICC Manufacturer (EUM) is also indirectly involved as it handles the issuance of the membership certificate that enables the authentication of the eSIM to the SM-DS and SM-DP+.

The main steps that take place whenever the user changes the subscription associated to an eSIM are as follows (more details are provided in Section 2.3.3):

1. *Profile Generation*: The MNO asks a specific SM-DP+ to generate the eSIM new profile. The latter will be securely stored until its download by the corresponding eSIM.
2. *Notification Registration*: Once the profile has been generated, the SM-DP+ sends a notification registration to the SM-DS. It aims to inform the SM-DS of the availability of a new

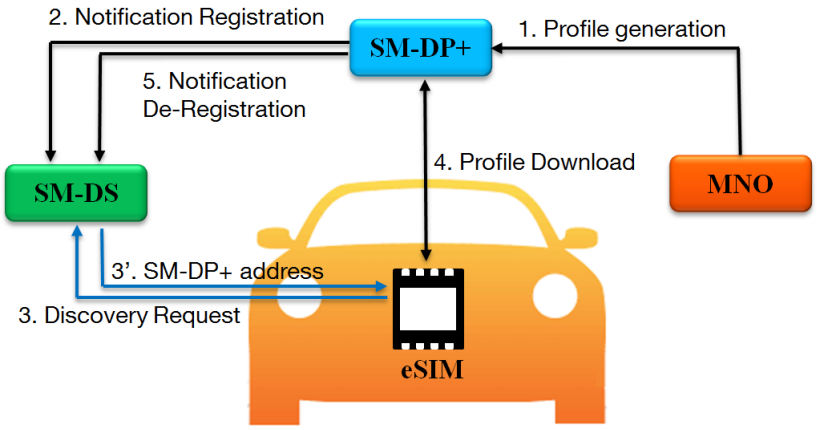

Figure 7.1: Remote Profile Provisioning Procedure (Automotive use case)

profile for the eSIM identified by EID in a given SM-DP+.

3. *Discovery Request*: The eSIM queries its default SM-DS to know whether there is a pending new profile for it. If so, after authenticating, and identifying, the eSIM, the SM-DS provides it with the address of the SM-DP+ that has generated its new profile.

4. *Profile Download*: The eSIM contacts the relevant SM-DP+ so as to download its new profile. After mutually authenticating each other, the eSIM is finally securely provided with its new network profile.

5. *Notification De-Registration*: Following a successful profile download, the SM-DP+ notifies the SM-DS. Thus, the latter can delete the corresponding notification as it is no longer useful.

7.3 Requirements

In this section, we define the different requirements that an anonymous authentication and identification protocol for eSIM must fulfill. The latter are classified in two categories: security requirements and functional requirements.

In the sequel, we make the following assumption: the SM-DS is assumed to be honest but *curious*. That is, the SM-DS correctly follows the protocol but may try to learn some additional information from received notifications and requests.

7.3.1 Security requirements

The security properties that an anonymous authentication and identification protocol shall satisfy are as follows:

- *Consistency*: A discovery request sent by a legitimate eSIM (for which a new profile has been generated by an SM-DP+) must necessarily match with one of the notifications stored in the default SM-DS.

- *Unlinkability*: The SM-DS should not be able to (1) link the different subscription changes associated with the same eSIM (*i.e.* same EID), or (2) retrieve the EID of the eSIM that generated a given discovery request.

7.3.2 Functional requirements

In addition to the security requirements stated above, the protocol must fulfil the following two functional requirements:

- *Efficiency*: The privacy-preserving protocol must not negatively impact the user's experience. Therefore, it ought to be as efficient as the original protocol (which entails some privacy issues).

- *No-pairings*: The protocol is intended for eSIMs which, we recall, cannot handle pairing computations (or even computations in either $\mathbb{G}_2$ or $\mathbb{G}_T$). Therefore, the proposed solution should not require any pairing computations on the eSIM side.

7.4 Our anonymous authentication and identification protocol for eSIM

In this section, we first give an overview on our authentication and identification protocol. Then, we detail its different phases while depicting the main ones (*i.e. Certificate Issuance* and *Discovery Request*) in Figure 7.2. Recall that our protocol is based on our BB-based Pre-DAA scheme which is proven secure in the random oracle model (see Chapter 4 and more precisely Section 4.4).

7.4.1 Overview

Our privacy-friendly authentication and identification protocol for eSIM mainly consists of the following four phases. (1) A *Setup* phase where the system public parameters as well as required keys are defined. (2) Upon the manufacture of an eSIM, a *Certificate Issuance* phase is triggered so as to provide the eSIM with a group membership certificate. The latter will enable the anonymous authentication of the eSIM to the different other entities. (3) At each new subscription, the MNO initiates the *Profile Generation* phase by asking the SM-DP+ to create a profile for the corresponding eSIM. Once the profile has been generated, the SM-DP+ notifies the discovery server (SM-DS) by providing it with a set `S` including legitimate identification values associated to the new subscription. (4) To get its new profile, the eSIM must first know which SM-DP+ holds it. To this end, it transmits an anonymous request to its default SM-DS. If the request is authenticated, the eSIM obtains the address of the corresponding SM-DP+ server (*i.e.* the one that has generated its new profile). This last phase is referred to as *Discovery Request*. Once it has been successfully completed, the eSIM can download its new profile by getting in touch with the relevant SM-DP+.

7.4.2 Protocols description

Hereinafter, we detail the different phases of our anonymous authentication and identification protocol. Its main building block is our BB-based Pre-DAA scheme detailed in Chapter 4 Section 4.4 and which requires no pairing computations on the platform side.

7.4.2.1 Setup

Let $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, h, g_0, f, \tilde{g}_0, \mathcal{H}, e)$ denote the system public parameters where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of prime order p, a k-bit prime, (h, g_0, g_1, g_2, f) are random generators of $\mathbb{G}_1$ and $\tilde{g}_0$ is a random generator of $\mathbb{G}_2$, $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a type-3 bilinear map. In the sequel, all computations on exponents are computed modulo p.

Each eUICC manufacturer (EUM) also selects a random value $y \in_R \mathbb{Z}_p^*$ as its private key sk and compute the associated public key $pk = (Y, W)$ where $Y = g_0^y$ and $W = \tilde{g}_0^y$. Indeed, each EUM holds a distinct public/private key pair and eSIMs are organized into groups according to their manufacturer which acts as a group manager (*i.e.* membership certificate issuer).

7.4.2.2 Certificate Issuance

As previously mentioned, each manufactured eSIM is uniquely identified by its eUICC identifier (EID). Upon its manufacture, the eSIM randomly selects $s \in_R \mathbb{Z}_p^*$ then, computes $C_{s_1} = g_1^{EID'}$ and $C_{s_2} = g_2^s$ where EID' is a 160-bit identifier derived from EID. Next, it builds a ZKPK $\pi_1 = \text{PoK}\{\alpha_1, \alpha_2 : C_{s_1} = g_1^{\alpha_1} \wedge C_{s_2} = g_2^{\alpha_2}\}$ that it sends to the EUM along with C_{s_1} and C_{s_2}.

Upon their receipt, the EUM first checks the validity of π_1. If so, it randomly selects $s', r \in_R \mathbb{Z}_p^*$ and computes $A = (C_{s_1} C_{s_2} g_2^{s'} h)^{\frac{1}{y+r}}$. Then, it provides the eSIM with s', r and A as well as a ZKPK π_2 ensuring that A is well-formed. The proof π_2 is defined as $\pi_2 = \text{PoK}\{\gamma : B = A^\gamma \wedge Y =$

$g_0^{\gamma}\}$ where $B = C_{s_1} C_{s_2} g_2^{s'} h \cdot A^{-r} = A^{y}$. If the check of π_2 validity succeeds, the eSIM stores the triple (A, r, s_u), where $s_u = s + s'$, as its membership certificate ζ. The latter will subsequently enable the eSIM to be anonymously authenticated to other entities.

Public Input: pp, $pk = (Y, W)$

(1) Certificate Issuance

eSIM		**EUM**
Private Input: EID		**Private Input:** $y \in_R \mathbb{Z}_p^*$
Choose $s \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $C_{s_1} \leftarrow g_1^{EID'}$ and $C_{s_2} \leftarrow g_2^{s}$		
Build $\pi_1 = \text{PoK}\{\alpha_1, \alpha_2 : C_{s_1} = g_1^{\alpha_1} \wedge C_{s_2} = g_2^{\alpha_2}\}$	$\xrightarrow{C_{s_1}, C_{s_2}, \pi_1}$	**Check** π_1 **Choose** $r, s' \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $A \leftarrow (C_{s_1} C_{s_2} g_2^{s'} h)^{\frac{1}{y+r}}$
Check π_2 $\zeta \leftarrow (A, r, s_u)$ where $s_u \leftarrow s + s'$	$\xleftarrow{A, r, s', \pi_2}$	**Build** $\pi_2 = \text{PoK}\{\gamma : Y = g_0^{\gamma} \wedge A^{\gamma} = C_{s_1} C_{s_2} \cdot g_2^{s'} h \cdot A^{-r}\}$

(2) Discovery Request

eSIM		**SM-DS**
	Additional Public Input: d_c	
Private Input: (A, r, s_u), EID		**Private Input:** y
Choose $t, l \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $B_0 \leftarrow A^l$, $T = \mathcal{H}(d_c)^{EID'}$ $C \leftarrow g_1^{lEID'} g_2^{ls_u} h^l B_0^{-r}$, $E \leftarrow C^{\frac{1}{l}} f^t$		
Build $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \delta_2, \theta, \gamma : E = C^{\alpha} f^{\beta} \wedge E \cdot h^{-1} = g_1^{\delta_1} g_2^{\delta_2} B_0^{\lambda} f^{\beta} \wedge C = E^{\theta} f^{\gamma} \wedge T = \mathcal{H}(d_c)^{\delta_1}\}$	$\xrightarrow{B_0, C, E, T, \pi_3}$	**Verify** that $e(C, \tilde{g}_0) = e(B_0, W)$ **Check** π_3
	$\xleftarrow{Add_{SM-DP+}}$	**Query database** for an entry with T
Details of the proof π_3**:** **Choose** $a_1, a_2, \ldots, a_7 \xleftarrow{R} \mathbb{Z}_p^*$ **Compute** $t_1 \leftarrow C^{a_1} f^{a_2}$, $t_2 \leftarrow g_1^{a_3} g_2^{a_4} B_0^{a_5} f^{a_2}$ $t_3 \leftarrow E^{a_6} f^{a_7}$, $t_4 \leftarrow \mathcal{H}(d_c)^{a_3}$		
Compute $c = \mathcal{H}(t_1, t_2, t_3, t_4, Ch)$	$\xleftarrow{Ch}$	**Choose** $Ch \in_R \mathbb{Z}_p^*$
Compute $R_1 \leftarrow a_1 + c/l$ $R_2 \leftarrow a_2 + ct$, $R_3 \leftarrow a_3 + c \cdot EID'$ $R_4 \leftarrow a_4 + cs_u$, $R_5 \leftarrow a_5 - \frac{cr}{l}$ $R_6 \leftarrow a_6 + cl$, $R_7 \leftarrow a_7 - ctl$	$\xrightarrow{c, R_1, \ldots, R_7}$	**Compute** $t'_1 = C^{R_1} f^{R_2} E^{-c}$ $t'_2 = g_1^{R_3} g_2^{R_4} B_0^{R_5} f^{R_2} H^{-c}$ $t'_3 = E^{R_6} f^{R_7} C^{-c}$ $t'_4 = \mathcal{H}(d_c)^{R_3} T^{-c}$ **Check if** $c = \mathcal{H}(t'_1, t'_2, t'_3, t'_4, Ch)$

Figure 7.2: Certificate Issuance and Discovery Request Protocols

7.4.2.3 Profile Generation

Following each subscription change, the MNO asks a given SM-DP+ server to generate the eSIM new profile. As soon as the profile is created, the SM-DP+ must notify the SM-DS that it holds a new profile for a given eSIM. This shall be done in a privacy-preserving way (*i.e.* the SM-DS should not be able to know the EID of the relevant eSIM).

