Design of privacy preserving cryptographic protocols for mobile contactless services

Ghada Arfaoui

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE MATHEMATIQUES, INFORMATIQUE, PHYSIQUE THEORIQUE ET INGENIERIE DES SYSTEMES

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

THÈSE présentée par :

Ghada ARFAOUI

soutenue le : **23 Novembre 2015**

pour obtenir le grade de : **Docteur de l'université d'Orléans**

Discipline/ Spécialité : Informatique

Conception de protocoles cryptographiques préservant la vie privée pour les services mobiles sans contact

THÈSE dirigée par :

Pascal BERTHOMÉ	Professeur des Universités, INSA Centre Val de Loire

RAPPORTEURS :

Olivier PEREIRA	Professeur, Université Catholique de Louvain
Peter RYAN	Professeur, Université de Luxembourg

JURY :

Mirian HALFELD FERRARI ALVES	Professeur des Universités, Université d'Orléans, Présidente du Jury
Hervé CHABANNE	Professeur associé, Télécom ParisTech / Morpho
Saïd GHAROUT	Ingénieur de recherche, Orange Labs
Jean-François LALANDE	Maître de conférences, INSA Centre Val de Loire
Jacques TRAORÉ	Ingénieur de recherche, Orange Labs

To my parents.

Acknowledgments

First of all, I would like to express my gratitude to my advisors: Professor Pascal Berthomé, Doctor Saïd Gharout, Doctor Jean-François Lalandes and Doctor Jacques Traoré. I appreciate all your contributions of time and ideas in order to make my Ph.D experience productive. Thank you for all your priceless advices on my research or my career.

Besides to my advisors, my thanks also go to the members of my PhD committee.

I also have to thank Doctor Jean-Michel Combes and Professor Maryline Laurent for giving me the opportunity to join, for the first time, Orange Labs security department as an intern.

My thanks to all the members of the NPS team for their kindness, jokes, fun, professional and personal advices…. for everything. I have had three great years with you.

I would also like to thank Antoine, Kim and Marc for proofreading some chapters of this thesis.

Finally, my sincere thanks go to my friends and special thanks to my family for their love and encouragement. Hope you forgive me when I have not been there because of my Ph.D.

Contents

List of Figures

List of Tables

Abbreviations

NFC	**N**ear **F**ield **C**ommunication
RFID	**R**adio **F**requency **ID**entification
REE	**R**ich **E**xecution **E**nvironment
TEE	**T**rusted **E**xecution **E**nvironment
OS	**O**perating **S**ystem
AVISPA	**A**utomated **V**alidation of **I**nternet **S**ecurity **P**rotocols and **A**pplications
SE	**S**ecure **E**lement
UICC	**U**niversal **I**ntegrated **C**ircuit **C**ard
SIM	**S**ubscriber **I**dentity **M**odule
SD	**S**ecurity **D**omain
ISD	**I**ssuer **S**ecurity **D**omain
SSD	**S**upplementary **S**ecurity **D**omain
CASD	**C**ontrolling **A**uthority **S**ecurity **D**omain
EAL	**E**valuation **A**ssurance **L**evel
DL	**D**iscrete **L**ogarithm
SDL	**S**ymmetric **D**iscrete **L**ogarithm
OMDL	**O**ne-**M**ore **D**iscrete **L**ogarithm
DDH	**D**ecisional **D**iffie-**H**ellman
XDH	e**X**ternal **D**iffie-**H**ellman
q-SDH	**q**-**S**trong **D**iffie-**H**ellman
q-DDHI	**q**-**D**ecisional **D**iffie-**H**ellman **I**nversion
DCR	**D**ecisional **C**omposite **R**esiduosity
LRSW	**L**ysyanskaya, **R**ivest, **S**ahai and **W**olf
TA	**T**rusted **A**pplication in the context of TEE migration
TA	**T**ransport **A**uthority in the context of transport service
RA /OA	**R**evocation / **O**pening **A**uthority
POK	**P**roof **O**f **K**nowledge
ZKPK	**Z**ero **K**nowledge **P**roof of **K**nowledge
SOK	**S**ignature **O**f **K**nowledge
BB-signature	**B**oneh-**B**oyen signature
CL-certificate	**C**amenisch-**L**ysyanskaya certificate
DAA	**D**irect **A**nounymous **A**ttesttaion
MAC	**M**essage **A**uthenticated **C**ode

Notation

$x \xleftarrow{\$} X$	x is chosen uniformly at random from the set X
$\mathbb{Z}_p$	The set of positive integers x less than *p-1*
$\{0,1\}^*$	The set of all finite length binary strings
$\{0,1\}^k$	The set of all binary strings of length k
$\mathbb{G}$	A cyclic group of prime order p
$1_{\mathbb{G}}$	The identity element of $\mathbb{G}$
λ	Security parameter
pp	Public Parameter
DB	Database
$\mathbf{ID}_U$	Identifier of a user
$\mathbf{B}_k$	A serial number of a m-ticket having k as an index
$\mathbf{Tick}_k$	A m-ticket of index k
$\mathbf{Pr}[R_1,\ldots, R_n:\mathbf{E}]$	The probability of the event E after the execution of events R_1..., R_n
$\mathcal{A}$	Adversary
$\mathcal{C}$	Challenger
$\mathcal{A}(\boldsymbol{keys}, \boldsymbol{DB}: \boldsymbol{oracles})$	An adversary who receives the keys "*keys*", has only **read access** to the databases "*DB*" and can query the oracles "*oracles*"

Introduction

(The english version is below)

Plusieurs études, telles que par exemple celles effectuées par International Data Corporation (IDC), montrent que le marché des appareils mobiles ne cesse de se développer et que le nombre d'applications téléchargées augmente de manière exponentielle. L'IDC prévoit que 1,5 milliards de smartphones seront vendus d'ici la fin de 2017 [3] et que les téléchargements d'applications atteindront 178 milliards en 2017 [4].

De même, la multiplicité des services pour les utilisateurs mobiles augmente de façon spectaculaire. Presque tout peut être fait via des applications mobiles. On peut distinguer deux types majeurs d'applications mobiles: les applications d'entreprise et les applications personnelles. Les applications d'entreprise permettent à un utilisateur d'avoir accès aux services de l'entreprise comme l'accès à des documents confidentiels de cette entreprise. Quant aux applications personnelles, elles offrent divers services quotidiens à l'utilisateur.

Parmi les applications personnelles les plus répandues, on trouve celles dédiées au divertissement. Un utilisateur peut télécharger une application mobile pour jouer à un jeu, écouter de la musique, regarder des vidéos et ainsi de suite. Il peut également utiliser une application mobile comme un moyen de communication, que ce soit en voix, vidéo ou à travers les réseaux sociaux. Outre ces applications, il existe aussi des applications bancaires. Ces dernières sont téléchargées par l'utilisateur soit pour surveiller son compte bancaire en temps réel soit pour effectuer des transactions de paiement en utilisant son smartphone. Dans ce cas, ces applications peuvent interagir avec un serveur distant et manipuler les informations bancaires de l'utilisateur, son identité ainsi que de nombreuses autres données privées comme les journaux de transactions effectuées. Parmi les applications mobiles émergentes, on compte aussi les applications dédiées au transport. En utilisant la possibilité d'effectuer des transactions sans contact des smartphones actuels, elles permettent à l'utilisateur d'acheter et de valider un titre de transport dématérialisé, comme un carnet de billets ou un abonnement. Comme les applications bancaires, les applications de transport public manipulent des données privées de l'utilisateur telles que son identité, des attributs associés à son âge, sa situation, par exemple, un étudiant ou une personne handicapée, le type de son titre de transport etc.

La croissance rapide de l'utilisation des applications et services mobiles ainsi que l'aspect sensible et privé des données manipulées par ces applications, produisent une situation attrayante pour les attaquants. Un attaquant peut être un utilisateur frauduleux, une entité externe, ou bien un fournisseur de service ou d'application malveillant. Les attaques qui peuvent être effectuées par ces attaquants sont soit contre le service, soit contre

l'utilisateur. Ci-après, nous détaillons les problématiques liées à chaque type d'attaques et nous donnons un aperçu de nos objectifs et nos contributions.

Les attaques contre l'utilisateur sont principalement liées aux problématiques de respect de la vie privée. Un attaquant externe peut essayer de capturer des données privées et des secrets de l'utilisateur. De même, un fournisseur de services peut essayer de traquer les utilisateurs dans leur vie quotidienne en fonction de leur utilisation du service. Par exemple, dans les transports publics, des violations contre la vie privée sont fréquemment soulevées. En 2012, le transporteur belge STIB a reçu le prix Big Brother pour sa carte sans contact (MoBIB) [5]. En effet, ce prix met en avant les entreprises et les gouvernements qui portent atteinte à la vie privée. Un autre exemple marquant illustre ce problème: certains avocats spécialisés dans le divorce ont utilisé les registres de paiement de l'E-ZPass et Fast Lane afin de poursuivre les maris de leurs clientes et prouver qu'ils ont triché sur leurs localisations à un moment et une date donnés [6].

Étant donné que plusieurs solutions industrielles déployées pour les infrastructures de transport public présentent d'importants problèmes liés au respect de la vie privée, les scientifiques se sont penchés sur cette question. Beaucoup de solutions cryptographiques ont été proposées afin d'assurer la protection de la vie privée, principalement pour garantir l'intraçabilité des voyages de l'utilisateur. Cependant, ces solutions ont généralement négligé les exigences fonctionnelles du service. En effet, la validation du titre de transport (billet ou abonnement électronique) doit être réalisée en moins de $300\ ms$ *[7]. Cette contrainte de temps devient difficile à respecter sur des plateformes d'implémentation très sécurisées ayant des ressources limitées à l'instar des cartes SIM.*

Dans ces circonstances, nous avons choisi de construire un service de transport réaliste, respectant la vie privée, et se basant sur de nouveaux outils cryptographiques améliorés. Dans cette thèse, nous commençons par montrer les obstacles pour construire des protocoles qui respectent la vie privée. Nous étudions les travaux liés au service de transport public en normalisation, dans les solutions industrielles et dans la littérature. En particulier, nous détaillons une vulnérabilité que nous avons découverte dans le protocole de Rupp et al. qui a été publié à la conférence FC 2013 [2]. Ensuite, nous présentons l'architecture du mobile ainsi que les hypothèses de sécurité associées à chaque composant. Ensuite, nous distinguons deux cas d'usage dans le service de transport public: (1) le cas d'usage "m-pass" (abonnement) où un utilisateur peut utiliser le système de transport d'une manière illimitée, mais pendant une période de temps donnée, et (2) le cas d'usage "m-ticketing" (billetterie) où un utilisateur a un accès limité au réseau de transport en fonction du nombre de ses m-tickets (billets électroniques sur mobile). Nous commençons par la formalisation des exigences en termes de sécurité et de respect de la vie privée du cas d'usage "m-pass", et nous définissons aussi ses exigences fonctionnelles. Plus tard, nous présentons notre solution d'abonnement intraçable, basée sur les schémas de signature de type Camenisch-Lysyanskaya, et prouvons formellement qu'elle satisfait toutes les exigences définies précédemment. Pareillement, nous formalisons les exigences en termes de sécurité et de respect de la vie privée du cas d'usage "m-ticketing", et définissons ses exigences fonctionnelles. Malgré les ressemblances que peuvent avoir les exigences du cas d'usage "m-ticketing" avec celles du cas d'usage "m-pass", elles doivent être mises en œuvre différemment puisque le cas d'usage "m-ticketing" présente plus de contraintes: l'anonymat et l'intraçabilité des m-tickets séparément en plus de l'anonymat et l'intraçabilité du carnet des m-tickets. Ensuite, nous présentons notre solution de billetterie intraçable basée sur les schémas de signature Boneh-Boyen et les preuves d'appartenance à un ensemble. Notre protocole assure

également le post-paiement. Lorsqu'on active ce mode, la vie privée de l'utilisateur ne doit pas être remise en cause. En outre, le système doit être en mesure d'empêcher, ou au moins de détecter, toute tricherie de l'utilisateur. Enfin, nous prouvons formellement que notre solution satisfait toutes les exigences définies précédemment.

Afin d'atteindre nos objectifs dans le cas d'usage "m-ticketing", nous proposons plusieurs optimisations cryptographiques aux preuves d'appartenance à un ensemble. Notre système permet à un prouveur de prouver, sans divulgation de données, qu'il détient un secret appartenant à un ensemble public donné. Ces preuves sont faites sans nécessiter de calculs de couplage, principalement du côté du prouveur. En effet, dans les constructions précédentes de preuves d'appartenance à un ensemble, le prouveur a besoin d'effectuer des calculs de couplage qui sont coûteux, en particulier pour les dispositifs à ressources limitées comme la carte SIM. Cette contribution est d'un intérêt indépendant et peut être utilisée à d'autres fins lorsqu'une telle preuve est nécessaire dans un contexte où les ressources sont limitées.

En plus de la conception et de la formalisation des protocoles de transport respectant la vie privée, nous proposons une implementation de ces protocoles sur une carte SIM et montrons qu'ils satisfont les exigences fonctionnelles, notamment la stricte contrainte de temps.

En ce qui concerne les attaques contre le service, elles visent à introduire des troubles dans le service. Un attaquant externe peut essayer de perturber le service ou même le rendre indisponible. Cela peut se produire, par exemple, en contrôlant à distance l'appareil de l'utilisateur à travers un malware. De même, un utilisateur malveillant peut tenter de compromettre l'application mobile, et donc compromettre la sécurité du service, afin de tricher. Par exemple, dans un service donnant accès à un système de transport, il peut essayer de modifier le montant d'un voyage ou de dépenser plusieurs fois le même billet. Les problèmes de sécurité décrits précédemment sont réalisables, car les environnements mobiles, hébergeant les applications mobiles, sont vulnérables. Différents types de failles sont fréquemment détectés. En 2013, l'analyse de Kaspersky Security a indiqué que 148,778 malwares mobiles ont été détectés [8]. Dans ces circonstances, une nouvelle famille d'environnements mobiles, appelé environnements d'exécution de confiance, est en train d'émerger afin de fournir une plus grande protection aux applications, et donc, aux services fournies. Un environnement d'exécution de confiance (Trusted Execution Environment, TEE) est défini comme un système d'exploitation sécurisé s'exécutant parallèlement au système d'exploitation standard, appelé également environnement d'exécution riche (Rich Execution Environment, REE), et possédant ses propres CPU et mémoires. Le TEE devrait héberger des applications de confiance, comme les applications bancaires ou de transport, qui fournissent des services sécurisés à des applications mobiles fonctionnant au sein du REE.

Dans ce contexte, nous cherchons à étudier et améliorer les fonctions de sécurité offertes par cette nouvelle famille d'environnements mobiles. Nous avons identifié deux problèmatiques. Tout d'abord, l'architecture du TEE ainsi que celle de son écosystème ne sont pas suffisamment spécifiées: le TEE est généralement considéré comme une boîte noire et seuls les API sont définies. Deuxièmement, dans un environnement multi-appareils (l'utilisateur a plusieurs appareils), l'utilisateur ne peut pas facilement et en toute sécurité migrer ses services d'un TEE à un autre. Dans cette thèse, nous montrons qu'actuellement, les protocoles de migration de TEE présentent des problématiques de confidentialité et de sécurité. Nous proposons ensuite une nouvelle architecture du

TEE, organisée en différents domaines de sécurité avec des droits différents, et un nouvel écosystème avec de nouveaux rôles. Avec ce nouvel écosystème et architecture du TEE, nous présentons notre protocole de migration qui assure la confidentialité des TEEs transférés et qui est basé sur un schéma de proxy de rechiffrement. Ensuite, nous le validons à l'aide d'un outil de validation automatisé de protocoles de sécurité, AVISPA [9].

Cette thèse est organisée en neuf chapitres.

Le chapitre 2 fournit une introduction aux services mobiles sans contact. Il détaille également le service de transport public et ses exigences. Ensuite, il décrit l'architecture d'un mobile actuel ainsi que ses moyens de sécurité.

Dans le chapitre 3, nous détaillons les composantes de base et les hypothèses cryptographiques que nous utiliserons dans les chapitres suivants.

Nous introduisons au chapitre 4 une nouvelle preuve d'appartenance à un ensemble.Cette preuve est plus efficace car elle ne nécessite pas de calcul de couplage côté prouveur.

Dans le chapitre 5, nous donnons plus de détails sur les travaux liés aux solutions de transport public et décrivons l'architecture du mobile qui constitue un cas d'utilisation dans la suite de la thèse.

Le chapitre 6 introduit une solution d'abonnement mobile respectant la vie privée dans le cadre du service de transport public avec son implémentation sur une carte SIM.

Le chapitre 7 introduit une solution de billetterie mobile, efficace et respectant la vie privée, pour les services de transports publics. Ce protocole utilise pleinement les résultats du chapitre 4. Nous montrons aussi comment est assuré un post-paiement sécurisé. Nous terminons ce chapitre en présentant les mesures de performance de notre implémentation sur une carte SIM.

Le chapitre 8 décrit un protocole de migration de TEE, sécurisé et respectant la vie privée.

Le chapitre 9 présente des conclusions, résume nos principales contributions et discute les perspectives, les défis et les futures orientations du travail de cette thèse.

Chapter 1

Introduction

Studies such as International Data Corporation (IDC) ones, show that the market of mobile devices is continuously increasing along with the number of downloaded applications. The IDC expects that 1.5 billion smartphones will be sold by the end of 2017 [3] and that application downloads will reach 178 billion by 2017 [4].

Likewise, the multiplicity of services for mobile users is dramatically increasing. Nearly everything can be done via mobile applications. Figure 1.1 gives a subset of mobile services and applications. There are two major types of mobile applications: corporate applications and personal ones. The corporate applications enable a user to have access to corporate services like the access to confidential documents of the company in which he works. Regarding the personal applications, it offers various daily services to the user.

Some of the most famous personal applications are for entertainment. A user can download a mobile application to play a game, listen to music, watch videos and so forth. He may also use a mobile application as a mean of communication, either voice and video or through social networks. Besides these applications, there are banking applications as well. The latters are downloaded by the user either to monitor his banking account in real-time or to perform payment transactions using his smartphone. In such a case, these applications may remotely interact with a back-end server and would handle the banking information of the user, his identity as well as many other private data like transactions logs. Other emerging applications are the transport ones. They enable the user to buy and validate, using a smartphone enabling contactless transactions, a mobile transport product, i.e., tickets or pass. Similar to banking applications, transport applications handle private user data such as his identity, attributes associated to his age, situation, e.g., a student or a disabled, the kind of his transport product and so on.

The rapid growth of the mobile applications and services use, along with the critical and private aspect of the data being handled by these applications, produce an attractive situation for attackers. An attacker can be a fraudulent user, an external entity, a malicious service or application provider. The attacks that can be performed by these attackers are either against the service or against the user. Hereafter, we detail the issues related to every type of attack as well as an overview of our objectives and contributions with respect to it.

Attacks against the user include privacy issues. An external attacker may try to capture the user's private data and secrets. Likewise, a service provider may try to track users

FIGURE 1.1: Mobile Services and Applications

in their daily life based on their use of the service. For instance, in public transport, privacy violations are frequently raised. In 2012, the Belgium transport operator STIB has received the Big Brother awards for its contactless card (MoBIB) [5]. Another example is the divorce lawyers who used the E-ZPass and Fast Lane toll records to track their clients husbands and prove that they were cheating about their location at a given time and date [6].

Given the serious privacy problems of the deployed industrial solutions for transport systems, scientists have looked into this issue. Many cryptographic solutions have been proposed in order to ensure the privacy requirements, namely the unlinkability of the user's trips. However, these solutions have usually overlooked the functional requirements of the service. In a nutshell, the validation of the transport product (mobile pass or mobile ticket) must occur in less than 300 ms [7]. However, the user's device, in particular the SIM card, is resource constrained.

In these circumstances, we chose to build a practical and privacy-preserving transport service based on new and enhanced cryptographic tools. In this thesis, we begin by showing the barriers to build privacy-preserving protocols. We study the related work to public transport service in standardization work, industrial solutions and literature. In addition, we provide details of an attack we discovered against Rupp et al. protocol which has been published at FC 2013 [2]. Then, we present our mobile architecture and trust assumptions with respect to the mobile components. Afterwards, we distinguish two use cases of the public transport service: (1) the mobile pass use case where a user can use the transport system in a limitless way but for a given period of time, and (2) the mobile ticketing use case where a user has a limited access to the transport network according to the number of his m-tickets. We start by formalizing the security and privacy requirements of the m-pass use case, and define its functional requirements. Later, we introduce our untraceable mobile pass solution based on Camenisch-Lysyanskaya signature schemes and formally prove that it satisfies the requirements previously defined.

Similarly to the mobile pass, we formalize the security and privacy requirements of the mobile ticketing use case, and define its functional requirements. These requirements, even though they look similar to the mobile use case, must be set up differently since the mobile ticketing use case presents more constraints: the privacy of the m-tickets separately in addition to the privacy of the book of m-tickets. Then, we introduce our untraceable mobile ticketing solution based on Boneh-Boyen signature schemes and set-membership proofs. Our protocol also enables post-payment. When enabling this mode, the user's privacy must not be questioned. In addition, the system must be able to prevent, or at least detect, any user cheating. Finally, we formally prove that it satisfies the requirements previously defined.

In order to achieve our objectives regarding the untraceable mobile ticketing use case, we propose several cryptographic optimizations to the set-membership proofs. Our scheme enables a prover to prove, in a zero knowledge way, that he holds a secret belonging to a given public set without requiring pairing computations, mainly at the prover side. Indeed, in the previous constructions of set-membership proofs, the prover requires pairing computations which are costly, especially for resource constrained devices like the SIM card. This contribution is of independent interest and can be used for other purposes where such a proof is required in a resource constrained environment.

In addition to the design and formalization of the privacy-preserving transport protocols, we propose an implementation of these protocols on a SIM card and show that it satisfies the functional requirements, in particular the stringent time constraint.

Regarding the attacks against the service, they aim at introducing troubles into the service. An external attacker may try to disrupt the service or even make it unavailable. This can occur, for instance, by remotely controlling the user's device through a malware within it. Likewise, a malicious user may try to compromise the mobile application, and hence compromise the security of the service, in order to cheat. For instance in a service giving access to a transport system, he may try to change the amount of a journey or double spend the same ticket. The security issues described previously are achievable because the mobile environments, hosting the mobile applications, are vulnerable. Different types of deficiencies are frequently detected. In 2013, Kaspersky Security analysis indicates that 148.778 mobile malware were detected [8]. In these circumstances, a new family of mobile environments, called Trusted Execution Environments, is emerging in order to provide more protection to applications, and hence, the services they provide. A Trusted Execution Environment is defined as a secure Operating System running alongside the standard Operating System, also called Rich Execution Environment and has its own CPU and memories. The Trusted Execution Environment should host Trusted Applications like banking or transport applications that provide secure services to mobile applications running within the Rich Execution Environment.

In this context, we aim at investigating and improving the security features provided by this new family of mobile environments. We identified two issues. First, the TEE architecture and ecosystem are not finely detailed: the TEE is usually considered as a black box and only APIs are defined. Second, in a multidevice environment (the user has many devices), the user cannot easily and securely migrate his TEE services from a device to another one. In this thesis, we show that current TEE migration protocols do have privacy and security issues. We then propose a new TEE architecture organized into different security domains with different rights and define a new TEE ecosystem with new entities roles. With the new TEE ecosystem and architecture, we present

our TEE migration protocol that ensures the confidentiality of the transferred TEE and is based on proxy re-encryption scheme. Next, we validate this protocol using the automated security protocol validation tool, AVISPA.

This thesis is organized into nine chapters.

Chapter 2 provides an introduction to contactless mobile services. It also details the public transport service and its requirements. Then, it describes the current mobile architecture along with its security features.

In Chapter 3, we detail the cryptographic assumptions and building blocks that we will use in the following chapters.

We introduce a new practical set-membership proof that does not require pairing computations at the prover side, as shown in Chapter 4.

In Chapter 5, we give details on works related to transport solutions and describe the mobile architecture that we will consider for our proposals.

Chapter 6 introduces a privacy-preserving mobile pass solution in the context of public transport service together with its implementation on a SIM card.

Chapter 7 introduces a practical and privacy-preserving mobile ticketing solution for public transport services. This protocol utilizes the new set-membership proof introduced in Chapter 4. We also show how to enable a secure post-payment mode. We end this chapter by presenting the performance measurements of our SIM card implementation.

Chapter 8 provides a secure and privacy-preserving TEE migration protocol.

Chapter 9 draws conclusions, summarizes our major contributions and discusses perspectives, challenges and future work directions.

Chapter 2

Contactless Mobile Services

Contents

Dans ce chapitre, nous nous intéressons aux services mobiles sans contact. Après avoir donné un aperçu des différents services, nous nous concentrons au domaine d'application des transports publics car ils présentent des contraintes opérationnelles fortes, particulièrement lorsque l'objectif est d'en élaborer une version sans contact. Nous détaillons les deux cas d'usage spécifiques, c'est-à-dire, l'abonnement et la billetterie électronique. Sachant que les services mobiles vont être implémentés sur des téléphones, il convient donc de comprendre l'architecture d'un téléphone afin de mieux cerner les enjeux de sécurité. Nous décrivons donc l'architecture d'un téléphone mobile. Nous détaillons, plus précisément, la technologie de communication à champ proche (Near Field Communication, NFC), les cartes SIM et les environnements d'exécution mobiles, en particulier les environnements d'exécution de confiance (Trusted Execution Environment, TEE).

2.1 Introduction

Since mobile devices are becoming widespread, service providers have adapted their services in order to maintain their consumers and attract new ones. Many services have been, and many others will be, made available via the user's mobile device. That is not all. Service providers make use of all the available mobile technologies in order to provide a better user experience, and hence attract more users. Consider, for instance, the Near Field Communication technology, it has been used in the deployment of contactless mobile services.

In this chapter, we aim to shed light on the context of contactless mobile services. We identify three areas: the contactless technology namely the Near Field Communication technology, the relevant services and the mobile devices. We begin by describing the history of the Near Field Communication technology, starting from its ancestors (contactless cards) to its beginnings, while identifying the issues that followed its expansion. Later, we give further details about the functioning of this technology and its main relevant services. As a user needs a (NFC-enabled) mobile in order to use a NFC service, we felt that it is important to study the mobile architecture and investigate the current mobile security features.

This chapter is organized as follows. In Section 2.2, we give a brief history about the Near Field Communication (NFC) technology and an overview of the contactless mobile services. In Section 2.3, we focus on the mobile public transport service: we detail its phases, entities and challenges. Finally, we describe the smartphone architecture in Section 2.4 and conclude the chapter in Section 2.5.

2.2 Contactless Mobile Services

Contactless services first appeared with transit contactless smart cards in Seoul, South Korea (1995) [10] and in Hong Kong with the Octopus card (1997) [11]. These cards have the size of a current banking card and contain a chip that communicates with the reader in proximity through electromagnetic induction. Then, in 2003, Gemplus introduced the first contactless card, GemCombiXplore, for mobile handsets [12]. This card has the same format as a SIM card and communicates through a RFID (Radio Frequency Identification) antenna embedded within the mobile device. GemCombiXplore has been adopted for e-purse, debit/credit, loyalty and contactless mobile payment applications for instance in public transport services. This enables users to replace the paper-based tickets with their mobile devices equipped with the GemCombiXplore card. This card is among the contactless implementations introduced before 2004 which presented a huge success. However, these implementations still present a major drawback: compliance and interoperability issues. Therefore, in 2004, NXP Semiconductors, Sony and Nokia founded the NFC Forum [13], a non-profit industry association in order to ensure the interoperability between devices and services by promoting the standardization of the *Near Field Communication* technology (NFC for short).

The NFC is a short-range wireless technology [14]. It can be seen as the radio-frequency identification (RFID) successor by providing a shorter range for security purposes. Indeed, NFC standards are based on existing RFID standards including ISO/IEC 14443 [15] and cover communications protocols and data exchange formats [16]. On the market, the NFC had a relatively slow start. The first NFC enabled mobile was brought to market in 2006 [17]. It is the Nokia 6131. Four years later, Samsung commercialized the Samsung Nexus S, the first NFC enabled Andoid smartphone [17]. Afterwards, the NFC market has dramatically increased to reach 416 million units in 2014 [18].

According to Information Handling Services (IHS) Technology, this trend will continue. It is estimated that, from 2013 to 2018, NFC-enabled mobiles will rise by 325 percent [18]. In other words, by 2018, two mobiles in three will be NFC-enabled compared to the one in four at the moment. The IHS study continues by pointing out a major challenge for the current NFC market: the development of NFC services and applications.

The main NFC services are presented in Figure 2.1. In all these services, a NFC-enabled device becomes the way to use the service: a mobile application replaces the credit card, the identity card, the professional card, the transport pass, the paper-based tickets, and so forth. We identify three main families of NFC services: mobile commerce (m-commerce), identification, and corporate services.

Corporate services include physical access control, time attendance, and secure PC log-on. In these services, a user, using his NFC-enabled device, has access to corporate resources such as a building, a lab or an internal network. Indeed, at the entrance of the company building, a lab or when attempting to use the internal network of the company, the NFC-enabled device, with some credentials, would exchange with a NFC reader in order to prove the legitimacy of the user. This kind of services has two major security issues:

1. proving the legitimacy of the user: a malicious user may try to use his credentials in order to access a resource without authorization,
2. and proving that the user is the owner of the used credentials: a malicious user may try to forge credentials or use the credentials of another user.

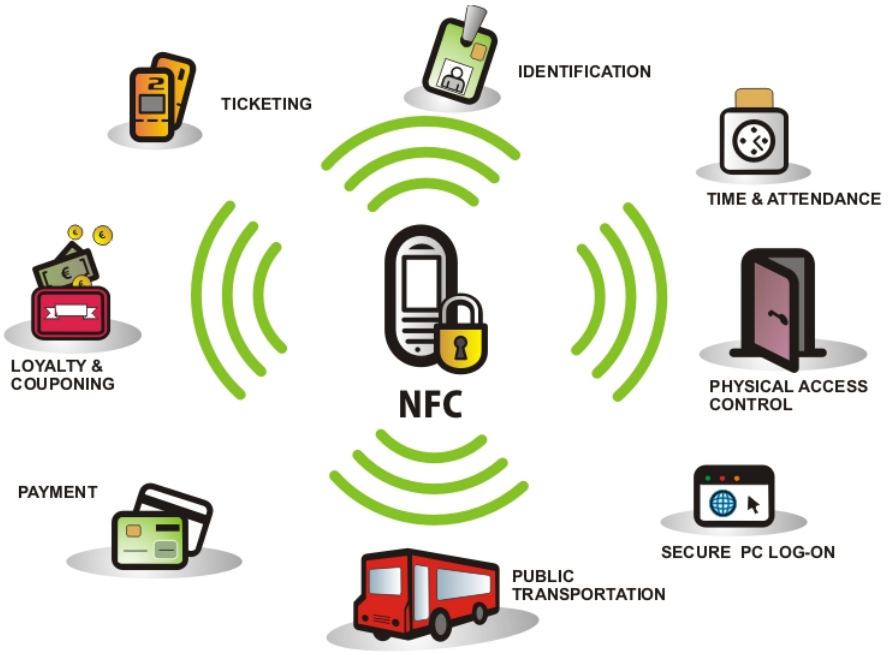


FIGURE 2.1: Contactless Mobile Services

In identification services, a user may use his NFC-enabled device as an identity authentication tool. Similarly to the corporate services, the user's NFC-enabled device, with some credentials, would exchange with a NFC reader. However, in these cases unlike the corporate services, the device reader exchanges enable to authenticate the identity of the user. The security issues of the identification services have some kind of similarities with corporate services issues:

1. proving that the used identity is not spoofed: the user is the right owner of the used identity, hence the used credentials,

2. and proving that the used identity, hence the used credentials, is not tampered. In addition, some identification use cases may bring out privacy issues especially when the service requires only a part of the user's identity.

M-Commerce services involve sale and purchase transactions using a NFC-enabled mobile device. We distinguish three main use cases: Mobile payment, Loyalty and couponing, and Public transport. In all these use cases, like the services described previously in this section, the user's NFC-enabled device, with some credentials, would exchange with a NFC reader. This latter can be within the payment equipment (the mobile payment use case), within the tradesman device (the loyalty and couponing use case) or within the turnstile located at the entrance of a public transport station (the public transport use case). Moreover, the public transport use case is more expanded than the other use cases and will continuously grow up [19]. Therefore, in this thesis, we give further attention to the public transport use cases. Hereafter, we give further details about it.

2.3 Mobile Transport Service

A public mobile transport service involves two main entities: a transport operator and a user. The *user* benefits from a transport service using his transport product (a mobile pass (m-pass) or a book of mobile tickets (m-tickets)). The *transport operator* encompasses: the transport back-end server and a validator. The transport back-end registers the users to the transport service and provides them with the necessary transport products. The validator contains a NFC reader and initiates a control with the smartphone of the user when he is traveling in the transport system, in order to check the authenticity and the validity of his transport product.

After downloading the transport application on the user's mobile device, the user performs three main phases, as described in Figure 2.2.

User registration and application setup. During this phase, a user gives a proof of his identity (e.g., a student card, an identity card or a passport) to the transport operator. If this verification succeeds, the transport operator will register the user at the transport service.

Product provisioning. A user chooses a product (e.g., a monthly transport pass or a book of 10 m-tickets) and the related personal attributes (e.g., the age of the user) in order to load the product into his smartphone. Additionally, if such a choice is possible in the transport system, the user chooses the geographical area of his product and the associated validity period. The product is then loaded onto the user's device.

Product validation. The user shows his smartphone at the entrance gate of the transport system in order to validate the product. To achieve this, the validator initiates a communication with the user's smartphone. Then, the validator checks the authenticity of the product and grants or denies access to the user. If a validation conflict occurs (two products are eligible to be validated), then the user has to be involved in selecting the adequate product. Finally, the smartphone and the validator record the event.

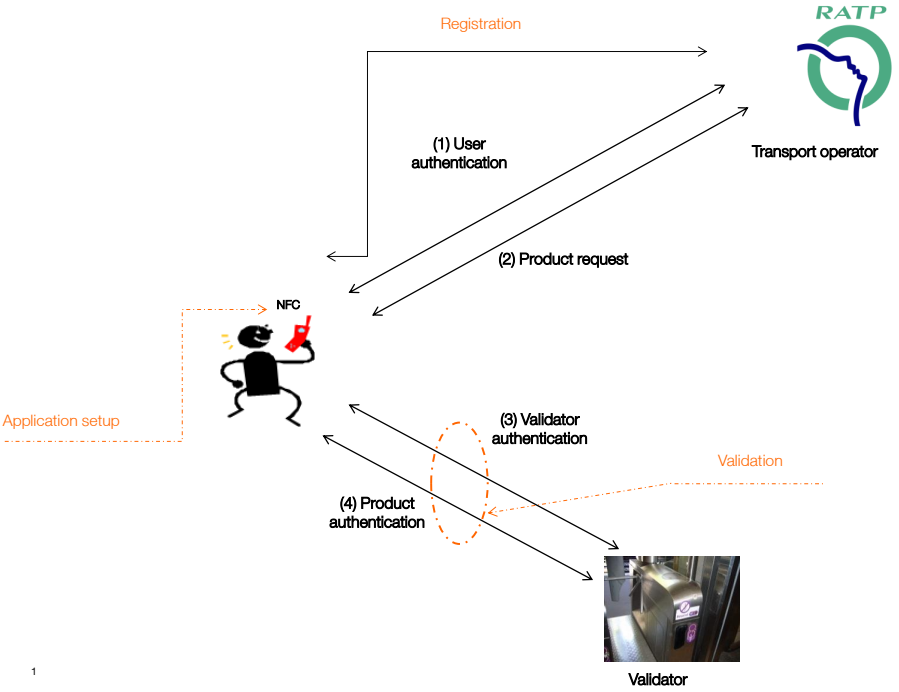


FIGURE 2.2: Public Transport Service Phases

The major security requirements for a public transport service are the correctness, the unforgeability and the non frameability. In a system which behaves correctly and in a secure manner, a user, who owns a valid transport product, should be able to access the transport network. In an unforgeable system, it should be impossible for anyone to forge fake products. In a non frameable system, no one can spoof the identity of a honest user and falsely accuse him of having used his transport product.