To do so, the SM-DP+ computes the set $\mathtt{S} = \{\mathcal{H}(d_s)^{EID'}, \mathcal{H}(d_s{+}1)^{EID'}, \ldots, \mathcal{H}(d_s{+}Max)^{EID'}\}$ such that d_s and Max respectively correspond to the subscription date and the time period during which the new profile remains available for download (for example, $Max = 7$ for an availability of a week). Of course, the subscription date and time period during which the profile is available can be set according to different time units (*e.g.* hours, days, etc.). The set $\mathtt{S}$ along with Add_{SM-DP+}, which denotes the address of the SM-DP+ that generated the new profile, are forwarded to the SM-DS.

7.4.2.4 Discovery Request

To download its new profile, the eSIM must first know which SM-DP+ holds it (*i.e.* the address of the SM-DP+ that has generated it). To this end, it generates an anonymous profile request that it transmits to the discovery server (SM-DS).

A request initiated at a given date, d_c, consists of the tag $T = \mathcal{H}(d_c)^{EID'}$ along with a signature σ on it. To generate σ, the eSIM proceeds as if it is creating a BB-based Pre-DAA signature with respect to the basename $\texttt{bsn} = d_c$ and for $m = \varnothing$. More specifically, the eSIM randomly picks $l, t \in \mathbb{Z}_p^*$ and computes $B_0 = A^l$, a randomized version of its membership certificate. Then, it computes $C = (g_1^{EID'} g_2^{s_u} h)^l B_0^{-r} = B_0^y$ and $E = C^{\frac{1}{t}} f^t$. Thus, σ is defined as $\sigma = (B_0, C, E, \pi_3)$ where $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \delta_2, \theta, \gamma : E = C^\alpha f^\beta \wedge H = E \cdot h^{-1} = g_1^{\delta_1} g_2^{\delta_2} \cdot B_0^\lambda \cdot f^\beta \wedge C = E^\theta f^\gamma \wedge T = \mathcal{H}(d_c)^{\delta_1}\}$.

Upon the receipt of a profile request, the SM-DS first verifies that $e(C, \tilde{g}_0) = e(B_0, W)$. If so, it checks the validity of the proof π_3. The request is considered valid only if both checks succeed. In such case, the SM-DS queries its database to verify whether there is an entry with the same tag $T = \mathcal{H}(d_c)^{EID'}$. If it is the case, it provides the eSIM with Add_{SM-DP+}, the address of the corresponding SM-DP+, from which it can download its new profile. Otherwise, the discovery request is discarded.

7.5 Security Analysis

The security of our anonymous authentication and identification protocol mainly relies on the security of our BB-based Pre-DAA scheme (see Section 4.4.3). Hereinafter, we state the assumptions under which our protocol satisfies the required security properties.

- ***Consistency.*** It follows by inspection. Indeed, if the same EID is used to generate both the set $\texttt{S}$ and the discovery request, then one of the elements of $\texttt{S}$ will be equal to the tag $T = \mathcal{H}(d_c)^{EID'}$ included in the discovery request.

- ***Unlinkability.*** The sets $\texttt{S}$ and $\texttt{S}'$ sent to the SM-DS following two subscription changes associated with the same eSIM are respectively defined as $\texttt{S} = \{\mathcal{H}(d_s)^{EID'}, \mathcal{H}(d_s+1)^{EID'}, \ldots, \mathcal{H}(d_s + Max)^{EID'}\}$ and $\texttt{S}' = \{\mathcal{H}(d_{s'})^{EID'}, \mathcal{H}(d_{s'}+1)^{EID'}, \ldots, \mathcal{H}(d_{s'} + Max)^{EID'}\}$ where d_s and $d_{s'}$ are the corresponding dates of subscriptions. Therefore, under the DDH assumption and as long as the eSIM does not make two subscription changes within Max days[1], our anonymous authentication and identification protocol ensures the first condition in the ROM. In fact, if two subscription changes are made within Max days, then the default SM-DS will receive two sets $\texttt{S}$ and $\texttt{S}'$ such that $\texttt{S} \cap \texttt{S}' \neq \varnothing$. Thus, the SM-DS will be able to link the eSIM subscription changes. As for the second condition, it follows from the anonymity property of our Pre-DAA scheme as the signature produced in our anonymous authentication and identification protocol is exactly the same as the one of the BB-based Pre-DAA scheme. We recall that our Pre-DAA scheme ensures anonymity in the ROM under the XDH assumption (see Section 4.4.3). Therefore, our protocol provides unlinkability, in the ROM, under the XDH assumption.

7.6 Performance Assessment

To show the efficiency and suitability of our protocol for SIM cards, we implemented the eSIM side of the *Discovery Request* protocol on an NFC-enabled standard SIM card. Hereinafter, we provide the details of our testbed environment and the corresponding timing results.

[1]This is a realistic assumption since eSIM subscription changes are not as frequent.

7.6.1 Testbed Environment

The eSIM side of the *Discovery Request* protocol was implemented on a Javacard 2.2.2 SIM card, GlobalPlatform 2.2 compliant, embedded in a Samsung galaxy S3 NFC smartphone. We used the same card as in Chapter 5 (*i.e.* its only particularity, compared to javacard specifications, is some additional API provided by the card manufacturer enabling operations in modular and elliptic curve arithmetic). The SM-DS side of the *Discovery Request* protocol was run on a PC (Intel Xeon CPU, 3.70GHz). The communication between the SIM card in the smartphone and the PC was done in NFC using a standard PC/SC reader (an Omnikey 5321). Finally, for the implementation, we used a 256-bit prime "pairing friendly" Barreto-Naehrig elliptic curve.

7.6.2 Implementation Results

Table 7.1 below gives timing results of the implementation of the *Discovery Request* protocol both in the case where the smartphone is switched on or off. The protocol is split into two parts so as to provide better performance results: an *off-line* part that can be run in advance by the card and an *on-line* part that needs to be performed on-line as it depends on the verifier's challenge.

The *Signature generation* (by card) refers to the total time, from the applet selection to the request reception, including the sending of the challenge Ch and the basename `bsn` (*i.e.* d_c in this use case) by the verifier (*i.e.* SM-DS), but excluding the proof verification. *Battery-On* denotes a switched on phone whereas *Battery-Off* refers to a powered-off phone either by the user, or because the battery is flat. In this latter situation, as stated by NFC standards, NFC-access to the SIM card is still possible, but with deteriorated performances.

Off-line computations are assumed to be automatically launched by the smartphone (Battery-On) following a signature generation and in anticipation for the next one. It is worth mentioning that all computations on the SIM side are entirely performed by the card. In fact, the smartphone is only used to trigger the protocol and to power the card. As regards to *On-line* computations, they refer to the calculation of the values T and t_4, the hash c as well as $R_1, \ldots, R_7$ involved in the proof π_3. They can be potentially carried out even with a battery-off phone (provided that they are triggered from an external card reader).

Table 7.1: Timings ((min-max) average) in milliseconds (ms) of the implementation of the *Discovery Request* protocol

Off-line part (card)	
Battery-On	**Battery-Off**
(741-767) 754 ms	(2294-2362) 2331 ms

On-line part		
Signature generation (card)		*Signature verification (PC)*
Battery-On	**Battery-Off**	(4-13) 6 ms
(180-191) 186 ms	(434-464) 446 ms	

Total On-line part	
Battery-On	**Battery-Off**
(184-204) 192 ms	(438-477) 452 ms

In our implementation, we assumed that `bsn` (*i.e.* the current date d_c in this use case) is not known in advance but rather provided to the card on-line. Consequently, the card has two scalar multiplications to compute *on-line*. For use cases where the `bsn` can be known in advance, these multiplications may be included in the *off-line* computation part, hence resulting in much better performances. Indeed, one scalar multiplication on our card takes around 60 ms.

7.7 Conclusion

Using our BB-based Pre-DAA scheme introduced in Section 4.4 as main building block, we proposed an anonymous authentication and identification protocol for embedded SIMs. Our

scheme aims to enable the authentication of an eSIM to an MNO independent entity known as Discovery Server while preserving the privacy of its owner. Thereby, we show one of the potential applications of our practical Pre-DAA schemes which require no pairing computations (or even computations on $\mathbb{G}_2$ or $\mathbb{G}_3$) on the platform side.

In the next chapter, based on our sequential aggregate MAC scheme, we design an efficient coercion-resistant electronic scheme where voter's credentials can be efficiently updated so as to enable credentials revocation as well as votes in multiple elections.

Chapter 8

An Efficient Coercion-Resistant Remote Electronic Voting Scheme

Contents

In this chapter, we propose a new coercion-resistant remote electronic voting scheme suitable for real polls as, unlike previous proposals, it has linear time complexity. Our scheme relies on credentials generated based on the algebraic $\mathsf{MAC}_{\mathsf{GGM}}$ scheme due to Chase *et al* [CMZ14]. Through the use of our sequential aggregate MAC scheme introduced in Section 4.3, voters' credentials can be efficiently updated so as to enable credentials revocation and allow eligible electors to vote in subsequent elections.

This work has been published in the paper "*Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant*" [ABBT16] at VOTING'16.

8.1 Remote Electronic Voting

Internet voting offers a better voting experience since voters can cast their votes from their computers or even their smartphones. By eliminating the need to visit polling places, it may

attract more voters and thus increase voter turnout. In addition, it improves the efficiency of vote tallying.

These benefits motivated countries such as Estonia and Switzerland to adopt it in real world elections. However, it is still not widely spread. This is particularly due to many inherent concerns such as selective DDoS attacks on the election server, malware attacks on the voter client as well as risks entailed by the lack of private polling booths [US 15].

In this work, we mainly focus on the latter concern while assuming that the voting devices are trustworthy; therefore, votes will be cast-as-intended. Indeed, adversaries may leverage the lack of private polling booths to perform coercion or vote-selling attacks. Consequently, electronic voting schemes ought to address this issue that remained a challenge for many years.

To this end, Juels, Catalano and Jakobsson (JCJ) [JCJ05] introduced an essential property known as *coercion-resistance*. It considers the different actions that a coercer could undertake, namely constrain a voter to cast a given vote, force her to reveal her private voting credential and subsequently vote on her behalf, or keep her from voting. They also proposed the first coercion-resistant scheme based on anonymous credentials. To be able to vote, an eligible voter is beforehand provided with a valid credential. Under coercion, she can use a fake credential instead of her valid one. Thereby, she deceives any adversary about her true vote intention as a coercer is unable to distinguish the fake credential from the valid one. Unfortunately, JCJ's scheme is inefficient for large scale voting scenarios as, for N ballots, the complexity of the tallying is in $O(N^2)$.

To cope with this issue, we propose an efficient coercion-resistant voting scheme which has linear time complexity and thus, is suitable for real polls. Our scheme relies on credentials generated based on the recent algebraic MAC scheme due to Chase et al. [CMZ14]. We prove that, even though a part of our credentials are made publicly known, a coercer is unable to distinguish a valid credential from a fake one. Furthermore, our scheme allows talliers to check credentials validity even though they are encrypted. Using our sequential aggregate MAC scheme introduced and proven secure in Section 4.3, eligible voters' credentials can be efficiently updated to allow them to vote in new elections. Credentials of voters who are no longer eligible to vote can be revoked as well. Thanks to these improvements, coercion-resistance is obtained almost for free as the proposed scheme is just slightly slower than non coercion-resistant classical mix-net based voting schemes.

8.2 Related Work

To enhance JCJ's voting system [JCJ05] and provide linear time tallying, some coercion-resistant schemes were then proposed. In what follows, we briefly review the most promising proposals.

Araújo, Foulle and Traoré [AFT07] were the first to propose a coercion-resistant electronic voting scheme that achieves linear time complexity. Nevertheless, their proposal does not support multiple elections. Indeed, at each new election, the voter has to visit the registration place in order to obtain her credential associated to the new poll.

To address this shortcoming, Araújo and Traoré [AT13] later proposed another scheme that allows credentials revocation and multiple elections. To issue a new credential, their scheme requires the registration authorities to jointly generate a BBS [BB04] group signature. Unfortunately, up to now, there is no practical solution to compute such a signature in a distributed manner, which makes the scheme impractical for real polls. It is, however, noteworthy to mention that they also proposed a generic technique to identify valid, but illegitimate, voting credentials that a majority of colluding registrars could compute. Even though such an event is unlikely, their generic technique applies to other voting schemes including the one that we introduce in this chapter.

Finally, Clark *et al.* [CH11] and Spycher *et al.* [SKHS11] proposed two different approaches to tackle the coercion-resistance issue. However, both schemes do not really have linear time complexity. They truly achieve it only if the level of anonymity is lowered. More specifically, a voter's ballot is indistinguishable from a small set of ballots and not from all the received ones.

8.3 Remote Electronic Voting System Framework

In this section, we first introduce the different parties involved in an electronic voting scheme. Then, we indicate the assumptions that are made in this work. Next, we give an overview of the five phases of our remote electronic voting scheme.

8.3.1 Stakeholders

A remote electronic voting system involves as participants a set of registration authorities $\mathcal{R}$ known as *registrars*, a set of tallying authorities $\mathcal{T}$ referred to as *talliers* and a set of *voters* $\mathcal{V}$. For security reasons, the roles of both registrars and talliers are generally distributed among a large group of authorities. Thus, they must collaborate in order to generate a new credential or tally the votes (*i.e.* none of them can generate a new credential or decrypt a vote on his own).

8.3.2 Assumptions

In this work, we assume that the voting device (*i.e.* PC or smartphone) is trusted. Thus, electors' votes will be cast-as-intended. That is, the vote encrypted in the received ballot matches the elector's choice.

Our voting scheme also makes use of a Bulletin Board (BB), which is a publicly accessible memory with an appendive-write access (*i.e.* one can only append data to BB, but he is not allowed to erase or overwrite existing data).

8.3.3 System Framework

Our coercion-resistant voting scheme mainly consists of five main phases, namely *Setup*, *Registration*, *Voting*, *Pre-Verification* and *Tallying*. (1) During the *Setup* phase, election parameters as well as required keys are generated. (2) To be able to vote, an eligible voter must register through a *Registration* phase. After proving her identity, she receives a unique and valid credential (associated to a secret s) that is issued by the registration authorities. (3) Later, during *Voting* phase, the voter uses her credential and the secret s to generate a ballot that she sends via an anonymous channel. (4) Before tallying votes, a *Pre-Verification* phase is carried out to remove both erroneous ballots as well as duplicate votes. (5) Once done, the tallying authorities perform the *Tallying* phase where valid votes are decrypted so as to compute election result and publish them on the Web Bulletin Board (WBB).

These five phases can be modelled through the following algorithms and protocols:

1. *Setup* aims to initialize the system parameters as well as the required keys. It consists of the following two algorithms:

   - `PubParam`(1^k): On input a security parameter k, this probabilistic algorithm outputs the system public parameters pp.
   - `KeyGen`(pp): On input the system public parameters pp, this probabilistic algorithm outputs the private/public key pairs of the tallying and the registration authorities: the talliers share the private/public key pair (sk_T, pk_T) whereas the registrars obtain (sk_R, pk_R), which is also shared among talliers.

2. *Registration*

   `Register`$(\mathcal{V}(ID_v, pp), \mathcal{R}(sk_R, pp))$: Given a voter's identity ID_v, this probabilistic algorithm outputs a unique (and valid) credential σ associated to a secret s cooperatively chosen by the registrars $\mathcal{R}$.

3. *Voting*

   `Voting`$(\mathcal{V}(\sigma, s, v, O, pk_T))$: This algorithm, which is executed by a voting device, takes as input the public key of the talliers pk_T, the candidate slate O, the voter's choice v as well as her credential σ and the associated secret s. It outputs a ballot B which includes an encrypted version of the voter's choice.

4. *Pre-Verification*

   `Pre-Verification`$(\mathcal{T}(\mathcal{BB}, sk_T, n_C))$: This algorithm, which is performed by the talliers, takes as input the set $\mathcal{BB}$ of all ballots cast on the WBB. Once this algorithm performed, duplicate ballots and ballots with invalid proofs are discarded, "some parts" of the remaining ballots are removed as well. The set of remaining ballots is denoted by $\mathcal{BB}'$.

5. *Tallying*

   `Tallying`$(\mathcal{T}(\mathcal{BB}', sk_T, n_C, \{pk_i\}_{i=1}^{i=n_V}))$: This algorithm, which is performed by the talliers, takes as input the remaining set of ballots $\mathcal{BB}'$. It tallies the valid votes and outputs the election result.

8.4 Security and Privacy Requirements

In this section, we recall the properties that a remote voting scheme must ensure to be both secure and privacy-friendly. In addition to the usual *correctness* property, which guarantees the proper functioning of the system, three main properties must be satisfied: *verifiability*, *eligibility* and *coercion-resistance*. In what follows, we briefly define them while focusing on the *coercion-resistance* property as it is the most relevant to our work (see [JCJ02, CGK+16] for formal definitions of *correctness* and *verifiability*).

8.4.1 Security Requirements

8.4.1.1 Verifiability

According to the considered entity, we can distinguish two types of verifiability, namely individual and universal. If both properties are satisfied and all voters are assured that the ballots they cast encrypt their real choices, then the scheme is said to provide *end-to-end* verifiability. Hereinafter, we briefly define both categories of verifiability.

Individual verifiability. Roughly speaking, individual verifiability enables the voter to verify that her ballot has been included in the election Web Bulletin Board (WBB), which is publicly accessible.

Universal verifiability. Informally, universal verifiability ensures that (1) only valid votes (*i.e.* votes cast by legitimate voters) are considered when computing vote outcome, and (2) the election result is consistent with the ballots published on the WBB (*i.e.* only invalid ballots have been discarded). As its name implies, this property may be checked by everyone.

8.4.1.2 Eligibility

Roughly speaking, *eligibility* ensures that the following two statements are met: (1) only eligible voters can cast a vote, and (2) every voter can cast one, and only one, vote. Thereby, ballot stuffing is avoided. It is noteworthy to mention that everyone should be able to check these two statements.

8.4.2 Privacy Requirements

8.4.2.1 Coercion-Resistance

Roughly speaking, *coercion-resistance* ensures that a voter can deceive the coercer about her true vote intention by making him believe that she behaved as instructed while it was actually not the case. Furthermore, it guarantees that no voter can obtain any information proving for whom she voted (this is known as *receipt-freeness*). Thereby, both coercion and vote-selling can be avoided. More formally, as defined by Juels, Catalano and Jakobsson [JCJ05], the coercion-resistance experiment between a PPT static[1] adversary $\mathcal{A}$ and a voter $\mathcal{V}_O$ targeted by $\mathcal{A}$ for a coercion attack is detailed in Figure 8.1 where:

- n_V, n_C and n_A respectively denote the total number of eligible voters, the number of candidates and the number of corrupted voters (*i.e.* under the control of the adversary $\mathcal{A}$). Thus, the number of uncertain votes (*i.e.* honest ones) is $n_U = n_V - n_A - 1$.
- V and $\vec{X}$ respectively denote the set of corrupted voters and the election outcome;
- D_{n,n_C} denotes a probability distribution that models the state of knowledge of the adversary about honest voters' intentions;
- $\texttt{fakekey}(pk_R, sk_j, pk_j)$ is a function that takes as input the public key of the registration authorities and the private/public key pair (*i.e.* voting credential) of the voter j. It outputs a fake key $\tilde{sk}$;
- $B_j = \texttt{Voting}\{(sk_j, pk_T, n_C, \beta)\}$ represents a ballot cast using the voting key sk_j and according to the distribution D_{n,n_C}, and where the selected choice is β.

More precisely, a coin is flipped and its outcome is represented by a bit b. The behavior of the coerced voter $\mathcal{V}_O$ depends on the value of b. If $b = 0$, then $\mathcal{V}_O$ (who is modeled as a function that selects a ballot either from a slate representing all the n_C candidates or a blank ballot denoted by $\varnothing$) casts a ballot encrypting the choice β (which was specified by $\mathcal{A}$) and provides $\mathcal{A}$ with a fake voting key (*i.e.* she attempts to evade the coercion attack). If $b = 1$, the voter provides $\mathcal{A}$ with her "real" voting key and does not cast any ballot (*i.e.* she submits to $\mathcal{A}$'s coercion). $\mathcal{A}$'s goal is to guess the value of b (*i.e.* determine if $\mathcal{V}_O$ has, indeed, cast a ballot or not).

The adversary's advantage $\texttt{Adv}^{\text{c-resist}}_{ES,\mathcal{A}}$ is defined as

$$\texttt{Adv}^{\text{c-resist}}_{ES,\mathcal{A}} = |\texttt{Pr}[\texttt{Exp}^{\text{c-resist}}_{ES,\mathcal{A}} = 1] - \texttt{Pr}[\texttt{Exp}^{\text{c-resist-ideal}}_{ES,\mathcal{A}} = 1]| \tag{8.1}$$

where $\texttt{Exp}^{\text{c-resist-ideal}}_{ES,\mathcal{A}}$ is the ideal coercion-resistance experiment depicted in Figure 8.2. In the latter, the adversary denoted by $\mathcal{A}'$ cannot cast ballots by himself but rather enumerates the choices of corrupted voters. Furthermore, all what $\mathcal{A}'$ sees is the vote outcome. The experiment $\texttt{Exp}^{\text{c-resist-ideal}}_{ES,\mathcal{A}}$ involves an *ideal* function denoted $\texttt{ideal} - \texttt{Tallying}$, which tallies ballots cast by honest voters in a normal manner and tallies the ballots cast by $\mathcal{A}'$ in a *special way*. Indeed, for each ballot $\mathcal{A}'$ has cast, it determines the underlying private key sk_i and if it turns out that sk_i

[1] He selects the voters to corrupt before the execution of the protocol.

$\mathtt{Exp}^{\text{c-resist}}_{ES,\mathcal{A},}(1^k, n_V, n_A, n_C)$

1. $\mathcal{BB} \leftarrow \varnothing$; $V \leftarrow \mathcal{A}$(voter names)
2. For $i \in \{1, \ldots, n_V\}$, $(sk_i, pk_i) \leftarrow \mathtt{register}(sk_R, i)$
3. $(\beta, j) \leftarrow \mathcal{A}(\{sk_i\}_{i \in V})$
4. If $|V| \neq n_A$ or $j \notin \{1, 2, \ldots, n_V\} \setminus V$ or $\beta \notin \{1, 2, \ldots, n_C\} \cup \varnothing$, then return 0
5. $b \in \{0, 1\}$
6. If $b = 0$, then $\tilde{sk} \leftarrow \mathtt{fakekey}(pk_R, sk_j, pk_j)$ and $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{B_j\}$ where $B_j = \mathtt{Voting}\{(sk_j, pk_T, n_C, \beta)\}$ else, $\tilde{sk} \leftarrow sk_j$
7. For $i \in \{1, \ldots, n_V\} \setminus V$ and such that $i \neq j$, $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{B_i\}$ where $B_i = \mathtt{Voting}\{(sk_i, pk_T, n_C, D_{n_U, n_C})\}$
8. $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{\tilde{B}\} \cup \{B_k\}_{k \in V}$ where $\tilde{B} = \mathcal{A}(\tilde{sk}, \mathcal{BB})$ and $B_k = \mathtt{Voting}\{(sk_k, pk_T, n_C, D_{n_U, n_C})\}$
9. $(\vec{X}, P) \leftarrow \mathtt{Tallying}(sk_T, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V})$
10. $b' \leftarrow \mathcal{A}(\vec{X}, P)$
11. If $b' = b$, then return 1.
12. Return 0.