In addition to the security requirements, privacy requirements are also needed especially because of the privacy violations described previously in Chapter 1. Considering the privacy of a user requires that he remains anonymous in his daily use of the transport system. However, under exceptional circumstances, e.g., a fraud or a murder, it should be possible to lift his anonymity, thus providing a form of revocable anonymity. Moreover, a privacy-preserving transport system should also ensure the unlinkability of the user's actions. In particular, the transport operator might be tempted to link all the trips performed by the same user, even if these trips are "anonymous". If the different actions of a user can be linked, then there is a high risk of profiling. Some of these actions combined together might play the role of quasi-identifiers, thus leading to a potential de-anonymization of the user. For instance, existing technologies such as U-prove may fail to provide the unlinkability property if the user does not go online regularly to refresh his pseudonym [20]. This was further confirmed by a study of Vives-Guasch and co-authors [21], which shows that most of current proposals in the literature use pseudonyms in order to achieve revocable anonymity but do not prevent the tracking of these pseudonyms.

Furthermore, the transport service has a stringent functional requirement: the total transaction time of a product validation should not exceed 300 ms [7].

Previously in this section, we made abstraction of the transport product which can be a mobile pass (m-pass) or a book of mobile tickets (m-tickets). This is because whatever the product is, the entities, the phases and the major security, privacy and functional requirements are the same. However, ensuring these requirements in the case of the m-pass is completely different from the m-ticketing case. Moreover, according to the product, other requirements may arise.

In Table 2.1, we detail the differences between the m-pass and the m-ticketing use cases. The way of usage and the structure of the transport product vary between the m-pass and m-ticketing use cases. On the one hand, one m-pass can infinitely be used during a given period of time. On the other hand, a book of N m-tickets enables a user to have N trips. The granularity of the m-ticketing use case implies more privacy issues compared to the m-pass use case. Indeed, a m-ticket is identified by an index $n \in N$ and a serial number. In order to ensure the same privacy level in the m-pass and m-ticketing use cases, we must ensure the anonymity of a m-ticket index in addition to the unlinkability of the m-tickets and the user's anonymity.

	M-pass	**M-ticketing**
Type	Access control	Access control
User's privacy	1. Anonymity of the user 2. Unlinkability between the different m-pass usages	1. Anonymity of the user 2. Unlinkability between the serial numbers of different m-tickets 3. Anonymity of a m-ticket index
Structure	1 m-pass	N m-tickets
Trips	∞	1 ticket per trip

TABLE 2.1: Comparison between the M-Pass and M-Ticketing use cases

Whether in the m-pass or m-ticketing use case, we assume that the user has an NFC-enabled smartphone that will host the transport application. Knowing that current mobile environments are vulnerable and that an attacker or a malicious user may attempt to compromise the transport application in order to make the service unavailable or to cheat, it is important to secure the running code of the transport application. Therefore, in the following section, we investigate the current available security features that can be provided by a smartphone.

2.4 Secure Mobile Architecture

Figure 2.3 gives an overview of the current smartphone architecture. Besides to the NFC component, the UICC (i.e., SIM card) and the mobile environment (Mobile OS), today, a new family of embedded environments, usually called Trusted Execution Environments (TEE), has emerged. Further, we give further details about the NFC technology provided by the NFC component and the security features of the mobile device, i.e., SIM card and TEE. We then investigate the differences between a TEE and a SIM card in order to get a feel for the implementation of our transport application in these environments

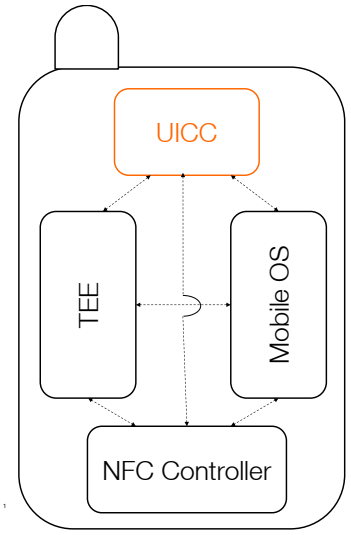


FIGURE 2.3: Current Smartphone Architecture

2.4.1 Near Field Communication

The Near Field Communication technology enables wireless interactions between devices within a range less than 4 centimeters with a maximum communication speed of 424 kbps [14]. The NFC technology proposes three communication modes: Reader/Writer mode, Peer-to-peer mode and Card emulation mode.

Reader/Writer mode [22] enables NFC-enabled devices to read or write information from or to inexpensive NFC tags. The NFC tag is powered by a magnetic field in order to answer the request of the mobile device. In the reader mode, the NFC tag turns back the stored data that usually consists on a URL address or a couponing service. In the writer mode, the mobile device writes data on the NFC tag. If the tag already contains any data, it will be overwritten. The NFC tags are usually embedded in smart posters and displays such as in cinemas in order to get information about a film, or in a museum to get information about a monument.

Peer-to-peer mode [22] enables two NFC-enabled devices to communicate with each other in order to exchange information such as business cards or share files like private photos. This mode can also be used between users in order to transfer money or tickets between wallets. The two devices communicate over a bidirectional half duplex channel which enables only one device to transmit information at a given time: when one device is transmitting, the other one has to listen and can start the transmission after the first one finishes.

Card emulation [22] mode enables NFC-enabled devices to act like a traditional contactless smart cards. A NFC-enabled device is able to embed multiple smart card applications: credit card, loyalty card, access card, transit card or tickets. This mode is advantageous for such use cases in particular for large scale deployment. Indeed, when using this technology, no renewal of the service infrastructure, such as the payment equipments, is needed. Currently supported communication interfaces for the card emulation mode are ISO/IEC 14443 Type A and Type B, and FeliCa.

2.4.2 Universal Integrated Circuit Card

According to ETSI TR 102 216 [23], a Universal Integrated Circuit Card (UICC) is *a smart card that conforms to the specifications written and maintained by the ETSI Smart Card Platform project.* This smart card is used for mobile telecommunications. GlobalPlatform specified a smartcard architecture [1] which is valid for UICC. According to the GlobalPlatform Card specification [1], a smart card is organized into isolated areas called Security Domain (SD) as presented in Figure 2.4. A security domain is defined as an on-card representative of an off-card entity that provides security services such as encryption and decryption for the applications of the off-card entity.

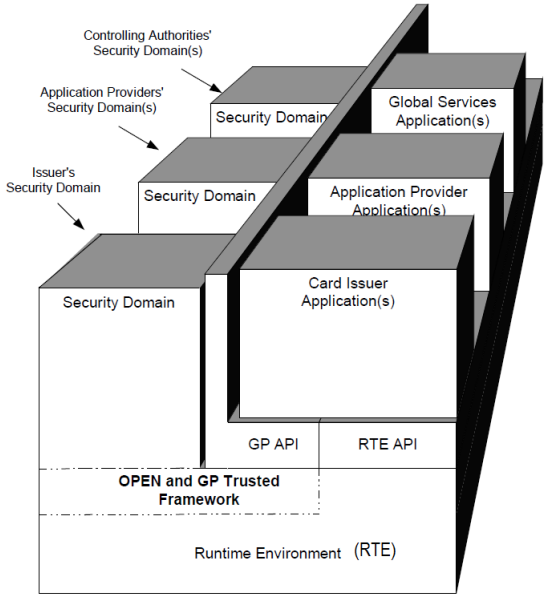


FIGURE 2.4: Smart Card Architecture [1]

Three types of security domains, hence three types of off-card entities, are defined by the specifications: Issuer Security Domain (ISD) which is the on-card representative of the Card Administrator, Supplementary Security Domains (SSD) which are the on-card services / applications providers such as the bank or the transport operator and Controlling Authority Security Domain (CASD) which is a Supplementary Security Domain with particular rights. CASD has a neutral role in the card and contains the card private key. It also performs operation for other Security Domains in the card. If it exists, it would enforce the security of the card and the trust relationship with off-card entities. Finally, the SIM card (UICC) is considered as a secure element that can resist high physical attacks as it could be certified Common Criteria Evaluation Assurance Level 4 Augmented (EAL4+) [24]. With this certification, we are assured that a running code within a SIM card has the highest levels of protection.

In addition to the SIM card, a mobile phone encompasses another secure environment called Trusted Execution Environment. We give further details about it in the following section.

2.4.3 Trusted Execution Environment

A Trusted Execution Environment [25] (TEE for short) is a hardware environment with a Secure Operating System which is isolated and completely separated from the mobile platform. This new family offers a complete execution environment; more secure than the traditional mobile Operating Systems (also called Rich OS, or REE for Rich Execution Environment), with sufficient memory and storage capabilities. In other words, TEE is a separate execution environment that runs alongside the REE and provides security services to this rich environment.

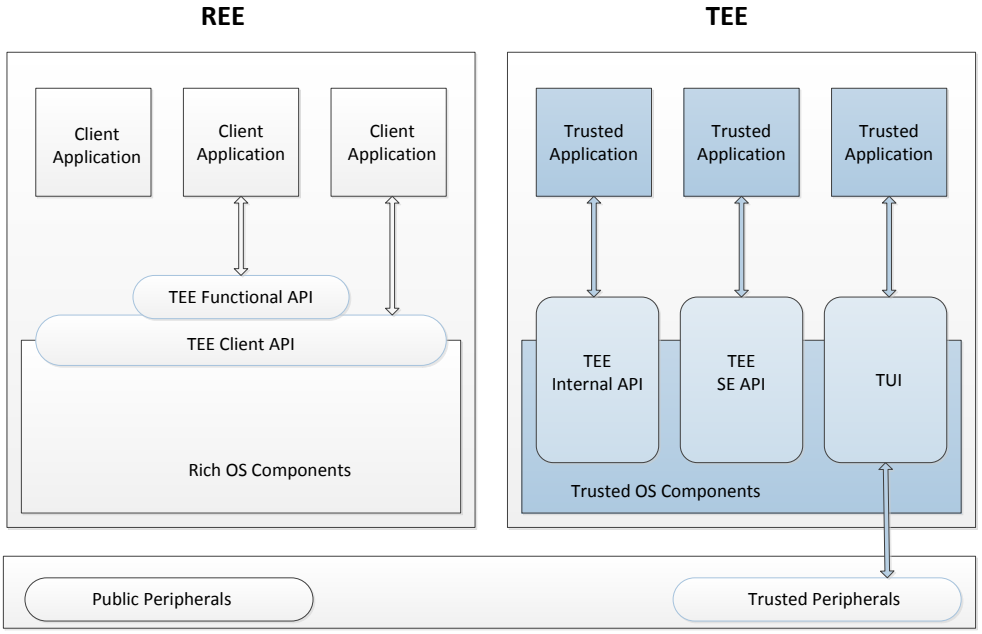


FIGURE 2.5: TEE Software Architecture

As shown in Figure 2.5, the TEE consists of Hardware Secure Resources, Trusted OS, Trusted Applications and TEE APIs. There are four classes of **hardware secure resources**: (1) Hardware Keys also called Platform Keys, (2) Secure functions such as secure storage, secure entry and secure display, (3) Cryptographic accelerators and (4) Interfaces to communicate with tamper resistant components such as Secure Element or NFC-enabled Secure Element. The **Trusted OS** is not only a holder of the Trusted Applications which run on top of it but also a supporting facility provider for application developers. In order to guarantee the root of trust of the TEE, it is authenticated and then isolated from the Rich OS during the secure boot process. A **Trusted Application** (TA), called also *Trustlet* [25], is an application running inside the TEE which provides security services to Client Applications, i.e., applications running in top of the REE. A Trusted Application may be a DRM application, a payment application, a corporate application, or a transport application. Regarding the **TEE APIs**, we distinguish two classes: public and private. Public APIs, i.e.,TEE Client API and TEE Functional API, provide a communication interface to Client Application which enables exchanges with TEE. However, Private APIs, i.e., TEE Internal API, TEE Secure Element API (TEE

SE API) and Trusted User Interface (TUI), provide communication interfaces to Trusted Applications enabling exchanges with Secure Element Applications and usage of TEE resources. Finally, the TEE is about to be certified EAL2.

Additional details about TEEs can be found in [26], [27].

2.4.4 Comparison

Compared to the SIM card, the TEE is worthwhile only with respect to the memory size, hence, the computing speed. The TEE platform should provide roughly some mega bytes of memory, unlike the SIM card which cannot exceed a dozen of kilobytes. Indeed, the TEE would use the smartphone infrastructure which makes it more efficient compared to the SIM card which is a resource constrained component. Using the smartphone infrastructure can also be considered as a major drawback: the TEE cannot run when the smartphone is off or when the battery is flat. On the other hand, a NFC-enabled SIM card can run (slowly) and exchange with a NFC reader even if the smartphone is off. This is because it can be powered by the reader. Functioning in any kind of situation is important in services such as public transport: the user must be able to use his mobile ticket or pass even if he has no battery. Regarding the security level, the SIM card offers more security to the software running within it as it is a tamper-resistant secure element which can be certified EAL4+. This would enable the integrity of the running code and stored credentials. Finally, another issue of TEE remains in their implementation. TEE is an emerging environment, hence, its implementations are very unstable and the developing tools are not always available.

	REE	**TEE**	**NFC-enabled SIM Card**
Security level	-	EAL2	EAL4+
Memories	≈ GBs	≈ some MBs	≈ some KBs
Flexibility	×	×	√
Ease of development	√	×	√
NFC, if battery off	×	×	√

TABLE 2.2: Comparison between a REE, a TEE and a NFC-enabled SIM Card

Because the SIM card offers the highest security, can run even when the smartphone battery if off or down and presents more development easiness, we choose to implement our transport application within a NFC-enabled SIM card. We would like to emphasize that even if in the context of this phD, we did our implementations on a SIM card, the proposed protocols can be easily implemented on a TEE (if available).

2.5 Conclusion

In this chapter, we described the contactless mobile services and their main security challenges. Then, we focused on the public transport service as it is the most expanded. We detailed its main phases along with the relevant, security, privacy and functional

issues. We distinguished two use cases of the public transport service: the m-pass and m-ticketing which presents more constraints that the m-pass.

In this thesis, we aim to build a secure and privacy-preserving public transport system. Regardless of the use case, the system must ensure its security and a high privacy level for users while fulfilling the stringent time constraint (transport product validation $< 300\ ms$). The security requirements encompass the unforgeability of the transport product, the correctness and non-frameability of the system. The privacy requirements consist of the user's anonymity and the unlikability of a user's trips. We also choose to implement our protocols within a NFC-enabled SIM card. This choice is made owing to the study of the smartphone security feature in the previous section. Indeed, the SIM card enables a higher protection for its running code and is operational even if the smartphone battery is down or off (this would enable the use of the transport product even when the battery is down or off).

Our implementations also comply with a TEE platform. However, unlike the SIM card, the TEE cannot be physically moved from one smartphone to another one. In such circumstances, a new problem arises: how can the user securely migrate his credentials, if the service is implemented within a TEE and he changes his smartphone? We give a solution to this problem in Chapter 8.

In next chapter, we present the cryptographic building blocks that we will use later in our constructions in Chapters 4, 6, 7 and 8.

Chapter 3

Cryptographic Foundations and Basic Protocols

Contents

Dans ce chapitre, nous présentons quelques outils mathématiques de base, à savoir, les groupes, les corps, les corps finis, les courbes elliptiques et les applications bilinéaires de type couplages. Nous spécifions ensuite quelques fonctions de base utilisées dans les schémas cryptographiques: les machines de Turing, quelques définitions sur la complexité, les fonctions à sens unique, les fonctions de hachage, les fonctions pseudo-aléatoires et les engagements. En outre, nous décrivons les hypothèses calculatoires que nous utiliserons dans les chapitres suivants pour prouver la sécurité de nos protocoles. Enfin, nous présentons les schémas cryptographiques qui nous seront utiles dans la suite de la thèse, en particulier, la fonction pseudo aléatoire introduite par Dodis et Yampolskiy, des schémas de chiffrement, i.e., chiffrement d'ElGamal, chiffrement de Paillier, chiffrement à seuil et proxy de rechiffrement, des schémas de signature, i.e., signature RSA, signature Boneh-Boyen et signature Camenisch-Lysyanskaya, et enfin les preuves d'appartenance à un ensemble.

3.1 Introduction

In this chapter, we first provide an introduction to some mathematical and cryptographic tools that will be used later in this thesis. Then, we announce the computational assumptions on which relies the security of our protocols. Finally, we present the basic cryptographic schemes, that we will use in our protocols in Chapter 4, Chapter 6, Chapter 7, and Chapter 8. Readers familiar with these topics may skip this chapter.

3.2 Preliminaries

In this section, we describe some algebraic tools, namely groups, fields, finite fields, elliptic curves and bilinear maps, that we will use in this thesis.

3.2.1 Mathematical Tools

3.2.1.1 Groups

Definition 1 (Group). *A group $\langle \mathbb{G}, * \rangle$ is a nonempty set $\mathbb{G}$ along with an operation "$*$" that maps $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}$. This mapping satisfies the following properties:*

- ***Associativity:*** $\forall (x, y, z) \in \mathbb{G}^3$, $(x * y) * z = x * (y * z)$.
- ***Identity element:*** $\exists!\ e \in \mathbb{G}\ /\ \forall x \in \mathbb{G},\ x * e = e * x = x$. *$e$ is called identity element of $\mathbb{G}$.*
- ***Inverse:*** $\forall x \in \mathbb{G}, \exists!\ y_x \in \mathbb{G}\ /\ x * y_x = y_x * x = e$. *$y_x$ is called inverse of x.*

Moreover, $*$ may be **commutative**: $\forall (x, y) \in \mathbb{G}^2$, $x * y = y * x$. In this case, the group is called *abelian* or *commutative*.

A group $\mathbb{G}$ is called *finite* if $|\mathbb{G}|$ is finite. $|\mathbb{G}|$ is called the *order* of $\langle \mathbb{G}, * \rangle$ (or simply of $\mathbb{G}$).

A group is called *cyclic* if: $\exists g \in \mathbb{G}$ such that

$$\forall x \in \mathbb{G}, \exists n \in \mathbb{Z}\ /\ x = \underbrace{g * g * g * ... * g * g}_{n\ times}$$

g is called a *generator* of $\mathbb{G}$. If $\mathbb{G}$ is finite and has a prime order, all the elements of the group except the identity element are generators of $\mathbb{G}$.

There are two possible notations: additive and multiplicative notations. The additive notation is used only if $*$ is commutative.

In the additive notation, the symbol"$*$" is replaced by "$+$", the identity element is denoted by "0", and the inverse of an element x is denoted by $-x$.

In the multiplicative notation, the symbol "$*$" is replaced by "$\times$" or ".", the identity element is denoted by "1", and the inverse of an element x is denoted by x^{-1}.

In this thesis, we consider a specific abelian group which is an elliptic curve. We give further details about this latter in Section 3.2.1.3. In order to finely define the elliptic curves, we first introduce the concepts of field, finite field and field characteristic.

3.2.1.2 Fields

Definition 2 (Field). *A field* $\langle K, +, . \rangle$ *is a nonempty set* K *along with two operations "+" (called addition) and "." (called multiplication) that map* $K \times K$ *to* K*. This mapping satisfies the following properties:*

- $\langle K, + \rangle$ *is an abelian group of identity element denoted by "0".*
- $\langle K^*, . \rangle$ *is a group of identity element denoted by "1" and where* K^* *denotes* $K - \{0\}$*.*
- *The additive identity element "0" and the multiplicative identity element "1" are distinct (*$0 \neq 1$*).*
- ***Distributive multiplication over addition:*** $\forall a, b, c \in K$*,* $a.(b + c) = (a.b) + (a.c)$ *and* $(a + b).c = (a.c) + (b.c)$

Moreover, "." may be commutative. In this case, the field is called *commutative*.

A *field characteristic* is defined as the smallest number n such that $1 + 1 + ... + 1$ (n times) is equal to 0. If this sum never reaches 0, the field is said to have a characteristic equal to zero.

A field K is called *finite* if $|K|$ is finite. $|K|$ is called the *order* of the field.

Any finite field $\langle K, +, . \rangle$ satisfies the following properties:

- K is commutative.
- The multiplicative group $\langle K^*, . \rangle$ is cyclic.
- The order of K is in the form p^n, where p is the characteristic and is prime, and n is an integer $\geqslant 1$. Conversely, for any prime integer p and any integer $n \geqslant 1$, there exists one and only one field (up to isomorphism) such that $|K| = p^n$. This field is denoted by $\mathbb{F}_{p^n}$.

3.2.1.3 Elliptic Curves

The elliptic curves over finite fields have been introduced in cryptography by Miller in 1985 [28] and by Koblitz [29] in 1987. Compared to multiplicative groups of a finite field, these curves offer the same security level with smaller elements. Moreover, the implementations based on elliptic curves require less resources which make it suitable for resource constrained environments like the SIM cards.

It is defined as follows:

Definition 3 (Elliptic curve over a finite field). *Let K be a finite field. The elliptic curve E(K) over K is the set of couples (x, y) verifying the Weierstrass equation:*

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in K$*, together with the "point at infinity"* $\mathcal{O}$*.*

Hence,

$$E(K)=\{(x,\ y) \in K^2,\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}$$

If K has a characteristic different from 2 or 3, the Weierstrass equation for E may be expressed in the following short form:

$$E: y^2 = x^3 + ax + b \text{ where } a, b \in K \text{ and } 4a^3 + 27b^2 \neq 0.$$

As mentioned previously, an elliptic curve has the structure of an abelian group. Indeed, we can define a point addition law, denoted "+", and consider the point at infinity $\mathcal{O}$ as the identity element. We can apply the *tangent and secant* method to geometrically perform addition of two points on the curve.

In the sequel, we use the short form of Weierstrass equation.

Definition 4 (Addition law - tangent and secant method). *For all points P and Q on the elliptic curve E(K), the addition of P and Q meets the following rules:*

- $-\mathcal{O} = \mathcal{O}$
- *Q = -P implies that* $Q + P = \mathcal{O}$
- $\mathcal{O} + P = P$ *and* $P + \mathcal{O} = P$
- $P = (x_1, y_1) \neq O$ *implies that* $-P = (x_1, -y_1)$
- *If* $P, Q \neq \mathcal{O}$ *and* $Q \neq -P$*, the tangent and secant method enables to define the point P+Q as follows:*
  - *If* $P \neq Q$*, let R be the unique third intersection point (when accounting for multiplicity) of the line containing P and Q, and E(K).*
  - *If* $P = Q$*, let R be the second intersection point of the tangent line to the curve at this point and E(K).*

  then $P + Q = -R$*, as described in Figure 3.1a and Figure 3.1b.*

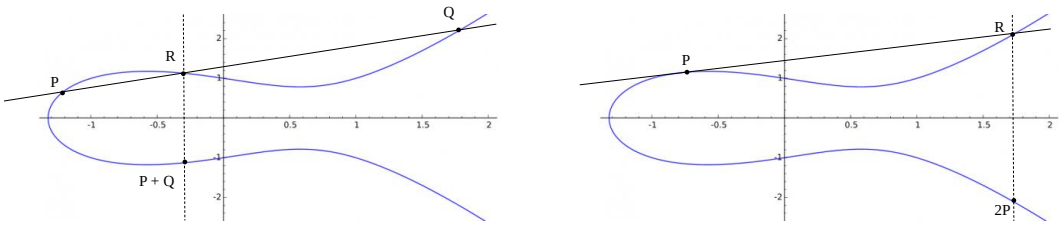
The addition law of $(E(K), +)$ can be described algebraically as well as geometrically. If K has a characteristic different from 2 or 3, the addition of the points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ is denoted $R = (x_R, y_R)$ and defined as follows if we consider the short Weierstrass equation.

$$\begin{aligned} x_R &= \lambda^2 - x_P - x_Q \\ y_R &= \lambda(x_P - x_R) - y_P \end{aligned}$$

where λ is defined as follows:

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

For further details about the elliptic curves, we refer to the book of Silverman [30]. This book also details bilinear maps on elliptic curves that we use in the construction of our protocols and present below.



(A) Adding Two Points on an Elliptic Curve (B) Doubling a point on an elliptic curve

FIGURE 3.1: Addition Operations on an Elliptic Curve

3.2.1.4 Bilinear Maps

Early in the nineties, bilinear maps have been used as means of attacking cryptosystems [31, 32]. Later in 2000, Joux [33] presented the first cryptographic protocol using such maps. This protocol is a tripartite variant of the Diffie–Hellman protocol. It enables an efficient key exchange between three participants using only one exchange. Since then, new cryptographic systems appeared and are now able to provide new properties that were either unachievable or very costly.

In the following, we give the definition of a bilinear map.

Definition 5 (Bilinear map). *Let p be a prime number and $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative groups of order p. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is said bilinear if it satisfies the following properties:*

- ***Bilinearity:*** *For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.*
- ***Non-degeneration:*** *For $g_1 \neq 1_{\mathbb{G}_1}$ and $g_2 \neq 1_{\mathbb{G}_2}$, $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.*
- ***Computability:*** *e is efficiently computable. (Efficiency will be defined in Section 3.2.2.2.)*

In cryptography, there are two *pairings* that are used for the design of bilinear maps: the Weil pairing [34] introduced by André Weil in 1940 for other purposes than the cryptography and the Tate pairing [35]. This latter is twice as fast as the Weil pairing. It is also used on more curves than the Weil pairing and even on non elliptic curves. By misuse of language, the bilinear maps based on pairings are often called pairings.

Usually, the groups are chosen as follows. $\mathbb{G}_1$ is a sub-group of an elliptic curve on a finite field. $\mathbb{G}_2$ is a sub-group of the multiplicative group of a finite field.

We would like to emphasize that, in this thesis, we use the pairing as a "black box". Moreover, when designing a cryptographic system using pairings, it is important to check that the used curves provide the functionalities required by this system.

3.2.2 Basic Functions

In this section, we start by defining the Turing machines that will be used in complexity definitions. Then, we define some basic functions that we will use later in this thesis.

3.2.2.1 Turing Machines

The Turing machines have been introduced by Turing in 1937 [36]. These machines can be used to simulate any computer algorithm and are formally defined as follows.

Definition 6 (Turing Machine). *A Turing machine is defined by a 7-tuple (Σ, $\mathcal{Q}$, σ, δ, Δ, 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.*
- σ*:* $\mathcal{Q} \times \Sigma \rightarrow \Sigma$ *is the writing function.*
- δ*:* $\mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ *is the state changing function.*
- Δ*:* $\mathcal{Q} \times \Sigma \rightarrow \{L, R\}$ *is the transition function where L is a left shift and R is a right shift.*
- $q_0 \in$ *is the initial state.*
- $\mathcal{F} \subset \mathcal{Q}$ *is the set of final states.*

On an input $\mathcal{I}$, a Turing machine can have various types of complexity. In this thesis, we focus on Time Complexity, denoted by TM($\mathcal{I}$). It is commonly estimated by counting the number of Turing machine steps from the initial to the final state. In addition, it is often desirable that a Turing machine runs in polynomial time. In the subsequent section, we give further details about this complexity time class.

3.2.2.2 Basic complexity definitions

In this section, we give the definitions of polynomial time complexity, negligible function and, negligible and overwhelming probability.

A Turing machine (or an algorithm) is said to be of polynomial time if its running time is upper bounded by a polynomial expression in the size, n, of the input for the algorithm. Formally speaking, a polynomial time complexity is defined as follows:

Definition 7 (Polynomial Time Complexity). *Let $\mathcal{M}$ be a Turing machine. We note $T_{\mathcal{M}}(n) = sup\{T_{\mathcal{M}(\mathcal{I})}, |\mathcal{I}| = n\}$.*

The time complexity of $\mathcal{M}$ is called polynomial if

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

In literature, polynomial time is often synonym of *efficient* and a problem is called *hard* if there is no known polynomial time algorithm to solve it.

Definition 8 (Negligible Function). *A function $\nu : \mathbb{N} \to \mathbb{R}^+$ is called negligible, if*

$$\forall c \in \mathbb{N}^*,\ \exists k_c \in \mathbb{N} \setminus \forall k \geqslant k_c,\ \nu(k) < k^{-c}$$

We can define the concept of negligible and overwhelming probability using the negligible functions.

Definition 9 (Negligible and Overwhelming Probability). *Let P be a probability depending on a positive integer k (called security parameter). P is said negligible if it is a negligible function of k. P is said overwhelming if the probability 1-P is negligible.*

3.2.2.3 One-Way Function

One way functions are a fundamental tool for cryptography. Indeed, it is a function that is easy to compute, but hard to invert. Its definition is based on negligible functions, i.e., functions that are asymptotically smaller than the inverse of any polynomial.

Definition 10 (One-way function). *Let $f : \{0,\ 1\}^* \to \{0,\ 1\}^*$ be a function. f is called a one-way function if it satisfies the following requirements:*

- *There is an efficient algorithm, that when has $x \in \{0,\ 1\}^*$ on input, outputs $f(x)$.*
- *There is no efficient algorithm that can inverse f. Formally speaking, for any efficient algorithm $\mathcal{A}$ and for all k sufficiently large, the probability*
  $$Pr[x \leftarrow \{0,\ 1\}^k;\ y \leftarrow f(x);\ x' \leftarrow \mathcal{A}(1^k, y) :\ f(x') = f(x)]$$
  is negligible.

Among the families of the one-way functions, there are hash functions which are frequently used in cryptography. Therefore, we give further details about then in the following section.

3.2.2.4 Hash Function

Originally, hash functions were used, in computer science, for data storing. They also enable to map binary strings of arbitrary finite length to reduce any message into a particular set, e.g., an elliptic curve, a multiplicative group of a finite field... Informally speaking, a hash function is defined as follows:

$$\text{H: } \{0,\ 1\}^* \to F$$

where F is a set such that $|F| \leqslant 2^k$ with k small enough. For instance, F can be the set $\{0,\ 1\}^k$.

In this thesis, we use a specific kind of hash functions: cryptographically secure hash functions. These functions ensure some particular security properties.

Definition 11 (Cryptographically secure hash function). *A cryptographically secure hash function H is a hash function which satisfies the following properties:*

- ***One-way:*** *H is a one-way function.*
- ***Weak collision resistant:*** *For a given $m \in \{0,\ 1\}^*$, it is hard to find $m' \neq m$ such that $H(m) = H(m')$ [37]. In other words, for a given $m \in \{0,\ 1\}^*$ and for any efficient algorithm $\mathcal{A}$, the probability*
$$Pr[m' \leftarrow \mathcal{A} : H(m) = H(m') \wedge m' \neq m]$$
is negligible.
- ***Strong collision resistant:*** *It is hard to find m and $m' \in \{0,\ 1\}^*$ with $m \neq m'$ such that $H(m) = H(m')$ [38]. In other words, for any efficient algorithm $\mathcal{A}$, the probability*
$$Pr[m, m' \leftarrow \mathcal{A} : H(m) = H(m') \wedge m' \neq m]$$
is negligible.

Another useful building block in cryptography is the pseudo-random function. We give its definition in the following section.

3.2.2.5 Pseudo-Random Function

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

Definition 12 (Pseudo-random function family). *Let m and l be two polynomial functions, k be a security parameter and $\mathcal{R}_k$ be the set of functions:*
$$\{0,\ 1\}^{m(k)} \rightarrow \{0,\ 1\}^{l(k)}.$$
$\mathcal{F}_k = \{F_s\}_{s \in \{0,\ 1\}^k}$ is 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 has the following properties:

- ***Efficiency:*** *Consider s and x, there is an efficient algorithm that can assess $F_s(x)$*
- ***Pseudo-random:*** *For any adversary $\mathcal{A}$ such that the number of its requests to F is polynomially limited, for all k, the probability*
$$|\underset{s \in \{0,1\}}{Pr}[\mathcal{A}(F_s) = 1] - \underset{R \in \mathcal{R}_k}{Pr}[\mathcal{A}(R) = 1]|$$
is negligible.

3.2.2.6 Commitment

A commitment allows a party to commit to a message m (*secret*) to another party. Later, the commitment can be opened by having the issuer of the commitment send additional information to the party to which he has committed to. Formally speaking, we define a commitment as follows.

Definition 13 (Commitment). *A commitment scheme considers the three algorithms:*

- `Setup`*: a probabilistic algorithm that generates the parameters and keys. It takes on input an integer k and outputs the public parameters and key params.* $(params) \leftarrow$ `Setup`(1^k).

- `Commit`*: the commitment algorithm that takes on input the public parameters params, a message m and the public key of the scheme. It outputs the commitment C of the message m and the data about the decommitment r.* $(r,\ C) \leftarrow$ `Commit`$(params)$

- `Decommit`*: the decommitment algorithm that takes on input a commitment C, a message m and the data r. It outputs* 1*, if C is a commitment of m, otherwise, it outputs* 0*.*
  $\{0,\ 1\} \leftarrow$ `Decommit`$(C,\ m,\ r)$

A commitment scheme must satisfy the following security properties:

- ***Hiding property*:** the commitment C does not reveal any information on the secret message m to the other party.

- ***Binding property*:** the commitment prevents the committing party from changing the message, that he has committed, at a later stage.

3.2.3 Cryptographic Primitives

In this section, we formally define the main cryptographic primitives that we will use later in this thesis. We also describe and classify the attacks that can occur against these primitives. The definitions provided in this section only refer to the so called public key cryptography. We start by the concept of public key encryption and digital signature that have been introduced by Diffie and Hellman in their paper "New directions in cryptography" [39]. Then, we present the proof of knowledge.

3.2.3.1 Public Key Encryption

In cryptography, the encryption enables a user A to transform a message such that only the authorized user B can read it. The key used for encryption, the public key of the user B, can be published, while the key used for decryption, the secret key of the user B, must be kept secret.

Any user can send a ciphertext to the user B provided that he knows the public key of the user B. Only the user B can retrieve the plaintext from the ciphertext.

Definition 14 (Public key encryption scheme). *A public key encryption scheme is defined by three algorithms:*

- `Setup`*: a probabilistic algorithm that takes on input an integer k and outputs the public parameters params, a public key pk and its relevant private key sk.*
  $(params,\ pk,\ sk) \leftarrow$ `Setup`(1^k)

- `Enc`*: this algorithm is often probabilistic. It takes on input a message* $m \in \{0,\ 1\}^*$ *and a public key pk and outputs a ciphertext c.*
  $c \leftarrow$ `Enc`$(params,\ m,\ pk)$

- `Dec`*: a deterministic algorithm that takes on input a ciphertext c and a private key sk and outputs the encrypted message m.*
  $m \leftarrow \texttt{Dec}(params,\ c,\ sk)$

Any encryption scheme must satisfy the two following security properties:

- ***Validity:*** a message encrypted using a public key pk must be properly decrypted using the corresponding private key sk.
- ***Confidentiality:*** Given a ciphertext, no information can be revealed regarding the encrypted message without the knowledge of the private key.

In this thesis, we use a particular refinement of encryption called proxy re-encryption. This notion first appeared in 1998 when Blaze et al [40] introduced the notion of "atomic proxy cryptography", in which a semi-trusted proxy computes a function that converts ciphertexts for Alice (computed under Alice's public key pk_A) into ciphertexts for Bob (ciphertexts can be opened using the private key of Bob sk_B) without seeing the underlying plaintext. Later, many works such as [41] have been done in order to improve the security and the performance of this kind of schemes.

Definition 15 (Proxy re-encryption scheme). *A proxy re-encryption scheme is defined by the following algorithms:*

- `Setup`*: a probabilistic algorithm that takes on input an integer k and outputs the public parameters params, a public key pk and its relevant private key sk.*
  $(params,\ pk,\ sk) \leftarrow \texttt{Setup}(1^k)$
- `ReKeyGen`*: let D_i be a set that consists of the public key pk_i of the user i or the private key sk_i of the user i or (pk_i, sk_i). This algorithm takes on input the public parameters params, D_A^{Source} of the user A and D_B^{Target} of the user B and outputs the re-encryption key $R_{A\rightarrow B}$ from A to B.*
  $R_{A\rightarrow B} \leftarrow \texttt{ReKeyGen}(params,\ D_A^{Source},\ D_B^{Target})$
- `Enc`*: this algorithm is often probabilistic. It takes on input a message $m \in \{0,\ 1\}^*$ and a public key pk and outputs a ciphertext c_{pk} for this public key.*
  $c_{pk} \leftarrow \texttt{Enc}(params,\ m,\ pk)$
- `Re-enc`*: this algorithm takes on input the public parameters params, the re-encryption key $R_{A\rightarrow B}$ and the ciphertext c_{pk_A} of the user A. It outputs a ciphertext c_{pk_B} of the same message (without knowledge of the message content) for the user B or $\perp$ it the ciphertext is not valid (for the re-encryption key).*
  $\{c_{pk_B}, \perp\} \leftarrow \texttt{Re-enc}(params, R_{A\rightarrow B}, c_{pk_A})$
- `Dec`*: a deterministic algorithm that takes on input a ciphertext c_{pk} and a private key sk and outputs the encrypted message m or $\perp$ if the ciphertext is not valid.*
  $\{m,\ \perp\} \leftarrow \texttt{Dec}(params,\ c_{pk},\ sk)$

A proxy re-encryption scheme has additional security properties which are:

- ***Non-transitive:*** The proxy, alone, cannot re-delegate decryption rights. For example, from $R_{A\rightarrow B}$ and $R_{B\rightarrow C}$, he cannot produce $R_{A\rightarrow C}$.

- ***Non-transferable:*** The proxy and a set of colluding delegatees cannot re-delegate decryption rights. For example, from $R_{A\to B}$, D_B, and D_C, they cannot produce $R_{A\to C}$.

The security of an encryption scheme (or any cryptographic scheme) also depends on the attackers goals and capabilities. An attacker chooses his victim (a user) and gets the public key of this victim. If the attacker aims to retrieve the plaintext corresponding to some cirphertext, we can distinguish two types of attacks:

- ***Total break:*** given the public key of the victim, the attacker deduces the private key of the victim. He can then decrypt any ciphertext of this victim.
- ***Partial break:*** the attacker retrieves a plaintext corresponding to a given ciphertext without knowing the private key of the victim. Thus, he may not be able to decrypt other ciphertext.

Instead of retrieving the plaintext corresponding to a ciphertext, an attacker may just try to retrieve some information about the plaintext. in such a case, he would attack the following properties:

- ***Non-Malleability (NM):*** an attacker is not able to transform a ciphertext c of an unknown message m into another ciphertext c' of a known message m' such that m and m' are bound in a certain way.
- ***Semantic security:*** given only a ciphertext, an attacker can not determine any information on the plaintext.
- ***Indistinguishability (IND):*** an attacker can not differentiate two ciphertext of two different messages.

Attacks may also be classified based on what type of information the attacker has:

- ***Chosen Plaintext Attack (CPA):*** the attacker can obtain the ciphertexts for plaintexts that are previously chosen. This is possible as the encryption key is public.
- ***Chosen Ciphertext Attack (CCA1):*** the attacker has a system that enables him to retrieve the decryption of chosen ciphertexts under an unknown key. Using these information, the attacker may try to retrieve the private key used for the decryption.
- ***Adaptative Chosen Ciphertext Attack (CCA2):*** this is an interactive form of chosen-ciphertext attack. The ciphertexts to be decrypted may change during the attack.

Usually, a scheme having the security property XX with respect to the attack YY is called XX-YY. For instance, an indistinguishable encryption scheme against the adaptative chosen ciphertext attack is IND-CCA2.

3.2.3.2 Digital Signature

Informally, a digital signature ensures the authenticity and integrity of a message. On one hand, anyone must be able to verify a signature when given the public key of the signer and the message. On the other hand, only the signer, i.e., the party that knows the secret key, must be able to compute a signature.

Definition 16 (Digital signature scheme). *A digital signature scheme is defined by the following algorithms:*

- `Setup`*: a probabilistic algorithm that takes on input an integer* k *and outputs the public parameters* $params$*, a public key* pk *and its corresponding private key* sk*.* $(params,\ pk,\ sk) \leftarrow \texttt{Setup}(1^k)$.
- `Sign`*: this algorithm is often probabilistic. It takes on input a message* $m \in \{0,\ 1\}^*$ *and a private key* sk*, and outputs a signature* σ *of* m*.* $\sigma \leftarrow \texttt{Sign}(param,\ m,\ sk)$.
- `Verify`*: a deterministic algorithm that takes on input a signature* σ*, a message* m *and a public key* pk*. It outputs* 1 *if* σ *is s signature of* m*, otherwise it outputs* 0*.* $\{0,\ 1\} \leftarrow \texttt{Verify}(params,\ \sigma,\ m,\ pk)$

Any type of the signature scheme must satisfy the two following security properties:

- ***Validity:*** a signature of a message using a private key *sk* can be verified by anyone using the corresponding public key *pk*.
- ***Integrity:*** the signature of a message prevents any change of the message at a later stage. Indeed, any change of the message after the signature invalidates the signature.
- ***Non-repudiation:*** a signer that has signed a message cannot at a later time deny having signed it.

In this thesis, we use a particular refinement of signature, called group signature. Using this type of signature, a member of a group can anonymously sign a message on behalf of the group. However in case of a dispute, the anonymity of a group signature can be lifted by one or several designated trusted authorities (called revocation authorities). This concept has been introduced by Chaum and van Heyst [42] in 1991.

A variant of group signature, called list signature, has been introduced by Canard and his co-authors [43]. With a list signature scheme, it becomes possible, when specific conditions are met, to link signatures produced by a member of the group. In particular, this link becomes possible if the signatures have been produced during a given sequence of time, i.e., a specific time period. A similar technique has later been used in [44] and is called Direct Anonymous Attestation (DAA). The main difference between DAA and list signatures is that in DAA, the role of the signer is split between a main signer with limited computational and memory capabilities, e.g., a Trusted Platform Module (TPM) or a SIM card, and an assistant signer, who is computationally more powerful but is considered less secure, e.g., a mobile phone containing the SIM card.

Definition 17 (Group signature scheme). *A group signature scheme considers three type of participants:*

- *A group manager that registers a user to the group.*
- *A member of a group who firstly requires registration of the group manager. Once registered, a member of a group can sign anonymously on behalf of the group.*
- *A revocation authority that is able to lift the anonymity of a given signature.*

A group signature scheme is a signature scheme defined by five algorithms.

- `Setup`*: a probabilistic algorithm that takes on input an integer k and outputs the public parameters $params$, the group's public pk_G, a group manager private key sk_{MG}, the revocation authority's key sk_{RA}.*
  $(params, pk_G, sk_{MG}, sk_{RA}) \leftarrow \texttt{Setup}(1^k)$
- `Join`*: an interactive protocol between a user and a group manager. At the end of this protocol, the user becomes a member of the group and gets secret key sk_U and a membership certificate to the group.*
- `Sign`*: a probabilistic algorithm that takes on input a message m, a membership certificate to a group and a secret key sk_U, and outputs a signature σ of m.* $\sigma \leftarrow \texttt{Sign}(param,\ m,\ sk_U)$
- `Verify`*: a deterministic algorithm that takes on input a message m, a signature σ and a group's public key pk_G, and outputs 1 if the signature is valid, otherwise, it outputs 0.* $\{0,\ 1\} \leftarrow \texttt{Verify}(params,\ \sigma,\ m,\ pk_G)$
- `Open`*: an algorithm that takes on input a message m, a valid signature σ, a group's public key pk_G and the revocation authority's key sk_{RA}, and outputs the identity of signer.* $ID_U \leftarrow \texttt{Open}(m,\ \sigma,\ pk_G,\ sk_{RA})$

A group signature scheme has additional security properties which are:

- ***Soundness and completeness:*** this property is a variant of the validity property mentioned above. A signature generated by a group member can be verified by anyone using the group public key.
- ***Anonymity:*** no one, except the revocation authority, can lift the anonymity of a signature and retrieve the identity of the signer.
- ***Unlinkability:*** given two messages and their corresponding signatures, no one, except the revocation authority, can decide whether the signatures have been generated by the same user or not.
- ***Non-framebility:*** no one can falsely accuse a group member of signing a message.
- ***Coalition resistance:*** any valid signature must be linked to group member.

Like the encryption scheme, the security of a signature scheme depends on the attackers goals and capabilities. We can distinguish four types of attacks:

- ***Total break:*** like in the encryption scheme, the attacker deduces the private key from the public key of the victim.
- ***Universal forgery:*** the attacker can generate a valid signature σ of any given message m without knowledge of the private key.
- ***Existential forgery:*** the attacker can generate at least one message/signature pair (m, σ), where σ was not produced by the victim and without knowledge of the private key.
- ***Selective forgery:*** the attacker can generate a message/signature pair (m, σ) where m has been chosen by the adversary prior to the attack and without knowledge of the private key. m must be fixed before the start of the attack and may be chosen to have interesting mathematical properties with respect to the signature algorithm.

Attacks may also be classified based on what type of information the attacker has:

- ***Attack with no message:*** the attacker has only the public key of the victim.
- ***Known Message Attack (KMA):*** the attacker has the public key of the victim and a set of message/signature.
- ***Adaptive Chosen Message Attack (CMA):*** the attacker has the public key of the victim and can get the signature of any message of his choice. The messages are progressively selected.

3.2.3.3 Proofs of Knowledge

In cryptography, a proof of knowledge (POK) is an interactive proof that enables a prover to prove that he has a given data. In 1986, Fiat and Shamir proposed a heuristic method to transform a three steps protocol in a non-interactive knowledge signature [45]. Indeed, the challenge sent by the verifier is replaced by the hash of the public parameter and a commitment. We focus on a variant called Zero-knowledge proof of knowledge.

Zero-knowledge proof of knowledge In a zero-knowledge proof, a prover convinces interactively a verifier that he knows a set of secret values $(\alpha_1, \alpha_2, ..., \alpha_n)$ verifying a given relation $\Re$ without revealing anything else. Such a proof will be denoted, in the sequel:

$$ZKPK((\alpha_1, \alpha_2, ..., \alpha_n) : \Re(\alpha_1, \alpha_2, ..., \alpha_n)).$$

A ZKPK should satisfy the following properties:

- ***Complete:*** a valid prover is accepted with overwhelming probability.
- ***Sound:*** a false prover should be rejected with overwhelming probability.
- ***Zero-knowledge:*** no information about the secret is revealed.

These ZKPK can be made non-interactive using the generic transformation introduced by Fiat and Shamir [45]. The resulting non-interactive proofs, sometimes called *signatures of knowledge* [46]. Such a signature of knowledge, for the relation $\Re$, will be denoted:

$$SOK((\alpha_1, \alpha_2, ..., \alpha_n) : \Re(\alpha_1, \alpha_2, ..., \alpha_n)).$$

3.3 Security

Proving the security of a cryptographic protocol is a major interest because it is a requirement to validate the protocol. In public key cryptography, proving the security of a scheme follows three steps:

- defining the security properties to be ensured.
- defining the computational assumption to be considered.
- presenting a reduction of the scheme security to a hard problem already defined in a computational assumption.

In this section, we recall the computational assumptions that we use later in this thesis and present the different models in which an adversary can be placed to conduct attacks.

3.3.1 Computational Assumptions

The security of our protocols, described in the next chapters, relies on the difficulty of mathematical problems resulting from the Number theory. In the following, we present the Discrete Logarithm (DL) assumption, the Decisional Diffie-Hellman (DDH) assumption, the eXternal Diffie-Hellman (XDH) assumption, the q-Strong Diffie-Hellman (q-SDH) assumption, the q-Decisional Diffie-Hellman Inversion (q-DDHI) assumption, the Decisional Composite Residuosity (DCR) assumption and the Lysyanskaya, Rivest, Sahai and Wolf (LRSW) assumption.

3.3.1.1 Discrete Logarithm (DL) Assumption

Definition 18 (Discrete Logarithm (DL) Assumption). *Let $\mathbb{G}$ be a cyclic group of prime order q, g be a generator of $\mathbb{G}$ and y be an element of $\mathbb{G}$, Discrete Logarithm assumption states that it is hard to find the integer $x \in \mathbb{Z}_q$ such that $y = g^x$ holds. The integer x is called the discrete logarithm of y in the base g and denoted $log_g(y)$.*

References for the discrete logarithm problem and the existing algorithm for solving include the survey of Odlyzko [47] and McCurley [48].

3.3.1.2 Symmetric Discrete Logarithm (SDL) Assumption

The Symmetric Discrete Logarithm (SDL) Assumption has been formalized in [49].

Definition 19 (Symmetric Discrete Logarithm (SDL) Assumption). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q, G_1 be an element of $\mathbb{G}_1$, and G_2 be an element of $\mathbb{G}_2$, Symmetric Discrete Logarithm Assumption states that it is hard to find the integer x given a tuple (G_1, $[x]G_1$, G_2, $[x]G_2$).*

3.3.1.3 One-more Discrete Logarithm (OMDL) Assumption

The One-more Discrete Logarithm (OMDL) Assumption has been introduced and formalized by Bellare, Namprempre, Pointcheval and Semanko in [50].

Definition 20 (One-more Discrete Logarithm (OMDL) Assumption). *Let G be a multiplicative group of prime order p, OMDL assumption states that given a random generator $g \in G$, a challenge oracle that produces a random group element Y_i when queried and a discrete logarithm oracle (with respect to g), it is hard to find, after t queries to the challenge oracle (where t is chosen by the adversary) and at most $t-1$ queries to the discrete logarithm oracle, the discrete logarithms of all t elements Y_i.*

3.3.1.4 Decisional Diffie-Hellman (DDH) Assumption

The Decision Diffie-Hellman problem was first mentioned in [51] where Stephan Brands presented a offline electronic payment system, even though there are earlier cryptographic systems that implicitly rely on the hardness of this problem (e.g. [52, 53]).

Definition 21 (Decisional Diffie-Hellman (DDH) Assumption). *Let $\mathbb{G}$ be a cyclic group of prime order p, DDH assumption states that given a random generator $g \in \mathbb{G}$, two random 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.*

3.3.1.5 eXternal Diffie-Hellman (XDH) Assumption

This assumption was firstly introduced by Boneh, Boyen and Shacham in the long version of [54] to prove the security of the presented scheme (a group signature scheme).

Definition 22 (eXternal Diffie-Hellman (XDH) Assumption). *Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear map, XDH assumption states that DDH assumption holds in $\mathbb{G}_1$.*

3.3.1.6 q-Strong Diffie-Hellman (q-SDH) Assumption

This assumption has been introduced by Boneh and Boyen in [55] in order to prove the security of the presented scheme (a group signature scheme).

Definition 23 (q-Strong Diffie-Hellman (q-SDH) Assumption). *Let $\mathbb{G}_1$ be a cyclic group, q-SDH assumption states that it is hard, given (g, g^y, g^{y^2},...,g^{y^q}) $\in \mathbb{G}_1^{q+1}$, to output a pair $(x, g^{1/(y+x)})$.*

3.3.1.7 q-Strong Diffie-Hellman I (q-SDH-I) Assumption

Definition 24 (q-Strong Diffie-Hellman I (q-SDH-I) Assumption). *Let $\mathbb{G}_1$ be a multiplicative group of prime order p, q-SDH-I assumption states that given three random generators (g_0, g_1, h)$\in \mathbb{G}_1^3$, a value $W' = g_0^{\gamma}$, an oracle $\mathcal{O}$ that on input a value $s \in \mathbb{Z}_p$ outputs a pair $(r, A = (g_1^s h)^{1/(\gamma+r)})$ with $r \in \mathbb{Z}_p$, it is hard to output a new triplet $(r', s', A' = (g_1^{s'} h)^{1/(\gamma+r')})$, with $(r', s') \in \mathbb{Z}_p^2$ such that s' has never been queried to $\mathcal{O}$.*

3.3.1.8 q-Decisional Diffie-Hellman Inversion (q-DDHI) Assumption

Definition 25 (q-Decisional Diffie-Hellman Inversion (q-DDHI) Assumption). *Let $\mathbb{G}$ be a multiplicative group of prime order p, q-DDHI assumption states that, given a random generator $g \in \mathbb{G}$ and the values $(g, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^q}) \in \mathbb{G}^{q+1}$, for a random $\alpha \in \mathbb{Z}_p$ and a candidate $X \in \mathbb{G}$, it is hard to decide whether $X = g^{1/\alpha}$ or not.*

3.3.1.9 The Decisional Composite Residuosity (DCR) Assumption

This assumption has been introduced by Paillier [56] in order to prove the security of the proposed encryption scheme.

Definition 26 (Decisional Composite Residuosity (DCR) Assumption). *DCR assumption states that it is hard to distinguish $\mathbb{Z}_{n^2}^n$ from $\mathbb{Z}_{n^2}^*$, where $\mathbb{Z}_{n^2}^n = \{z \in \mathbb{Z}_{n^2}^*, \exists y \in \mathbb{Z}_{n^2}^* : z = y^n mod n^2\}$.*

3.3.1.10 LRSW Assumption

This assumption has been introduced by Lysyanskaya, Rivest, Sahai and Wolf [57] in order to prove the security of the proposed scheme.

Definition 27 (LRSW Assumption). *Let $\mathbb{G}$ be cyclic group of prime order q and g be a generator of $\mathbb{G}$. the assumption LRSW states that, given $X = g^x$, $Y = g^y$ and an oracle $O_{X,Y}^{LRSW}$ that takes on input a message $m \in \mathbb{Z}_q$ and outputs $A = (a, a^y, a^{x+mxy})$ such that a is randomly chosen, it is hard to compute $A' = (a', a'^y, a'^{x+mxy})$ for another value a', if m has not been given as input to the oracle.*

After defining our computational assumption, we move to the third step of the process "proving the security of a scheme" which is "presenting a reduction of the scheme security to a hard problem".

3.3.2 Provable Security

In this section, we present the concept of reductionist security and the different security models that can be used to prove the security of a cryptographic scheme.

3.3.2.1 Reductionist Security

The main idea of the reductionist security is to formally prove that if an adversary $\mathcal{A}$ breaks a security property of a scheme, he can be used to solve a hard problem like those we have stated in the previous section. The first precepts of provable security have been introduced by Goldwasser and Micali [58] in 1984 for encryption. Later with Rivest, they defined this new concept for signature [59, 60]. Though polynomial, the proposed reductions had a computational high cost. Later several improvements occurred. In 1996, Bellare and Rogaway introduced the exact security approach [61] where the estimate of computational complexities of adversarial tasks is more accurate than what we would have with polynomial equivalence. Then, Ohta and Okamoto introduced the concrete security approach [62] enabling more efficiency. More recently, Pointcheval, in 2001, introduced the concept of practical security [63].

If we consider an adversary $\mathcal{A}$ reaching his goal in time t and a reduction that solves the hard problem in time $f(t)$, we can define three security concepts:

- Asymptotic security, if f is polynomial (bounded by a t polynomial).
- Exact security, if f is explicit.
- Practical security, if f is "small", e.g., linear.

If the reduction is polynomial, attacking the protocol is said to be hard as solving the hard problem.

In order to have the best possible reduction, we should precisely define the goals of the attacker. Therefore, we describe, in the following section, the different security models that can be used to define the environment where an attacker can perform his attack.

3.3.2.2 Security Models

In this section, we distinguish two security models: the Random Oracle Model and the standard Model.

Random Oracle Model In cryptography, random oracles can be used as an ideal replacement for hash functions in schemes where strong randomness assumptions are needed of the hash function's output. In such a case, the attacker cannot deduce any information from the hash values that it has already been obtained and it can not build hashes for new values. In such a model, the security of the scheme does not depend of the specified hash functions. A scheme is secure if an attacker solves a hard problem in order to break it. The concept of random oracles has been introduced by Fiat and Shamir [45] in 1986 and then modeled by Belleare and Rogaway [64] in 1993.

Standard Model The Random Oracle model has been criticized as very restrictive, even if it is widely used. Canetti, Goldreich and Halevi [65, 66] demonstrated that there are secure schemes under the random oracle model such that any implementation of the oracles makes them insecure. Bellare, Boldyreva and Palacio made proposals, in [67],

that ensure some properties only in the random oracle model. This assumption implies that the security of the scheme cannot be proven. In order to mitigate such problem, a new model has been introduced: Standard model. Unlike the random oracle model, the standard model does not consider any assumption regarding the used hash functions.

3.4 Cryptographic Schemes

In this section, we present cryptographic schemes that we will use in our constructions of the thesis.

3.4.1 Pseudo-Random Function

In this thesis, we use a pseudo-random function (Definition 12) that has been introduced by Dodis and Yampolskiy [68, 69]. Their construction works as follows. Let $\mathbb{G}$ be a cyclic group with prime order p, g_t a generator of $\mathbb{G}$, $s \in \mathbb{Z}_p$ a private secret. The pseudo-random function (PRF for short) F_s takes as input a message $k \in \mathbb{Z}_p$ and outputs $F_s(k) = g_t^{1/(s+k+1)}$.

The properties of this construction, given in definition 12, are ensured under the q-DDHI assumption in $\mathbb{G}$.

3.4.2 Pedersen Commitment

In this thesis, we use a commitment scheme that has been introduced by Pedersen in [70]. This commitment scheme provides perfect hiding and satisfies the binding property under the DL assumption.

`Setup` The parameters are:

- $\mathbb{G}$ a cyclic group with prime order p.
- g_1 and h_T are generators of $\mathbb{G}$.

`Commit`

- Let m be the message to be committed.
- Choose a random value $\nu \in \mathbb{Z}_p^*$
- Compute $C = g_1^m \times h_T^\nu$

C is a commitment of the message m.

3.4.3 Public Key Encryption

In this thesis, we use two public key encryption schemes (Definition 14), i.e., the ElGamal encryption scheme and the Paillier encryption scheme. Actually, we apply the threshold version of these schemes. We also utilize a proxy re-encryption scheme (Definition 15) based on ElGamal cryptosystem.

In this section, we detail the ElGamal and Paillier encryption schemes. We give the definition of a threshold cryptosystem and describe the proxy re-encryption scheme based on ElGamal cryptosystem.

3.4.3.1 ElGamal

The ElGamal [71] encryption scheme is a public key algorithm based on the Diffie-hellman [39]. The variant, that we use it in our m-ticketing system, works as follows.

`Setup`
The parameters are:

- $\mathbb{G}$ a cyclic group with prime order p.
- g_T a random generator of $\mathbb{G}$.

The private key is formed by $x \in \mathbb{Z}_p^*$ and the corresponding public key consists of the elements $(g_T, h = g_T^x)$

`Enc`
The ElGamal ciphertext of a message $m \in \mathbb{G}$ is computed as follows:

- Choose a random number $r \in \mathbb{Z}_p^*$.
- Compute $C_1 = g_T^r$ and $C_2 = m \times h^r$

The encryption of m is $(C_1,\ C_2)$.

`Dec`
The decryption of the ciphertext (C_1, C_2) is obtained through the following computation:

$$m = C_2 / C_1^x$$

The ElGamal cryptosystem is semantically secure under the Decisional Diffie-Hellman (DDH) assumption.

3.4.3.2 Paillier

We use the Paillier encryption scheme [56] in our m-ticketing system. It works as follows.

`Setup`
The parameters are:

- a and b random primes such that $a, b > 2, a \neq b, |a| = |b|$ and $gcd(ab, (a-1)(b-1)) = 1$
- $n = ab$
- $\Pi = lcm(a-1, b-1)$
- $K = \Pi^{-1} \ mod \ n$
- $g_P = (1+n)$

The public key is $pk = (n, g_P)$ and the corresponding secret key is $sk = (a, b)$.

`Enc`
To encrypt a message $m \in \mathbb{Z}_n$, a user performs the following operations:

- Choose a random number $r \in \mathbb{Z}_n^*$
- Compute $C = g_P^m r^n \ mod \ n^2$

The encryption of the message m is C.

`Dec`
The decryption algorithm is as follows:

$$m = ((C^{\Pi K} mod \ n^2) - 1)/n \ mod \ n^2$$

The Paillier cryptosystem, described above, provides semantic security against chosen-plaintext attacks (IND-CPA) under the Decisional Composite Residuosity (DCR) assumption.

3.4.3.3 Threshold Cryptosystem

In a threshold version, the ElGamal public key and its corresponding private key [72] (respectively the Paillier public key and its corresponding private key [73]) are cooperatively generated by n parties; however, the private key is shared among the parties. In order to decrypt a ciphertext, a minimal number of t (the threshold) out of n parties is necessary.

3.4.3.4 Proxy Re-Encryption Scheme

We use the proxy re-encryption building block in our TEE migration protocol. Blaze et al. introduced the notion in [40]. These approach is based on the ElGamal cryptosystem and it works as follows.

`Setup`
The parameters are:

- $\mathbb{G}$ a cyclic group with prime order q.
- g a generator of $\mathbb{G}$.

- a and b randomly chosen in $\mathbb{Z}_q^*$.

The private and public key of Alice are (a, g^a) and the private and public key of Bob are (b, g^b).

`ReKeyGen`
The re-encryption key is computed as follows: $R_{A\to B} = b/a = ba^{-1}\ mod\ q$

`Enc`
To encrypt a message $m \in \mathbb{G}$ that can be decrypted by Alice, a user does the following operations:

- Choose a random integer $r \in \mathbb{Z}_q^*$.
- Compute $g^r m$.
- Compute g^{ar}.

The ciphertext of Alice is $C_A = (g^r m,\ g^{ar})$

`Re-enc`
To re-encrypt the ciphertext of Alice (C_A), in other words, to convert a ciphertext of Alice (C_A) to a ciphertext of Bob (C_B), the proxy re-encryption computes $(g^{ar})^{R_{A\to B}}$.

$$(g^{ar})^{R_{A\to B}} = (g^{ar})^{b/a} = g^{br}$$

The ciphertext of Bob is $C_B = (g^r m,\ g^{br})$.

`Dec`
Bob can retrieve the plaintext corresponding to the ciphertext $C_B = (g^r m,\ g^{br})$ by computing $m = \frac{g^r m}{(g^{br})^{1/b}}$.

3.4.4 Digital Signature

In this thesis, we use group signature scheme (Definition 17) namely Boneh-Boyen signature and Camenisch-Lysyanskaya signature. We also utilize the RSA signature.

In this section, we detail these signature schemes.

3.4.4.1 RSA

The RSA signature schemes has been introduced together with the RSA encryption scheme [74]. It is based on the hash and sign concept and it is secure owing to the Full Domain Hash [64].

`Setup`
The parameters are:

- $p,\ q \in \mathbb{N}$ two prime integers.
- $n = pq$.

- e an integer co-prime with $\phi(n) = (p-1)(q-1)$ where ϕ is Euler's totient function.
- $d = e^{-1}\ mod\ \phi(n)$.
- $H : \{0,\ 1\}^* \rightarrow \mathbb{Z}_n$ a hash function.

The private key is d and the public key is $(n,\ e)$.

`Sign`
To sign a message $m \in \mathbb{Z}_n$, a user carries the following operations:

- Compute $m' = H(m)$.
- Compute $\sigma = m'^d\ mod\ n$

The signature of the message m is σ.

`Verify`
The signature σ is valid if $H(m) = \sigma^e\ mod\ n$.

3.4.4.2 Boneh-Boyen Signature

Based on the scheme proposed by Boneh and Boyen [54, 55, 75], it is possible to prove knowledge of a signature on a message, without revealing the signature nor the message. There are two variants of this signature scheme: with and without pairing. We detail these variants in the sequel.

Boneh-Boyen Signatures with pairings
`Setup`
The parameters are:

- $\mathbb{G}$ a cyclic group with prime order p.
- e a bilinear pairing.
- g_1 and g_2 two generators of $\mathbb{G}$.
- $y \in \mathbb{Z}_p$.

The private key is y and the public key is $Y = g_2^y$.

`Sign`
The signature of a message $m \in \mathbb{Z}_p$ is obtained by computing

$$\sigma = g_1^{1/(y+m)}.$$

`Verify`
A signature σ of m is valid if:

$$e(\sigma, Y g_2^m) = e(g_1, g_2).$$

Boneh-Boyen Signature without pairings
`Setup`

- $\mathbb{G}$ is a cyclic group with prime order p.
- e is a bilinear pairing.
- g_1 and g_2 two generators of $\mathbb{G}$.
- $y \in \mathbb{Z}_p$.

The private key is y and the public key is $Y = g_2^y$.

`Sign`
Similarly to Boneh-Boyen signatures with pairings, the signature A of a message $m \in \mathbb{Z}_p$ is obtained by computing

$$A = g_1^{1/(y+m)}.$$

This implies that $A^y = g_1 A^{-m}$.

`Verify`
The signature A of the message m is valid if the proof π is valid where π proves 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: $ZKPK(y : Y = g_2^y \wedge A^y = g_1 A^{-m})$. Such a proof of equality of discrete logarithms has been introduced in [76].

Theorem 1. The Boneh-Boyen (BB for short) signature scheme without pairings is existentially unforgeable under a weak chosen message attack under the q-Strong Diffie-Hellman (q-SDH) assumption, in the random oracle model.

Proof 1 (sketch). Under the q-Strong Diffie-Hellman assumption, it is impossible to find a message m (which was not given to the signing oracle) and a value A such that $A = g_1^{1/(y+m)}$, as proved in [55, 75]. Moreover, in the random oracle model, the proof of knowledge π is unforgeable [55, 75], which concludes the proof.

3.4.4.3 Camenisch-Lysyanskaya signatures

The signature schemes, proposed by Camenisch and Lysyanskaya [77], are equipped with protocols providing various properties.

One of these protocols allows a signature to be issued on the messages that are not known to the signer, but to which the signer only knows a commitment. Informally, in a protocol for signing a committed value, we have a signer with public key pk, and the corresponding secret key sk, and a user who queries the signer for a signature. The common input to the protocol is a commitment C on secret values $(m_1, m_2, \ldots, m_l)$ known only by the user. At the end of this protocol, the user obtains a valid CL-signature $= \mathsf{Sign}(m_1, m_2, \ldots, m_l)$ and the signer learns nothing about $(m_1, m_2, \ldots, m_l)$.

`Setup`
The parameters are:

- $\mathbb{G}$ and $\mathtt{G}$ are cyclic groups of prime order p.
- g is a generator of $\mathbb{G}$ and $\mathtt{g}$ is a generator of $\mathtt{G}$.
- e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathtt{G}$.
- $x \in \mathbb{Z}_p$.
- $y \in \mathbb{Z}_p$.
- $z_i \in \mathbb{Z}_p$ where $1 \leqslant i \leqslant l$.
- $X = g^x$.
- $Y = g^y$.
- $Z = g^{z_i}$ where $1 \leqslant i \leqslant l$ where l is the number of user's secret values.
- $W_i = Y^{z_i}$ where $1 \leqslant i \leqslant l$. The values W_i will be used to prove the knowledge of the signature.

The private key is $sk = (x, y, \{z_i\})$ where $1 \leqslant i \leqslant l$ and the public key is $pk = (p,\ \mathbb{G},\ \mathtt{G},\ g,\ \mathtt{g},\ e,\ X,\ Y,\ \{Z_i\},\ \{W_i\})$ where $1 \leqslant i \leqslant l$.

`Sign`
The signature protocol is an interactive protocol between a user $\mathcal{U}$ who has on input $(m_0, ..., m_l)$ and a signer $\mathcal{S}$ who has on input the private key $sk = (x, y, \{z_i\})$.

1. $\mathcal{U}$ computes a commitment M on the messages $(m_0, ..., m_l)$

$$M = g^{m_0} \Pi_{i=1}^{l} Z_i^{m_i}.$$

2. $\mathcal{U}$ sends a zero-knowledge proof of knowledge of the opening of the commitment $POK((\mu_0, ..., \mu_l) : M = g^{\mu_0} \Pi_{i=1}^{l} Z_i^{\mu_i})$.
3. $\mathcal{S}$ computes the signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ where
   - $\alpha \in \mathbb{Z}_p$.
   - $a = g^{\alpha}$.
   - $A_i = a^{z_i}$ such that $1 \leqslant i \leqslant l$.
   - $b = a^y$.
   - $B_i = A_i^y$ such that $1 \leqslant i \leqslant l$.
   - $c = a^x M^{\alpha xy}$.
4. $\mathcal{S}$ sends the signature σ

`Verify`
To verify a signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ on the messages $(m_0, ..., m_l)$, the user checks that

- $\{A_i\}$ were formed correctly: $e(a, Z_i) = e(g, A_i)$.

- b and $\{B_i\}$ were formed correctly: $e(a, Y) = e(g, b)$ and $e(A_i, Y) = e(g, B_i)$ such that $1 \leqslant i \leqslant l$.
- c is formed correctly: $e(X, a).e(X, b)^{m_0}.\Pi_{i=1}^{l} e(X, B_i)^{m_i} = e(g, c)$.

This protocol is secure under the LRSW assumption.

Another protocol allows to prove knowledge of a signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ on a tuple of messages $(m_1, m_2, \ldots, m_l)$ without releasing any information on the corresponding signature. Each message can either be revealed to the verifier, sent in a committed form or he might have no information about it. The prover

1. Computes and sends a blinded version of his signature σ: Choose random r, $r_0 \in \mathbb{Z}_p$, and blind the signature to form $\tilde{\sigma} = (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \hat{c}, 1 \leqslant i \leqslant l)$ such that
   - $\tilde{a} = a^r$
   - $\tilde{b} = b^{r'}$
   - $\tilde{A}_i = A_i^r$ where $1 \leqslant i \leqslant l$.
   - $\tilde{B}_i = B_i^{r'}$ where $1 \leqslant i \leqslant l$.
   - $\hat{c} = c^{rr'}$
2. let v_x, v_{xy}, v_s and $V_{(xy,i)}$ such that
   $$v_x = e(X, \tilde{a}),\ v_{xy} = e(X, \tilde{b}),\ v_s = e(g, \hat{c}) \text{ and } V_{(xy,i)} = e(X, \tilde{B}_i)$$
3. The prover and verifier compute these values (locally) and then carry out the following zero-knowledge proof protocol:
   $$POK((\mu_0, ..., \mu_l), \rho : (v_s)^{\rho} = v_x (v_{xy})^{\mu_0} \textstyle\prod_{i=1}^{l})(V_{(xy,i)})^{\mu_i}$$
4. The verifier accepts the proof above if:
   - $\{\tilde{A}_i\}$ were formed correctly: $e(\tilde{a}, Z_i) = e(g, \tilde{A}_i)$.
   - $\tilde{b}$ and $\{\tilde{B}_i\}$ were formed correctly: $e(\tilde{a}, Y) = e(g, \tilde{b})$ and $e(\tilde{a}, Y) = e(g, \tilde{B}_i)$ such that $1 \leqslant i \leqslant l$.
   - the proof is correct.