Figure 8.1: Coercion-Resistance Security Experiment

is not assigned to a corrupted user, then the corresponding vote is not counted. Depending on the value b, $\mathtt{ideal} - \mathtt{Tallying}$ proceeds as follows: if $b = 0$, $\mathtt{ideal} - \mathtt{Tallying}$ discards ballots cast using $\tilde{sk}$; if $b = 1$, $\mathtt{ideal} - \mathtt{Tallying}$ includes in the final tally a ballot cast using $\tilde{sk}$.

An election scheme ES is coercion-resistant if, for any adversary $\mathcal{A}$, any parameters n_U and n_C, and any distribution D_{n_U, n_C}, the adversary's advantage is negligible. That is, $\mathcal{A}$ learns nothing more than the election outcome $\vec{X}$.

$\mathtt{Exp}^{\text{c-resist-ideal}}_{ES,\mathcal{A}}(1^k, n_V, n_A, n_C)$

1. $\mathcal{BB} \leftarrow \varnothing$; $V \leftarrow \mathcal{A}'$(voter names)
2. For $i \in \{1, \ldots, n_V\}$, $(sk_i, pk_i) \leftarrow \mathtt{register}(sk_R, i)$
3. $(\beta, j) \leftarrow \mathcal{A}'()$
4. If $|V| \neq n_A$ or $j \notin \{1, 2, \ldots, n_V\} \setminus V$ or $\beta \notin \{1, 2, \ldots, n_C\} \cup \varnothing$, then return 0
5. $b \in \{0, 1\}$
6. If $b = 0$, then $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{B_j\}$ where $B_j = \mathtt{Voting}\{(sk_j, pk_T, n_C, \beta)\}$
7. $\tilde{sk} \leftarrow sk_j$
8. For $i \in \{1, \ldots, n_V\} \setminus V$ and such that $i \neq j$, $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{B_i\}$ where $B_i = \mathtt{Voting}\{(sk_i, pk_T, n_C, D_{n_U, n_C})\}$
9. $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{\tilde{B}\} \cup \{B_k\}_{k \in V}$ where $\tilde{B} = \mathcal{A}(\tilde{sk}, \mathcal{BB})$ and $B_k = \mathtt{Voting}\{(sk_k, pk_T, n_C, D_{n_U, n_C})\}$
10. $(\vec{X}, P) \leftarrow \mathtt{ideal} - \mathtt{Tallying}(sk_T, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V})$
11. $b' \leftarrow \mathcal{A}'(\vec{X}, P)$
12. If $b' = b$, then return 1.
13. Return 0.

Figure 8.2: Coercion-Resistance-Ideal Security Experiment

8.5 Our Coercion Resistant Electronic Voting Scheme

In this section, we first provide an overview of our coercion-resistant voting scheme. Then, we detail its different phases while explaining how it enables multiple elections as well as credentials revocation.

8.5.1 Overview

To be able to vote, an eligible voter must first register by proving her identity. Once done, she obtains a unique and valid credential σ that is associated to a secret s only known to her (*i.e.* the voter). This credential, which is made publicly available, is issued by the registration authorities. Using it, as well as the associated secret, the voter can generate a ballot B that contains her choice in an encrypted form. More precisely, the ballot consists of a randomized version of the voter's credential, the ciphertext of her choice as well as two NIZKPKs proving the validity of the ballot. Upon its generation, the ballot is cast via an anonymous channel. If the voter is under coercion, she can cast a fake ballot using an invalid secret s' without the adversary being able to distinguish it from a valid one. Before tallying votes, some pre-verifications must be carried out so as to remove both erroneous ballots (*i.e.* ballots with invalid ZKPKs) as well as duplicate votes. Once done, vote tallying may be performed. To do so, the talliers first send the ballots that got through the pre-verification phase to a verifiable Mix-net [Cha81, JJR02] and then anonymously identify valid ones (*i.e.* ballots published with valid credentials). Finally, they cooperatively decrypt the associated ciphertexts to recover valid votes and publish the election result on the WBB.

8.5.2 Our Efficient Coercion-resistant Voting Scheme

8.5.2.1 Setup

Let O be the set of eligible options (candidates) and $v \in O$, one of these options. Let $\mathbb{G}$ be a cyclic group with a prime order p and $g_1, g_2, o \in \mathbb{G}$ be three generators such that o is the public generator selected for this election. The talliers share the threshold Modified ElGamal key pair (sk_T, pk_T) where $sk_T = (y_1, y_2) \in_R \mathbb{Z}_p^*$ and $pk_T = (g_1, g_2, h = g_1^{y_1} g_2^{y_2})$. As for the registrars, they jointly select and share a secret key $sk_R = (x_0, x_1) \in_R \mathbb{Z}_p^{*2}$ associated to the public values $(C_{x_0}, X_1 = h^{x_1})$ where $C_{x_0} = g^{x_0} h^x$ such that $x \in_R \mathbb{Z}_p^*$. This key is also shared among talliers.

8.5.2.2 Registration

Once her eligibility proved (using her electoral card for example), the voter obtains a unique and valid voting credential σ generated as follows. After cooperatively choosing $s \in_R \mathbb{Z}_p$ and $u \in_R \mathbb{G}\backslash\{1\}$, the registrars jointly compute $\sigma = (u, u')$ where $u' = u^{x_0 + s x_1}$. It is then provided, through an untappable channel, to the voter along with the secret value s and a Designated Verifiable Proof (DVP) that σ is a valid credential on s. As previously mentioned in Section 3.3.5.1, a DVP can only convince the corresponding voter and nobody else. So, it is completely useless in case of coercion and even for vote-selling. Concurrently, the credential σ is stored in the database DB, which contains all the valid credentials, while s is kept as a secret only known to the voter.

In case of coercion, the voter would deceive the coercer by revealing her credential along with a fake value s' without the coercer noticing it. Indeed, as shown hereinafter in Lemma 8.1, under the DDH assumption, the coercer cannot decide whether s' is valid with respect to the voter's credential σ or not (*i.e.* decide if a given triplet $(s', u, u' = u^{x_0 + s x_1})$ is a valid MAC on s or not). It is worth mentioning that the generic technique proposed in [AT13] can be subsequently used

so as to detect any vote cast with a valid, but illegitimate, credential that is computed by a set of colluding malicious registrars.

Lemma 8.1. *Under the* DDH *assumption, it is infeasible to decide whether* $s' \stackrel{?}{=} s \bmod p$ *from* s', $C_{x_0} = g^{x_0}h^x$, $X_1 = h^{x_1}$, $u = h^b$, $u' = u^{x_0+sx_1}$ *where* s, s', x, x_0, x_1, $b \in_R \mathbb{Z}_p^*$ *and* g, h *are two random generators of* $\mathbb{G}$.

Proof (sketch). Suppose that we have an oracle deciding whether $s' \stackrel{?}{=} s \bmod p$, given s', $X_1 = h^{x_1}$, $C_{x_0} = g^{x_0}h^x$, $u = h^b$, $u' = u^{x_0+sx_1}$, for $s, s', x, x_0, x_1, b \in_R \mathbb{Z}_p^*$ and g, h are two random generators of $\mathbb{G}$. Then, we show how to decide whether $c \stackrel{?}{=} x_1 b \mod p$ given $h, \alpha = h^{x_1}, \beta = h^b$ and $\gamma = h^c$ for $x_1, b, c \in_R \mathbb{Z}_p^*$, hence contradicting the DDH assumption.

The reduction is as follows. Set $C_{x_0} = g^{x_0}h^x$ for $x_0, x \in_R \mathbb{Z}_p^*$, choose $s' \in_R \mathbb{Z}_p^*$ and give $s', C_{x_0}, X_1 = \alpha, u = \beta, u' = u^{x_0}\gamma^{s'}$ to the oracle. We have the following two cases:

- Case 1. If $c = x_1 b \mod p$, then $u' = u^{x_0+s'x_1}$.
- Case 2. If $c \neq x_1 b \mod p$, then $c = x_1 b(1 + c')$ for some $c' \neq 0 \mod p$ (since $x_1 \neq 0$ and $b \neq 0$) and $u' = u^{x_0+sx_1}$ with $s = s'(1 + c') \neq s'$ (since $s' \neq 0$).

Therefore, given $s', C_{x_0}, X_1 = \alpha, u = \beta, u'$, the oracle will tell whether $s' \stackrel{?}{=} s$, from which we can decide whether $c \stackrel{?}{=} x_1 b$.

In the particular case where $s = 0$, even a computationally unbounded adversary will not be able to figure out whether $u' \stackrel{?}{=} u^{x_0}$. This is due to the fact that the Pedersen's commitment $C_{x_0} = g^{x_0}h^x$ perfectly hides the value x_0. □

8.5.2.3 Voting (First Election)

To vote in a first election, the voter first picks a random $r \in_R \mathbb{Z}_p$ and generates $\sigma_r = (u^r, u'^r) = (w, w')$, a randomized version of her credential σ. Then, she chooses her candidate $v \in O$ and casts her vote which consists of the ballot $B = \langle E_T[v], w, w', E_T[w^s], o^s, P\rangle$ where P is a set of NIZKPKs ensuring that the ballot is well formed whereas $E_T[v]$ and $E_T[w^s]$ denote the Modified ElGamal encryptions of respectively v and w^s using the talliers' public key pk_T. As for P, it includes both $\pi_1 = PoK\{\alpha : B_4 = E_T[w^\alpha] \wedge B_5 = o^\alpha\}$ related to the knowledge of s and $\pi_2 = PoK\{\beta : B_1 = E_T[\beta] \wedge \beta \in O\}$ proving that v belongs to the set O.

8.5.2.4 Pre-Verification

This phase should be performed by Talliers prior to the tallying one detailed later on. It aims to verify votes posted on the WBB so as to remove ballots with invalid ZKPKs as well as duplicate votes. It is, however, worth mentioning that, during this phase, ballots with invalid credentials are not discarded yet. More precisely, this phase is composed of the following four steps:

1. *Verifying proofs.* For each posted ballot, the proofs P are verified to remove ballots with invalid proofs.
2. *Removing duplicates.* By comparing all o^s values, duplicates votes (*i.e.* ballots published using the same secret s) are removed. The policy, in this case, could be to keep the last one.
3. *Reconstruction of the credential.* For each ballot, the talliers cooperatively compute the modified ElGamal ciphertext $E_T[w]$ of w. Owing to ElGamal homomorphic property and through the use of $E_T[w]$ and $E_T[w^s]$ as well as the shared secret values x_0 and x_1, the ciphertexts $E_T[w^{x_0}]$ and $E_T[(w^s)^{x_1}]$ can be jointly computed. Thereby, the talliers can

compute $E_T[w^{x_0+sx_1}] = E_T[w^{x_0}] \cdot E_T[w^{sx_1}]$. By dividing the second component of this ciphertext by w', they obtain $C = E_T[w^{x_0+sx_1}]/w'$. If the credential $\sigma_r = (w, w')$ is valid, then C should be equal to $E_T[1]$, a ciphertext of 1.

4. *PET pre-test.* In this last step, a Plaintext Equivalence Test (PET)[2] [JJ00] is performed on credentials. To this end, C is cooperatively raised to a random value $\alpha \in_R \mathbb{Z}_p$. For a valid credential σ, $D = C^{\alpha}$ should be equal to $E_T[1^{\alpha}] = E_T[1]$. Note that D is still kept encrypted to prevent any information leakage, especially in case of coercion.

8.5.2.5 Tallying

To compute election results, the talliers perform the following three steps:

1. *Mixing tuples.* The tuples $\langle D, E_T[v]\rangle$ that succeeded all pre-verifications are sent to a verifiable Mix-net (which takes as input a set of ballots and shuffle them). The output is then published on the WBB.