Another appealing feature of some of these CL-signature schemes is that the signatures, they produce, can be randomized. Indeed, given a valid CL-signature σ on a message m, anyone can compute σ', another valid signature on m. Moreover, σ and σ' are unlinkable under the Decision Diffie-Hellman (DDH) assumption (*i.e.*, given σ and σ' (but not m), it is hard to decide whether these two signatures have been issued on the same message).

Finally, CL-signatures have been widely used to design anonymous credentials or DAA-like schemes.

3.4.5 Set-Membership Proof

A set-membership proof allows a prover to prove, in a zero-knowledge way, that his secret lies in a given public set. Such proofs can be used, for instance, in the context of electronic voting, where the voter needs to prove that his secret vote belongs to the set of all possible candidates.

Recent propositions of set-membership proofs [78, 79] follow the same idea:

- Let Φ be a public set.
- Let CA be a designated authority that
  - owns a pair of signature keys (sk_{CA}, pk_{CA}),
  - produces public signatures $\sigma_i = SIGN_{sk_{CA}}(i)$ on each element i of the public set Φ (and only on these elements).
- Let Σ be the public set of the signatures σ_i.
- The proof of knowledge of a secret $i \in \Phi$ consists in proving the knowledge of a (public) signature on the secret i (which will be only possible if i belongs to the interval Φ) without revealing i or the used signature.
$$POK(i, r, \sigma_i : C = g^i h^r \wedge \sigma_i = SIGN_{sk_{CA}}(i))$$

Solutions in [78, 79] require the prover to perform pairing computations. However, these cryptographic tools are not often convenient for many constrained platforms like Secure Elements, SIM cards, etc. Indeed, these computations require many resources, hence, much time.

3.5 Conclusion

In this chapter we began by presenting some mathematical and cryptographic tools that we need in this thesis. Then, we specified the computational assumptions on which relies the security of our constructions. Finally, we described some cryptographic schemes like group signature schemes that are usually used in e-poll and e-ticketing systems in order to ensure users' privacy, namely anonymity and unlinkability, when using the service. In such use cases, usually the user has a resource constrained device, however, privacy-enhancing cryptographic tools are usually costly. In the next chapters, we introduce optimizations and adaptations of these cryptographic tools in order to find the right balance between privacy and efficiency.

At the end of this chapter, we described set-membership proofs. Current implementations are very costly as it requires pairing computations at the prover side (user's device). In the next chapter, we introduce a practical set-membership proof that does not require pairing computations at the prover side. This implementation will be used later in our m-ticketing protocol.

Chapter 4

A New Practical Set-Membership Proof

Contents

Dans ce chapitre, nous présentons une partie des résultats [80] qui ont été publiés dans la revue "Proceedings on Privacy Enhancing Technologies (PoPETs)" de la conférence PETS 2015. Nous introduisons une nouvelle preuve d'appartenance d'un élément à un ensemble. Cette preuve, contrairement aux constructions antérieures décrites dans le chapitre précédent, ne nécessite pas de calculs de couplage du côté du prouveur et dans certains cas du côté du vérifieur également. Cette nouvelle preuve nous sera utile dans la construction d'un protocole efficace et respectant la vie privée dans le Chapitre 7.

4.1 Introduction

In this chapter, we present a new set-membership proof protocol. Our set-membership proof is a zero knowledge proof of knowledge which allows a prover to convince any verifier that a committed integer k lies in a public discrete set Φ. This zero knowledge proof of knowledge can be made non-interactive using the Fiat-Shamir heuristic and the random oracle model [45]. The commitment scheme that we use is the one proposed by Pedersen [70].

Our set-membership proof bears some similarities with the protocol proposed by Camenisch et al. [81]. Yet, our set-membership proof does not require the prover (a SIM card and/or a smartphone in our setting) nor the verifier (in a specific scenario) to perform pairing computations which are quite costly, in the order of seconds, e.g. for the following microprocessors: 17.9s on an ATmega [82], 2.5s on a Philips HiPerSmart$^{\text{TM}}$MIPS [83], 1.9s on an MSP430X [84]. Our set-membership proof is also more efficient and conceptually simpler than the recent one proposed by Canard et al. at EuroPKI 2013 [79]. Their scheme which involves several verifiers, who share a decryption key, seems to be

tailored to specific applications like e-voting where the verifiers (tallying authorities) own such a key to open ballots. As regards our set-membership proof, we will use it in the construction of our m-ticketing protocol (Chapter 7) in order to efficiently prove that a ticket index belongs to the public set of indexes.

This chapter is organized as follows. In Section 4.2, we detail the new set-membership proof. Then, we prove the security of our implementation in Section 4.3. We draw the conclusion of this chapter in Section 4.4.

4.2 Set-Membership Proof Protocol

The public parameters of our set-membership proof are

- $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$: three (multiplicative) bilinear groups of prime order p.
- e: a bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.
- g, g_1 and h_T: three generators of $\mathbb{G}_1$.
- g_3: a generator of $\mathbb{G}_2$.
- H: a hash function, for instance, SHA-256.

Our protocol considers a designated authority, a verifier and a prover. The designated authority (which may or may not be the verifier) randomly chooses a value $y \in \mathbb{Z}_p^*$ (its private key) and publishes the corresponding public key $Y = g_3^y$. After generating its key, this designated authority can issue BB-signatures on each element of the public set Φ. Let k denote an element of the set Φ and A_k the BB-signature on the value k, i.e., $A_k = g^{1/(y+k)}$. The set of all message-signature pairs (k, A_k), that we denote by $\sum$, is published by the designated authority.

$$\Phi = \{k\}, \sum = \{(k, A_k)\}_{k\in\Phi} = \{(k, g^{1/(y+k)})\}_{k\in\Phi}$$

In Figure 4.1, we detail our new protocol that enables to prove that a value k committed by the prover in the commitment Com belongs to the set Φ. This comes down to proving knowledge of the BB-signature A_k.

Remark 1: Proving knowledge of a valid BB-signature

$$A_k = g^{1/(y+k)}$$

on a value k committed in

$$Com = g_1^k h_T^\nu,$$

without revealing either k or the signature and without using pairings on the prover's side can be done as follows.

The prover first randomizes A_k by choosing a random value $l \in \mathbb{Z}_p^*$ and computing $B = A_k^l$. Since $A_k = g^{1/(y+k)}$ this implies that:

$$B = A_k^l = (g^l)^{1/(y+k)}$$

and then that

$$B^{y+k} = g^l.$$

If we note $B_1 = B^{-1}$ and $D = B^y$, it implies that:

$$D = B_1^k g^l.$$

As a result, in order to prove that the value k committed in Com belongs to the set Φ, the prover just has to send B and D to the verifier and compute the following zero knowledge proof of knowledge:

$$\Pi = POK(k, \nu, l : Com = g_1^k h_T^\nu \wedge D = B_1^k g^l).$$

Prover		**Verifier**
Public input: public parameters, sets Φ and $\sum$ **Private Input:** k		**Public Input:** public parameters, sets Φ and $\sum$ **Private Input:** y
Pre-computations		
Choose $\nu \in Z_p^*$ **Compute:** $Com = g_1^k h_T^\nu$ **Pick** the valid BB signature corresponding to the element $k : A_k = g^{1/(y+k)}$ **Choose** $l \in Z_p^*$ **Compute(see remark 1):** $B = A_k^l$; $B_1 = B^{-1}$; $D = B_1^k g^l$ **Choose** $k_1, l_1, r_1 \in Z_p^*$ **Compute:** $Com_1 = g_1^{k_1} h_T^{r_1}$; $D_1 = B_1^{k_1} g^{l_1}$		
Real time computations		
Compute: $c = H(Com, B, D, Com_1, D_1, ch)$; $s_1 = k_1 + c \times k \ mod \ p$; $s_2 = r_1 + c \times v \ mod \ p$ $s_3 = l_1 + c \times l \ mod \ p$;	$\xleftarrow{ch}$	**Choose** $ch \in Z_p^*$ (A random challenge)
$\Pi = Com, B, D, s_1, s_2, s_3$	$\xrightarrow{\Pi}$	**Check** that $B \neq 1_{\mathbb{G}_1}$ • If the verifier and the designated entity are the same entity. This implies that the verifier holds the private signature key y **(First case)**. Hence the prover does not send the value D. The verifier can compute it: $D = B^y$ and goes to (*) • Otherwise, if the verifier doesn't know the private signature key y **(Second case)**, then, it checks that $e(D, g_3) = e(B, Y)$ and goes to (*) **(*) Compute:** $\tilde{C} = g_1^{s_1} h_T^{s_2} Com^{-c}$ and $\tilde{D} = B_1^{s_1} g^{s_3} D^{-c}$ **Check** that: $c = H(Com, B, D, \tilde{C}, \tilde{D}, ch)$

FIGURE 4.1: A New Efficient Set-Membership Proof

The computation of the zero knowledge proof of knowledge Π is described in Figure 4.1.

Upon receiving Π, the verifier proceeds to the verification. We distinguish two cases.

In the *first* case, the verifier and the designated authority are the same. Thus, the verifier holds the private signature key y. Consequently, the verification of Π occurs without requiring any pairing computations. This implies that our set-membership proof does not require pairing computations for both the prover and the verifier.

In the *second* case, the verifier does not have the private signature key y of the designated authority. Then, the verifier needs to perform some pairing computations in order to verify Π. Nevertheless, the prover still has no pairing computations to perform. This is particularly interesting for the m-ticketing use case.

4.3 Security Analysis

Theorem 2. If the $|\Phi|-$SDH assumption holds, then the protocol in Figure 4.1 is a zero-knowledge argument of set-membership for a set Φ.

Proof 2 (sketch).
The **completeness** of the protocol follows by inspection.

The **soundness** follows from the extraction property of the underlying proof of knowledge and the unforgeability of the underlying signature scheme. In particular, the extraction property implies that for any prover P^* that convinces V with probability ϵ, there exists an extractor which interacts with P^* and outputs a witness (k, v, l) with probability poly(ϵ). Moreover, if we assume that the extractor input consists of two transcripts:

$$\{Y, \sum, Com, B, D, c, \tilde{c}, s_1, \tilde{s}_1, s_2, \tilde{s}_2, s_3, \tilde{s}_3\}$$

the witness can be obtained by computing (all the computations are done mod p):

$$k = (s_1 - \tilde{s}_1)/(c - \tilde{c}),$$
$$\nu = (s_2 - \tilde{s}_2)/(c - \tilde{c}),$$
$$l = (s_3 - \tilde{s}_3)/(c - \tilde{c})$$

The extractor succeeds when $(c - \tilde{c})$ is invertible in $\mathbb{Z}_p$. If the following check

$$D = B^y$$

holds , this implies that:

$$D = B^y = B_1^k g^l$$

and then

$$B^y B^k = g^l.$$

Let us denote by $A_k = B^{1/l}$. Note that we necessarily have $l \neq 0$, otherwise this would imply that the prover knows the secret value y (which would be equal to $-k \ mod \ p$). We therefore have

$$A_k^{y+k} = g$$

and consequently that

$$A_k = g^{1/(y+k)}$$

So the prover knows a valid BB signature on the value k. This implies that $k \in \Phi$.

Also note that if $D = B^y$ then this implies that the Prover P only knows one representation of D in the base B_1 and g. Otherwise this would imply that P knows the private key y. Indeed, suppose that P knows two representations of D in the base B_1 and g. Let us denote by (k, l) and $(\tilde{k}, \tilde{l})$ these two representations. Since $\mathbb{G}_1$ is a prime order group and $B_1 \neq 1$ and $g \neq 1$, this implies that $k \neq \tilde{k}$ and $l \neq \tilde{l}$. Since

$$D = B_1^k g^l = B_1^{\tilde{k}} g^{\tilde{l}},$$

this implies that

$$B_1^{k-\tilde{k}} = g^{l-\tilde{l}}$$

and that

$$B_1^{(k-\tilde{k})/(l-\tilde{l})} = g.$$

Let us denote by

$$\tilde{y} = (k - \tilde{k})/(l - \tilde{l}).$$

then

$$D = B^y = B_1^k g^l = B^{-k} B^{-(l\tilde{y})}.$$

Since $\mathbb{G}_1$ is a prime order group this implies that

$$y = -k - l\tilde{y}.$$

If $k \notin \Phi$, then P^* can be directly used to mount a weak-chosen attack against the BB signature scheme with probability poly(ϵ) of succeeding. Thus ϵ must be negligible.

Finally, to prove **(honest-verifier) zero-knowledge property**, we construct a simulator *Sim* that will simulate all interactions with any (honest verifier) V^*.

1. *Sim* retrieves Y and $\sum$ from V^*.
2. *Sim* randomly chooses $k \in \Phi$ and $l \in \mathbb{Z}_p^*$ and computes $B = A_k^l$, $B_1 = B^{(-1)}$ and $D = B_1^k g^l$.
3. *Sim* randomly chooses $c, s_1, s_2, s_3 \in \mathbb{Z}_p^*$ and computes $Com_1 = g_1^{s_1} h_T^{s_2} Com^{-c}$ and $D_1 = B_1^{s_1} g^{s_3} C^{-c}$.
4. *Sim* outputs $S = \{Com, B, D, Com_1, D_1, c, s_1, s_2, s_3\}$

Since $\mathbb{G}_1$ is a prime order group, then the blinding is perfect in the first two steps and S and V^*'s view of the protocol are statistically indistinguishable.

4.4 Conclusion

In this chapter, we introduced a new set-membership proof that enables a prover to prove that he owns an element of a public set without revealing it. Unlike the previous constructions, this proof does not require pairing computations at the prover side. However, when the verifier does not know the designated authority private key (the verifier and the designed authority are distinct), the verifier may need to do some pairing computations. We will use our new set-membership proof in the construction of our m-ticketing protocol in Chapter 7. Prior that, we outline the related work to public transport service in the next chapter.

Chapter 5

Mobile Transport Service: State of The Art

Contents

Dans ce chapitre, nous décrivons l'état de l'art concernant les systèmes de transport public et spécifions les entités d'un système de transport ainsi que l'architecture du téléphone mobile et le modèle de sécurité, utilisés dans les deux chapitres suivants. En ce qui concerne l'état de l'art, nous étudions, dans un premier temps, les protocoles proposés par les standards et les solutions commerciales. Ensuite, nous nous intéressons aux solutions de la littérature qui ont proposé des protocoles assurant certaines propriétés liées au respect de la vie privée des utilisateurs. En particulier, nous décrivons une vulnérabilité, que nous avons découverte, du protocole proposé par Rupp et al. [2] qui a été publié à FC'13. Concernant les entités de notre système, nous considérons l'utilisateur, l'autorité de transport et une nouvelle entité qu'on appelle "autorité de révocation". Cette dernière est complètement indépendante de l'autorité de transport et est capable de révoquer l'anonymat des utilisateurs et des titres de transport. Quant aux phases d'un système de transport, nous en distinguons quatre: initialisation du système, enregistrement d'un utilisateur, demande d'un titre de transport et validation de celui-ci. Nous précisons également que l'application transport sera implémentée sur une carte SIM. Enfin, nous définissons, d'une manière informelle, le modèle de sécurité associé à un système de transport. Ce modèle sera formalisé pour les deux cas d'usage dans les chapitres suivants.

5.1 Introduction

In this Chapter, we detail related work on privacy-preserving solutions for public transport service. We begin by describing the CALYPSO standard in Section 5.2. In Section 5.3, we present some commercial mobile contactless solutions for transport service while showing to which extent they guarantee the users' privacy. We review proposed privacy-preserving transport systems in the literature in Section 5.4. These studies have helped us to identify the important security and privacy properties required to build privacy-preserving mobile solutions for transport system. We give further details about the Rupp et al. [2] attack that we discovered in Section 5.5. In Section 5.6 and Section 5.7, we give an overview of our transport system. We present the architecture overview and the entities of the system as well as the trust model. We also recall the main phases of a transport system. These phases will be personalized according to the use case, i.e., m-pass or m-ticketing use case. Section 5.8 concludes the chapter.

5.2 Standardization

The CALYPSO standard [85, 86] (ISO 24014-1:2007 norm) introduced the contactless aspects across a European consortium composed of transport operators such as the Belgium STIB and the French RATP. CALYPSO specifies the details of all the transactions related to e-ticketing for contactless transport services from the purchase of the ticket/pass to their uses. This standard is very precise, in particular when describing operations such as the card authentication, the validator authentication and the messages exchanged between these two entities. Moreover, performance issues such as the computational time are also included in this standard, particularly for contactless smartcards.

5.3 Commercial Transport Solutions

Many commercial mobile transport solutions have recently emerged. For instance, the service Touch and Travel (T&T) [87] has been proposed by the German federal railways Deutsche Bahn since 2008 and the mobile contactless public transport service based on the Octopus card [88, 89] has been proposed by PCCW-HKT, one of Hong Kong's major mobile network operators and Octopus Cards Limited (OCL) since 2013.

The T&T service is one of the first contactless mobile ticketing services. A user must start the T&T application upon departure. Once he reaches his destination, the user must indicate the end of his trip. Designating departure and arrival stations can be performed by using a touchpoint if one is available at the station, relying on the GPS as captured by his smartphone or based on the network cell location.

Regarding the Octopus card, it is a popular contactless smart card used for public transport and retail payment in Hong Kong. Such cards, which can be seen as electronic purses, have unique identifier for all the user's transactions [90].

The solutions mentioned above do not satisfy all the privacy requirements. they are limited to only protect the users' privacy against outsiders and not against the transport

authority. Indeed, they suffer from the linkability issue, as it enables the transport authority to track users.

5.4 Mobile Privacy-Preserving Solutions for Public Transport

In this section, we give an overview of solutions for NFC transport services in the literature. Most of them do not address the anonymity and unlinkability issues, and the few who do either are not yet efficient enough to cope with the constraint of running in less than 300 ms, or present privacy weaknesses.

In the follow-up papers [91, 92], Tamrakar and Ekberg proposed a contactless m-ticketing transport service. The main idea of this service is that a user, who has already subscribed to the service, can buy some credits and then purchases tickets. The ticketing application is authorized to locally generate a ticket before then charging the user account. Like for T&T service, the user has to specify the departure and arrival stations. Depending on the trust assumptions that are made, the authors propose two architectures relying or not on a trusted execution environment [26]. While this solution complies with the main functional requirements of public transport service, a user's anonymity and the unlikability of his trips are not ensured. Indeed, the user personal data, such as his identifier and location, are protected against outsiders but revealed to the service provider.

Heydt-Benjamin et al. were the first to propose a cryptographic framework for transport service [93]. The authors' approach uses e-cash, anonymous credentials, and proxy re-encryption [40]. They discuss the challenges that a privacy-preserving transit system should solve. Using cryptographic transit tickets should not disable basic capabilities like cloning detection, virtual ticket backups or transfers, ticket revocation. The efficiency of the use of a virtual ticket, especially over contactless technologies, is also important.

Derler et al., authors of [94], have analyzed the efficiency of a selective disclosure protocol in the latest smartphones. In a nutshell, a selective disclosure protocol allows a user to reveal that he possesses some property linked to his identity, but without revealing his identity itself or any unnecessary information. With respect to the ticketing use case, they used a one time-show protocol, which is the Brands DSA-based protocol [95, 96]. Thanks to this protocol, during the product validation phase, the user is able to prove the validity of his credentials without disclosing them. Even though this system provides anonymous tickets, the m-tickets, hence the users' trips, remain linkable. Moreover, the ability to recover the identity of a user (by an independent entities such as a judge) is not provided. In the other hand, Derler et al. implementation introduces an important delay ($2 - 15s$) even if it is shared between a mobile (Nokia 6131) and a secure element (an embedded G&D smart card). This is because their group signature implementation uses pairing based cryptography.

In [97], Isern-Deya and his co-authors proposed an automatic fare collection system for temporal and spatial services, which is based on the BBS group signature [54, 55]. Thus, they provided revocable anonymity and unlinkability for the users of the service. The main drawback of this solution is that although it has been implemented on a smartphone, the waiting time at the entrance and exit of the transport network is still prohibitive, i.e., in the order of seconds.

More recently, Rupp et al. [2] proposed at FC 2013 a privacy preserving pre-payment protocol with refund for public transport. The main goal of Rupp et al. is to minimize the number of coins a user needs to pay for a trip. This would minimize the number of computationally expensive spending transactions. To this end, the user withdraws a single coin be worth the most expensive ride in the coin credential. Then, the user will be refunded. The verification procedure (of the refund) is too costly to handle on constrained devices. The lack of verification could lead to serious privacy weaknesses: a malicious transport authority could link users' trips. We detail the Rupp et al. protocol and its privacy weakness in the following section in order to illustrate the difficulty to build a protocol ensuring a high level of privacy.

In Table 5.1, we classify the analyzed proposals depending on:

1. Anonymity: the ability to identify users during the use of the transport service by outsiders or by the transport authority,
2. Unlinkability: the ability to track a user either over one journey (entrance and exit of the journey) or over different journeys, by outsiders or the transport authority,
3. Hardware: the hardware that has been used for the implementation, if the proposal has been implemented, and
4. Efficiency: a proposal is efficient if the validation time is less than $300ms$.

Ref.	Anonymity	Unlinkability	Hardware	Efficiency
Tamrakar et al. [91, 92]	×	×	Mobile / TEE	√
Heydt-Benjamin et al. [93]	√	√	×	×
Derler et al. [94]	√	×	Mobile and Secure Element	×
Isern-Deya et al. [97]	√	√	Mobile	×
Rupp et al. [2]	√	×	Secure Element	×

TABLE 5.1: Comparison of the analyzed proposals

To summarize, balancing users' privacy with efficiency and security needs of the transport authority is not an easy task. In the next section, we give further details about the privacy weakness of Rupp et al. protocol, published in FC 2013, that we discovered.

5.5 The Privacy Weakness of Rupp et al. Solution

In this section, we give further details about the payment protocol for public transport proposed by Rupp et al. at FC 2013 [2] and called P4R. Then, we present an attack that enables a malicious transport authority to break users' privacy.

At first, the user buys a trip authorization token (TAT_i) which consists of an (extended) coin in Brands' scheme [95, 96]. This token enables the user to perform one trip. Then, the user receives a fresh refund token RT.

$$RT = SN_{RT} \text{ such that } SN_{RT} \leftarrow G$$

The user sets a variable R to 1 and v to 0. R will be used to aggregate blinding factors and v is the refund amount. At the entry turnstile, the user shows a TAT_i and proves its ownership. Right after this, the user receives a refund calculation token (RCT_i) which

consists of a MAC on TAT_i, a timestamp and the reader ID. At the turnstile of the exit, the user shows his RCT and proves the ownership of the TAT_i contained within it. In addition, he collects refund on his RT, as described in Figure 5.1. A refund value ω is represented as a ω-times BLS signature [98] $SN_{RT}^{d^\omega}$ such that d is the secret BLS signature key of the transport authority. Later, the user redeems his RT. This consists on sending a blinded version of his RT ($RT' = RT^r$), SN_{RT}, R and the collected amount υ to the transport authority. The latter checks the validity of both SN_{RT} and the signature,

$$e(SN_{RT}^R, h^{d^\upsilon}) \stackrel{?}{=} e(RT', h) \textbf{ (2)}.$$

Collecting Refund		
User		**Exit turnstile**
$r \leftarrow \mathbb{Z}_p^*$	$\xleftarrow{\omega}$	Determine the refund ω based on the received information in RCT
$RT' = RT^r$	$\xrightarrow{RT'}$	
$\upsilon = \upsilon + \omega$, $R = Rr \ mod \ p$, $RT = RT''$	$\xleftarrow{RT''}$	$RT'' = RT'^{d^\omega}$ **(1)**
Redeeming Refund		
User		**Transport authority**
$r \leftarrow \mathbb{Z}_p^*$, $RT' = RT^r$, $R = Rr \ mod \ p$	$\xrightarrow{SN_{RT}, RT', \upsilon, R}$	check validity of SN_{RT}, $\upsilon < p-1$, $e(SN_{RT}^R, h^{d^\upsilon}) \stackrel{?}{=} e(RT', h)$ **(2)**

FIGURE 5.1: The refund protocols of Rupp et al. solution [2]

In [2], Rupp et al. assume that, for efficiency reasons, users don't verify their RTs. Indeed, verifications of refund tokens (step 1 in Figure 5.1) is too costly and can not be handled by constrained devices such as SIM cards. This lack of verification could lead to serious privacy weaknesses: a malicious transport authority could link users' trips.

More precisely, for a given station, the transport authority is able to identify all the users who left at this station. To this end, in step (1) in Figure 5.1, instead of computing

$$RT'' = RT'^{d^\omega},$$

the transport authority chooses a variable t,

$$t \leftarrow \mathbb{Z}_p^*$$

and computes

$$RT'' = RT'^{\mathbf{t}d^\omega}.$$

The user will not detect that RT'' is improperly calculated because there is no verification on the user's side. The refund token RT will then carry on the exponent t till the redeeming refund step, i.e., $RT' = SN_{RT}^{\mathbf{t^j}Rd^\upsilon}$ where j corresponds to the number of times the user left the targeted station, R is the product of all the variables r and υ is the sum of all the refunds ω. At the redeeming refund step, upon receiving the serial number SN_{RT}, the blinded version of the refund token RT', the refund amount υ, and the aggregate blinding factor R, the transport authority will as usual check the validity of SN_{RT} and whether the amount is within the allowed range $[0, p-1]$. Now, in order to verify the signature and distinguish users who exited at the relevant station, the transport authority starts by checking the relation (2). If (2) holds, this implies that the user did not left the targeted station. Otherwise, the transport authority checks:

$$e(SN_{RT}^{\mathbf{t^j}R}, h^{d^\upsilon}) = e(SN_{RT}, h)^{\mathbf{t^j}Rd^\upsilon} = e(SN_{RT}^{\mathbf{t^j}Rd^\upsilon}, h) \quad = e(RT', h) \textbf{ (3)}$$

To do so, the transport authority will repeat the computations with different values of j until the relation (3) holds. When

$$e(SN_{RT}^{t^j R}, h^{d^v}) \stackrel{?}{=} e(RT', h) \textbf{ (3)}$$

holds, this implies that the user left the targeted station j times.

In order to monitor several stations at the same time, the transport authority will have a list of variables t, e.g., t_A for station A, t_B for station B, etc. At the redeeming refund step, the transport authority will test different values of t until it finds the ones that satisfy the relation

$$e(SN_{RT}^{\prod t_x^{j_x} R}, h^{d^v}) \stackrel{?}{=} e(RT', h) \textbf{ (4)}$$

where t_x characterizes a station x and j_x represents the number of exits at the station x. For instance, if the user left the station A four times, the station B twice and the station M once, relation (4) will be as follows:

$$e(SN_{RT}^{t_A^4 t_B^2 t_M^1 R}, h^{d^v}) \stackrel{?}{=} e(RT', h).$$

In this way, the transport authority will be able to identify the users that left a pool of targeted stations but also the number of times they exited at these stations: this clearly implies that users' trips are *linkable*.

However, solving relation (4) and finding the right tuples (t_x, j_x), becomes quickly a complex and unmanageable task when the number of targeted stations increased (more than three for example). To mitigate this, a malicious transport authority could only monitor two stations at the same time and periodically change these monitored stations. In this way, the transport authority will be able to monitor a large number of stations and will have a global overview of the users' journeys which breaks the privacy property of Rupp et al. system. We would like to emphasize that privacy holds in P4R if we assume that the transport authority is "honest but curious", i.e., it will not deviate from the refund protocol, but such an assumption is too strong in the context of transport systems [99].

In order to mitigate this issue, the user should check the received refund (Step (1) in Figure 5.1). This verification implies pairing computations. However, in constraint environments such as the SIM card, this is not feasible yet. A straightforward solution would be to delegate these computations to the smartphone. Such a solution, however, has two main drawbacks. On the one hand, delegating computations to an untrusted environment, i.e., the smartphone which could be compromised by spywares, will affect the user's privacy. On the other hand, verifying the received refund token would lead to inefficient transactions at exit turnstiles.

This attack shows how difficult is to construct an efficient protocol (validation time less than 300 ms) over a resource constrained device like the SIM card while ensuring the highest privacy level.

5.6 Architecture

In this section, we give the different entities of our public transport system. These entities are the same for both the m-pass and m-ticketing use cases. Then, we detail our architecture. Finally, we recall the main phases in a public transport service.

In our public transport system, i.e., m-pass or m-ticketing system, we consider three different entities. The user (U) is the owner of an NFC-enabled smartphone and wants to use the transport service via a transport application. The transport authority (TA) is the manager of the transport service. Besides these two entities, we consider a third actor which is a revocation authority (RA). It can revoke the transport products, i.e., m-pass or m-ticket, and users' anonymity, and is completely independent of the transport authority. This role may be split between several authorities in such a way that they should all agree and cooperate before recovering a user's identity.

At the user's side, as shown in Figure 5.2, we consider an untrusted mobile platform (smartphone) with an incorporated trusted secure element. We use a SIM card as a tamper resistant secure element since it could be certified EAL4+ [24]. Using a SIM card will ensure the integrity of the implementation. Indeed, the transport cardlet manages the cryptographic credentials of the transport application and executes required cryptographic computations.

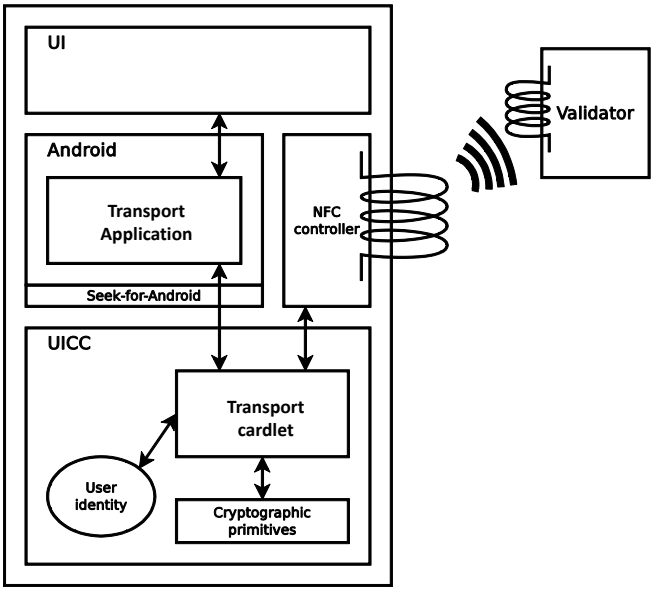


FIGURE 5.2: Architecture Overview

The main phases of a public transport service, i.e., m-pass service and m-ticketing service, are *system initialization*, *user registration*, *product request* and *product validation*.

- *System initialization*: during this phase, the transport authority initializes the public parameters and keys of the transport system.
- *User registration*: during this phase, the user will register himself to transport service.
- *Product request*: during this phase, the user will get his transport product from the transport authority, e.g., a monthly m-pass, a set of 10 m-tickets. if the transport system is functioning in a pre-payment way, the user must pay his product during this phase. Otherwise, the transport service operates in a post-payment way. In this case, the user will be charged later.

- *Product validation*: during this phase, the user has to interact with the validator in order to prove the authenticity of his transport product.

5.7 Trust Model

In this section, we informally describe the attacker model in a public transport service. We give more details in Chapter 6 with respect to the mobile pass use case and in Chapter 8 for the m-ticketing use case.

We assume that the attacker has full control over communication channels. He has the ability to compromise any user's smartphone, i.e., mobile operating system. Moreover, the attacker can compromise one or more users. In such a case, he can read / modify keys stored on or read / modify programs running within the SIM card of the corrupted user. However, we assume that he is not able to read / modify keys stored on or read / modify programs execution that take place within the SIM card of an uncorrupted user. We also suppose that the attacker can compromise the transport authority but not the revocation authorities.

5.8 Conclusion

In this chapter, we presented the related work to transport system and showed, through the privacy attack against the Rupp et al. protocol, that it is not obvious to find the right balance between the user's privacy and the solution's efficiency. We also described our mobile platform architecture and recall the transport service phases. We informally defined the attacker model as well. This model will be formally and finely specified in Chapter 6 for the m-pass use case and in Chapter 7 for the m-ticketing use case.

In the next chapter, we introduce our m-pass solution together with its formal security analysis and performance results.

Chapter 6

A Practical and Privacy-Preserving Mobile Pass

Contents

Dans ce chapitre, nous présentons les résultats [100] qui ont été publié dans la revue "ICST Trans. Mobile Communications Applications". Nous proposons un service mobile sans contact respectant la vie privée des utilisateurs: l'identité d'un utilisateur ne peut pas être liée à ses actions lors de l'utilisation du système de transport. Notre protocole est basé sur une variante de schéma de signature de groupe: les signatures Camenisch-Lysyanskaya. Même si un utilisateur est censé rester anonyme et ses actions intraçables, nous avons conçu une technique pour lever son anonymat dans des circonstances exceptionnelles. Nous avons également implémenté un prototype qui démontre que notre solution répond aux principales exigences fonctionnelles des systèmes de transport: à savoir que la validation d'un titre de transport doit s'effectuer en moins de 300 ms, et ce, même si la batterie du smartphone est épuisée. En outre, nous avons formellement validé la sécurité de notre protocole dans le modèle de l'oracle aléatoire.

6.1 Introduction

In this chapter, we propose a privacy-preserving contactless mobile pass solution, in which a user's identity cannot be linked to his trips when using the transport system. In particular, an eavesdropper and the transport provider are not able to track the journeys of the user. To achieve this objective, we have developed a privacy-preserving protocol for a contactless mobile pass, whose security is based on the combination of a secure element in a smartphone and on a privacy-enhancing cryptographic protocol. Even if a user should remain anonymous and his actions unlinkable in his daily journeys, we have designed a technique for lifting his anonymity in extreme circumstances. This exceptional procedure can be called for instance under the order of a judge, in case of a fraud or a murder.

This chapter is organized as follows. At first, we detail the framework of our m-pass system in Section 6.2 and our protocol in Section 6.3. Then, we introduce our functional requirements and formalize the security and privacy requirements in Section 6.4. We show that our protocol meets the requirements, that we set previously, in Section 6.5. Finally, we give details about the performance results of our solutions in Section 6.6 and conclude this chapter in Section 6.7.

6.2 Framework of an M-Pass System

We model our m-pass system onto four phases. Besides the phases described in Section 5.6, we introduce a new phase called "revocation" phase.

First of all, the m-pass system parameters and keys are initialized during the *System initialization* phase. Then, the user registers to the transport service (user registration) and retrieves his m-pass credentials (product request) during the *Registration* phase. In the *validation* phase, the user proves his legitimacy to use the transport service without disclosing his identity. Finally, the *revocation* phase consists in revoking a user (i.e., retrieving a user's identity based on an anonymous m-pass) and revoking an m-pass (i.e., denying access to the transport service based on a user's identity).

Next, we model these phases with various algorithms and protocols (Figure 6.1) executed by the entities described in Section 5.6. In the m-pass protocol, we rename the revocation authorities by *Opening authorities* (OA).

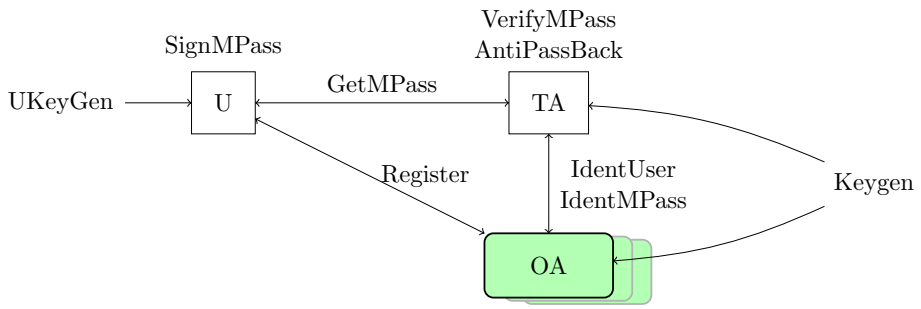


FIGURE 6.1: M-Pass Framework Overview

System initialization

`Setup`(1^λ): This probabilistic algorithm outputs pp, a description of the system parameters. We assume that pp are implicit to the other algorithms, and that they include λ, the security parameter. They are also an implicit input to the adversary, we will then omit them.

`Keygen`(pp): This probabilistic algorithm outputs the two following secret/public key pairs: (sk_o, pk_o) for the opening authorities and (sk_t, pk_t) for the transport authority. The system public key gpk is eventually set as (pp, pk_t, pk_o).

Registration

`UKeygen`(gpk, ID_U): This probabilistic algorithm outputs a secret key sk_u for a user's identifier ID_U.

`Register`(U(gpk, sk_u, ID_U), OA(gpk, sk_o, DB_o)): This is an interactive protocol between a new user with his identifier ID_U and his private key sk_u, and the opening authorities that take on input its private key sk_o and its secured database DB_o where the user will be registered. If the user accepts the protocol, he gets a registration proof μ. If the opening authorities accept the protocol, the user's identity, i.e., his identifier and a commitment of his private key, will be stored within the database DB_o.

`GetMPass`(U(gpk, sk_u, μ), TA(gpk, sk_t, DB_t)): This is an interactive protocol between a registered user with his private key sk_u and his registration proof μ, and the transport authority with its private key sk_t and DB_t. If the transport authority accepts the protocol, its output is a CL-certificate (A, B, C, D) and a commitment C_1 of the user's secret key sk_u. C_1 and the user identifier are stored within the database DB_t.

Validation

`SignMPass`(gpk, sk_u, bsn, rc): This is a probabilistic algorithm that takes as input a registered user private key sk_u, a time slot bsn and a random challenge rc. It outputs a signature σ.

`VerifyMPass`(gpk, σ, bsn, rc): This is a deterministic algorithm that outputs 0 if the signature σ with respect to bsn and rc is valid. Otherwise, it outputs $\perp$.

Revocation

`AntiPassBack`(σ_1, σ_2): This is a deterministic algorithm that takes on input two valid signatures σ_1 and σ_2 (which have been generated during the execution of the algorithm `SignMPass`). If σ_1 and σ_2 have been generated by the same user during the same time slot bsn, it outputs 0. Otherwise, it outputs $\perp$.

`IdentUser`(σ, DB_o, DB_t): This is a deterministic algorithm that takes as input a signature σ (which has been generated during the execution of the algorithm `SignMPass`), the secured database of the opening authorities DB_o and the secured database of the transport authority DB_t. It outputs the identifier ID_U of the registered user who generated the signature σ. If the user who generated the signature σ is not registered, it outputs $\perp$.

`IdentMPass`(C_1, σ, bsn, rc): This is a deterministic algorithm that takes on input C_1 a commitment of sk_u (a user's secret), a signature σ (which has been generated during the

execution of the algorithm `SignMPass`), a time slot bsn, and a message rc. It outputs 0, if σ is a signature on bsn and rc, and corresponds to committed secret on C_1. Otherwise, it outputs $\perp$.

6.3 Untraceable Mobile Pass

The main idea of our solution is to use a DAA scheme in order to enable an enrolled user to be anonymous inside the group of other subscribers of the m-pass service. More precisely, our solution is based on the DAA scheme of Brickell et al. [44], which we adapted in order to handle revocation features. For the sake of simplicity in this section, we define our protocol by taking into account only one group of subscribers (e.g., subscribers for an m-pass valid in zone 1 of Paris.): the group of persons having an m-pass for using the public transport system. We treat the generic case of several groups later.

6.3.1 M-Pass Protocol

System initialization The transport authority considers two hash functions, $\mathcal{H}$ and $\mathcal{H}_1$, modeled as random oracles:

$$\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^k,$$
$$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1.$$

Then, he picks up randomly a generator G_1 (respectively G_2) of $\mathbb{G}_1$ (respectively $\mathbb{G}_2$):

$$G_1 \xleftarrow{\$} \mathbb{G}_1 \ ,$$
$$G_2 \xleftarrow{\$} \mathbb{G}_2 \ .$$

The public parameters of the group are gpp such that:

$$gpp \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{H}, \mathcal{H}_1, e, G_1, G_2).$$

(recall that e is a bilinear map and $\mathbb{G}_T$ is a multiplicative cyclic group of order p, *cf.* Chapter 3). The transport authority generates his public and private keys (pk_t, sk_t) to sign messages such that:

$$sk_t \leftarrow (x, y) \xleftarrow{\$} \mathbb{Z}_p \times \mathbb{Z}_p \ ,$$
$$pk_t \leftarrow (X, Y) \leftarrow ([x]G_2, [y]G_2) \ .$$

Similarly, the opening authorities generate its public and private signature keys (pk_o, sk_o) and its public and private Paillier encryption keys (pk_p, sk_p) such that:

$$pk_p \leftarrow (n, g_p) \ ,$$
$$sk_p \leftarrow (a, b)$$

where a, b, n and g_p are computed as in Section 3.4.3.2.

In consequence, the group public key pk_g is set up such that:

$$pk_g = (gpp, pk_t, pk_o, pk_p)$$

where gpp is the public parameters of the group. The keys pk_t and pk_o are used to verify signatures issued respectively by the transport authority and the opening authorities. The key pk_p is used for encryption.

Regarding the user, his private key sk_u is the sum of two random secret values: s' and s''. s' is randomly chosen by the m-pass cardlet from $\mathbb{Z}_p$. s'' is randomly chosen by the opening authorities from $\mathbb{Z}_p$ then securely sent to the m-pass cardlet. In such a way,

no computation in $\mathbb{G}_2$ will be required to the cardlet (as the cardlet cannot do such computations). Moreover, neither the smartphone nor the Opening authorities have the user's private key. In Figure 6.2, we detail the generation protocol of a user's private key sk_u.

Smartphone		**Smartcard Cardlet**		**Opening authorities**
Public input: pk_g Private Input:		Public input: pk_g Private Input: s'		Public Input: pk_g Private Input: sk_o, s'', DB_o
$r'_2 \xleftarrow{\$} \mathbb{Z}_p^*$, **Compute:** $B'_2 = [r_2']G_2$, $C'_2 = [s']G_2$, $c'_2 = \mathcal{H}(C'_2, B'_2)$, $s'_2 = r'_2 - c'_2\ s' mod\ p$, $\Pi'_2 = SOK(s' : C'_2 = [s']G_2)$ $= (c'_2, s'_2)$	$\xleftarrow{s'}$ $\xrightarrow{C'_2, \Pi'_2}$	$s' \xleftarrow{\$} \mathbb{Z}_p$	$\xrightarrow{C'_2, \Pi'_2}$	**Compute:** $\tilde{B}'_2 = [c'_2]C'_2 + [s']G_2$ **If** $c'_2 \stackrel{?}{=} \mathcal{H}(C'_2, \tilde{B}'_2)$ **then,** $s'' \xleftarrow{\$} \mathbb{Z}_p$ **else abort.** **Compute:** $C''_2 = [s'']G_2$, **then,** $C_2 = C'_2 + C''_2$ $= [s']G_2 + [s'']G_2$ $= [s' + s'']G_2$
		Compute: $sk_u = s' + s''$	$\xleftarrow{C_2, s''}$	$= [sk_u]G_2$

FIGURE 6.2: User's Private Key Generation Protocol

User registration During the registration phase, described in Figure 6.3, the m-pass cardlet establishes two secure channels with the opening and transport authorities.

First, the registration of the user to the opening authorities consists of two steps. **(1)** The m-pass cardlet computes a commitment C_1 on the private key sk_u of the user and a signature of knowledge Π_1 proving the knowledge of the private key sk_u used to compute C_1. Then, it sends C_1, C_2, Π_1 along with B_1 (this value enable the verification of Π_1), to the opening authorities.

The opening authorities verify that these computations are correct. If this verification succeeds, it means that C_1 and C_2 are commitments on the same secret value (sk_u here) known by the user. In this case, the opening authorities send back a signature μ of C_1 and stores the triplet (C_1, C_2, μ) in the database DB_o, which has to remain private and secure.

(2) Then, upon receiving the signature μ, the m-pass cardlet sends C_1, μ, C_0 (the Paillier encryption of the user's private key sk_u) and a signature of knowledge Π'_1 proving that the user knows the secret encrypted in C_0 and used to compute C_1, to the transport authority, which then checks the validity of the signature (using pk_o) and the proof. If the verifications are successful, the transport authority computes and sends a blind CL-certificate [77] (A, B, C, D) on the private key of the user (recall that CL-signature schemes allow a signature to be issued on messages that are not known to the signer, but to which the signer only knows a commitment). We would like to mention that for a given sk_u, a user can have only one m-pass (one blind CL-certificate (A, B, C, D)). If

`Register`: **Registration within the database of the opening authorities**		
Smartcard cardlet		**Opening authorities**
Public input: pk_g, C_2 Private Input: sk_u		Public Input: pk_g Private Input: sk_o, DB_o
$r \xleftarrow{\$} \mathbb{Z}_p^*$ **Compute:** $B_1 = [r]G_1$ $C_1 = [sk_u]G_1$ $c = \mathcal{H}(C_1, B_1)$ $s = r - c\ sk_u mod\ p$ $\Pi_1 = SOK(sk_u : C_1 = [sk_u]G_1)$ $= (c, s)$	$\xrightarrow{C_1, C_2, \Pi_1}$	**Compute:** $\tilde{B}_1 = [c]C_1 + [s]G_1$ **If** $c \stackrel{?}{=} \mathcal{H}(C_1, B_1)$ **and** $e(C_1, G_2) \stackrel{?}{=} e(G_1, C_2)$ **Then Compute:** $\mu = Signature(sk_o, C_1)$
	$\xleftarrow{\mu}$	**Store** (C_1, C_2, μ) **in** DB_o

`GetMPass`: **Registration within the database of the transport authority**		
Smartcard cardlet		**Transport authority**
Public input: pk_g, μ Private Input: sk_u		Public Input: pk_g Private Input: sk_t, DB_t
$j \xleftarrow{\$} \mathbb{Z}_n^*$ **Compute**: $C_0 = g_p^{sk_u} j^n\ mod\ n^2$ $\Pi'_1 = SOK(sk_u : C_1 = [sk_u]G_1 \wedge C_0 = g_p^{sk_u} j^n\ mod\ n^2)$	$\xrightarrow{C_0, C_1, \mu, \Pi'_1}$	**If verify**(pk_o, μ) **and verify** Π'_1 **Then Compute a signature on** C_1: $a \xleftarrow{\$} \mathbb{Z}_p$ $A = [a]G_1$ $B = [y]A$ $C = [x](A + D)$ $D = [a.y]C_1$
	$\xleftarrow{A, B, C, D}$	**Store** $(user, C_1)$ **in** DB_t

FIGURE 6.3: Registration Protocols

the user wants an other m-pass, he should use a new private key sk'_u. At the end, the transport authority saves (ID_{user}, C_1) inside his database DB_t.

The databases DB_o and DB_t will be used only for de-anonymizing or blacklisting of users as detailed later in Section "Countermeasures". Once the registration phase is completed, user's registration data are stored in both the transport and opening authorities databases. He is now allowed to use his m-pass application to access the transport service.

M-pass validation The validation of the m-pass, described in Figure 6.4, is used by the user to prove that he is entitled to use the transport service without disclosing his identity. The process is split into two parts: the first part (called precomputation in Figure 6.4) can be performed in advance (before the actual validation) while the second part (called m-pass validation in Figure 6.4) should be done during the validation itself at the gate.

When the validation at the gate occurs, the validator sends a random challenge rc and a time slot bsn to the m-pass cardlet that will respond with a DAA σ on these

Precomputations: randomization of the CL-Certificate	
Smartcard cardlet	**Gate**
Public input: pk_g Private Input: sk_u	Public Input: pk_g, bsn
$l \xleftarrow{\$} \mathbb{Z}_p$ **Compute:** $R = [l]A$, $S = [l]B$, $T = [l]C$ **and** $W = [l]D$ $k \xleftarrow{\$} \mathbb{Z}_p$ **Compute:** $R_2 = [k]S$	
M-pass validation	
Compute: $J = \mathcal{H}_1(bsn)$ $R_1 = [k]J$ $K = [sk_u]J$ $c = \mathcal{H}(J, K, R, S, T, W, R_1, R_2, rc)$ $s = k + c.sk_u \ modp$	$\xleftarrow{rc,\ bsn}$ rc $\xleftarrow{\$} \{0,1\}^\gamma$
	$\xrightarrow{\sigma=(R,S,T,W,K,c,s)}$ **Compute:** $J' = \mathcal{H}_1(bsn)$ $R_1 = [s]J' - [c]K$, $R_2 = [s]S - [c]W$ **Verify:** $e(R, Y) \stackrel{?}{=} e(S, G_2)$ $e(T, G_2) \stackrel{?}{=} e(R + W, X)$ $c \stackrel{?}{=} \mathcal{H}(J', K, R, S, T, W, R_1, R_2, rc)$ **If the user is blacklisted, then access is denied. Otherwise, access is granted.**

FIGURE 6.4: `SignMPass` / `VerifyMPass` - Validation Protocol

challenges. The bsn message is constant during a fixed period of time (typically 5 minutes) and enables the implementation of the anti-pass back feature described in Section "Countermeasures". During the validation phase, the m-pass cardlet has very few computations to perform, only the ones that involve the secret key sk_u, the basename bsn and the challenge rc. The m-cardlet first computes $J = \mathcal{H}_1(bsn)$ and a commitment K on sk_u with respect to the base J: $K = [sk_u]J$. It then computes a signature of knowledge (c, s) proving that it knows the discrete logarithm of K with respect to the base J. Afterwards, it communicates the DAA signature σ on rc and bsn to the gate. This DAA σ consists of a randomized CL-certificate (R, S, T, W), pre-computed by the m-pass application, along with J, K, c, and s. Roughly speaking, the DAA signature σ is a signature of knowledge (*cf.* Section 3.2.3.3) on rc and bsn, proving that (R, S, T, W) is a CL-certificate on the secret value committed in K and that the m-pass cardlet knows this committed value. Then, the validator verifies that the signature σ is valid and is not blacklisted. If the verification is successful, the validator grants access to the user to the transport network.

Groups A group of users is defined by a group public key pk_g. We suggest to regularly update this key to ensure that it has only a limited validity period. Indeed, the users have to update their CL-certificate every time the group public key is updated. A user, who did not update his certificate accordingly, will not any more be considered as a member of the group and thus will not be able to generate a valid signature during the

validation phase. Moreover, by regularly updating the group public key, we limit the size of the blacklist containing the revoked m-passes.

For the sake of simplicity, we have considered so far a single group of users (i.e., a unique group public key), in order to distinguish a legitimate user from an illegal one. In practice, we have to distinguish different kinds of m-passes and consequently different groups of users. A m-pass is often characterized by its type (e.g., student, senior, disabled) and its fare zones (i.e., holders of such fare zone are entitled to travel on.). Thus, potentially a validator gate may embed several group public keys. For example, in the situation in which three groups of users exist: {student, fare zone 1-2}, {student, fare zone 1-5} and {student, fare zone 2-3}, a validator located in a zone 3 station will accept the users of the groups {student, fare zone 1-5} and {student, fare zone 2-3}. Thus, in order to verify the validity of all types of m-passes, a validator may have to store several group public keys.

If the number of groups is huge, storing their respective group public keys will be quite inefficient. We therefore propose that, during the validation phase, the m-pass cardlet sends to the validator the group it belongs to. In this way, the verification time will be constant even if the number of group public keys is large.

6.3.2 Countermeasures

In order to ensure the security of our privacy-preserving m-pass solution, we provide, in this section, an "anti-passback" feature prohibiting consecutive uses of the same m-pass in the same place and during a given period of time, a "de-anonymization" protocol to retrieve the identity of a user in case of emergency or fraud as well as a revocation solution which allows to revoke m-passes.

Anti-passback The anti-passback feature of a transport service denies the access to a user that has already used the m-pass during the previous minutes or seconds depending on how the system is configured. For example, the entrance gate should deny the access if two users try to enter consecutively using the same transport pass.

During the validation phase, the validator stores the signature σ_1 for a time slot bsn. If the user tries to re-use his m-pass during this time slot, the m-pass cardlet will compute and send a new signature σ_2. The validator is able to detect that σ_1 and σ_2 are computed by the same m-pass cardlet by comparing the two parameters K of the signatures σ_1 and σ_2. If they are the same, this means that the two signatures σ_1 and σ_2 originate from the same m-pass cardlet.

When the gate changes the value of bsn, two DAA signatures cannot be linked any more using K. Thus, renewing bsn frequently is mandatory in order to guarantee that the actions of users cannot be linked by the transport authority.

De-anonymization The de-anonymization procedure, described in Figure 6.5, is an implementation of the algorithm `IdentUser`. It allows the transport authority to identify the m-pass (and consequently the holder of this pass) that has generated a particular signature. For instance, in case of a fraud, the logs contained in a gate can be used to identify the corresponding malicious users. Of course the transport authority cannot

de-anonymize users on his own, which would otherwise harm their privacy. Thus, the revocation of the anonymity of a signature cannot be done without the consent and the cooperation of the opening authorities.

First, the transport authority sends the targeted signature $\sigma = (R, S, T, W, J, K, c, s)$ to the opening authorities. Based on this signature and the secret database DB_o, the opening authorities search for the triplet (C_1, C_2, μ) such that $e(J, C_2)$ matches with $e(K, G_2)$. If the equality holds, this means that C_2 and K are two commitments on the same secret value (*i.e.*, sk_u in this case). Then, it replies with the corresponding C_1. Finally, based on the database DB_t and the received C_1, the transport authority finds out the identity of the user ID_{user}.

Transport authority		**opening authorities**
Public input: pk_g, DB_t, σ Private Input: sk_t		Public Input: pk_g Private Input: sk_o, DB_o
	$\xrightarrow{\sigma}$	**Find** (C_1, C_2, μ) in DB_o **such that:** $e(J, C_2) = e(K, G_2)$
Knowing C_1 **Find** (ID_{user}, C_1) in DB_t	$\xleftarrow{C_1}$	

FIGURE 6.5: `IdentUser` - User De-Anonymization Protocols

Blacklist management The blacklisting procedure, described in Figure 6.6, uses the `IdentMPass` algorithm. It allows the transport authority to revoke an m-pass such that it cannot access anymore to the transport service. For example, a user that has lost his smartphone can ask the transport authority to blacklist his m-pass, or in case of a fraud the de-anonymized m-pass can be blacklisted by the transport authority.

If the transport authority wants to exclude a user, the authority seeks for the relevant commitment, C_1, of the user in the database DB_t and sends it to the opening authorities. Upon receiving C_1, the opening authorities retrieve the corresponding commitment C_2 and computes (for all bsn starting from the time the m-pass has been blacklisted): $e(\mathcal{H}_1(bsn), C_2)$. The number of values $e(\mathcal{H}_1(bsn), C_2)$ computed could appear to be large, but it stays limited as we propose to renew regularly the group public keys to implement the validity duration of all m-passes. Thus, depending on the frequency of changes of bsn and of group public keys, the size of the computed set can vary.

Finally, the set of computed values $e(\mathcal{H}_1(bsn), C_2)$ is sent back to the transport authority that forwards it to all the validators of the transport system. Thus during the validation phase, the validator will receive a signature $\sigma = (R, S, T, W, J, K, c, s)$ from an m-pass. It will check the validity of the signature and that the m-pass is not blacklisted. To this end, the validator will check whether $e(K, G_2)$ matches with one of the computed values $e(\mathcal{H}_1(bsn), C_2)$. If the equality holds, this means that C_2 and K are two commitments on the same secret value (*i.e.*, sk_u in our context), and thus the user is blacklisted and the validator denies access to this user. The validator will then typically send a specific message to the m-pass cardlet to lock it. Clearly, this locking message has to be authenticated to thwart denial-of-service attacks.

Transport authority		**Opening authorities**
Public input: pk_g, DB_t, ID_{user} Private Input: sk_t		Public Input: pk_g Private Input: sk_o, DB_o
Knowing ID_{user} **Find** (ID_{user}, C_1) in DB_t	$\xrightarrow{C_1}$	**Knowing** C_1 **Find** (C_1, C_2, μ) in DB_o
$\forall \sigma$, if $\exists e(\mathcal{H}_1(bsn), C_2)$ **such that** $e(K, G_2) = e(\mathcal{H}_1(bsn), C_2)$, **the validator denies the access to the user.**	$\xleftarrow{e(\mathcal{H}_1(bsn), C_2)}$	$\forall bsn$, **Compute** $e(\mathcal{H}_1(bsn), C_2)$

FIGURE 6.6: Protocol for Blacklisting a User.

6.4 Requirements

We consider three types of requirements: *functional* requirements which are "efficiency" and "versatility", *security* requirements which consist in "correctness", "traceability" and "Non-frameability", and *privacy* requirements which comprises "unlinkability".

6.4.1 Functional Requirements

Efficiency An m-pass systems must fulfill functional requirements imposed by transport operators [7], in particular the validation of an m-pass must be performed in less than $300ms$. We must consider that a SIM card has limited computation capabilities. In particular, pairing APIs are not available on current SIM cards.

Versatility The user smartphone and the validator cannot be assumed to be connected to a back-end server during the validation phase. This enables the user to use his m-pass in any kind of situation, especially in areas with low connectivity, e.g., underground, or if the battery of his mobile is flat.

6.4.2 Security and Privacy Model

We formally define the security and privacy properties (except the correctness property) of our m-pass protocol in which the attack capabilities of a probabilistic polynomial time adversary $\mathcal{A}$ are modeled by providing him access to some oracles. In the sequel, $\mathcal{HU}$ will denote the set of honest users and $\mathcal{MU}$ the set of corrupted users. We assume that $\mathcal{A}$ receives all the exchanged messages in our system. $\mathcal{A}$ acts as an active adversary as regards to the messages issued by malicious users and as a passive adversary with respect to honest users.

$\mathcal{O}Register_{HU}$ is an oracle that will be used by an adversary in order to register honest users. By calling this oracle with ID_U as argument, the adversary adds a new user. The oracle runs $sk_u \leftarrow \texttt{UKeygen}(gpk, ID_U)$ and adds ID_U to the set $\mathcal{HU}$. The private key sk_u is kept secret and a commitment of sk_u is returned to the adversary.

$\mathcal{O}Register_{MU}$ is an oracle that will be used by an adversary in order to register malicious users. The adversary calls this oracle with argument the identifier ID_U of a user and sets his private key to sk_u. The identity ID_U is added to the set $\mathcal{MU}$.

$\mathcal{O}CorruptUser(ID_U)$ is a user secret key oracle enabling the adversary to obtain the private key sk_u of a user $ID_U \in \mathcal{HU}$. The oracle sends sk_u to $\mathcal{A}$.

$\mathcal{O}GetMPass_U$ is an oracle that runs the user's side in the `GetMPass` protocol. This oracle will be used by an adversary playing the role of a malicious TA. The adversary gives to the oracle an identity ID_U of an honest registered user. If the user accepts the protocol, the adversary gets a CL-certificate of the user.

$\mathcal{O}GetMPass_T$ is an oracle that runs the transport authority side in the `GetMPass` protocol. This oracle will be used to simulate the execution of the protocol between a user (corrupted or not) and an honest TA.

$\mathcal{O}SignMPass(ID_U, bsn, rc)$ is an oracle that takes as input the identifier ID_U of an honest and registered user (he has one valid CL-certificate), a fresh time slot bsn and a challenge rc, and outputs a signature σ on bsn and rc. The oracle records (ID_U, σ, bsn, rc) in a list *Set*.

$\mathcal{O}IdentUser(\sigma)$ is an oracle that takes on input a signature σ and outputs the identifier ID_U of the user who generated the signature σ.

$\mathcal{O}IdentMPass(C_1, \sigma, bsn, rc)$ is an oracle that takes on input C_1 a commitment of sk_u (a user's secret, a valid signature σ (which has been generated during the execution of the algorithm `SignMPass`), a time slot bsn and a message rc. It indicates whether the signature has been generated on bsn and rc, and using the committed secret on C_1.

Correctness Informally speaking, our protocol is correct if (1) a valid m-pass enables to generate valid signatures, (2) honestly generated signatures are accepted, (3) a validated signature enables opening authorities to identify the user who generated it, (4) opening authorities can link all the signature generated by a given registered user that obtained a valid m-pass, and (5) valid signatures generated by the same user during the same time slot bsn are linkable.

Unlinkability Informally speaking, it should be impossible, except for the opening authorities, to track an m-pass obtained by a user, in particular, to distinguish which of users produced a given signature for every bsn to link two signatures validated by the same user. For this, an adversary has full control over the transport authority (in particular it owns the private key sk_t) and all the users except two honest users i_0 and i_1. The adversary can initiate the `IdentUser` protocol over any signature and can get the user's identity behind it, except for the signatures generated by i_0 and i_1. He can also initiate the `IdentMPass` protocol for all the users except for i_0 and i_1. We define the unlinkability experiment $\mathbf{Exp}_{\mathcal{A}}^{unlink}(1^\lambda)$ in Figure 6.7. The scheme is unlinkable if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage $\mathbf{Adv}_{\mathcal{A}}^{unlink-b}(1^\lambda) = |Pr[\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda) = b] - 1/2|$ is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^{\lambda})$

1. $pp \leftarrow Setup(1^{\lambda})$; $\mathcal{HU} \leftarrow \varnothing$; $\mathcal{MU} \leftarrow \varnothing$
2. $(gpk, sk_t, sk_o) \leftarrow \texttt{Keygen}(pp)$
3. $(i_0, i_1, bsn, rc) \leftarrow \mathcal{A}(gpk, sk_t, DB_t$: $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}GetMPass_U$, $\mathcal{O}SignMPass$, $\mathcal{O}IdentUser$, $\mathcal{O}IdentMPass)$
4. If i_0 or $i_1 \in \mathcal{MU}$ then return $\perp$.
5. If $\mathcal{O}SignMPass$ has been requested on i_0 or i_1 with bsn then return $\perp$.
6. $b \leftarrow \{0, 1\}$
7. $\sigma_b \leftarrow \texttt{SignMPass}(gpk, sk_{i_b}, bsn, rc)$
8. $b' \leftarrow \mathcal{A}(gpk, sk_t, DB_t, \sigma_b, \sigma_{1-b}$: $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}GetMPass_U$, $\mathcal{O}SignMPass$, $\mathcal{O}IdentUser$, $\mathcal{O}IdentMPass)$
9. If $\mathcal{O}CorruptUser$ has been requested on i_0 or i_1 then return $\perp$.
10. If $\mathcal{O}SignMPass$ has been requested on bsn on either i_0 or i_1, then return $\perp$.
11. If $\mathcal{O}IdentUser$ has been requested on σ_b then return $\perp$.
12. If $\mathcal{O}IdentMPass$ has been requested on(i_0, σ_b, bsn, rc) or (i_1, σ_b, bsn, rc) then return $\perp$.
13. return b'

FIGURE 6.7: Unlinkability Security Experiment

Traceability Informally speaking, it should be impossible for anyone to validate an m-pass, i.e., generate a valid signature σ, such that the algorithm `IdentUser` returns $\perp$, i.e., an identifier ID_U that doesn't appear in the transport authority database DB_t. We formally define the traceability experiment $\mathbf{Exp}_{\mathcal{A}}^{trac}(1^{\lambda})$ in Figure 6.8. The scheme is traceable if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability $\texttt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{trac}(1^{\lambda}) = 1]$ is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{trac}(1^{\lambda})$

1. $pp \leftarrow Setup(1^{\lambda})$; $\mathcal{HU} \leftarrow \varnothing$; $\mathcal{MU} \leftarrow \varnothing$
2. $(gpk, sk_t, sk_o) \leftarrow \texttt{Keygen}(pp)$
3. $(\sigma, bsn, rc) \leftarrow \mathcal{A}(gpk$: $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}GetMPass_T)$.
4. If $\texttt{VerifyMPass}(gpk, \sigma, bsn, rc) = \perp$ then return 0.
5. If $\texttt{IdentUser}(\sigma, DB_o, DB_t) = \perp$ return 1 else return 0.

FIGURE 6.8: Traceability Security Experiment

Non-Frameability Informally speaking, it should be impossible for anyone to falsely accuse a honest user of having used his m-pass. We formally define the non-frameability experiment $\mathbf{Exp}_{\mathcal{A}}^{Nfra}(1^{\lambda})$ in Figure 6.9. The scheme is non-frameable, if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability
$\texttt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{Nfra}(1^{\lambda})=1]$ is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{Nfra}(1^{\lambda})$

1. $pp \leftarrow Setup(1^{\lambda})$; $\mathcal{HU} \leftarrow \varnothing$; $\mathcal{MU} \leftarrow \varnothing$; $Set \leftarrow \varnothing$
2. $(gpk, sk_t, sk_o) \leftarrow \texttt{Keygen}(pp)$
3. $(\sigma, bsn, rc) \leftarrow \mathcal{A}(gpk, sk_t, sk_o, DB_t, DB_o: \mathcal{O}Register_{HU}, \mathcal{O}Register_{MU}, \mathcal{O}CorruptUser, \mathcal{O}GetMPass_U, \mathcal{O}SignMPass)$
4. If $\texttt{VerifyMPass}(gpk, \sigma, bsn, rc) = \perp$ or $\texttt{IdentUser}(\sigma, DB_o, DB_t) = \perp$ then return 0
5. If $\texttt{IdentUser}(\sigma, DB_o, DB_t) = ID_U \in \mathcal{HU}$ and $(ID_U, \sigma, bsn, rc) \notin Set$ then return 1 else return 0.

FIGURE 6.9: Non-frameability Security Experiment

6.5 Security Analysis

We prove that our m-pass protocol provides the security and privacy properties defined in Section 6.4.

Forking Lemma. We use the Forking Lemma [101] in our proofs, to prove that an adversary $\mathcal{A}$ is not able to produce a new valid signature σ (see Figure 6.4) unless he knows the secret sk_u corresponding to the valid CL-certificate (R, S, T, W) and the discrete logarithm of K in the base J.

Using the notation of [101], if an adversary is able to produce a valid signature σ (rc, bsn, σ_1, h, σ_2) where $\sigma_1 = (J, K, R, S, T, W, R_1, R_2, rc)$, $h = \mathcal{H}(J, K, R, S, T, W, R_1, R_2, rc)$, and $\sigma_2 = s$ then an adversary $\mathcal{A}'$ can produce two valid signatures (rc, bsn, σ_1, h, σ_2) and (rc, bsn, σ_1, h', σ'_2) such that $h \neq h'$.

Theorem 3 (Unlinkability). Our m-pass protocol satisfies the unlinkability requirement, in the random oracle model, under the XDH assumption.

Proof 3 (sketch). We give a security proof as a sequence of games, using Shoup's methodology [102]. We will only give a high-level description of the initial game (**Game 0**) and brief descriptions of the modifications between successive games.

Game 0: This is the original attack game with respect to a given efficient adversary $\mathcal{A}$.

The Challenger $\mathcal{C}$ can construct the public parameter gpp for $\mathcal{A}$. It also chooses the keys for the Paillier encryption scheme and the ones for the transport and opening authorities. $\mathcal{C}$ sends gpp and sk_t to $\mathcal{A}$ and answer his requests as follows:

- $\mathcal{O}Register_{HU}$ requests: the Challenger $\mathcal{C}$ randomly chooses $sk_u \in \mathbb{Z}_p^*$, computes $C_1 = [sk_u]G_1$ and, $C_2 = [sk_u]G_2$, returns (C_1, μ) to the adversary $\mathcal{A}$ and stores (C_1, $C_2 = [sk_u]G_2$, μ) in DB_o.
- $\mathcal{O}Register_{MU}$ requests: $\mathcal{C}$ randomly chooses $s'' \in \mathbb{Z}_p^*$ and sends it to $\mathcal{A}$ along with a signature μ on the commitment $C_1 = [s' + s'']G_1$ provided by $\mathcal{A}$ (where s' is a secret value chosen by $\mathcal{A}$). $\mathcal{C}$ stores (C_1, $C_2 = [s' + s'']G_2$, μ) in DB_o.
- $\mathcal{O}CorruptUser$ requests: the Challenger $\mathcal{C}$ retrieves sk_u and gives it to $\mathcal{A}$.
- $\mathcal{O}GetMPass_U$ requests: $\mathcal{C}$ retrieves sk_u and proceeds normally to obtain a certificate on sk_u

- $\mathcal{O}SignMPass$ requests: the Challenger $\mathcal{C}$ has all the values (sk_u, A, B, C, D) that has been issued in $\mathcal{O}GetMPass_U$ / $\mathcal{O}Register_{HU}$ requests. Therefore, $\mathcal{C}$ can generate signatures on any bsn and rc.
- $\mathcal{O}IdentUser(\sigma)$ requests: the Challenger $\mathcal{C}$ uses the algorithm `IdentUser` to identify the user who generated σ.
- $\mathcal{O}IdentMPass(C_1, \sigma, bsn, rc)$ requests: the Challenger uses the algorithm `IdentMPass` to determine whether σ has been generated using the committed secret in C_1 or not.