2. *Identifying valid votes.* For each tuple, the ciphertext D is jointly decrypted. If the plaintext is equal to 1, then the credential σ_r and the associated ballot are considered as valid. Otherwise, the ballot is said invalid and is, thus, discarded.

3. *Decrypting and counting votes.* Finally, for each remaining valid ballot, $E_T[v]$ is cooperatively decrypted in order to count the votes. The obtained result is then published on the WBB.

For subsequent elections, or in the case where some voters are no longer eligible to vote, the authorities should be able to update eligible voters' credentials without requiring them to register again. In what follows, we explain how to efficiently update the credentials through the use of our sequential aggregate MAC scheme introduced in Section 4.3.

8.5.3 Multiple Elections and Credentials Revocation.

For every new election, the registrars must generate both a specific election identifier and a new pair of keys. For the ith election, this pair is defined as $(x_i, X_i = h^{x_i})$ where $x_i \in_R \mathbb{Z}_p^*$ is shared among registrars and talliers. Hereinafter, we detail the case of a second election identified by e_I and which associated key pair is $(x_2, X_2 = h^{x_2})$.

For each initial credential $\sigma = (u, u') \in DB$ belonging to an eligible voter, the registrars jointly select a random value $t \in_R \mathbb{Z}_p$, compute $\sigma_2 = (u^t, (u'u^{e_I x_2})^t) = (w, w' = w^{x_0+sx_1+e_I x_2})$ and update DB. Then, the new database is published to enable eligible voters to get their new credential. These changes are irrelevant except for the pre-verification phase whose third step is henceforth defined as follows:

- *Reconstruction of the credential.* First, the talliers cooperatively encrypt w to get $E_T[w]$. Then, as previously and thanks to ElGamal homomorphic property, they jointly compute the three ciphertexts: $E_T[w^{x_0}]$, $E_T[w^{sx_1}]$ and $E_T[w^{e_I x_2}]$ using e_I, $E_T[w]$ and $E_T[w^s]$ as well as their shared secret keys x_0, x_1 and x_2. Thereby, the talliers can compute $E_T[w^{x_0}] \cdot E_T[w^{sx_1}] \cdot E_T[w^{e_I x_2}] = E_T[w^{x_0+sx_1+e_I x_2}]$. By dividing the second component of the ciphertext by w', they obtain $C = E_T[w^{x_0+sx_1+e_I x_2}]/w'$. For a valid credential, C should be equal to $E_T[1]$, a ciphertext of 1.

[2] A PET aims to check if two ciphertexts correspond to a same plaintext or not, without revealing it.

8.6 Security Analysis

In this section, we provide proof sketches ensuring that our remote electronic voting system satisfies the security and privacy requirements defined in Section 8.4.

8.6.1 Verifiability

Theorem 8.2. *Our voting scheme provides the* end-to-end verifiability *requirement.*

Proof (sketch). We recall that a scheme ensures end-to-end verifiability if ballots are cast as intended and both individual and universal verifiability are satisfied.

- *Individual Verifiability*: Recall that, during the registration phase, the voter is provided with a DVP which ensures the validity of her credential. Besides, since she knows (w, w', o^s) where $o \in \mathbb{G}$ is the generator associated to the election, $w = u^r$ and $w' = (u')^r$, the voter can search in the WBB whether there is a ballot containing the same values (*i.e.* o^s, w and w'). Thereby, she can verify if her ballot has been taken into account in the WBB or not. Thus, our scheme ensures individual verifiability.
- *Universal Verifiability*: Every step of the tallying phase (PET, Mix-net, ZKPK and duplicate removal) is publicly verifiable. Thus, anyone can check that the election outcome corresponds to the ballots published on the WBB and, in particular, that only invalid ballots with invalid credentials have been discarded. Therefore, our scheme provides universal verifiability.

Thus, since we assumed that votes are cast as intended, we can conclude that our voting scheme ensures the end-to-end verifiability requirement. □

8.6.2 Eligibility

Theorem 8.3 (Eligibility)**.** *Our voting scheme satisfies the* eligibility *requirement, in the random oracle model, under the assumption that* $\mathsf{MAC}_{\mathsf{GGM}}$ *is UF-CMVA secure.*

Proof (sketch). Intuitively, the first condition of eligibility (*i.e.* only an eligible voter can cast a vote) follows from the unforgeability of the $\mathsf{MAC}_{\mathsf{GGM}}$ and the fact that ballots with invalid proofs or credentials are discarded. The second condition (*i.e.* a voter can only cast one vote) is also satisfied since duplicates are removed in the second step of the pre-verification phase. Consequently, only one vote per credential (valid or fake) will be processed during the tallying phase. Therefore, our voting scheme ensures the eligibility requirement.

□

8.6.3 Coercion-Resistance

Theorem 8.4 (Coercion-Resistance)**.** *Our voting scheme satisfies the* coercion-resistance *requirement, in the random oracle model, under the* DDH *assumption.*

Proof (sketch). Recall that the adversary's goal is to guess the behavior of the coerced voter $\mathcal{V}_O$ during the run of the election system. That is, if (1) $\mathcal{V}_O$ provided him with a fake voting key and cast a ballot with choice β, or (2) $\mathcal{V}_O$ has revealed her "real" (*i.e.* valid) voting key and abstained from voting. To prove that an election system provides coercion-resistance, we must show that $\mathcal{A}$ succeeds with probability negligibly close to that of a PPT adversary $\mathcal{A}'$ interacting with an *ideal* election system. In particular, $\mathcal{A}'$ is a passive adversary and his only input is $\vec{X}$: the final tally of votes cast by honest voters, and the number of discarded ballots (*i.e.* ballots built with invalid credentials).

Hereinafter, we prove coercion-resistance using Shoup's game hopping technique [Sho04]. We only provide a rather high level description of the initial game (Game 0) along with brief descriptions of the transitions between successive games.

Game 0: It corresponds to the real attack game with respect to an efficient adversary $\mathcal{A}$.

1. **Setup**: The Simulator $\mathcal{S}$ randomly chooses the system public parameters $(\mathbb{G}, p, h, g, o)$ and selects a randomized candidate slate $O = \{v_i\}_{i=1}^{n_C}$ as well as two random values $x_0, x_1 \in_R \mathbb{Z}_p^*$ as the registrars secret key sk_R. The values $C_{x_0} = g^{x_0}h^x$ and $X_1 = h^{x_1}$, where $x \in_R \mathbb{Z}_p^*$, are published. $\mathcal{S}$ also randomly picks the talliers private key sk_T and computes the associated public key pk_T.

2. **Register**(ID_{v_i}): $\mathcal{S}$ uses the private key $sk_R = (x_0, x_1)$ to generate a credentials $(u, u' = u^{x_0+s_i x_1})$ associated to a random secret s_i. Note that $\mathcal{S}$ keeps a list $\mathtt{L}$ of all generated credentials and their associated secrets.

3. **Adversarial corruption**: The adversary $\mathcal{A}$ selects V, the set of n_A voters that he wants to corrupt, the voter j to be coerced as well as the target vote β. If any of them is not correctly selected (for instance, $|V| > n_A$), then the simulation aborts.

4. **Coin flip**: A coin $b \in \{0, 1\}$ is flipped.

5. **Credential disclosure**: $\mathcal{S}$ provides $\mathcal{A}$ with the set of credentials $\{\sigma_i\}_{i \in V}$ and associated secrets as well as the credential $\sigma_j = (u_j, u'_j)$ and an associated secret s for the targeted voter j (*i.e.* the voter who will be coerced). If $b = 1$, then $s = s_j$, otherwise s is chosen at random.

6. **Honest voter simulation**: $\mathcal{S}$ uses the public key $pk_T = (g_1, g_2, h)$ of the modified ElGamal cryptosystem to generate the ballots of the honest voters. It also provides the NIZK proofs.

7. **Adversarial ballot posting**: $\mathcal{A}$ posts a set of ballots B_0.

8. **Decrypting $\mathcal{A}$'s ballots**: $\mathcal{S}$ first verifies the NIZK proofs associated to each ballot. Let B_1 be the set of ballots with valid proofs. For each credential whose associated secret is known to $\mathcal{A}$, $\mathcal{S}$ decrypts the ballot.

9. **Pre-verification simulation**: $\mathcal{S}$ simulates the behavior of honest tallying authorities as follows:

   - **Proof checking**: Let $E_0 = B_1 \cup \mathcal{A}_0$ where A_0 are the ballots cast by honest voters. $\mathcal{S}$ behaves as an honest tallier by discarding ballots with invalid proofs. Let E_1 be the set of ballots with valid proofs.
   - **Removing duplicates**: $\mathcal{S}$ simulates the elimination of duplicate ballots. Let E_2 be the resulting ballot list.
   - **Credential reconstruction**: As it holds the keys (x_0, x_1), $\mathcal{S}$ can reconstruct credentials through the use of ElGamal homomorphic property.
   - **PET pre-test**: $\mathcal{S}$ performs plaintext equivalent test on credentials.

10. **Tallying simulation**: Using the key sk_T, $\mathcal{S}$ perfectly simulates this phase.

At the end of the game, $\mathcal{A}$ outputs a bit b'. Let S_0 denote the event that $b = b'$ in this game and S_i be the event that $b = b'$ in Game i. We have

$$\mathtt{Pr}[S_0] = \mathtt{Pr}[\mathtt{Exp}_{ES,\mathcal{A}}^{\text{c-resist}} = 1] \tag{8.2}$$

Game 1: This is the same game as **Game 0** except that we modify the way the challenger generates the ballots of honest voters. The latter are generated as follows. $\mathcal{S}$ chooses a random value τ in $\mathbb{Z}_p^*$ and computes $h_1 = g_1^\tau$ and $h_2 = g_2^\tau$. Thus, the quadruple (g_1, g_2, h_1, h_2) is a DH quadruple. Using this quadruple and h, $\mathcal{S}$ generates the ballots of honest voters. More precisely, it creates each ciphertext of the ballot as follows: for instance, to create the ciphertext encrypting the voting option $v_k = g_1^{t_k}$, $\mathcal{S}$ chooses $s_i \in_R \mathbb{Z}_p^*$ at random and sets $(c_1 = h_1^{s_i}, c_2 = h_2^{s_i}, c_3 = h_1^{s_i y_1} h_2^{s_i y_2} v_k)$. Since the quadruple (g_1, g_2, h_1, h_2) is a DH quadruple, then this ciphertext has the right distribution. Thus, we have

$$\mathtt{Pr}[S_1] = \mathtt{Pr}[S_0]. \tag{8.3}$$

Game 2: This is the same game as **Game 1** except that for each ballot in E_1, $\mathcal{S}$ runs the extractor for the associated proofs of knowledge to extract the secret s associated to the randomized credential (w, w'). If the extractor fails, then $\mathcal{S}$ outputs $\perp$ and aborts. Let us denote by F this failure event. By the Difference Lemma [Sho04], we have

$$|\mathtt{Pr}[S_2] - \mathtt{Pr}[S_1]| \leq \mathtt{Pr}[F] \tag{8.4}$$

The probability $\mathtt{Pr}[F]$ is bounded by the knowledge error of the proofs of knowledge (which is a negligible quantity denoted by $v(k)$). Therefore, we have

$$|\mathtt{Pr}[S_2] - \mathtt{Pr}[S_1]| \leq v(k) \tag{8.5}$$

Note that we tacitly assumed that the ballots are cast sequentially rather than concurrently so as to enable the rewinding of $\mathcal{A}$ in order to extract the secrets. In a concurrent setting, we could use verifiable encryption.