Challenge phase: The adversary $\mathcal{A}$ outputs two honest users (i_0, i_1), a message rc, and a basename bsn. $\mathcal{C}$ randomly chooses a value b from $\{0, 1\}$ and issues a valid signature σ_b on rc and bsn on behalf of i_b. The goal of $\mathcal{A}$ is to guess the value b. After having received its challenge, $\mathcal{A}$ can query $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}GetMPass_U$, $\mathcal{O}SignMPass$, $\mathcal{O}IdentUser$ and $\mathcal{O}IdentMPass$ but with the following restrictions: it cannot query:

- the $\mathcal{O}CorruptUser$ oracle on i_0 or i_1,
- $\mathcal{O}SignMPass$ oracle on i_0 or i_1,
- $\mathcal{O}IdentUser$ oracle on σ_b,
- $\mathcal{O}IdentMPass$ oracle on (i_0, σ_b, bsn, rc) or (i_1, σ_b, bsn, rc)

At the end of the game, the adversary $\mathcal{A}$ outputs a bit b', its guess. Let S_0 be the event that $b = b'$ in this game and S_i the event that $b = b'$ in the game i. We have,

$$|Pr[S_0] - 1/2| = \mathbf{Adv}_{\mathcal{A}}^{unlink-b} = |Pr[\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda)] - 1/2|$$

Game 1: this game is the same as **Game 0** except that we replace the value $K = [sk_{i_b}]J$ in the signature σ_b by a random value of the group $\mathbb{G}_1$ and simulate the "signature of knowledge" σ_b . Such a simulated proof can be easily done in the random oracle model using standard techniques. Under the XDH assumption, the adversary $\mathcal{A}$ cannot detect this change. Indeed, we can easily construct an XDH distinguisher $\mathcal{D}$, as proven below in **Claim 1** (further details about **Claim 1** are given after the unlinkability proof), with XDH-advantage satisfying:

$$|Pr[S_0] - Pr[S_1]| \leqslant q_H \times q_R \times \mathbf{Adv}_{\mathcal{D}}^{XDH}(1^\lambda) \quad (1)$$

where $\mathbf{Adv}_{\mathcal{D}}^{XDH}(1^\lambda)$ is the XDH-advantage of $\mathcal{D}$ in solving the XDH problem and q_H (respectively q_R) a bound on the number of hash queries (respectively $\mathcal{O}Register_{HU}$ requests). In the sequel, we will use the simplified notation $\mathbf{Adv}_{XDH}$ to denote the XDH-advantage of some efficient algorithm in solving the XDH problem.

Game 2: This is the same game as **Game 1** except that we replace the CL-certificate (R, S, T, W) on sk_{i_b} by a CL-certificate on a random value and then simulate the "signature of knowledge" σ_b. Such a simulated proof can easily be done in the random oracle model using standard techniques. Under the XDH assumption, $\mathcal{A}$ cannot detect this change. Indeed, we can easily construct an XDH distinguisher $\mathcal{D}$, as proven below in **Claim 2** (further details about **Claim 2** are given after the unlinkability proof), with XDH-advantage satisfying:

$$|Pr[S_1] - Pr[S_2]| \leqslant q_R \times \mathbf{Adv}_{XDH} \quad (2)$$

In the **Game(2)**, the unlinkability-challenger $\mathcal{C}$ gives no information (in a strong information theoretic sense) to $\mathcal{A}$ about the bit b (since the CL-certificate and the "tag" K have been replaced by random values). Therefore we have:

$$Pr[S_2] = 1/2$$

We can now give an upper bound for $\mathbf{Adv}_{A}^{unlink-b}(1^\lambda)$:

$$\mathbf{Adv}_{\mathcal{A}}^{unlink-b}(1^\lambda) = |Pr[\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda)] - 1/2| = |Pr[S_0] - Pr[S_2]|$$

We have:

$$|Pr[S_0] - Pr[S_2]| = \Sigma_{j=0}^{j=1}|Pr[S_j] - Pr[S_{j+1}]| \leq q_H \ \times \ (q_R + 1) \ \mathbf{Adv}_{XDH}$$

Therefore under the XDH assumption $\mathbf{Adv}_{\mathcal{A}}^{unlink-b}(1^\lambda)$ is negligible.

We can then conclude that our proposed m-pass protocol satisfies the unlinkability requirement, in the random oracle model, under the XDH assumption.

Claim 1: $|Pr[S_0] - Pr[S_1]| \leq q_H \ \times \ q_R \ \times \ \mathbf{Adv}_{XDH}$

Proof of Claim 1 (sketch): let $\mathcal{A}$ be an adversary who breaks the unlinkability requirement of our m-pass protocol with non-negligible advantage. We will construct an algorithm $\mathcal{D}$, using $\mathcal{A}$ as an oracle, which solves the XDH problem. Let $\mathcal{X}$ be $\mathcal{D}$'s challenger in the XDH-game. $\mathcal{X}$ selects a bit d and sends to $\mathcal{D}$ an XDH-triplet if $d = 1$ and a random one if $d = 0$. Let $(G_1, \mathcal{A} = [a]G_1, \mathcal{B} = [b]G_1, \mathcal{C} = [c]G_1) \in G_1^4$ be $\mathcal{D}$'s XDH-challenge. $\mathcal{D}$ has to decide whether $d = 1$ (i.e. $c \equiv ab \ mod \ p$) or not. $\mathcal{D}$ first defines the public parameters *gpp* of the m-pass protocol from the ones of the XDH-challenge, where $\mathcal{H}$ and $\mathcal{H}_1$ are random oracles under the control of $\mathcal{D}$. $\mathcal{D}$ also chooses the keys for the Paillier encryption scheme and the ones for the transport and opening authorities. $\mathcal{D}$ randomly selects i^* in $[1;\ q_R]$ and j^* in $[1;\ q_H]$. $\mathcal{D}$ sends *gpp* and sk_t to $\mathcal{A}$ and answers its requests as follows:

- $\mathcal{O}Register_{HU}$ requests: for the i-th request, $\mathcal{D}$ will acts as follows:
  1. if $i \neq i^*$, then it randomly chooses $sk_u \in \mathbb{Z}_p^*$, computes $C_1 = [sk_u]G_1$ and, $C_2 = [sk_u]G_2$, returns $(C_1,\ \mu)$ to the adversary $\mathcal{A}$ and stores $(C_1,\ C_2 = [sk_u]G_2,\ \mu)$ in DB_o.
  2. if $i = i^*$, $\mathcal{D}$ randomly chooses α and β in $\mathbb{Z}_p^*$, computes $C_1 = [\alpha a]G_1 + [\beta]G_1 = [\alpha a + \beta]G_1$ (where $[a]G_1$ comes from the XDH challenge) and returns $(C_1,\ \mu)$ to the adversary $\mathcal{A}$.
- $\mathcal{O}Register_{MU}$ requests: $\mathcal{D}$ randomly chooses $s'' \in \mathbb{Z}_p^*$ and sends it to $\mathcal{A}$ along with a signature μ on the commitment $C_1 = [s' + s'']G_1$ provided by $\mathcal{A}$ (where s' is a secret value chosen by $\mathcal{A}$). $\mathcal{D}$ stores $(C_1,\ C_2 = [s' + s'']G_2,\ \mu)$ in DB_o.
- $\mathcal{O}CorruptUser$ requests: if $ID_{U_i} \neq ID_{U_{i^*}}$, $\mathcal{D}$ retrieves sk_{u_i} and gives it to the adversary $\mathcal{A}$ and aborts otherwise.
- $\mathcal{O}GetMPass_U$ requests: if $ID_{U_i} \neq ID_{U_{i^*}}$, $\mathcal{D}$ retrieves sk_{u_i} and proceeds normally to obtain a certificate on sk_{u_i}. Else, it randomly chooses a value $C_0 \in \mathbb{Z}_{n^2}$ and simulates the proof Π'_1 (this is possible in the random oracle model) and obtains a blind CL-certificate $(A,\ B,\ C,\ D)$ on $C_1 = [\alpha a + \beta]G_1$.

- $Hash$ requests: upon receiving the j-th request on bsn, $\mathcal{D}$ proceeds as follows: if no previous request was made on bsn, then we distinguish two cases: if $j \neq j^*$, then $\mathcal{D}$ selects $\delta \in \mathbb{Z}_p^*$ and outputs $\mathcal{H}_1(bsn) = [\delta]G_1$. Else, it selects $\gamma \in \mathbb{Z}_p^*$ and outputs $\mathcal{H}_1(bsn) = [b]G_1 + [\gamma]G_1 = [b+\gamma]G_1$, where $[b]G_1$ comes from the XDH challenge. If bsn has already been queried then $\mathcal{D}$ returns the same output as the first request.

- $\mathcal{O}SignMPass$ requests: upon receiving a query on (ID_U, rc, bsn), $\mathcal{D}$ proceeds as usual if $ID_{U_i} \neq ID_{U_{i^*}}$, else we distinguish the two following cases:

  1. If bsn is the one that was requested in the j^*-th hash request then $\mathcal{D}$ aborts.
  2. Else, it chooses a fresh value $\delta \in \mathbb{Z}_p^*$ (if bsn has not already been queried), computes $\mathcal{H}_1(bsn) = [\delta]G_1$ and K=$[\delta]C_1$=$[\delta(\alpha a + \beta)]G_1$. $\mathcal{D}$ simulates the signature of knowledge σ (which is possible in the random oracle model), outputs σ and stores (σ, rc, bsn) in Set.

- $\mathcal{O}IdentUser(\sigma)$ requests: upon receiving a query on (σ, rc, bsn), $\mathcal{D}$ checks if (σ, rc, bsn) belongs to Set. If so, it returns $ID_{U_{i^*}}$. Else it uses the `IdentUser` algorithm, to identify the user who generated σ.

- $\mathcal{O}IdentMPass(C_1, \sigma, bsn, rc)$ requests: $\mathcal{D}$ checks if (σ, rc, bsn) belongs to Set. If so, it returns "1" to $\mathcal{A}$, which means that the signature comes from $ID_{U_{i^*}}$. Otherwise, it uses the algorithm `IdentMPass` to determine whether σ has been generated using the committed secret on C_1 or not.

Challenge phase: $\mathcal{A}$ outputs two honest identities ID_{i_0} and ID_{i_1}, a message rc and a basename bsn. If a hash request on this basename was already submitted during the j-th request then $\mathcal{D}$ aborts if $j \neq j^*$. $\mathcal{D}$ randomly selects a bit b and aborts if $ID_{i_b} \neq ID_{U_{i^*}}$. Else, it issues a signature σ_b on rc and bsn on behalf of i_b as follows: if no hash request was submitted on bsn, then it selects $\gamma \in \mathbb{Z}_p^*$ and outputs $\mathcal{H}_1(bsn) = [b]G_1 + [\gamma]G_1 = [b+\gamma]G_1$. It randomly chooses $l \in \mathbb{Z}_p^*$ to randomize the blind CL-certificate (A, B, C, D) on $C_1 = [\alpha a + \beta]G_1$: ($R = [l]A$, $S = [l]B$, $T = [l]C$, $W = [l]D$). It then computes $K = [\alpha]\mathcal{C} + [\beta]\mathcal{B} + [\alpha\gamma]\mathcal{A} + [\beta\gamma]G_1$ and simulates the "signature of knowledge" $\sigma_b = (R, S, T, W, K)$. Note that if $c \equiv ab \ mod \ p$, the signature σ_b issued by $\mathcal{D}$ in the challenge phase is indistinguishable from an original one; else it is a random one.

After the Challenge phase, $\mathcal{D}$ proceeds as in the previous phase. At the end of the game, the adversary outputs a bit b', its guess. If $b' = b$, then $\mathcal{D}$ returns 1 to $\mathcal{X}$ and 0 otherwise. Let F be the event that $\mathcal{D}$ did not abort. We have

$$Pr[d' = 1/d = 1] = Pr[S_0 \wedge F]$$

and

$$Pr[d' = 1/d = 0] = Pr[S_1 \wedge F]$$

therefore:

$$|Pr[S_0] - Pr[S_1]| \leq q_H \ \times \ q_R \ \times \ \mathbf{Adv}_D^{XDH}(1^\lambda)$$

Claim 2: $|Pr[S_1] - Pr[S_2]| \leqslant q_R \ \times \ \mathbf{Adv}_{XDH}$

Proof of Claim 2 (sketch): the proof is similar to the proof of **Claim 1**; only the construction of the signature σ_b differs in the two proofs. We will construct an algorithm $\mathcal{D}$, using $\mathcal{A}$ as an oracle, which solves the XDH problem. Here again, let us denote by $\mathcal{X}$ $\mathcal{D}$'s challenger in the XDH-game and by (G_1, $\mathcal{A} = [a]G_1$, $\mathcal{B} = [b]G_1$, $\mathcal{C} = [c]G_1$)$\in G_1^4$ its XDH-challenge. $\mathcal{D}$ has to decide whether $d = 1$ (i.e. $c \equiv ab \ mod \ p$) or not. $\mathcal{D}$ first defines

the public parameters gpp of the m-pass protocol from the ones of the XDH-challenge, where $\mathcal{H}$ and $\mathcal{H}_1$ are random oracles under the control of $\mathcal{D}$. $\mathcal{D}$ also chooses the keys for the Paillier encryption scheme and the ones for the transport and opening authorities. In particular, $\mathcal{D}$ knows $sk_t = (x, y)$ the private key of the transport authority. Let q_R be a bound on the number of $\mathcal{O}Register_{HU}$ requests. $\mathcal{D}$ randomly selects i^* in [1; q_R], sends gpp and sk_t to $\mathcal{A}$, and answers its requests as follows:

- $\mathcal{O}Register_{HU}$ requests: for the i-th request, $\mathcal{D}$ will acts as follows:
  1. if $i \neq i^*$, then it randomly chooses $sk_u \in \mathbb{Z}_p^*$, computes $C_1 = [sk_u]G_1$ and, $C_2 = [sk_u]G_2$, returns (C_1, μ) to the adversary $\mathcal{A}$ and stores (C_1, $C_2 = [sk_u]G_2$, μ) in DB_o.
  2. if $i = i^*$, $\mathcal{D}$ randomly chooses α and β in $\mathbb{Z}_p^*$, computes $C_1 = [\alpha a]G_1 + [\beta]G_1 = [\alpha a + \beta]G_1$ (where $[a]G_1$ comes from the XDH challenge) and returns (C_1, μ) to the adversary $\mathcal{A}$.
- $\mathcal{O}Register_{MU}$ requests: $\mathcal{D}$ randomly chooses $s'' \in \mathbb{Z}_p^*$ and sends it to $\mathcal{A}$ along with a signature μ on the commitment $C_1 = [s' + s'']G_1$ provided by $\mathcal{A}$ (where s' is a secret value chosen by $\mathcal{A}$). $\mathcal{D}$ stores (C_1, $C_2 = [s' + s'']G_2$, μ) in DB_o.
- $\mathcal{O}CorruptUser$ requests: if $ID_{U_i} \neq ID_{U_{i^*}}$, $\mathcal{D}$ retrieves sk_{u_i} and gives it to the adversary $\mathcal{A}$ and aborts otherwise.
- $\mathcal{O}GetMPass_U$ requests: if $ID_{U_i} \neq ID_{U_{i^*}}$, $\mathcal{D}$ retrieves sk_{u_i} and proceeds normally to obtain a certificate on sk_{u_i}. Else, it randomly chooses a value $C_0 \in \mathbb{Z}_{n^2}$ and simulates the proof Π'_1 (this is possible in the random oracle model) and obtains a blind CL-certificate (A, B, C, D) on $C_1 = [\alpha a + \beta]G_1$.
- Hash requests: upon receiving the j-th request on bsn, $\mathcal{D}$ proceeds as follows: if no previous request was made on bsn, then $\mathcal{D}$ selects $\delta \in \mathbb{Z}_p^*$ and outputs $\mathcal{H}_1(bsn) = [\delta]G_1$. If bsn has already been queried then $\mathcal{D}$ returns the same output as the first request.
- $\mathcal{O}SignMPass$ requests: upon receiving a query on (ID_U, rc, bsn), $\mathcal{D}$ proceeds as usual if $ID_{U_i} \neq ID_{U_{i^*}}$, else, if bsn has not been queried, it chooses a fresh value $\delta \in \mathbb{Z}_p^*$, computes $\mathcal{H}_1(bsn) = [\delta]G_1$ and $K{=}[\delta]C_1{=}[\delta(\alpha a + \beta)]G_1$. $\mathcal{D}$ simulates the signature of knowledge σ (which is possible in the random oracle model), outputs σ and stores (σ, rc, bsn) in Set.
- $\mathcal{O}IdentUser(\sigma)$ requests: upon receiving a query on (σ, rc, bsn), $\mathcal{D}$ checks if (σ, rc, bsn) belongs to Set. If so, it returns $ID_{U_{i^*}}$. Else, it uses the `IdentUser` algorithm, to identify the user who generated σ.
- $\mathcal{O}IdentMPass(C_1, \sigma, bsn, rc)$ requests: $\mathcal{D}$ checks if (σ, rc, bsn, rc) belongs to Set. If so, it returns "1" to $\mathcal{A}$, which means that the signature comes from $ID_{U_{i^*}}$. Otherwise, it uses the algorithm `IdentMPass` to determine whether σ has been generated using the committed secret in C_1 or not.

Challenge phase: $\mathcal{A}$ outputs two honest identities ID_{i_0} and ID_{i_1}, a message rc and a basename bsn. $\mathcal{D}$ randomly selects a bit b and aborts if $ID_{i_b} \neq ID_{U_{i^*}}$. Else it issues a signature σ_b on rc and bsn on behalf of i_b as follows: if no hash request was submitted on bsn, then it selects $\delta \in \mathbb{Z}_p^*$ and $\gamma \in \mathbb{Z}_p^*$ and outputs $\mathcal{H}_1(bsn) = [\delta]G_1$. It then

computes the values (R, S, T, W, K) of the signature σ as follows: $R = [b + \gamma]G_1$ (where $[b]G_1$ comes from the XDH challenge), $S = [y(b + \gamma)]G_1$, $T = [x](R + W)$ and $W = [y]([\alpha]\mathcal{C} + [\beta]\mathcal{B} + [\alpha\gamma]\mathcal{A} + [\beta\gamma]G_1)$. It then randomly chooses K in the group $\mathbb{G}_1$ and simulates the "signature of knowledge" $\sigma_b = (R, S, T, W, K)$. Note that if $c \equiv ab \ mod \ p$, the blind CL-certificate (R, S, T, W) on C_1 issued by $\mathcal{D}$ in the challenge phase is indistinguishable from an original one; else it is a CL-certificate on a random one.

After the Challenge phase, $\mathcal{D}$ proceeds as in the previous phase. At the end of the game, the adversary outputs a bit b', its guess. If $b' = b$, then $\mathcal{D}$ returns 1 to $\mathcal{X}$ and 0 otherwise. Let F be the event that $\mathcal{D}$ did not abort. We have

$$Pr[F] \geq 1/q_R$$

We also have:

$$Pr[d' = 1/d = 1] = Pr[S_1 \wedge F]$$

and

$$Pr[d' = 1/d = 0] = Pr[S_2 \wedge F]$$

therefore:

$$|Pr[S_1] - Pr[S_2]| \leq q_R \ \times \ \mathbf{Adv}_D^{XDH}(1^\lambda)$$

Theorem 4 (Traceability). Our m-pass protocol satisfies the traceability requirement, in the random oracle model, under the LRSW assumption.

Proof 4 (sketch). Let $\mathcal{C}$ be the Challenger of the adversary $\mathcal{A}$ in the traceability experiment. $\mathcal{C}$ will use $\mathcal{A}$ to break the LRSW assumption. The Challenger $\mathcal{C}$ has access to a CL-certificate oracle. The Challenger has on input the public key of transport authority pk_t. $\mathcal{C}$ chooses also the keys for Paillier encryption scheme. It can, then, construct the public parameters *gpp* for $\mathcal{A}$ and answer the requests of the adversary $\mathcal{A}$ as follows:

- $\mathcal{O}Register_{HU}$ requests: the Challenger $\mathcal{C}$ randomly chooses sk_u and computes $C_1 = [sk_u]G_1$, and $C_2 = [sk_u]G_2$.
- $\mathcal{O}Register_{MU}$ requests: the Challenger $\mathcal{C}$ randomly chooses s'' and sends this value to $\mathcal{A}$. It then sends a signature μ on the value C_1 provided by $\mathcal{A}$.
- $\mathcal{O}CorruptUser$ requests: $\mathcal{C}$ gives sk_u to the adversary $\mathcal{A}$.
- $\mathcal{O}GetMPass_T$ requests: if $ID_U \in \mathcal{HU}$, $\mathcal{C}$ retrieves sk_u. If $ID_U \in \mathcal{MU}$, $\mathcal{C}$ decrypts C_0 and retrieves sk_u. The Challenger $\mathcal{C}$ then calls the CL-certificate oracle with the input sk_u. The CL-certificate oracle returns (A, B, C, D). $\mathcal{C}$ gives (A, B, C, D) to the adversary.
- $\mathcal{O}SignMPass$ requests: the Challenger $\mathcal{C}$ has all the values (sk_u, A, B, C, D) that has been issued in $\mathcal{O}Register_{HU}$ / $\mathcal{O}GetMPass_T$ requests. Therefore, $\mathcal{C}$ can generate signatures on any *bsn* and *rc*.

Eventually, the adversary $\mathcal{A}$ outputs a valid signature $\tilde{\sigma} = (\tilde{R}, \tilde{S}, \tilde{T}, \tilde{W}, \tilde{K}, \tilde{c}, \tilde{s})$ on a challenge rc and a basename bsn such that the algorithm `IdentUser` on σ does not return an ID of a registered user (ID does not match with the databases DB_t, DB_o), and consequently, the CL-certificate ($\tilde{R}$, $\tilde{S}$, $\tilde{T}$, $\tilde{W}$) certifies an unknown value $\tilde{sk}_u$ that has not been requested to the CL-certificate oracle

Using the Forking Lemma, the replay technique and the soundness property, the Challenger can extract $\tilde{sk}_u$ from $\tilde{\sigma}$, and $(\tilde{R}, \tilde{S}, \tilde{T}, \tilde{W})$ is a valid CL-certificate on $\tilde{sk}_u$. This is a forgery because the value $\tilde{sk}_u$ has (by definition) never been asked to the CL-certificate oracle. Thus, $\mathcal{C}$ breaks the LRSW assumption.

Therefore, our m-pass protocol is traceable under the LRSW assumption.

Theorem 5 (Non-Frameability). Our m-pass protocol satisfies the non-frameability requirement, in the random oracle model, under the SDL assumption.

Proof 5 (sketch). Let $\mathcal{A}$ be an adversary against the non-frameability requirement. We construct a reduction $\mathcal{C}$ using $\mathcal{A}$ against SDL problem challenges. Let $(G_1, \tilde{C}_1 = [s]G_1, G_2, \tilde{C}_2 = [s]G_2)$ be the SDL challenge. The Challenger $\mathcal{C}$ must output s.

Let q_R be the bound on the number of $\mathcal{O}Register_{HU}$ requests. $\mathcal{C}$ selects $i^* \in [1, q_R]$, construct the public parameters *gpp* for $\mathcal{A}$ as in the initialisation phase, and answers its queries as follows.

$\mathcal{A}$ chooses the keys for the opening and transport authorities.

- $\mathcal{O}Register_{HU}$ requests: when $\mathcal{A}$ makes the i^{th} $\mathcal{O}Register_{HU}$ query to register a user, $\mathcal{C}$ will run the user's private key generation protocol with a fresh value s' if $i \neq i^*$. When $i = i^*$, $\mathcal{C}$ sets $C'_2 = \tilde{C}_2$ and simulates the proof Π'_2 (this is possible in the random oracle model). It then computes $C_2 = \tilde{C}_2 + [s'']G_2 = [s + s'']G_2$ where s'' is provided by $\mathcal{A}$. Let $C_1 = \tilde{C}_1 + [s'']G_1$. $\mathcal{C}$ simulates the proof Π_1 and obtains from $\mathcal{A}$ a signature μ.
- $\mathcal{O}Register_{MU}$ requests: $\mathcal{C}$ does not need to do anything.
- $\mathcal{O}CorruptUser$ requests: $\mathcal{C}$ returns the corresponding sk_u of the user ID_{U_i} if $i \neq i^*$ and aborts otherwise.
- $\mathcal{O}GetMPass_U$ requests: $\mathcal{C}$ acts normally if $ID_{U_i} \neq ID_{U_{i^*}}$. Otherwise, it randomly chooses C_0, simulates the non-interactive proof Π'_1, and obtains from $\mathcal{A}$ a valid CL-certificate (A, B, C, D) on $\tilde{sk}_u = s + s''$.
- $\mathcal{O}SignMPass(ID_U, bsn, rc)$ requests: the challenger $\mathcal{C}$ acts normally if $ID_{U_i} \neq ID_{U_{i^*}}$. Otherwise, $\mathcal{C}$ picks a random value β in $\mathbb{Z}_p^*$, computes $J = \mathcal{H}_1(bsn) = [\beta]G_1$ (this is possible in the random oracle model), and then $K = [\beta]C_1 = [\beta(s + s'')]G_1 = \tilde{sk}_u J$. $\mathcal{C}$ then simulates the non-interactive proof σ.

Eventually, the adversary $\mathcal{A}$ outputs a valid signature $\tilde{\sigma} = (\tilde{R}, \tilde{S}, \tilde{T}, \tilde{W}, \tilde{K}, \tilde{c}, \tilde{s}))$ on a challenge $\tilde{rc}$ and a basename $\tilde{bsn}$ such that:

- the algorithm `IdentUser` on $\tilde{\sigma}$ returns an an identifier $\tilde{ID}_U \in \mathcal{HU}$, and
- the oracle $\mathcal{O}SignMPass$ has not been called to generate this signature,

If $\tilde{ID}_U \neq ID_{U_{i^*}}$ then $\mathcal{C}$ aborts. Using the Forking Lemma, the replay technique and the soundness property, the Challenger can extract $\tilde{sk}_u$ from $\tilde{\sigma}$. As $\tilde{sk}_u = s + s''\ mod\ p$, $s = \tilde{sk}_u - s''\ mod\ p$. $\mathcal{C}$ is then able to solve the SDL problem since it will not abort with a probability $1/q_R$.

Therefore, our m-pass protocol is non-frameable under the SDL assumption.

6.6 Implementation

In this section, we describe the prototype implementing our mobile pass system. The software components involved in the transport service are represented in Figure 5.2. The validator is simulated by a Java swing application connected to an NFC reader using the ISO14443B protocol. The cardlet is embedded within a SIM card. We used a regular Galaxy S3 smartphone running Android 4.1.2. The Android system does not need any modification as the Seek for Android patch [103] has been already added by Samsung in order to access secure elements. In the following, we give more technical details about the validator and the SIM card, the used cryptographic parameters, and the performance results as well.

6.6.1 Validator Details

The validator has been developed over a PC, i.e., an Intel(R) Xeon(R) with a E5-1620 CPU with 4 cores running at 3.70GHz under a 64-bit Linux OS, in Java and Scala. The Java software uses a native library for EC scalar multiplications and pairings. This library depends on libGMP for big integers computations and benefits from its assembly optimizations. Furthermore, computations are distributed between threads (at JVM level) to benefit from the multi-core architecture of the PC. Regarding the NFC reader of the validator, we used is an Omnikey 5321 dual interfaces.

6.6.2 SIM Card Details

The SIM card handles requests from the validator using the NFC contactless interface. Our SIM card is compliant with

- Global Platform Card specifications version 2.2.1,
- Javacard version 2.2.2 and
- Javacard virtual machine version 3.0.1 classic.

Those features allow to develop Java Card Applets so-called cardlets, to load them on the SIM Card and to execute them thanks to the Java Card virtual machine. In order to provide a very efficient signature computation, we chose a SIM Card equipped with an arithmetic coprocessor. Thanks to this latter, we were able to meet the timing performances required by the transport operators (typically less than 300 ms), by offering non standardized Java Card APIs to the transport cardlet. Those APIs allow to compute efficiently:

- modular operations on large integers like addition, subtraction, multiplication, exponentiation, etc.,
- arithmetic operations on elliptic curves like scalar multiplication and point addition.

Those customized APIs are mapped to assembly subroutines that drive the arithmetic co-processor directly.

6.6.3 Curve and Pairing Parameters

We used a 256-bit Barreto-Naehrig curve [104] over $\mathbb{F}_q$ since this family of curves provides an optimal size for $\mathbb{G}_1$ and $\mathbb{G}_2$ while preventing the MOV attack [105] due to their embedding degree of 12. G_1 is the group of $\mathbb{F}_q$ - rational points of order p and G_2 is the subgroup of trace zero points in $E(F_{q^{12}})[p]$. Our pairing is of type-3 [106].

The curve and pairing parameters that we used in the implementation of our protocols are:

$q = 824340166543009075205740409837836820394672829279961300246559122928892942645 93$

$p = 824340166543009075205740409837836820391801696809065871368966452554653091398 57$

$b = 5$

6.6.4 Performance Results

The validation of an m-pass occurs in two phases: pre-computations and real-time computations. The pre-computations consists in randomizing the CL-certificate (A, B, C, D). Regarding the real-time computations, it occurs when the user taps the validator with his smartphone. After detecting the smartphone and selecting the m-pass cardlet, the validator sends the basename bsn and the challenge rc to the m-pass cardlet. The signature, computed by the cardlet, is sent back to the validator that checks it and displays the result.

6.6.4.1 Pre-Computations

Some pre-computations operations are delegated to the transport application within the smartphone after receiving A, B, C and D from the cardlet. The smartphone application then generates a pool of pre-computed tokens by multiplying each variable A, B, C and D by a random l. Then, the pool of generated tokens is sent back to the cardlet.

1 signature	20 signatures
175 ms	3.47 s

TABLE 6.1: M-Pass Pre-Computation Timings

The elements that the card has to store for one signature are one $\mathbb{Z}_p$ element and 5 uncompressed points (such that four of them we store also one byte for compression). The total size of these elements is 356 *bytes*. In our implementation, we did pre-computations for 20 signatures. Regarding the timings, detailed in Table 6.1, the pre-computations for preparing one signature occurs in average in 175 s.

6.6.4.2 Real-Time Computations

Table 6.2 gives timings (average over more than 60 trials and standard deviation between parentheses) for all real-time computations (the whole validation protocol) which include the signature generation and verification. The timings include as well the NFC exchanges

duration. The performance of our solution is critical for the validation of the m-pass. The whole validation process takes in average 205.27 ms, which is quite efficient. It enables to add the authentication of the validator into a window of time of 300 ms.

We denote by "Battery-Off" a powered-off phone either by the user, or because the battery is flat. It is possible to use the service even if the smartphone has run out of battery. Indeed, the m-pass application hosted in Android is involved only in the pre-computation phase. Once this step is done, the validator interacts with the smartcard via the NFC controller and sends the energy via the NFC magnetic field. In this case, the smartcard runs the signature with lower performances. We measured a total transaction time of 596.81 ms.

	Signature+NFC	Verification	Total
Battery-On	184.24 ms (16.41)	20.79 ms (5.29)	205.27 ms (16.66)
Battery-Off	576.34 ms (39.33)		596.81 ms (39.38)

Table 6.2: M-Pass - Timings of Real-Time Operations

The signature generation consists in computing a signature $\sigma = (R, S, T, W, K, c, s)$ and the NFC communication time. The considered operations for generating the signature are only one hash value and lightweight operations in $\mathbb{Z}_p$. The size of the computed signature is 229 *bytes* (sent by 1 APDUs). Regarding the communication between a SIM card and a reader, it is slow, but the whole process (Signature+NFC) remains very fast, i.e., 184.24 ms on average.

6.7 Conclusion

In this chapter, we proposed a privacy-preserving mobile pass solution for transport service that can be embedded into an NFC-enabled smartphone. This solution enables users to use the transport service in an anonymous and unlinkable manner, preventing in particular the risk of being tracked by the transport authority. However, in exceptional circumstances (*e.g.*, under the injunction of a judge), the identity of the user associated to an anonymous validation can be de-anonymized and his rights to access the transport network revoked. Thus, this solution is flexible enough to ensure the standard security properties while providing a high level of privacy to users.

The cryptographic protocol behind our mobile pass solution is based on group signatures that are used when authenticating with the entrance gates of the transport network. When a user authenticates to the validator, he signs on a challenge on behalf of a group, thus hiding his identity from the transport operator. The unlinkability of the different actions of the user is obtained by changing regularly the challenge sent by the validator. In the situation in which a malicious service operator deliberately refuses to renew the challenge, we suggest to detect such an attack by collecting the received challenge using an Android application for implementing the detection.

The functional requirements of a transport use case mainly consist of the validation time constraint and the operating when the smartphone battery is flat. In this chapter, we showed that the proposed protocol fulfill these functional requirements whilst using

strong security parameters. This is by implementing a prototype using a standard NFC-enabled SIM card and gathering the validation timing over more than 60 trials. The m-pass validation can be completed in 205.27 ms when the mobile is switched on, and in 596.81 ms when the mobile is switched off or its battery is flat.

In the next chapter, we address the privacy problem of the mobile ticketing use case. Although this use case has similarities with the mobile pass use case, it presents more granularity and hence, preserving the same privacy level requires a new cryptographic solution.

Chapter 7

A Practical and Privacy-Preserving Ticketing Service

Contents

Dans ce chapitre, nous présentons une partie des résultats [80] qui ont été publiés dans la revue "Proceedings on Privacy Enhancing Technologies (PoPETs)" de la conférence PETS 2015. Nous concevons un protocole sécurisé pour le cas d'usage billetterie mobile (m-ticketing) sans contact (NFC) pour un service de transport public. Ce protocole respecte l'anonymat des utilisateurs et empêche les opérateurs de transport de tracer leurs clients. À cette fin, nous utilisons la nouvelle preuve d'appartenance à un ensemble, introduite dans le chapitre 4, et proposons également plusieurs optimisations du schéma de signature Boneh-Boyen, permettant l'amélioration de sa performance lors des transactions sans contact (NFC). Notre protocole permet également la révocation de l'anonymat d'un utilisateur ou d'un billet. Cette option est uniquement possible pour les autorités de révocation. Le protocole de m-ticketing offre aussi une plus grande flexibilité par rapport aux solutions précédentes car il permet le post-paiement et la validation hors ligne des m-tickets. Nous avons formellement validé la sécurité de notre protocole dans le modèle de l'oracle aléatoire. Par ailleurs, nous montrons que notre protocole remplit l'exigence

fonctionnelle imposée par les transporteurs: à savoir la validation d'un m-ticket doit se dérouler en moins de 300 ms, tout en utilisant des paramètres cryptographiques (taille des clés) adéquats. En effet, nous mettons en place un prototype utilisant une carte SIM NFC. La validation d'un m-ticket peut être complétée en 184.25 ms lorsque le mobile est allumé, et en 266.52 ms lorsque le mobile est éteint ou la batterie est vide.

7.1 Introduction

The second use case of the transport service is the mobile ticketing. Although this use case has some similarities with the m-pass use case, it has more granularity which applies new privacy issues. A user holds a book of max_{ticket} m-tickets that enables him to have max_{ticket} trips. Every m-ticket is identified by an index $k \in [1..max_{ticket}]$ and a serial number. In order to ensure a high privacy level, we must ensure the anonymity of the m-tickets indexes in addition to the user's anonymity and the unlinkability of the m-tickets serial numbers. The reader can note that the index problem does not exist in the mobile pass use case. Indeed, the m-pass use case can been seen as a simplification of the m-ticketing use case, and hence, ensuring the same privacy level is simpler. In Table 7.1, we summarize the major differences between both use cases.

	M-pass	**M-ticketing**
User's privacy	1. Anonymity of the user 2. Unlinkability between the different m-pass usages	1. Anonymity of the user 2. Unlinkability between the serial numbers of different m-tickets 3. Anonymity of a m-ticket index
Structure	1 m-pass	N m-tickets
Trips	∞	1 ticket per trip

TABLE 7.1: M-Pass vs M-Ticketing

In this chapter, we propose a new cryptographic protocol for the m-ticketing use case that provides strong authentication, strict anonymity, and unlinkability properties, whilst remaining efficient when implemented in constrained environments like SIM cards. Towards this goal, we use the new set-membership proof introduced in Chapter 4. The new set-membership proof does not require provers nor verifiers (but in a specific scenario for verifiers) to perform pairing computations, especially on the prover side. It is therefore particularly suitable for our (ticketing) setting where provers hold SIM cards that do not support such costly computations. We also propose several optimizations of Boneh-Boyen signature schemes which are of independent interest and increase the performance and efficiency of BB-signatures. Based on these cryptographic primitives, our m-ticketing protocol enables to securely validate an m-ticket without disclosing any personal information to the transport operator, even if the latter is malicious. We push forward a strict privacy requirement when using one of the numbered m-tickets: for example, if the user uses the m-ticket number 3 from the book of m-tickets [1..10], the transport operator will only be able to check that the used m-ticket is one among the ten m-tickets [1..10].