Game 3: This is the same game as **Game 2** except that $\mathcal{S}$ outputs $\perp$ and aborts if at least one of the extracted secret and associated credential (*i.e.* u, u', s where $u' = u^{x_0 + s x_1}$) have not been issued by $\mathcal{S}$ or belong to an honest voter. That is, $\mathcal{S}$ aborts if $\mathcal{A}$ succeeds in forging a valid credential and its associated secret. We recall that the credentials correspond to a $\mathsf{MAC}_{\mathsf{GGM}}$ on a secret s. Since the $\mathsf{MAC}_{\mathsf{GGM}}$ scheme, which is due to Chase *et al.* [CMZ14], is proven UF-CMVA, we have that

$$|\mathtt{Pr}[S_3] - \mathtt{Pr}[S_2]| \leq \mathtt{Adv}^{\mathsf{MAC}_{\mathsf{GGM}}}_{UF-CMVA}(1^k) \tag{8.6}$$

Game 4: This is the same game as Game 3 except that, instead of using a DDH quadruple, $\mathcal{S}$ uses a random quadruple (g_1, g_2, h_1, h_2) to generate the ballots of honest users. That is, h_1 and h_2 are defined as $h_1 = g_1^\tau$ and $h_2 = g_2^\delta$ with $\tau \neq \delta \pmod p$. Under the DDH assumption, $\mathcal{A}$ cannot distinguish this change. Indeed, we can easily construct a DDH distinguisher $\mathcal{D}_1$ with an advantage of solving the DDH problem satisfying

$$|\mathtt{Pr}[S_4] - \mathtt{Pr}[S_3]| \leq \mathtt{Adv}^{\mathcal{D}_1}_{\mathsf{DDH}}(1^k) \tag{8.7}$$

Note that, with such a change, the view produced by the simulation does not give any information (in a strong information theoretic sense) about the votes posted by honest parties (as modified ElGamal is semantically secure). Indeed, for any $\tilde{v} = g_1^{\tilde{t}}$, there exists a private key $(\tilde{y}_1, \tilde{y}_2)$ corresponding to the public key h such that the decryption algorithm on input a ciphertext (c_1, c_2, c_3) outputs $\tilde{v}$. Let z denote the discrete logarithm of h in base g_1 (*i.e.* $h = g_1^z = g_1^{\tilde{y}_1} g_2^{\tilde{y}_2}$), λ be the discrete logarithm of g_2 in the base g_1 (*i.e.* $g_2 = g_1^\lambda$), s the common discrete logarithm of c_1, c_2 in bases h_1 and h_2, and γ the discrete logarithm of c_3 in base g_1 (*i.e.* $c_3 = g_1^\gamma$). To find the private key $(\tilde{y}_1, \tilde{y}_2)$, we have to solve a system of linear equations:

$$\tilde{y}_1 + \lambda \tilde{y}_2 = z \tag{8.8}$$

$$\tau s\tilde{y}_1 + \lambda\delta s\tilde{y}_2 = \gamma - \tilde{t} \tag{8.9}$$

The equation (8.8) results from the fact that we should have $h = g_1^z = g_1^{\tilde{y}_1} g_2^{\tilde{y}_2}$ whereas equation (8.9) is due to the fact that (c_1, c_2, c_3) is an encryption of $\tilde{v}$ and, in particular, that $c_3 = h_1^{s\tilde{y}_1} h_2^{s\tilde{y}_2}\tilde{v}$. The determinant of this system is equal to $\lambda s(\delta - \tau)$ which is different from 0, since (g_1, g_2, h_1, h_2) is not a DH quadruple (and thus $\tau \neq \delta \pmod p$) and by definition, λ and s are different from 0 $(\text{mod } p)$. Consequently, there must exists a solution to the above system.

Game 5: In case $b = 0$, instead of sending a fake credential to $\mathcal{A}$, $\mathcal{S}$ sends him a valid one (but the ballot cast by $\mathcal{A}$ using this credential will not be included in the final tally). Under the DDH assumption, $\mathcal{A}$ cannot distinguish this change. Thus, we have

$$|\mathtt{Pr}[S_5] - \mathtt{Pr}[S_4]| \leq \mathtt{Adv}_{\mathsf{DDH}}(1^k) \tag{8.10}$$

The view produced by this last game gives no information, in a strong information theoretic sense, about the votes posted by honest voters. The only information that the adversary obtains is the final tally $\overrightarrow{X}$ of votes cast by the honest voters as well as the number of ballots discarded due to invalid proofs. Furthermore, in both cases (*i.e.* when $b = 0$ or $b = 1$), $\mathcal{A}$ receives a valid credential. Thus, we have

$$\mathtt{Pr}[S_5] = \mathtt{Pr}[\mathtt{Exp}^{\text{c-resist-ideal}}_{ES,\mathcal{A}} = 1] \tag{8.11}$$

It results from the above that

$$\begin{aligned}\mathtt{Adv}^{\text{c-resist}}_{ES,\mathcal{A}} &= |\mathtt{Pr}[\mathtt{Exp}^{\text{c-resist}}_{ES,\mathcal{A}} = 1] - \mathtt{Pr}[\mathtt{Exp}^{\text{c-resist-ideal}}_{ES,\mathcal{A}} = 1]| \\ &= |\mathtt{Pr}[S_0] - \mathtt{Pr}[S_5]|\end{aligned}$$

Thus, we can give the following upper bound for the adversary's advantage

$$\mathtt{Adv}^{\text{c-resist}}_{ES,\mathcal{A}} \leq \sum_{j=0}^{j=4} |\mathtt{Pr}[S_{j+1}] - \mathtt{Pr}[S_j| \leq v(k) + \mathtt{Adv}^{\mathsf{MAC}_{\mathsf{GGM}}}_{UF-CMVA}(1^k) + 2\ \mathtt{Adv}_{\mathsf{DDH}}(1^k)$$

Therefore, we can conclude that our electronic voting system satisfies the coercion-resistance requirement in the ROM under the DDH assumption and the UF-CMVA security of the $\mathsf{MAC}_{\mathsf{GGM}}$. □

8.7 Conclusion

In this chapter, we proposed an efficient coercion-resistant electronic voting scheme which has linear time complexity while supporting both credentials revocation as well as multiple elections. To do so, we rely on credentials generated using Chase *et al.* algebraic MAC scheme [CMZ14]. In order to allow revocation and multiple elections, voters' credentials are updated using our sequential aggregate MAC scheme introduced in Section 4.3.

Chapter 9

Conclusion

In this thesis, we aimed to combine two security properties, namely authentication and privacy, while dealing with resource constrained environments such as SIM cards. In this context, we introduced five cryptographic primitives, namely a partially blind signature scheme, an algebraic Message Authentication Code (MAC) scheme and a sequential aggregate MAC scheme as well as two Direct Anonymous Attestation schemes. Based on them, we designed four practical privacy-preserving protocols that are both efficient and suited for SIM cards.

First, we looked into the privacy issue of electronic payment systems for applications such as electronic Toll (eToll) and Electric Vehicle Charging (EVC) which have already attracted a lot of interest. Unfortunately, existing proposals have some shortcomings: they are either inefficient, or unsuitable for resource constrained environments, or do not support flexible prices. Therefore, we proposed a new electronic Cash system intended for this particular kind of applications, in which the same entity acts as both merchant and bank, referred to as *private eCash*. To this end, we designed a new partially blind signature scheme which, unlike previous proposals, allows multiple presentations of the same signature in an unlinkable way. Thanks to it, our eCash ensures the necessary security and privacy requirements while being efficient, suitable for SIM cards and enabling flexible prices. Indeed, its implementation on a standard SIM card showed that a transaction can be performed in only 205 ms.

Then, we focused on anonymous credentials, and more precisely on keyed-verification anonymous credentials. The latter are the secret key analogue of traditional anonymous credentials, where users obtain a credential, then prove its possession in an unlinkable way. In this respect, we proposed a practical Keyed-Verification Anonymous Credential system that can be easily turned into an efficient public-key anonymous credential system. To build it, we designed a new algebraic MAC scheme based on Boneh Boyen signature scheme where MACs can be publicly verifiable. Our KVAC system is many times faster than Idemix and almost as efficient as Microsoft's U-Prove whilst, contrary to U-Prove, providing multi-show unlinkability. Furthermore, unlike both KVAC systems introduced at CSS 2014 [CMZ14], its presentation proof is of complexity $O(1)$ in the number of group elements. Last but not least, it only requires the issuer to hold a single private key regardless of the number of attributes.

Next, we dealt with one of the most practical applications of anonymous signatures, namely Direct Anonymous Attestation (DAA). Our goal was to design an efficient DAA scheme which requires neither pairing computations, nor computations in $\mathbb{G}_2$ or $\mathbb{G}_T$, on the platform side in order to be suitable for SIM cards. With this aim in mind, we introduced two efficient DAA schemes where all computations on the platform side can be entirely carried out by the TPM. The first proposal is based on an algebraic MAC scheme due to Chase, Meiklejohn and Zaverucha, whereas the second is built upon a pairing-free variant of the Boneh and Boyen's signature scheme. Using the latter as a building block, we proposed a privacy-preserving authentication protocol for

embedded SIM so as to cope with a real issue that has arisen at the GSM Association (GSMA). Thereby, we showed how our DAA schemes can be of particular interest for M2M applications that involve a Secure Element (SE) such as a SIM card. We also demonstrated its efficiency by implementing it on a GlobalPlatform compliant SIM card.

Finally, we considered another application of anonymous credentials, specifically remote electronic voting systems. Our main purpose was to provide an efficient coercion-resistance remote electronic voting system that supports credentials revocation (*e.g.* those of voters who are no longer eligible to vote) as well as multiple elections. To do so, we proposed an efficient sequential aggregate Message Authentication Code scheme. Using it, legitimate voters' credentials can be efficiently updated so as to enable them to vote in new elections without having to revisit the registration place. Unlike other coercion-resistant electronic voting schemes, our proposal really combines practicality and linear time complexity whilst supporting multiple elections.

Our cryptographic schemes as well as privacy-preserving protocols are formally proven to ensure the necessary security and privacy requirements under classical computational assumptions.

Perspectives. In this thesis, we introduced some improvements to a set of cryptographic primitives and privacy-preserving protocols in order to enable their use when dealing with constrained environments such as SIM cards. Unfortunately, some M2M devices (*i.e.* sensors) are not as "powerful" as resource constrained SIM cards. So, it would be interesting to investigate potential additional enhancements of these primitives so as to adapt them, if possible, for such devices. One solution would consist in delegating the heaviest computations to a more powerful entity (*e.g.* a smartphone), if such an entity is available, without decreasing the security and privacy levels of the entire solution. Still, if the latter entity is not trusted, the device must verify that the computations were correctly performed. Thus, the computational cost of the result verification ought to be lower than that of computing it.

Furthermore, another M2M application, that is raising serious privacy concerns, is smart metering. Indeed, smart meters, which are currently being rolled-out, may enable the disclosure of several personal information such as sleeping routines, the type of devices being used by the user and even the TV channel he is viewing. So, it would be paramount to study this use case and see whether our proposals could be used to protect user's privacy.

Thesis Publications

Here is the list of the papers published, at international conferences and journals, during this thesis.

International Conferences

[BBD+16] *Private eCash in Practice.* (**FC 2016**)
A. Barki, S. Brunet, N. Desmoulins, S. Gambs, S. Gharout and J. Traoré.

[BBDT16] *Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials.* (**SAC 2016**)
A. Barki, S. Brunet, N. Desmoulins and J. Traoré.

[ABBT16] *Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant.* (**VOTING 2016**)
R. Araújo, **A. Barki**, S. Brunet and J. Traoré.

International Journal

[BBGT16] *M2M Security: Challenges and Solutions.* (**IEEE COMST 2016**)
A. Barki, A. Bouabdallah, S. Gharout and J. Traoré.

Patents

[BCGT15] *Procédé d'identification anonyme d'un module de sécurité.*
A. Barki, L. Coureau, S. Gharout and J. Traoré.