This Chapter is organized as follows. In Section 7.2, we detail the framework of our m-ticketing system. We describe our protocol and show how we ensure a secure post-payment in Section 7.3. Then, we formalize our functional, security and privacy requirements in Section 7.4 and formally demonstrate that our protocol ensures these requirements in Section 7.5. Finally, we give details about the performance results of our protocol in Section 7.6 and conclude in Section 7.7.

7.2 Framework of an M-Ticketing System

We recall that we consider three different entities in our m-ticketing system. The user (U) is the owner of an NFC-enabled smartphone and wants to use the transport service. The transport authority (TA) is the manager of the transport service. The revocation authority (RA) can revoke m-tickets and users' anonymity, and is completely independent of the transport authority. This role may be split between several authorities in such a way that they should all agree and cooperate before recovering the identity of an m-ticket holder.

An m-ticketing system consists of six different phases in our protocol. (1) The m-ticketing system parameters and keys are initialized during the *initialization*. (2) The *registration* phase enables a user to register to the transport service. (3) In the *permission token request* phase, a user gets a *permission token* allowing him to generate max_{ticket} m-tickets. (4) The *validation* phase consists in generating and validating an m-ticket. (5) The *revocation* phase enables to retrieve the identity of a user who generated a given m-ticket and the m-tickets obtained by a given user. Finally, (6) the *reporting* phase enables a user to report his usage (i.e. the m-tickets that he did not use) to the transport authority and the transport authority to detect any duplication of m-tickets (i.e., m-tickets that have been validated several times). In the sequel, these phases are modeled with various algorithms and protocols (Figure 7.1) executed by the above entities.

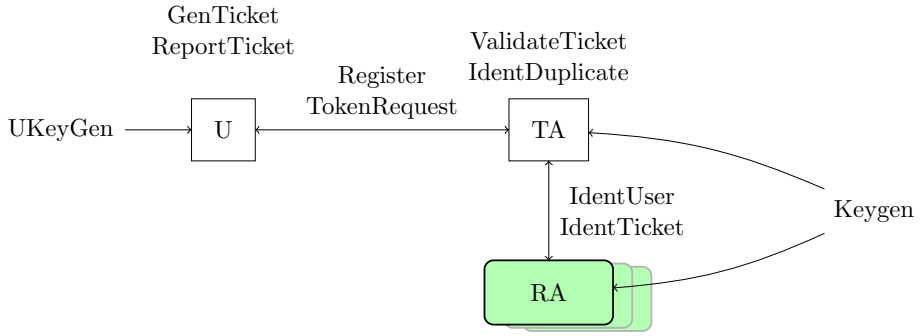


FIGURE 7.1: M-Ticketing Framework Overview

Initialization

`Setup`(1^λ): This probabilistic algorithm outputs pp a description of the system parameters. We assume that pp are implicit to the other algorithms, and that they include λ, the security parameter, and max_{ticket}, the number of m-tickets that each book of m-tickets contains. They are also an implicit input to the adversary, we will then omit them.

`Keygen`(pp): This probabilistic algorithm outputs the two following secret/public key pairs: (rsk, rpk) for the revocation authority and (tsk, tpk) for the transport authority. The system public key gpk is eventually set as (pp, tpk, rpk).

Registration

`UKeygen`(gpk, ID_U): This probabilistic algorithm outputs a secret/public key pair (usk, upk) for a user identifier ID_U.

`Register`($U(ID_U, upk), TA(DB_{REG})$): This is an interactive protocol between a new user that takes as input his identity ID_U and his public key upk, and the TA that takes as input the database DB_{REG} where the identifiers of the registered users will be stored. If TA accepts the protocol, the user's identity and public key are stored within DB_{REG}.

Permission token request

`TokenRequest`(U(upk, gpk), TA(tsk, gpk, DB_{REG})): This is an interactive protocol between a user that takes as input (upk, gpk), and the TA that takes as input (tsk, gpk, DB_{REG}). If the user accepts the protocol, his output is a permission token τ that will enable him to generate/validate a book of max_{ticket} m-tickets. If TA accepts the protocol, its output is a transcript $view$ of the protocol.

Validation

`GenTicket`(gpk, τ): This probabilistic algorithm takes as input a user's permission token τ and outputs an m-ticket $Tick_k$ with a serial number B_k such that $k \in [1..max_{ticket}]$.

`ValidateTicket`($gpk, Tick_k$): This deterministic algorithm takes as input a ticket $Tick_k$. If $Tick_k$ is valid, it outputs 1 and $Tick_k$ is stored within the database $DB_{UsedTickets}$ that will be used to detect the m-tickets that have been used several times. Otherwise, it outputs 0.

Revocation

`IdentUser`(rsk, DB_{REG}, $Tick_k$): This deterministic algorithm takes as input the private key of the revocation authority, rsk, the registration database of TA, DB_{REG}, and a valid m-ticket $Tick_k$. It outputs the identifier ID_U of the registered user who obtained $Tick_k$. If ID_U does not belong to DB_{REG}, it outputs $\perp$.

`IdentTicket`(rsk, $view$, ID_U): This deterministic algorithm takes as input the private key rsk, a user's identifier ID_U and a transcript $view$ of an execution of `TokenRequest` with this user. It outputs all the m-tickets that can be generated from the token obtained after the execution of the `TokenRequest` protocol that led to $view$.

Reporting

`ReportTicket`(τ): This algorithm is executed by a user with his permission token τ. The user generates all the unused m-tickets and collects them in a *usage report* R. R is then sent to the transport authority.

`IdentDuplicate`($B_k, DB_{UsedTickets}$): This deterministic algorithm takes as input B_k the serial number of a valid m-ticket $Tick_k$ and $DB_{UsedTickets}$, and outputs the number of occurrences of B_k in $DB_{UsedTickets}$.

7.3 Untraceable Mobile Ticketing System

At the beginning, as shown in Figure 7.2, the user registers at the transport service. Then, he retrieves a permission token A from the transport authority. This token enables the generation of max_{ticket} m-tickets (an m-ticket book[1]). The token A is a BB signature on a secret s known only by the user. The secret s identifies a type of m-tickets book.

At the validation phase, the user authenticates the validator and sends an m-ticket $Tick_k$: (B_k, E, Π). B_k is a unique serial number of the m-ticket, E is an ElGamal encryption of g_1^s, and Π is a ZKPK. Π proves that (1) the user has a valid BB signature on s without revealing neither s nor the signature A, (2) E is an encryption of g_1^s without revealing s and (3) k belongs to $[1..max_{ticket}]$ without revealing k. (3) is based on the new set-membership proof detailed in Chapter 4.

Finally, the user post-pays by regularly reporting unused m-tickets, i.e., by revealing the unused m-ticket serial numbers B_k to the TA. This enables to detect any malicious behaviour, i.e., validating more m-tickets than what were obtained, without breaking the user's privacy. Additional countermeasures allows to recover the user's identity and retrieve his m-tickets, with the agreement of the authorities.

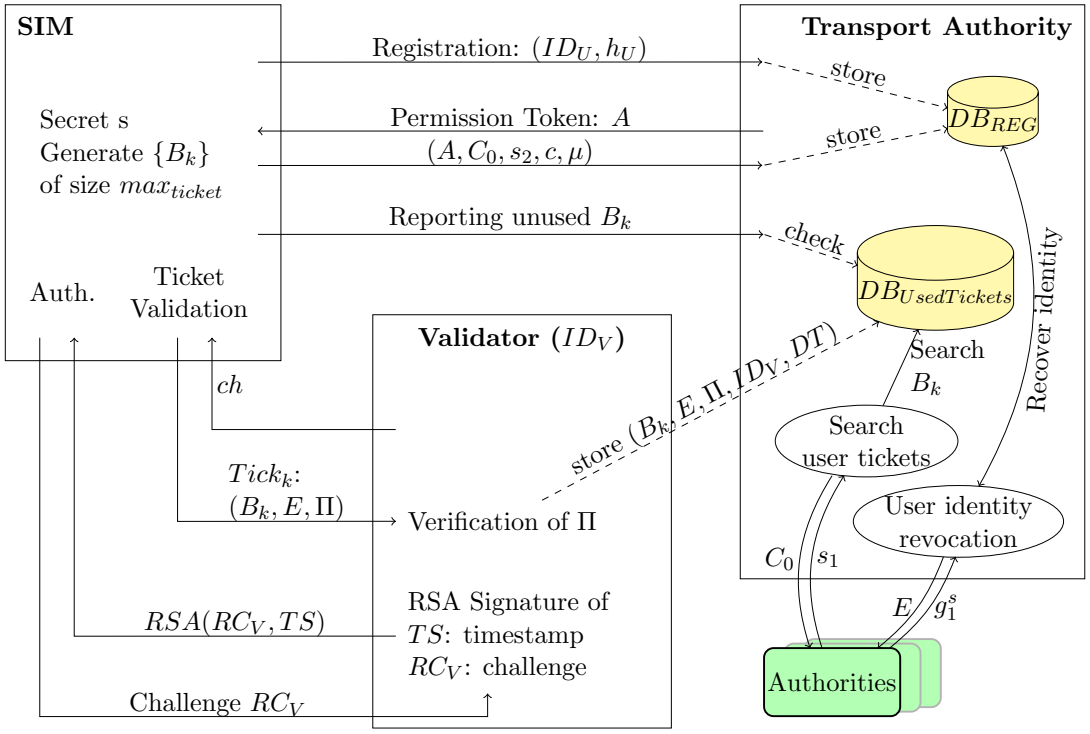


FIGURE 7.2: The M-Ticketing Protocol Overview

[1]We assume that m-tickets have a unique rate per area such as in Paris [107], Berlin [108] and Moscow [109] undergrounds. The user can have different types of m-tickets books such that every book is characterized by a rate, an area and the number of available m-tickets (max_{ticket}), e.g., a book of 10 m-tickets valid in area 1 and the price of one m-ticket is 1.30€.

7.3.1 M-ticketing Protocol

Our protocol is divided in four phases: system initialization, user registration, permission token request and validation. The later includes the validator authentication and the m-ticket validation.

System Initialization In the sequel, we consider, except when it is explicitly mentioned, bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ where all groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative and of prime order p. Let g, g_0, g_1, g_t, g_T, g_U, h, G, H be nine generators of $\mathbb{G}_1$ and g_2, g_3 two generators of $\mathbb{G}_2$. The user's SIM card will perform computations only in $\mathbb{G}_1$ (note that current SIM cards are not designed to handle pairing computations nor computations in $\mathbb{G}_2$ or $\mathbb{G}_T$.).

The revocation authorities set two pairs of keys: a private and public keys of the threshold ElGamal cryptosystem (pk_{RG}, sk_{RG}) and a private and public keys of the threshold Paillier cryptosystem (pk_{RP}, sk_{RP}).

We assume that the private keys are shared among the revocation authorities (e.g., using the technique proposed in [110] for El Gamal and the technique proposed in [73] for Paillier). In order to decrypt a ciphertext, a minimal number t (the threshold) of these authorities should cooperate.

The public key, pk_{RG}, consists of the elements $(g_T, h_T = g_T^{x_T})$, where g_T is a random generator of $\mathbb{G}_1$, and the corresponding private key, sk_{RG}, is formed by $x_T \in \mathbb{Z}_p^*$. Let a and b be random primes for which $a, b > 2, a \neq b, |a| = |b|$ and $gcd(ab, (a-1)(b-1)) = 1$; let $n = ab$, $\Pi = lcm(a-1, b-1)$, $K = \Pi^{-1}\ mod\ n$, and $g_P = (1+n)$; then the public key, pk_{RP}, is (n, g_P) and the secret key, sk_{RP}, is (a, b).

The transport authority (TA) sets for each group, i.e., a type of m-tickets book, a key pair of the BB signature scheme. The private key is a random element $\gamma \in \mathbb{Z}_p^*$ and the corresponding public key is $W' = g_0^{\gamma}$ and $W = g_2^{\gamma}$. The transport authority sets another key pair of the BB signature scheme which will be used for the set-membership proof. The private key is a random element $y \in \mathbb{Z}_p^*$ and the corresponding public key is $Y = g_3^{y}$.

The user owns a public/private key pair $(x_U \in \mathbb{Z}_p^*, h_U = g_U^{x_U})$. During the permission token request, the user obtains from TA a token $A = (g_1^s h)^{1/(\gamma+r)}$. The value s is jointly chosen by TA and the user (but only known by the user) whereas r is known by both entities.

User Registration We denote by ID_U a user's identity and DB_{REG} the database where the TA saves the identities of registered users. First, the user sends his public key h_U and his identity ID_U to TA. Then, he signs, using Schnorr signature scheme [111], a random challenge rc received from TA, using his private key x_U. If the signature is valid, TA saves the tuple (ID_U, h_U) in DB_{REG}. Then, the user securely introduces his banking credentials in order to be subsequently charged.

Permission Token Request The permission token request phase is detailed in Figure 7.3. A permission token $(A = (g_1^s h)^{1/(\gamma+r)}, r)$ consists of an (extended) BB-signature [77] on s (the secret member group key only known by the user, whereas r is

known by the user and TA). Thanks to his permission token, the user will be able to use max_{ticket} m-tickets. The value of max_{ticket} is set by TA and linked to the key pair $(\gamma, W = g_2^{\gamma})$ used by TA during the permission token request protocol. TA will use a different pair of keys for each possible value of max_{ticket}, i.e., for each group associated to each type of m-tickets book.

At the end of this phase, TA saves in DB_{REG} the *view* (A, C_0, s_2, c, μ) where C_0 is the Paillier encryption of s_1 such that the secret $s = s_1 + s_2$ and (c, μ) are the commitment and signature by the user of his secret s.

User		**Transport Authority**
Public input: The public parameters **Private Input:** x_U		**Public Input:** The public parameters **Private Input:** γ, DB_{REG}
Choose $s_1 \in \mathbb{Z}_p^*$, $j \in]0, n[$ **Compute:** $Com = g_1^{s_1}$; $C_0 = g_P^{s_1} j^n \ mod\ n^2$		
$\Pi_1 = POK(s_1, j : Com = g_1^{s_1} \wedge C_0 = g_P^{s_1} j^n \ mod\ n^2)$	$\xrightarrow{Com, \Pi_1, C}$	**Verify** Π_1
		Choose $s_2 \in \mathbb{Z}_p^*$ **Let** $s = s_1 + s_2 \ mod\ p$, then $Com \times g_1^{s_2} = g_1^{s_1} g_1^{s_2} = g_1^{s_1+s_2} = g_1^s = c$ **Choose** $r \in \mathbb{Z}_p^*$ **Compute:** $A = (g_1^s h)^{1/(\gamma + r)}$
Verify Π_2 **Compute:** $c = g_1^{s_1} g_1^{s_2} = g_1^{s_1+s_2} = g_1^s$; $\mu = Sign(x_U, A, g_1^{s_1}, g_1^{s_2})$	$\xleftarrow{A, r, g_1^{s_2}, \Pi_2}$	$\Pi_2 = POK(\gamma : W' = g_0^{\gamma} \wedge A^{\gamma} = c \times hA^{-r})$
Compute: $s = s_1 + s_2$	$\xrightarrow{\mu}$ $\xleftarrow{s_2}$	**Verify** μ **Save** (A, C_0, s_2, c, μ) associated to (ID_U, h_U) in DB_{REG}

FIGURE 7.3: The Protocol of a Permission Token Request

Validation In this phase, the validator is authenticated, the permission token is verified and finally the m-ticket is validated.

Validator Authentication

The validator authentication consists of a challenge / response protocol. The user sends a Random Challenge RC_V to the validator gate. Upon receiving RC_V, the validator replies with the RSA signature on a timestamp (TS) and the received challenge (we use a short public verification exponent v in order to have a fast verification in the SIM card).

Then, the m-ticketing cardlet checks the received signature. If it succeeds, the m-ticketing cardlet checks the validity of the permission token based on the timestamp (TS) sent by the validator. Indeed, if TS is lower than $\mathcal{D}$ $(TS < \mathcal{D})$, the permission token is still valid. Then, the cardlet checks whether the number of used m-tickets reached max_{ticket}, the number of authorized post-paid m-tickets. If a check fails, the m-ticketing cardlet aborts and the m-ticketing application displays a message to ask the user to renew the permission token.

M-ticket Validation

An m-ticket $Tick_k$ (indexed k) is characterized by a serial number $B_k = g_t^{1/(s+k+1)}$ along with an ElGamal encryption $E = (C_1 = g_T^a, C_2 = g_1^s \times h_T^a)$ of g_1^s. To prove the validity of a m-ticket, the user must prove that:

- k belongs to $\Phi = [1..max_{ticket}]$ using our new set-membership proof (described in Chapter 4),
- He knows a valid BB signature $A = (g_1^s h)^{1/(\gamma+r)}$ on s (without revealing s and the signature),
- E is an encryption of g_1^s.

Therefore a m-ticket $Tick_k$ is represented by (B_k, E, Π) where $\Pi = POK(k, a, s, r, A : B_k = g_t^{1/(s+k+1)} \wedge A = (g_1^s h)^{1/(\gamma+r)} \wedge C_1 = g_T^a \wedge C_2 = g_1^s \times h_T^a \wedge k \in [1..max_{ticket}])$.

Remark 2 (BB optimizations): Proving the knowledge of a valid BB-signature $A = (g_1^s h)^{1/(\gamma+r)}$ on a value s, without revealing s or the signature and without using pairings on the prover's side can be done as follows:

The prover first randomizes A by choosing a random value $\alpha \in \mathbb{Z}_p^*$ and computes $B_0 = A^\alpha$. Since $A = (g_1^s h)^{1/(\gamma+r)}$ this implies that $B_0 = A^\alpha = (g_1^{\alpha s} h^\alpha)^{1/(\gamma+r)}$ and then that:

$$B_0^{\gamma+r} = g_1^{\alpha s} h^\alpha \tag{7.1}$$

Let us note by $B' = B_0^{-1}$ and $C = B_0^\gamma$. From (7.1), we have:

$$C = g_1^{\alpha s} \times h^\alpha \times B'^r \tag{7.2}$$

As a result, in order to prove that he knows a BB-signature A on the value s, without revealing s nor the corresponding signature, the prover just has to send B_0 and C (that he can compute using (7.2)) to the verifier and prove that he knows the representation of C with respect to (g_1, h, B'): $\Pi_{BB} = POK(\alpha, s, r : C = g_1^{\alpha s} \times h^\alpha \times B'^r)$. The proof consists in B_0, C and Π_{BB} (and no pairing computations are needed on the prover's side to compute Π_{BB}). The verifier will have to check that $B_0 \neq 1_{\mathbb{G}_1}$, that $C = B_0^\gamma$ (via pairing computations or by using the key γ if it owns this key) and that Π_{BB} is valid. If all the verifications hold, the verifier will be convinced that the prover knows a valid BB-signature.

As described in Figure 7.4, the validator verifies the proof Π, saves the date and time of the operation DT, the serial number of the validated m-ticket B_k, the El Gamal encryption E (of g_1^s) and the proof Π. These verifications can be done in two ways. In the first case, the validator holds the private keys γ (the private key of TA) and y (the private key used during the set-membership[2]). Hence, he can perform the verification of Π without executing any pairing computations. In such a case, the protocol is run without any pairing computations either on the user side (SIM card) or on the validator side. In the second case, the validator does *not* hold the private keys γ and y. Therefore, in order to perform the verification of Π, the validator would execute pairing computations. We still achieve our goal, i.e., no pairing computations at the user side (SIM card).

[2]This situation does not question the security of our protocol because the validator belongs to the transport authority trusted area.

M-ticketing cardlet	**Validator**
Public input: The public parameters and the public keys of the revocation and transport authorities h_T, W, Y	**Public Input:** The public
The sets $\Phi = [1..max_{ticket}]$ and $\sum = \{A_1, A_2, ..., A_{max_{ticket}}\}$ where $A_i = g^{1/(y+i)}$ for $i \in \Phi$	The sets Φ and $\sum$
Private Input: $A = (g_1^s h)^{1/(\gamma+r)}$, k(The index of the ticket that will be used), s	**Private Input:** The private signature keys γ and y (in some scenario)
$\mathcal{D}$ day: validity end date of the permission token	

Pre-computations:

Compute $B_k = g_t^{1/(s+k+1)}$
Choose $a \in \mathbb{Z}_p^*$ and **Compute:** $C_1 = g_T^a, C_2 = g_1^s \times h_T^a$
Computation of elements involved in the proof that the user knows a valid signature on s
Choose $\alpha \in \mathbb{Z}_p^*$ and **Compute (see remark 2):** $B_0 = A^\alpha$; $B' = B_0^{-1}$; $C = g_1^{\alpha s} \times h^\alpha \times B'^r$
Choose $r_2, r_3, r_4, a_1, d_1, b_1, \alpha_1, t, t_1, t_3, t_4, t_5, \nu, d', f' \in \mathbb{Z}_p^*$
Compute: $T' = G^s H^{r_2}$; $\beta = \alpha s$; $T'' = T'^\alpha H^{r_3}$; $r_5 = r_3 + \alpha r_2 (mod\ p)$; $Com = g_1^k h_T^\nu$
Pick the valid BB signature corresponding to the element k: $A_k = g^{1/(y+k)}$
Choose $l \in \mathbb{Z}_p^*$ and **Compute:** $B = A_k^l$; $B_1 = B^{-1}$; $D = B_1^k g^l$
Choose $k_1, l_1, r_1 \in \mathbb{Z}_p^*$ and **Compute:** $Com_1 = g_1^{k_1} h_T^{r_1}$; $D_1 = B_1^{k_1} g^{l_1}$
$\delta = s + k$; $f = a + \nu$; $K = g_t B_k^{-1} = B_k^{s+k} = B_k^\delta$; $L = C_2 Com = g_1^{s+k} h_T^{a+\nu} = g_1^\delta h_T^f$
Computation of the witnesses
Compute: $C_1' = g_T^{a_1}$; $C_2' = g_1^{d_1} \times h_T^{a_1}$; $R' = G^{d_1} H^{t_1}$; $R'' = T'^{\alpha_1} H^{t_3}$;
$C' = g_1^{b_1} h^{\alpha_1} B'^t$; $T_4' = G^{b_1} H^{t_5}$; $K' = B_k^{d'}$; $L' = g_1^{d'} \times h_T^{f'}$

Real time computations

Validator authentication

Choose $RC_V \in \mathbb{Z}_p^*$ and **Check** that the number of used m-tickets $< max_{ticket}$ —RC_V→ TS = getTimeStamp() and **Compute:** $Signature_{RSA}$, the RSA signature on RC_V

Verify $Signature_{RSA}$ and **Check** that $TS < \mathcal{D}$ ←$Signature_{RSA}$— and TS

m-ticket Validation

Compute: c= $H(C, C_1, C_2, B_0, T', T'', Com, B, D, K, C_1', C_2', R', R'', C', T_4', Com_1, D_1, K', L', ch)$ ←ch— **Choose** $ch \in \mathbb{Z}_p^*$
$s_1 = k_1 + c \times k\ mod\ p$; $s_2 = r_1 + c \times \nu\ mod\ p$
$s_3 = l_1 + c \times l\ mod\ p$; $\omega_1 = a_1 + c \times a\ mod\ p$
$\omega_2 = d_1 + c \times s\ mod\ p$; $\omega_3 = t_1 + c \times r_2\ mod\ p$
$\omega_4 = b_1 + c \times \beta\ mod\ p$; $\omega_5 = \alpha_1 + c \times \alpha\ mod\ p$
$\omega_6 = t + c \times r\ mod\ p$; $\omega_8 = t_3 + c \times r_3\ mod\ p$
$\omega_{10} = t_5 + c \times r_5\ mod\ p$; $\omega_{11} = d' + c \times \delta\ mod\ p$
$\omega_{12} = f' + c \times f\ mod\ p$

Let $E = (C_1, C_2)$ and the proof $\Pi = (C, B_0, T', T'', Com, c, B, D, s_1, s_2, s_3, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_8, \omega_9, \omega_{10}, \omega_{11}, \omega_{12})$ —B_k, E, Π→ **Check** that $B \neq 1_{\mathbb{G}_1}$

- If The validator holds the private signature keys y and γ **(First case)**, the prover does not send the value D and C. The verifier can compute it: $D = B^y$, $C = B_0^\gamma$. Then goes to (*)
- Otherwise, if the validator doesn't know the private signature key y and γ **(Second case),** then, check that $e(D, g_3) = e(B, Y)$; $e(C, g_2) = e(B_0, W)$ and goes to (*)

(*) Compute: $\tilde{C}$=$g_1^{s_1}\ h_T^{s_2}\ Com^{-c}$; $\tilde{D}$=$B_1^{s_1}\ g^{s_3}\ D^{-c}$; $\tilde{C}_1$=$g_T^{\omega_1}\ C_1^{-c}$; $\tilde{C}_2$=$g_1^{\omega_2}\ h_T^{\omega_1}\ C_2^{-c}$; $\tilde{R}'$=$G^{\omega_2}\ H^{\omega_3}\ T'^{-c}$; $\tilde{C}'$=$g_1^{\omega_4}\ h^{\omega_5}\ B'^{\omega_6}\ C^{-c}$; $\tilde{R}''$=$T'^{\omega_5}\ H^{\omega_8}\ T''^{-c}$; $\tilde{T}_4$=$G^{\omega_4}\ H^{\omega_1 0}\ T''^{-c}$; $\tilde{K}'$=$B_k^{\omega_1 1}\ K^{-c}$; $\tilde{L}'$=$g_1^{\omega_1 1}\ h_T^{\omega_1 2}\ L^{-c}$

Check c= $H(C, C_1, C_2, B_0, T', T'', Com, B, D, K, \tilde{C}_1, \tilde{C}_2, \tilde{R}', \tilde{R}'', \tilde{C}', \tilde{T}_4', \tilde{C}, \tilde{D}, \tilde{K}', \tilde{L}', ch)$

Send (B_k, E, Π, ID_V, DT) to TA in order to be saved within the secured database $DB_{UsedTickets}$; ID_V is the identity of the validator and DT is the date and time of the transaction.

FIGURE 7.4: The validation Protocol

We emphasize that owing to our improvements on Boneh-Boyen based Camenisch-Lysyanskaya signature scheme, all the computations (required to validate an m-ticket) are performed by the SIM card. We do not need to outsource part of the computations to the powerful but untrusted mobile phone. Consequently, the user's privacy is ensured with respect to the smartphone itself. Even a compromised mobile, e.g., containing a spyware, cannot collect any information about our m-ticketing application computations.

At the end of a successful m-ticket validation, the validator sends (B_k, E, Π) to TA in order to be saved within a centralized and secured database $DB_{UsedTickets}$ jointly with his identity ID_V and the date and time of the transaction DT. In such a way, the TA will detect any malicious behaviour such that a multiple usage or cloning (cf. Paragraph "Countermeasures").

Theorem 6. The protocol in Figure 7.4 is a ZKPK of a permission token (A, r, s) and a value k such that $B_k = g_t^{1/(s+k+1)}$, $k \in [1..max_{ticket}]$ and $E = (C_1, C_2)$ is an El Gamal encryption of g_1^s.

Proof 6 (Sketch proof). As mentioned in Chapter 3, a zero knowledge proof of knowledge must be complete, sound and zero-knowledge.

The *completeness* follows by inspection.

Soundness: roughly speaking, the whole proof Π can be divided in three sub-proofs Π_1, Π_2 and Π_3, where:

$\Pi_1 = POK(\alpha, s, r, r_2, r_3, r_5, a : C = g_1^{\alpha s} \times h^{\alpha} \times B'^{r} \wedge T' = G^s H^{r_2} \wedge T'' = T'^{\alpha} H^{r_3} = G^{\alpha s} H^{r_5} \wedge C_1 = g_T^a \wedge C_2 = g_1^s \times h_T^a)$

$\Pi_2 = POK(k, \nu, s, r_2 : Com = g_1^k h_T^{\nu} \wedge T' = G^s H^{r_2} \wedge K = g_t B_k^{-1} = B_k^{s+k})$

$\Pi_3 = POK(k, \nu, l : Com = g_1^k h_T^{\nu} \wedge D = B_1^k g^l)$

If the verifier accepts the proof Π this means that: $D = B^y$ (1) and $C = B_0^{\gamma}$ (2).

Note that the proofs Π_1, Π_2 and Π_3 are classical variants of Schnorr's proof of knowledge. Using the "extractors" of these proofs of knowledge, we can retrieve k, α, s, r, l.

From (1) we have: $D = B^y = B_1^k g^l$. This implies that $B^y B^k = g^l$. Let us denote by $A_k = B^{1/l}$. (Note that $l \neq 0$, otherwise this would imply that the prover knows the secret value y). We therefore have $A_k^{y+k} = g$ and therefore that $A_k = g^{1/(y+k)}$. So the prover knows a valid BB signature on the value k. This implies that $k \in [1..max_{ticket}]$.

From (2) we have that $C = B_0^{\gamma} = g_1^{\alpha s} \times h^{\alpha} \times B'^{r}$. This implies that $B_0^{\gamma+r} = g_1^{\alpha s} \times h^{\alpha}$. Let us denote by $A = B_0^{1/\alpha}$, this implies that $A^{\gamma+r} = g_1^s \times h$ (Note that $\alpha \neq 0$, otherwise this would imply that the prover knows the secret value γ). So $A = (g_1^s h)^{1/(\gamma+r)}$. The prover therefore knows a valid permission token (A, r, s).

In conclusion, the prover knows a permission token (A, r, s) and a value k such that $B_k = g_t^{1/(s+k+1)}$, $k \in [1..max_{ticket}]$ and $E = (C_1, C_2)$ is an El Gamal encryption of g_1^s.

(Honest-verifier) Zero-Knowledge: since Π_1, Π_2 and Π_3 are classical ZKPK, we can easily construct simulators $Sim1$, $Sim2$ and $Sim3$ (respectively) for such proofs. From these simulators, it is straightforward to construct a simulator Sim that will simulate all interactions with any (honest) verifier V^*. Since $\mathbb{G}_1$ is a prime order group, then the

blinding would be perfect and the output of Sim and V^*'s view of the protocol would be statistically indistinguishable.

Revocation We distinguish two levels of revocation: the user's anonymity revocation and the m-tickets revocation. In the first case, the transport authority would like to get the user's identity corresponding to a given m-ticket. In the second case, the transport authority would like to revoke all the m-tickets of a given user, e.g., upon the request of the user further to the theft of his smartphone.

In order to recover a user's identity based on a used m-ticket $Tick_k = (B_k, E, \Pi)$, TA sends E to the revocation authorities. Together, the revocation authorities decipher E and retrieve the commitment of s (g_1^s). By using DB_{REG} the database of registered users and g_1^s, TA will find the user's identity in the tuples $(A, C_0, s_2, c, \mu, ID_U, h_U)$.

In order to retrieve the m-tickets of a user based on his identity $ID_{U'}$, first of all, TA must find the tuple $(A, C_0, s_2, c, \mu, ID_U, h_U)$ from DB_{REG} such that $ID_U = ID_{U'}$. Then, TA sends C_0 (The Paillier encryption of s_1) to the revocation authorities. Similarly to the first case, the revocation authorities decipher together C and retrieve s_1. Upon receiving s_1, TA computes $s = s_1 + s_2$, hence, it can retrieve all the m-tickets ($B_k = g_t^{1/(s+k+1)}$) of the user.

7.3.2 A Secure Post-Payment Process

One novelty of our m-ticketing protocol, in addition to the respect of users' privacy, is to give the ability to charge a user after the usage of m-tickets.

Regular Reporting In a post-payment approach, the m-ticketing application must report unused m-tickets to the back-end server, before the pre-defined D day. Thus, the regular reports does not question the privacy of the user. The following example gives further clarification. Suppose that the user retrieved a permission token of 5 m-tickets. Before D day, he used 4 m-tickets, i.e., m-tickets number 1, 2, 3 and 4. On D day, the m-ticketing application will report to the back-end server the m-ticket number 5 using a network connection. The report contains $B_5 = g_t^{1/s+5+1}$ and the proof $\Pi = POK(k, s, r, A : B_5 = g_t^{1/(s+5+1)} \wedge A = (g_1^s h)^{1/(\gamma+r)} \wedge 5 \in [1..5])$.

Regularly revealing the unused m-tickets enables the transport authority to (1) charge the user, (2) check the reliability and accuracy of the reports without questioning the user's privacy. Indeed, the unlinkability of user's m-tickets ($B_k = g_t^{1/(s+k+1)}, E, \Pi$), both during the use of the service and during the reporting phase, is ensured owing to the q-DDHI and DDH assumptions (cf. *unlinkabililty* proof in Section 7.5).

Countermeasures A malicious user's goal simply consists on not paying or paying less than what he is supposed to. To do so, he may try to block the reporting or attack the cardlet of the SIM.

If the user blocks network communications in order to block reporting, the SIM cardlet will refuse to issue m-tickets for a permission token after reaching the limit of max_{ticket},

or the limit of the time window controlled by $\mathcal{D}$. This countermeasure C relies on the security of the SIM card.

If the user performs a successful physical attack against the SIM card, which is extremely difficult [24], he may try to (1) defeat the countermeasures C and never do reports, or (2) use the same m-ticket several times or (3) report a used m-ticket. In the first case, the TA will notice that the user did not respect the D day limitation (reporting his usage before the D day). In the other cases, this will be detected in $DB_{UsedTickets}$ (two m-tickets with the same serial number). TA can, then, decide to break the user's anonymity and retrieve all his m-tickets usage history.

Thus, the user is forced to report his consumption and renew his permission token lest the service is unusable or TA breaks his anonymity.

Note that forging an m-ticket is not possible, even with a compromised SIM because of the *unforgeability* property.

7.4 Requirements

Similarly to the m-pass system, we consider three types of requirements: *functional* requirements which are "efficiency" and "versality", *security* requirements which consist in "correctness", "unforgeability" and "Non-frameability", and *privacy* requirements which comprises "unlinkability".

Obviously, the two transport system use cases, namely m-pass and m-ticketing, share the same requirements. However, as shown in what follows, these requirements are differently interpreted.

7.4.1 Functional Requirements

Efficiency Similarly to the m-pass system, a m-ticketing system must fulfil functional requirements imposed by transport operators [7], in particular the validation of an m-ticket must be performed in less than $300ms$. We must consider that a SIM card has limited computation capabilities. In particular, pairing APIs are not available on current SIM cards.

Versatility The mobile phone and the validator cannot be assumed to be connected to a back-end server during the validation phase. This enables the user to use an m-ticket in any kind of situation, especially in areas with low connectivity, e.g., underground, or if the battery of his mobile is flat. Moreover, the m-ticketing system must support both pre-payment and post-payment modes, i.e., an m-ticket can be prepaid or charged later (after its use and before a given date that we denote later by D day). In the post-payment mode, the transport authority must learn the total number of m-tickets used by each customer in order to charge them. This is not an easy task especially for anonymous and unlinkable m-tickets.