Submitted on November 23, 2015 at INPI under the number 1561268.

List of Tables

List of Figures

Bibliography

[3GP15] 3GPP. ecall data transfer; in-band modem solution; general description (release 13), 3GPP TS 26.267, v13.0.0, Dec 2015.

[3GP16] 3GPP, 3rd Generation Partnership Project. *Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA), V13.0.0*, 2016.

[ABBT16] Roberto Araújo, Amira Barki, Solenn Brunet, and Jacques Traoré. Remote electronic voting can be efficient, verifiable and coercion-resistant. In Jeremy Clark, Sarah Meiklejohn, Y.A. Peter Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, pages 224–232. Springer Berlin Heidelberg, 2016.

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045, pages 136–151. Springer Berlin Heidelberg, 2001.

[AF96] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *LNCS*, pages 244–251. Springer Heidelberg, 1996.

[AFT07] Roberto Araújo, Sébastien Foulle, and Jacques Traoré. A practical and secure coercion-resistant scheme for remote elections. In David Chaum, Miroslaw Kutylowski, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Frontiers of Electronic Voting*, volume 7311, pages 330–342. Schloss Dagstuhl, Germany, 2007.

[ALF+14] Man Ho Au, J.K. Liu, Junbin Fang, Z.L. Jiang, W. Susilo, and Jianying Zhou. A new payment system for enhancing location privacy of electric vehicles. *IEEE Transactions on Vehicular Technology*, 63(1):3–18, Jan 2014.

[ALT+15] Ghada Arfaoui, Jean-François Lalande, Jacques Traoré, Nicolas Desmoulins, Pascal Berthomé, and Saïd Gharout. A practical set-membership proof for privacy-preserving NFC mobile ticketing. *Proceedings on Privacy Enhancing Technologies*, abs/1505.03048, 2015.

[AMO08] Norio Akagi, Yoshifumi Manabe, and Tatsuaki Okamoto. An efficient anonymous credential system. In Gene Tsudik, editor, *FC 2008*, pages 272–286. Springer Berlin Heidelberg, 2008.

[AT99] Giuseppe Ateniese and Gene Tsudik. *Financial Cryptography: Third International Conference, FC'99 Anguilla, British West Indies, February 22–25, 1999 Proceedings*, chapter Some Open Issues and New Directions in Group Signatures, pages 196–211. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[AT13] Roberto Araújo and Jacques Traoré. A practical coercion resistant voting scheme revisited. In *Heather, J., Schneider, S., Teague, V. (eds.) Vote-ID 2013*, volume 7985 of *LNCS*, pages 193–209. Springer, Heidelberg, 2013.

[BB04] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer Berlin Heidelberg, 2004.

[BBD+16] Amira Barki, Solenn Brunet, Nicolas Desmoulins, Sébastien Gambs, Saïd Gharout, and Jacques Traoré. Private eCash in Practice. In *20th International Conference Financial Cryptography and Data Security, FC 2016, Bridgetown, Barbados, February 22-26, 2016, Revised Selected Papers*, pages –. Springer Berlin Heidelberg, 2016.

[BBDT16] Amira Barki, Solenn Brunet, Nicolas Desmoulins, and Jacques Traoré. Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials. In *Selected Areas in Cryptography - SAC 2016 - 23nd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, pages –. Springer Berlin Heidelberg, 2016.

[BBGT16] Amira Barki, Abdelmadjid Bouabdallah, Saïd Gharout, and Jacques Traoré. M2M Security: Challenges and Solutions. *IEEE Communications Surveys Tutorials*, 18(2):1241–1254, Secondquarter 2016.

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 132–145. ACM, 2004.

[BCGT15] Amira Barki, Laurent Coureau, Saïd Gharout, and Jacques Traoré. Procédé d'identification anonyme d'un module de sécurité, patent submitted on november 23, 2015 at inpi under the number 1561268, 2015.

[BF01] Dan Boneh and Matt Franklin. *Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, chapter Identity-Based Encryption from the Weil Pairing, pages 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[BFG+13] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N. P. Smart, and B. Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Secur.*, 12(3):219–249, June 2013.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 416–432, Berlin, Heidelberg, 2003. Springer-Verlag.

[BL07] Ernie Brickell and Jiangtao Li. Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, pages 21–30. ACM, 2007.

[BL10] E. Brickell and J. Li. Enhanced privacy id from bilinear pairing for hardware authentication and attestation. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 768–775, Aug 2010.

[BL11] Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. *IJIPSI*, 1(1):3–33, 2011.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. CCS '13, pages 1087–1098. ACM, 2013.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. *Advances in Cryptology — ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings*, chapter Short Signatures from the Weil Pairing, pages 514–532. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Proceedings of the 12th International Conference on Selected Areas in Cryptography*, SAC'05, pages 319–331, Berlin, Heidelberg, 2006. Springer-Verlag.

[BP97] Niko Baric and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. EUROCRYPT'97, pages 480–494. Springer-Verlag, 1997.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[BR96] Mihir Bellare and Phillip Rogaway. *Advances in Cryptology — EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings*, chapter The Exact Security of Digital Signatures-How to Sign with RSA and Rabin, pages 399–416. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[Bra93] Stefan A. Brands. An efficient off-line electronic cash system based on the representation problem., 1993.

[Bra94] Stefan Brands. Untraceable off-line cash in wallet with observers. In *CRYPTO '93*, pages 302–318. Springer-Verlag New York, Inc., 1994.

[BRT+10] Josep Balasch, Alfredo Rial, Carmela Troncoso, Bart Preneel, Ingrid Verbauwhede, and Christophe Geuens. PrETP: Privacy-preserving electronic toll pricing. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 5–5, 2010.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. *Topics in Cryptology – CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings*, chapter Foundations of Group Signatures: The Case of Dynamic Groups, pages 136–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–145. IEEE Computer Society, 2001.

[CCJT14] Sébastien Canard, Iwen Coisel, Amandine Jambert, and Jacques Traoré. New results for the practical use of range proofs. In Sokratis Katsikas and Isaac Agudo, editors, *Public Key Infrastructures, Services and Applications: 10th European Workshop, EuroPKI 2013*, pages 47–64. Springer Berlin Heidelberg, 2014.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Hee Jung Cheon, editors, *ASIACRYPT 2015*, pages 262–288. Springer Berlin Heidelberg, 2015.

[CDL16a] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. Cryptology ePrint Archive, Report 2016/663, 2016. http://eprint.iacr.org/2016/663.

[CDL16b] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *9th International Conference, TRUST 2016, Proceedings*, 2016.

[CDL16c] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In *Public-Key Cryptography - PKC 2016 - 19th Conference Proceedings*, pages 234–264, 2016.

[CF08] Xiaofeng Chen and Dengguo Feng. Direct anonymous attestation for next generation TPM. *JCP*, 3(12):43–50, 2008.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 209–218, New York, NY, USA, 1998. ACM.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In Moni Naor, editor, *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57. Springer Berlin Heidelberg, 2004.

[CGK+16] Veronique Cortier, David Galindo, Ralf Kuesters, Johannes Mueller, and Tomasz Truderung. Verifiability notions for e-voting protocols. Cryptology ePrint Archive, Report 2016/287, 2016. http://eprint.iacr.org/2016/287.

[CH91] David Chaum and Eugène Heyst. *Advances in Cryptology — EUROCRYPT '91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings*, chapter Group Signatures, pages 257–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

[CH11] Jeremy Clark and Urs Hengartner. Selections: Internet voting with over-the-shoulder coercion-resistance. In *FC 2011*, pages 47–61, 2011.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[Cha83] David Chaum. *Advances in Cryptology: Proceedings of Crypto 82*, chapter Blind Signatures for Untraceable Payments, pages 199–203. Springer US, Boston, MA, 1983.

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985.

[Che10] Liqun Chen. A daa scheme requiring less tpm resources. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *5th International Conference, Inscrypt 2009. Revised Selected Papers*, pages 350–365. Springer Berlin Heidelberg, 2010.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, pages 93–118. Springer Berlin Heidelberg, 2001.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *SCN 2002*, pages 268–289. Springer Berlin Heidelberg, 2003.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, pages 56–72. Springer Berlin Heidelberg, 2004.

[CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic macs and keyed-verification anonymous credentials. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1205–1216, New York, NY, USA, 2014. ACM.

[CP93] David Chaum and Torben Pryds Pedersen. *Advances in Cryptology — CRYPTO' 92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings*, chapter Wallet Databases with Observers, pages 89–105. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[CPS10] Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient daa scheme. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010. Proceedings*, pages 223–237. Springer Berlin Heidelberg, 2010.

[CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical report, 1997.

[CSS+09] Inhyok Cha, Y. Shah, A.U. Schmidt, A. Leicher, and M.V. Meyerstein. Trust in m2m communication. *Vehicular Technology Magazine, IEEE*, 4(3):69–75, Sept 2009.

[CU15] Liqun Chen and Rainer Urian. DAA-A: Direct anonymous attestation with attributes. In Mauro Conti, Matthias Schunter, and Ioannis Askoxylakis, editors, *8th International Conference, TRUST 2015, Proceedings*, pages 228–245. Springer International Publishing, 2015.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, pages 418–430. Springer Berlin Heidelberg, 2000.

[DF89] Yvo G. Desmedt and Yair Frankel. Threshold cryptosystems. In *Proceedings on Advances in Cryptology*, CRYPTO '89, pages 307–315, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[DGOW95] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs dishonest verifier in public coin zero-knowledge proofs. In Don Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings*, pages 325–338. Springer Berlin Heidelberg, 1995.

[DH06] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[DHKG11] Jeremy Day, Yizhou Huang, Edward Knapp, and Ian Goldberg. SPEcTRe: spot-checked private ecash tolling at roadside. In *WPES*, pages 61–68. ACM, 2011.

[dlPHV14] Antonio de la Piedra, Jaap-Henk Hoepman, and Pim Vullers. Towards a full-featured implementation of attribute based credentials on smart cards. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis Askoxylakis, editors, *CANS 2014*, pages 270–289. Springer International Publishing, 2014.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer Berlin Heidelberg, 2005.

[EG85] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[Eri11] Ericsson. More than 50 billion connected devices. In *284, 23-3149, Uen*, February 2011.

[ETS13] ETSI. Machine-to-machine communications (m2m); functional architecture. In *ETSI TS 102 690, V2.1.1*, October 2013.

[ETS14] ETSI. Machine-to-machine communications (m2m); mia, dia and mid interfaces. In *ETSI TS 102 921, V1.3.1*, september 2014.

[ETS16] ETSI. *Smart Cards; UICC-Terminal interface; Physical and logical characteristics V13.1.0*, 2016.

[Eur13] European Commission, Press Release. ecall: automated emergency call for road accidents mandatory in cars from 2015, June 2013.

[Eva11] Dave Evans. The internet of things, how the next evolution of the internet is changing everything. In *Cisco Internet Business Solutions Group (IBSG) white paper*, April 2011.

[FFK+11] Z.M. Fadlullah, M.M. Fouda, N. Kato, A. Takeuchi, N. Iwasaki, and Y. Nozaki. Toward intelligent machine-to-machine communications in smart grid. *Communications Magazine, IEEE*, 49(4):60–65, April 2011.

[FFS] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, pages 233–253. Springer Berlin Heidelberg, 2015.

[FP01] Pierre-Alain Fouque and David Pointcheval. *Advances in Cryptology — ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings*, chapter Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks, pages 351–368. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[FPS01] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In Yair Frankel, editor, *Financial Cryptography*, volume 1962 of *LNCS*, pages 90–104. Springer Berlin Heidelberg, 2001.

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. Cryptology ePrint Archive, Report 2009/146, 2009. http://eprint.iacr.org/.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In AndrewM. Odlyzko, editor, *Advances in Cryptology, CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Berlin Heidelberg, Santa Barbara, CA, USA, 1986.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[GF94] Hans-Georg Rück Gerhard Frey. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–874, 1994.

[Glo15] GlobalPlatform. GlobalPlatform Card Specification V2.3, 2015. http://www.globalplatform.org/specificationscard.asp, 2015.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, September 2008.

[GSM12] GSMA Mobile NFC. White Paper: Mobile NFC in Transport. http://www.uitp.org/public-transport/technology/Mobile-NFC-in-Transport.pdf, September 2012.

[GSM14a] GSMA. *SGP.01 Embedded SIM Remote Provisioning Architecture, Technical Specification, V3.1*, 2014.

[GSM14b] GSMA. *From concept to delivery: the M2M market today*, Feb 2014.

[GSM16a] GSMA. *SGP.02 Remote Provisioning Architecture for Embedded UICC, Technical Specification, V1.1*, 2016.

[GSM16b] GSMA. *SGP.21 RSP Architecture V2.0*, 2016.

[GSM16c] GSMA. *SGP.22 Remote SIM Provisioning Technical Specification; Version 1.1*, 2016.

[HT07] Emeline Hufschmitt and Jacques Traoré. Fair blind signatures revisited. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *LNCS*, pages 268–292. Springer Berlin Heidelberg, 2007.

[HZB+13] Gesine Hinterwälder, Christian T. Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. Efficient e-cash in practice: Nfc-based payments for public transportation systems. In *PETS 2013*, pages 40–59. Springer Berlin Heidelberg, 2013.

[IBM10] IBM. Specification of the identity mixer cryptographic library (revised version 2.3.0). IBM Research Report RZ 3730, 2010. http://domino.research.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/eeb54ff3b91c1d648525759b004fbbb1?OpenDocument.

[IET04] IETF: B. Aboda and L. Blunk and J. Vollbrecht and J. Carlson and H. Levkowetz. *Extensible Authentication Protocol (EAP), IETF RFC 3748*, 2004.

[IET08] IETF: P. Jayaraman and R. Lopez and Y. Ohba and M. Parthasarathy and A. Yegin. *Protocol for Carrying Authentication for Network Access (PANA) Framework, IETF RFC 5193*, 2008.

[IET10] IETF: E. Rescorla and M. Ray and S. Dispensa and N. Oskov. *Transport Layer Security (TLS) Renegotiation Indication Extension, IETF RFC 5746*, 2010.

[JCJ02] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. Cryptology ePrint Archive, Report 2002/165, 2002. http://eprint.iacr.org/2002/165.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES*, pages 61–70, 2005.

[JJ00] Markus Jakobsson and Ari Juels. *Mix and Match: Secure Function Evaluation via Ciphertexts*, pages 162–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.

[JKD12] Marek Jawurek, Florian Kerschbaum, and George Danezis. Sok: Privacy technologies for smart grids - a survey of options., 2012.

[JL00] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings*, pages 221–242, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*, pages 385–394, 2000.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. *Advances in Cryptology — EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa*, chapter Designated Verifier Proofs and Their Applications, pages 143–154. Springer Berlin Heidelberg, 1996.

[KL08] Jonathan Katz and Andrew Y. Lindell. Aggregate message authentication codes. In *Proceedings of the 2008 The Cryptopgraphers' Track at the RSA Conference on Topics in Cryptology*, CT-RSA'08, pages 155–169, Berlin, Heidelberg, 2008. Springer-Verlag.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.

[Law09] N. Lawson. Side-channel attacks on cryptographic software. *Security Privacy, IEEE*, 7(6):65–68, Nov 2009.

[LMY15] Weiwei Liu, Yi Mu, and Guomin Yang. An efficient privacy-preserving e-coupon system. In Dongdai Lin, Moti Yung, and Jianying Zhou, editors, *Information Security and Cryptology*, volume 8957 of *LCNS*, pages 3–15. Springer International Publishing, 2015.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. *Sequential Aggregate Signatures and Multisignatures Without Random Oracles*, pages 465–485. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[LRSW00] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. *Pseudonym Systems*, pages 184–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[Mau05] Ueli Maurer. *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings*, chapter Abstract Models of Computation in Cryptography, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[MDND15] Milica Milutinovic, Koen Decroix, Vincent Naessens, and Bart DeDecker. Privacy-preserving public transport ticketing system. In Pierangela Samarati, editor, *Data and Applications Security and Privacy XXIX*, volume 9149 of *LNCS*, pages 135–150. Springer International Publishing, 2015.

[Mil86] VictorS. Miller. Use of elliptic curves in cryptography. In HughC. Williams, editor, *Advances in Cryptology - CRYPTO'85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, 1986.

[MMCS11] Sarah Meiklejohn, Keaton Mowery, Stephen Checkoway, and Hovav Shacham. The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 32–32, 2011.

[MOV93] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, Sep 1993.

[MVO91] Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 80–89, 1991.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 427–437. ACM, 1990.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer Berlin Heidelberg, 1998.

[Pai99] Pascal Paillier. Low-cost double-size modular exponentiation or how to stretch your cryptoprocessor. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, pages 223–234, 1999.

[Paq13] Christian Paquin. U-Prove Technology Overview V1.1 (Revision 2). In Microsoft Technical Report, 2013. http://research.microsoft.com/pubs/166980/U-ProveTechnologyOverviewV1.1Revision2.pdf.

[PBB09] Raluca Ada Popa, Hari Balakrishnan, and Andrew J. Blumberg. VPriv: Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 335–350, 2009.

[Ped92] Torben Pryds Pedersen. *Advances in Cryptology — CRYPTO '91: Proceedings*, chapter Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, pages 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[Poi02] David Pointcheval. Practical security in public-key cryptography. In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2002.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[PS16] David Pointcheval and Olivier Sanders. *Short Randomizable Signatures*, pages 111–126. Springer International Publishing, Cham, 2016.

[PZ13] Christian Paquin and Greg Zaverucha. U-Prove cryptographic specification V1.1 (Revision 3). In Microsoft Technical Report, 2013. http://research.microsoft.com/pubs/166969/U-ProveCryptographicSpecificationV1.1.pdf.

[RBHP15] Andy Rupp, Foteini Baldimtsi, Gesine Hinterwälder, and Christof Paar. Cryptographic theory meets practice: Efficient and privacy-preserving payments for public transport. *ACM Trans. Inf. Syst. Secur.*, 17(3):10:1–10:31, March 2015.

[RHBP13] Andy Rupp, Gesine Hinterwälder, Foteini Baldimtsi, and Christof Paar. P4R: Privacy-preserving pre-payments with refunds for transportation systems. In Ahmad-Reza Sadeghi, editor, *FC 2013*, volume 7859 of *LNCS*, pages 205–212. Springer Heidelberg, 2013.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[Sch90] C. P. Schnorr. *Advances in Cryptology — EUROCRYPT '89: Workshop on the Theory and Application of Cryptographic Techniques Houthalen, Belgium, April 10–13, 1989 Proceedings*, chapter Efficient Identification and Signatures for Smart Cards, pages 688–689. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[Sch91] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sho97] Victor Shoup. *Advances in Cryptology — EUROCRYPT '97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings*, chapter Lower Bounds for Discrete Logarithms and Related Problems, pages 256–266. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[SKHS11] Oliver Spycher, Reto E. Koenig, Rolf Haenni, and Michael Schläpfer. A new approach towards coercion-resistant remote e-voting in linear time. In *FC 2011*, pages 182–189, 2011.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[US 15] US Vote Foundation. End-to-end verifiable internet voting. In *The future of voting, Expert Statement*, 2015.

[VA13] Pim Vullers and Gergely Alpár. Efficient selective disclosure on smart cards using idemix. In Simone Fischer-Hübner, Elisabeth de Leeuw, and Chris Mitchell, editors, *IDMAN 2013*, pages 53–67. Springer Berlin Heidelberg, 2013.

[Wei40] André Weil. Sur les fonctions algébriques de constantes finies. In *Comptes rendu de l'Académie des sciences*, volume 210, pages 592–594. 1940.

[YGS15] Shui Yu, Song Guo, and I. Stojmenovic. Fool me if you can: Mimicking attacks and anti-attacks in cyberspace. *Computers, IEEE Transactions on*, 64(1):139–151, Jan 2015.

[Yu12] Wei Yu. False data injection attacks in smart grid: Challenges and solutions. *NISTIR 7916*, 2012.

Abstract

Machine to Machine (M2M) applications are increasingly being deployed so as to enable a better management of resources and provide users with greater comfort. Unfortunately, they also entail serious security and privacy concerns. In this thesis, we focus on M2M security, and particularly on the authentication and privacy issues of M2M applications involving a SIM card.

In the first part of this thesis, we design five new cryptographic primitives, that are of independent interest, and formally prove that they meet the expected security requirements. More precisely, they consist of a partially blind signature scheme, a sequential aggregate Message Authentication Codes (MAC) scheme, an algebraic MAC scheme and two pre-Direct Anonymous Attestation (pre-DAA) schemes. Some of the proposed schemes aim to achieve a particular property that was not provided by previous constructions whereas others intend to improve the efficiency of state-of-the-art schemes. Our five schemes do not require the user's device to compute pairings. Thus, they are suitable for resource constrained environments such as SIM cards.

In a second part, we rely on these primitives to propose new privacy-preserving protocols. More specifically, we design an efficient private eCash system where the user can settle expenses of different amounts using a single reusable payment token. We also propose a protocol enabling anonymous authentication and identification of an embedded SIM (eSIM) to a Discovery Server (DS). Thereby, eSIMs can be remotely provisioned with their new network profiles while protecting users' privacy against a malicious discovery server. Furthermore, we rely on our algebraic MAC scheme to build a practical Keyed-Verification Anonymous Credentials (KVAC) system. Finally, based on our sequential aggregate MAC scheme, we introduce a remote electronic voting system that is coercion-resistant and practical for real polls. The security of our protocols is formally proven in the Random Oracle Model (ROM) under classical computational assumptions.

Résumé

Les applications Machine-to-Machine (M2M) sont de plus en plus déployées afin de fournir plus de confort aux utilisateurs et permettre une utilisation optimale des ressources. Toutefois, ces applications ne présentent pas que des avantages pour les utilisateurs. En effet, ces dernières peuvent engendrer des problèmes de sécurité, voire porter atteinte à la vie privée de leurs utilisateurs. Dans cette thèse, nous nous intéressons à la sécurité des applications M2M, et plus précisément à l'authentification et la préservation de la vie privée d'utilisateurs d'équipements M2M dotés d'une carte SIM.

Dans la première partie de ce mémoire, nous proposons cinq nouvelles primitives cryptographiques, à savoir un schéma de signature partiellement aveugle, deux schémas de codes d'authentification de messages, ainsi que deux schémas d'attestations anonymes. Ces nouvelles primitives sont plus efficaces, voire fournissent des fonctionnalités nouvelles par rapport aux schémas de l'état de l'art et sont adaptées aux environnements limités en ressources telles que les cartes SIMs.

Dans la deuxième partie, nous nous appuyons sur ces primitives pour construire de nouveaux protocoles préservant la vie privée des utilisateurs. Plus précisément, nous introduisons un nouveau système de paiement anonyme efficace avec lequel les utilisateurs peuvent effectuer plusieurs paiements, de manière complètement intraçable, en utilisant un unique « jeton » de paiement. Nous proposons également un protocole d'authentification et d'identification anonyme pour la nouvelle génération de cartes SIM, connue sous le nom d'Embedded SIM (eSIM). Ainsi, une eSIM peut récupérer son nouveau profil réseau tout en préservant la vie privée de son détenteur. Par ailleurs, en se basant sur notre schéma de codes d'authentification de messages, nous construisons un système d'accréditations anonymes. Enfin, nous spécifions un système de vote électronique efficace rendant inutile toute forme de coercition à l'encontre d'un électeur. La sécurité de toutes nos contributions est prouvée dans le modèle de l'oracle aléatoire sous des hypothèses classiques.