7.4.2 Security and Privacy Model

Besides the correctness property, we formally define three security and privacy properties of our m-ticketing protocol in which the attack capabilities of a probabilistic polynomial time adversary $\mathcal{A}$ are modeled by providing him access to some oracles. In the sequel, $\mathcal{HU}$ will denote the set of honest users and $\mathcal{MU}$ the set of corrupted users. We assume that $\mathcal{A}$ receives all the exchanged messages in our system. $\mathcal{A}$ acts as an active adversary as regards to the messages issued by malicious users and as a passive adversary with respect to honest users.

$\mathcal{O}Register_{HU}$ is an oracle that will be used by an adversary in order to register honest users. By calling this oracle with ID_U as argument, the adversary adds a new user. The oracle runs $(upk, usk) \leftarrow \texttt{UKeygen}(gpk, ID_U)$ and adds ID_U (along with upk) to the set $\mathcal{HU}$. The private key usk is kept secret and public key upk is returned to the adversary.

$\mathcal{O}Register_{MU}$ is an oracle that will be used by an adversary in order to register malicious users. The adversary calls this oracle with argument the identifier ID_U of a user and sets his public key to upk and his private key to usk. The identity ID_U (along with upk) is added to the set $\mathcal{MU}$.

$\mathcal{O}CorruptUser$ is a user secret key oracle enabling the adversary to obtain the private key usk of a user $ID_U \in \mathcal{HU}$. The oracle transfers ID_U to $\mathcal{MU}$ and returns usk.

$\mathcal{O}TokenRequest_U$ is an oracle that runs the user's side in the `TokenRequest` protocol. This oracle will be used by an adversary playing the role of a malicious TA. The adversary gives to the oracle an identity ID_U of an honest user and his public key upk. If the user accepts the protocol, the adversary is given a transcript view of the protocol.

$\mathcal{O}TokenRequest_T$ is an oracle that runs the transport authority side in the `TokenRequest` protocol. This oracle will be used to simulate the execution of the protocol between a user (corrupted or not) and an honest TA.

$\mathcal{O}GenTicket(ID_U, view)$ is an oracle that takes as input the identifier ID_U of an honest user and a transcript $view$ of an execution of the `TokenRequest` protocol with this user and outputs an m-ticket $Tick_k$ using a fresh index k that has not been used in a previous query of $\mathcal{O}GenTicket$ on ID_U and $view$. The oracle records $(ID_U, Tick_k)$ in a list Set.

$\mathcal{O}IdentTicket_T(ID_U, view)$ is an oracle that takes as input the identifier of a user ID_U and a transcript $view$ of an execution of `TokenRequest` with this user and outputs all the m-tickets that this user is able to generate.

$\mathcal{O}IdentUser_T(Tick_k)$ is an oracle that returns the identifier ID_U of the user who generated an m-ticket $Tick_k$.

$\mathcal{O}ReportTicket(ID_U, view)$ is an oracle that takes as input the identifier ID_U of an honest user and a transcript $view$ of a `TokenRequest` execution with this user and outputs the set of unused m-tickets. For each unused m-ticket $Tick_k$, the oracle records $(ID_U, Tick_k)$ in Set.

Correctness Informally speaking, our protocol is correct if (1) a valid permission token enables to generate valid m-tickets, (2) honestly generated m-tickets are accepted, (3) a validated m-ticket enables revocation authorities to identify the user who generated

it and (4) revocation authorities can retrieve all the m-tickets generated by a given registered user that obtained a valid permission token.

Unforgeability Informally speaking, it should be impossible for anyone (1) to validate more m-tickets than what he obtained i.e. an adversary who retrieved N tokens τ (N books of max_{ticket} m-tickets) should not be able to generate more that $N * max_{ticket}$ m-tickets; (2) to validate m-tickets such that the algorithm `IdentUser` returns $\perp$, i.e., an identifier ID_U that doesn't appear in DB_{REG}. We formally define the unforgeability experiment $\mathbf{Exp}_{\mathcal{A}}^{unforg}(1^{\lambda})$ in Figure 7.5. The scheme is unforgeable if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability $\mathtt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{unforg}(1^{\lambda}) = 1]$ is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{unforg}(1^{\lambda})$

1. $pp \leftarrow Setup(1^{\lambda})$; $\mathcal{HU} \leftarrow \varnothing$; $\mathcal{MU} \leftarrow \varnothing$
2. $(gpk, tsk, rsk) \leftarrow$ `Keygen`(pp)
3. $(\{Tick_{k_j}^j\}_{j=1}^{j=l}, \{R_i\}_{i=1}^{i=f}) \leftarrow \mathcal{A}(gpk$: $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}TokenRequest_T$, $\mathcal{O}GenTicket$, $\mathcal{O}Report\ Ticket)$. An R_i corresponds to a "usage report", i.e., a set on unused m-tickets.
4. Let DB be an empty database.
5. For j from 1 to l do {If `ValidateTicket`$(gpk, Tick_{k_j}^j)$ then store $Tick_{k_j}^j$ in DB}.
6. For i from 1 to f do {Validate the m-tickets of the report R_i and store valid unused m-tickets in DB}
7. For all $Tick_k$ in DB do {$b =$ `IdentDuplicate`(B_k, DB) where B_k is the serial number of the m-ticket $Tick_k$
   If b¿1, then delete all the duplicates of the m-ticket $Tick_k$.
   If `IdentUser`$(rsk, DB_{REG}, Tick_k)$ outputs $\perp$ then return 1 and aborts. }
8. If L the number of m-tickets that remained within DB is greater that $N * max_{ticket}$ ($L > N * max_{ticket}$) where N is the number of calls of the oracle $\mathcal{O}TokenRequest_T$ and max_{ticket} is the number of authorized m-tickets by token, then return 1 else return 0

FIGURE 7.5: Unforgeability Security Experiment

Non-Frameability Informally speaking, it should be impossible for anyone to falsely accuse an honest user of having spent an m-ticket. We formally define the non-frameability experiment $\mathbf{Exp}_{\mathcal{A}}^{Nfra}(1^{\lambda})$ in Figure 7.6. The scheme is non-frameable, if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability
$\mathtt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{Nfra}(1^{\lambda})=1]$ is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{Nfra}(1^{\lambda})$

1. $pp \leftarrow Setup(1^{\lambda})$; $\mathcal{HU} \leftarrow \varnothing$; $\mathcal{MU} \leftarrow \varnothing$; $Set \leftarrow \varnothing$
2. $(gpk, tsk, rsk) \leftarrow$ `Keygen`(pp)
3. $(Tick_k) \leftarrow \mathcal{A}(gpk, tsk, rsk, DB_{REG}, DB_{UsedTickets}$: $\mathcal{O}Register_{HU}, \mathcal{O}Register_{MU}, \mathcal{O}CorruptUser$, $\mathcal{O}TokenRequest_U$, $\mathcal{O}Gen\ Ticket$, $\mathcal{O}ReportTicket)$
4. If `ValidateTicket`$(gpk, Tick_k) = 0$ or `IdentUser`$(rsk,\ DB_{REG}, Tick_k) = \perp$ then return 0
5. If `IdentUser`$(rsk, DB_{REG}, Tick_k) = ID_U \in \mathcal{HU}$ and $(ID_U, Tick_k) \notin Set$ then return 1 else return 0.

FIGURE 7.6: Non-frameability Security Experiment

Unlinkability Informally speaking, it should be impossible, except for the revocation authorities, to trace the m-tickets obtained by a user, in particular: (1) to link m-tickets obtained during the permission token request phase to the validated/used ones; (2) to link two m-tickets validated by the same user or to decide whether two validated m-tickets have the same number/index or not; (3) to link validated m-tickets to non-used m-tickets reported by the user to the transport authority. For this, an adversary has full control over the transport authority (in particular it owns the private key tsk) and all the users except two honest users i_0 and i_1. The adversary can initiate the `IdentUser` protocol over any m-ticket and can get the user's identity behind it, except for the m-tickets generated by i_0 and i_1. He can also initiate the `IdentTicket` protocol for all the users except for i_0 and i_1. We define the unlinkability experiment $\mathbf{Exp}_{\mathcal{A}}^{unlink}(1^\lambda)$ in Figure 7.7. The scheme is unlinkable if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage $\mathbf{Adv}_{\mathcal{A}}^{unlink-b}(1^\lambda) = |Pr[\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda) = b] - 1/2|$ is negligible.

$\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda)$

1. $pp \leftarrow Setup(1^\lambda)$; $\mathcal{HU} \leftarrow \varnothing$; $\mathcal{MU} \leftarrow \varnothing$
2. $(gpk, tsk, rsk) \leftarrow$ `Keygen`(pp)
3. $(i_0,\ k_0,\ i_1,\ k_1) \leftarrow \mathcal{A}(gpk,\ tsk,\ DB_{REG},\ DB_{UsedTickets}$: $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}TokenRequest_U$ $\mathcal{O}GenTicket$ $\mathcal{O}IdentTicket_T$ $\mathcal{O}IdentUser_T$ $\mathcal{O}Report\ Ticket)$
4. If i_0 or $i_1 \in \mathcal{MU}$ then output $\perp$.
5. (a) let i_0 and i_1 run the protocol `TokenRequest` and get the permission tokens τ_0 and τ_1 and output $view_0$ and $view_1$.
   (b) $Tick_{k_b} \leftarrow$ `GenTicket`(gpk, τ_0) and $Tick_{k_{1-b}} \leftarrow$ `GenTicket`(gpk, τ_1), with $b \in \{0, 1\}$
6. $b' \leftarrow \mathcal{A}\ (gpk,\ tsk,\ DB_{REG},\ DB_{UsedTickets},\ Tick_{k_j},\ Tick_{k_{1-j}}$: $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}Token\ Request_T$ $\mathcal{O}GenTicket$ $\mathcal{O}IdentTicket_T$ $\mathcal{O}IdentUser_T$ $\mathcal{O}Report\ Ticket)$, with $j \in \{0, 1\}$
7. If $\mathcal{O}CorruptUser$ was requested on i_0 or i_1, or $\mathcal{O}IdentTicket_T$ was requested on $(i_0,\ view_0)$ or $(i_1,\ view_1)$ then output $\perp$.
8. If $\mathcal{O}Ident\ User_T$ was requested for $Tick_{k_j}$ or $Tick_{k_{1-j}}$, output $\perp$.
9. If $\mathcal{O}ReportTicket$ was requested for i_0 or i_1 and i_0 and i_1 did not validate the same number of m-tickets then output $\perp$.
10. Return b'

FIGURE 7.7: Unlinkability Security Experiment

7.5 Security Analysis

In this section, we prove that our m-ticketing protocol provides the security and privacy properties defined in Section 7.4.

Lemma 1. If the q-SDH assumption holds in $\mathbb{G}_1$ then the q-SDH-I assumption holds in $\mathbb{G}_1$.

Proof. See for example [112] for a proof of this Lemma.

Remark: The triplet (r', s', A') corresponds to a permission token of our m-ticketing protocol. In the sequel, we will call the oracle $\mathcal{O}$ a BB-signature oracle and the value s' the permission token secret (or token secret for short).

Forking Lemma. We use the Forking Lemma [101] in our proofs, to prove that an adversary $\mathcal{A}$ is not able to produce a new valid m-ticket $Tick_k$ (respectively a Schnorr's signature μ, see Figure 7.3) unless he knows all the underlying secrets (a, s, r, A, k) (respectively the secret x_U).

Using the notation of [101], if an adversary is able to produce a valid ticket $Tick_k$ $(k, \sigma_1, h, \sigma_2)$ where

σ_1=(C, C_1, C_2, B_0, T', T'', Com, B, D, K, C_1', C_2', R', R'', C', T_4', Com_1, D_1, K', L', ch),

h=H(C, C_1, C_2, B_0, T', T'', Com, B, D, K, C_1', C_2', R', R'', C', T_4', Com_1, D_1, K', L', ch) and

σ_2=(s_1, s_2, s_3, ω_1, ω_2,..., ω_{12})

then it can produce two valid m-tickets $(k, \sigma_1, h, \sigma_2)$ and $(k, \sigma_1, h', \sigma'_2)$ such that $h \neq h'$.

Theorem 7 (Unforgeability). Our m-ticketing protocol satisfies the unforgeability requirement, in the random oracle model, under the q-SDH assumption. The proof is detailed below.

Proof 7 (sketch of proof). Let $\mathcal{A}$ be an adversary who breaks the unforgeability requirement of our m-ticketing protocol with non-negligible probability. We will construct an algorithm $\mathcal{B}$, using $\mathcal{A}$ as an oracle, which breaks the q-SDH-I assumption. $\mathcal{B}$ receives on input from its challenger ($\mathbb{G}_1$, g_0, g_1, g_U h, $W' = g_0^{\gamma}$) the public parameters of the q-SDH-I challenge and has access to a BB-signature oracle. The other generators of the m-ticketing system are randomly chosen.

$\mathcal{B}$ also chooses the keys for the Paillier encryption scheme. It sends W' and the public key of the Paillier scheme to $\mathcal{A}$. The private and public keys of the El Gamal cryptosystem can be chosen either by $\mathcal{A}$ or $\mathcal{B}$. $\mathcal{B}$ is therefore able to construct the public parameters pp for $\mathcal{A}$ and can answer to its requests as follow:

- $\mathcal{O}Register_{HU}$ requests: $\mathcal{B}$ randomly chooses $x_U \in \mathbb{Z}_p$ and computes $h_U = g_U^{x_U}$.
- $\mathcal{O}Register_{MU}$ requests: $\mathcal{B}$ does not need to do anything.
- $\mathcal{O}CorruptUser$ requests: $\mathcal{B}$ gives x_U to $\mathcal{A}$.
- $\mathcal{O}TokenRequest_T$ requests: $\mathcal{B}$ plays as follows: $\mathcal{B}$ first receives a Paillier encryption C_0. It decrypts it (recall that $\mathcal{B}$ chose the private key for the Paillier encryption scheme) and retrieve the corresponding plaintext s_1. It then queries the BB-signature oracle on s where $s = s_1 + s_2$ and s_2 is randomly chosen by $\mathcal{B}$. The BB-signature oracle sends back a pair $(r, A = (g_1^s h)^{(1/(y+r)}$ with $r \in \mathbb{Z}_p$, and $\mathcal{B}$ transmits it to $\mathcal{A}$ along with the value s_2. So $\mathcal{B}$ perfectly simulates the $\mathcal{O}TokenRequest_T$ for $\mathcal{A}$.
- $\mathcal{O}GenTicket(ID_U, view)$ requests: since $\mathcal{B}$ knows the values of all the tokens (A, r, s) that have been issued during $\mathcal{O}TokenRequest_T$ requests, it can easily answer to $\mathcal{O}GenTicket$ queries. The simulation of this oracle is perfect.
- $\mathcal{O}ReportTicket(ID_U, view)$ requests: $\mathcal{B}$ proceeds as in an $\mathcal{O}GenTicket$ request for each unused m-ticket.

We differentiate two types of adversaries:

- **Type-1 Forger**: an adversary that outputs a valid m-ticket $Tick_k$ which cannot be linked to a registered user (corrupted or not).
- **Type-2 Forger**: an adversary that outputs more valid m-tickets than he obtained.

We show that both Forgers can be used to solve the q-SDH-I problem. However, the reduction works differently for each Forger type. Therefore, initially $\mathcal{B}$ chooses a random bit $c_{mode} \in 1, 2$ that indicates its guess for the type of forgery that $\mathcal{B}$ will output.

If $c_{mode} = 1$

Eventually, $\mathcal{B}$ outputs, with non-negligible probability, a valid m-ticket $Tick_{k'} = (B_{k'}, E', \Pi')$ such that the algorithm `IdentUser`, on the input $Tick_{k'}$, returns an unknown identifier ID (i.e., does not appear in DB_{REG}).

First case: $c = g_1^{s'}$ (the plaintext encrypted in E') has been queried during an $\mathcal{O}TokenRequest_T$ request but no corresponding signature μ has been produced. This means that the adversary didn't receive the value s_2 (where $s' = s'_1 + s_2$) from the $\mathcal{O}TokenRequest_T$ oracle. We know from the Forking Lemma and the proofs Π and Π_1 that $\mathcal{A}$ necessarily knows s' and s_2. Since only $g_1^{s_2}$ has been revealed by this oracle during the $\mathcal{O}TokenRequest_T$, $\mathcal{A}$ could be used to extract Discrete Logarithms. Therefore under the DL assumption, this first case could only occur with negligible probability.

Second case: E' is an encryption of a commitment $c = g_1^{s'}$ of a value s' that has not been queried during a $\mathcal{O}TokenRequest_T$ request. Therefore s' has not been queried to the BB-signature oracle either. Using the Forking Lemma and the soundness property of the proof Π_1, $\mathcal{B}$ is able to extract with non-negligible probability the secrets (a', s', r', A', k') underlying the m-ticket $Tick_{k'}$. $\mathcal{B}$ outputs the triplet (r', s', A') to its challenger of the q-SDH-I assumption and therefore breaks it.

If $c_{mode} = 2$

Eventually, $\mathcal{A}$ outputs, with non-negligible probability ξ, $L = N \times max_{ticket} + 1$ valid m-tickets [3] $\{Tick_{k_j}^j\}_{j=1}^{j=L}$, where N is the number of calls to the $\mathcal{O}TokenRequest_T$ oracle, max_{ticket} is the number of authorized m-tickets by token and $Tick_{k_j}^j = (B_{k_j}, E, \Pi)$. Let us denote by ($s_1$, s_2,..., s_N) the N token secrets submitted to the BB-signature oracle. W.l.o.g, we may assume that all these values are different (recall that a token secret s is jointly computed by $\mathcal{A}$ and $\mathcal{B}$). We therefore have $N \times max_{ticket}$ distinct token pairs (s_i, k) with $i \in \{1, ..., N\}$ and $k \in \{1, ..., max_{ticket}\}$. Let Γ be the set containing all these token pairs and $Tick_j^i$ the m-ticket corresponding to the token pair (s_i, j).

Among the L m-tickets output by $\mathcal{A}$, there is at least one m-ticket $Tick_{k*}$ for which the corresponding token pair **$(s*, k*)$ does not belong to** Γ. Otherwise this would mean that two m-tickets among these L m-tickets, e.g., $Tick_{k_1}$ and $Tick_{k_2}$, have the same token pair (since $L > N \times max_{ticket}$. Let us denote by ($s_1^*$, k_1) (respectively (s_2^*, k_2)) the token pair corresponding to $Tick_{k_1}$ (respectively $Tick_{k_2}$). Therefore the serial number $B_{k_1}^*$ of $Tick_{k_1}$ would be equal to $B_{k_2}^*$ the one of $Tick_{k_2}$: $B_{k_1}^* = g_t^{1/(s_1^*+k_1+1)} = g_t^{1/(s_2^*+k_2+1}$

[3] Without loss of generality, we do not make any distinction between a m-ticket and an unused m-ticket.

$= B^*_{k_2}$. This case cannot occur since all duplicates (i.e. m-tickets which have the same serial numbers) are discarded in the experiment $\mathbf{Exp}_{\mathcal{A}}^{unforg}(1^\lambda)$.

Suppose now that $s* \in \{s_1, s_2, \ldots, s_N\}$. Since $(s*, k*) \notin \Gamma$ this implies that $k* \notin \{1, ..., max_{ticket}\}$. Such a case will happen with negligible probability under the q-SDH assumption (see Theorem 2). Therefore $s* \notin \{s_1, s_2, ..., s_N\}$ and consequently has not been queried to the BB-signature oracle ($\mathcal{A}$ is in fact a Type-1 Forger). $\mathcal{B}$ then picks a random m-ticket $Tick_{k'}$ among the L m-tickets output by $\mathcal{A}$. With probability $1/L$, it has chosen $Tick_{k*}$. Using the Forking Lemma and the soundness property of the proof Π, $\mathcal{B}$ is able to extract with non-negligible probability the secrets (a', s', r', A', k') underlying the m-ticket $Tick_{k'}$. $\mathcal{B}$ outputs the triplet (r', s', A') and therefore breaks the q-SDH-I assumption with non-negligible probability ξ/L.

To complete the proof, we need to clarify why we discard duplicates in $\mathbf{Exp}_{\mathcal{A}}^{unforg}(1^\lambda)$. We consider that $Tick_k = (B_k, E, \Pi)$ and $Tick_{k'} = (B_{k'}, E', \Pi')$ are duplicates if their serial numbers are equal. Let us denote by (s, k) (respectively (s', k')) the token pair corresponding to the ticket $Tick_k$ (respectively $Tick_{k'}$). Since $B_k = B_{k'}$, we have $s + k = s' + k' \ mod \ p$.

First case: $(s, k) = (s', k')$

This implies that $Tick_k$ and $Tick_{k'}$ are in fact the same tickets. We can therefore discard one of them.

Second case: $(s, k) \neq (s', k')$

Case 2.1: s or $s' \notin \{s_1, s_2, ..., s_N\}$. This implies that $\mathcal{A}$ is a Type-1 Forger. Under the q-SDH-I assumption, this case will occur with negligible probability.

Case 2.2: s and $s' \in \{s_1, s_2, ..., s_N\}$. This implies that k and $k' \in \{1, ..., max_{ticket}\}$. Otherwise $\mathcal{A}$ could be used to break the q-SDH assumption (see the proof of Theorem 2). Since s and s' have been randomly chosen (they are jointly computed by $\mathcal{A}$ and $\mathcal{B}$), the probability that $s + k = s' + k' \ mod \ p$ is negligible. Therefore Case 2.2. will occur with negligible probability either.

Consequently, under the q-SDH assumption, only the first case could occur and we only discard tickets that come from the same token secret.

Theorem 8 (Non-Frameability). Our m-ticketing protocol is non-frameable, in the random oracle model, under the q-DDHI assumption. The proof is detailed below.

Proof 8 (sketch of proof). We assume that the challenger $\mathcal{C}$ in the experiment $\mathbf{Exp}_{\mathcal{A}}^{Nfra}$ (1^λ) receives a random instance $(g^{x_1}, g^{x_2}, ..., g^{x_l})$ of the one-more DL problem where g is a random generator in $\mathbb{G}_1$. $\mathcal{C}$ will run the adversary $\mathcal{A}$ as a subroutine and act as $\mathcal{A}$'s challenger in the non-frameability experiment. Because $\mathcal{C}$ is against the one-more DL assumption, he has access to a DL oracle. $\mathcal{C}$ picks three random values ξ_1, ξ_2 and ξ_3 in $\mathbb{Z}_p$ and computes $g_1 = g^{\xi_1}$, $g_U = g^{\xi_2}$ and $G = g_1^{\xi_3}$. The other generators of the m-ticketing system are randomly chosen. $\mathcal{A}$ chooses the keys tsk for the transport authority and rsk for the revocation authority and $\mathcal{C}$ chooses the key sk_{RP} for the Paillier encryption scheme. $\mathcal{C}$ is therefore able to construct the public parameters pp for $\mathcal{A}$ and can answer to its requests in the following way:

- $\mathcal{O}Register_{HU}$ requests: $\mathcal{C}$ answers using his input of the one-more DL problem. $\mathcal{C}$ puts $h_{U_i} = (g^{x_i})^{\xi_2} = g_U^{x_i}$

- $\mathcal{O}Register_{MU}$ requests: $\mathcal{C}$ does not need to do anything.
- $\mathcal{O}CorruptUser$ requests: $\mathcal{C}$ uses the DL oracle to give the corresponding x_i to the adversary.
- $\mathcal{O}TokenRequest_U$ requests: $\mathcal{C}$ first uses his input of the one-more DL problem to compute the commitment Com: $\mathcal{C}$ puts $Com = (g^{x_j})^{\xi_1} = g_1^{x_j}$. If the protocol doesn't abort then we put $s = x_j + s_2 \ mod \ p$ and $c = g_1^s$, where x_j is unknown to $\mathcal{C}$ and s_2 is provided by $\mathcal{A}$. In the random oracle model, $\mathcal{C}$ is able to perfectly simulate the proof of knowledge Π_1 as well as the Schnorr's signature μ.
- $\mathcal{O}GenTicket(ID_U, view)$ requests: All the values involved in the computation of an m-ticket $Tick_k$ can be easily simulated by $\mathcal{C}$ except T' and B_k. To compute $T' = G^s H^{r_2}$ where r_2 is a random value chosen by $\mathcal{C}$, it proceeds as follows: $T' = (Com \times g_1^{s_2})^{\xi_3} H^{r_2} = G^s H^{r_2}$. As $\mathcal{C}$ does not know the value s it cannot compute or simulate the value $B_k = g_t^{1/(s+k+1)}$. It will therefore choose a random value R and define B_k as R. In the random oracle model, $\mathcal{C}$ can simulate the proof of knowledge Π using standard techniques. The simulation is not perfect since we replace the value B_k by a random value R. However $\mathcal{A}$ will not detect this change unless he can solve the q-DDHI problem.
- $\mathcal{O}ReportTicket(ID_U, view)$ requests: $\mathcal{C}$ proceeds as in an $\mathcal{O}GenTicket$ request for each unused m-ticket.

Now, we assume that the adversary $\mathcal{A}$ manages to produce a valid m-ticket $Tick_k$ such that it breaks the non-frameability of our m-ticketing protocol and it mades t requests to the $\mathcal{O}CorruptUser$ oracle. This means that this m-ticket has not been obtained on a $\mathcal{O}GenTicket$ query and the `IdentUser` algorithm on $Tick_k$ outputs an identifier ID_U which is in $\mathcal{HU}$ along with a Schnorr's signature μ that proves that this m-ticket comes from a permission token obtained by this user[4] .

It follows from the Forking Lemma that if $\mathcal{A}$ is able to produce a valid m-ticket $Tick_k$ $(k, \sigma_1, h, \sigma_2)$ where

σ_1=(C, C_1, C_2, B_0, T', T'', Com, B, D, K, C_1', C_2', R', R'', C', T_4', Com_1, D_1, K', L', ch),

h=H(C, C_1, C_2, B_0, T', T'', Com, B, D, K, C_1', C_2', R', R'', C', T_4', Com_1, D_1, K', L', ch) and

σ_2=(s_1, s_2, s_3, ω_1, ω_2,..., ω_{12}) then it can produce two valid m-tickets $(k, \sigma_1, h, \sigma_2)$ and $(k, \sigma_1, h', \sigma_2)$ such that $h \neq h'$. Using the technique of replay and the soundness property of the proof Π, one is able to extract all the secret values (a, s, r, A, k).

First case: the value s corresponds to a permission token obtained during an $\mathcal{O}$ $TokenRequest_U$ on ID_{U_i} (i.e., g_1^s is equal to the value c produced during such a request). $\mathcal{C}$ outputs the t values x_i that comes from the requests to the DL oracle plus the value $x_j = s - s_2 \ mod \ p$ and so breaks the one-more DL assumption.

Second case: the value s does not correspond to a permission token obtained during an $\mathcal{O}TokenRequest_U$ on ID_{U_i} (meaning that all the values c generated during such requests are different from g_1^s). We know by the definition of the experiment that no

[4]We would like to emphasize that since the output of `IdentUser` can be publicly verifiable, a wrong `IdentUser` procedure is statistically negligible (even for a powerful adversary).

$\mathcal{O}CorruptUser$ oracle query (and consequently no DL oracle query) has been made on this identity. Therefore the public key h_{U_i} corresponding to ID_{U_i} is in the one-more DL problem input. It follows from the Forking Lemma that if $\mathcal{A}$ is sufficiently efficient to produce such a signature μ, then there exists an algorithm A' which can produce two Schnorr's signatures with non-negligible probability. Using the techniques of replay and soundness, $\mathcal{C}$ is able to extract the private key x_U used to generate the signature μ. $\mathcal{C}$ outputs the t values x_i, coming from the requests to the DL oracle, plus the value x_{U_i} and so breaks the one-more DL assumption.

We prove the non-frameability under the q-DDHI and one-more discrete logarithm assumptions [50]. We use in fact the OMDL assumption to get a better reduction, but the proof can also be done under the discrete logarithm assumption. As the q-DDHI assumption implies the DL one, we can conclude that our m-ticketing protocol is non-frameable, in the random oracle model, under the q-DDHI assumption.

Theorem 9 (Unlinkability). Our m-ticketing protocol satisfies the unlinkability requirement, in the random oracle model, under the q-DDHI and DDH assumptions. The proof is detailed below.

Proof 9 (sketch of proof). We prove the **unlinkability** under a slightly weaker model than the one presented in Section 7.4. Indeed, we consider a slightly different experiment in which the adversary cannot query the revocation oracle $\mathcal{O}IdentUser_T$. We rename this new requirement **Unlinkability***. We would like however to emphasize that the access to revocation functionalities will likely be carefully controlled in a real deployment of an m-ticketing system. Therefore, unlinkability* is a reasonable model to consider.

Anyway, in order to satisfy the original unlinkability requirement, we just need to replace the ElGamal encryption scheme, which only satisfies IND-CPA security, by an IND-CCA2 encryption scheme. It is well-known that by double encrypting the same message under an IND-CPA scheme and using a *simulation sound proof of equality* of plaintexts, we obtain an IND-CCA2 scheme. Therefore, by using the double ElGamal encryption scheme, which was proved IND-CCA2 in [113], our m-ticketing protocol would satisfy the original unlinkability requirement.

Let $\mathcal{A}$ be an adversary who breaks the unlinkability requirement of our m-ticketing protocol. W.l.o.g. we will consider that a query to the $\mathcal{O}ReportTicket$ oracle on (ID_U, *view*) for the report of j unused m-tickets is equivalent to j queries to the $\mathcal{O}GenTicket$ on (ID_U, *view*).

Let $m = max_{ticket}$. We will say that an adversary $\mathcal{A}$ against our unlinkability experiment $\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda)$ is a Type-(i, j) distinguisher, with $i \leq m-1$ and $j \leq m-1$, if $\mathcal{A}$ after having received its challenge (from its $\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda)$-challenger) makes at most i queries to the $\mathcal{O}GenTicket$ oracle on (i_0, $view_0$) and j queries to the $\mathcal{O}GenTicket$ oracle on (i_1, $view_1$). We can remark that a Type-(i, j) distinguisher, with $i \leq m-1$ and $j \leq m-1$, is also a Type-($m-1$, $m-1$) distinguisher. We may therefore, without loss of generality, restrict our attention to Type-($m-1$, $m-1$) distinguishers. In the sequel, we will thus assume that $\mathcal{A}$ is such an adversary. More precisely, we will suppose that after receiving $Tick_{k_b}$ and $Tick_{k_{1-b}}$, $\mathcal{A}$ arbitrarily queries the $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}IdentTicket_T$ and $\mathcal{O}TokenRequest_U$ oracles and then queries the $\mathcal{O}ReportTicket$ oracle on (i_0, $view_0$) and (i_1, $view_1$).

We give a security proof as a sequence of games, using Shoup's methodology [102]. We will only give a rather high-level description of the initial game (Game 0) and brief descriptions of the modifications between successive games.

Game 0: This is the original attack game with respect to a given efficient adversary $\mathcal{A}$.

The challenger $\mathcal{C}$ randomly chooses g, g_0, g_1, g_t, g_T, g_U, h, G, H nine generators of $\mathbb{G}_1$ and g_2, g_3 two generators of $\mathbb{G}_2$. It also chooses the keys for the Paillier encryption scheme as well as the ones for the transport authority and revocation authorities. $\mathcal{C}$ sends gpk and tsk to $\mathcal{A}$. $\mathcal{C}$ answers to $\mathcal{A}$'s requests as follows:

- $\mathcal{O}Register_{HU}$ requests: $\mathcal{C}$ randomly chooses $x_U \in \mathbb{Z}_p$ and computes $h_U = g_U^{x_U}$.
- $\mathcal{O}Register_{MU}$ requests: $\mathcal{C}$ does not need to do anything.
- $\mathcal{O}CorruptUser$ requests: $\mathcal{C}$ gives x_U to $\mathcal{A}$.
- $\mathcal{O}TokenRequest_U$ requests: $\mathcal{C}$ chooses a random value s_1 to obtain a valid permission token (A, r, s) and uses x_U to generate the signature μ.
- $\mathcal{O}GenTicket(ID_U, view)$ requests: $\mathcal{C}$ uses its permission token (A, r, s) and a fresh index k that has not been used in a previous query of $\mathcal{O}GenTicket$ on ID_U and $view$ and computes a ticket $Tick_k = (B_k, E, \Pi)$.
- $\mathcal{O}IdentTicket_T(ID_U, view)$ requests: $\mathcal{C}$ computes the m-tickets $Tick_k$ based on the secret s corresponding to the user identifier ID_U and gives them to $\mathcal{A}$.
- $\mathcal{O}ReportTicket(ID_U, view)$ requests: $\mathcal{C}$ proceeds as in a $\mathcal{O}GenTicket$ request for each unused m-ticket.

The adversary chooses two honest users i_0 and i_1, and two indexes k_0 and k_1. $\mathcal{C}$ runs the protocol `TokenRequest` with i_0 and i_1, and outputs the corresponding views, $view_0$ and $view_1$, to $\mathcal{A}$. It then generates two valid m-tickets: $Tick_{k_b} = (B_{k_b}, E_b, \Pi_b)$ for i_0 and $Tick_{k_{1-b}} = (B_{k_{1-b}}, E_{1-b}, \Pi_{1-b})$ for i_1 and send them to $\mathcal{A}$. The goal of $\mathcal{A}$ is to guess the value of b. After having received its challenge, $\mathcal{A}$ can queries the $\mathcal{O}Register_{HU}$, $\mathcal{O}Register_{MU}$, $\mathcal{O}CorruptUser$, $\mathcal{O}TokenRequest_U$, $\mathcal{O}GenTicket$, $\mathcal{O}IdentTicket_T$ and $\mathcal{O}ReportTicket$ oracles but with the following restrictions: it cannot query the $\mathcal{O}CorruptUser$ oracle on i_0 or i_1 or the $\mathcal{O}IdentTicket_T$ oracle on (i_0, $view_0$) or (i_1, $view_1$) and cannot query the $\mathcal{O}Report\ Ticket$ oracle on (i_0, $view_0$) or (i_1, $view_1$) if both users did not validate the same number of m-tickets (otherwise it can easily win this game).

At the end of the game, the adversary outputs a bit b', its guess. Let S_0 be the event that $b = b'$ in this game and S_i the event that $b = b'$ in game i. We have: $|Pr[S_0] - 1/2|$ $= \mathbf{Adv}_{\mathcal{A}}^{unlink-b}(1^\lambda) = |Pr[\mathbf{Exp}_{\mathcal{A}}^{unlink-b}(1^\lambda) = b] - 1/2|$

Let $\{Tick^i_{k^i_b} = (B^i_{k^i_b}, E^i_{k^i_b}, \Pi^i_{k^i_b})\}_{i=1}^{i=m-1}$ denote the $m-1$ unused (reported) m-tickets of i_0 and $\{Tick^i_{k^i_{1-b}} = (B^i_{k^i_{1-b}}, E^i_{k^i_{1-b}}, \Pi^i_{k^i_{1-b}})\}_{i=1}^{i=m-1}$ the ones of i_1, where the k^i_b's and the k^i_{1-b}'s belongs to $\{1, ..., m\}$.

- **Game(0,b)**: This is the same game as **Game 0** except that we replace the El Gamal ciphertext E_b by an encryption of a random value and then simulate the

proof Π_b. Such a simulated proof can easily be done in the random oracle model using standard techniques. Under the DDH assumption, $\mathcal{A}$ cannot detect this change. Indeed, we can easily construct a DDH distinguisher $\mathcal{D}$ with DDH-advantage satisfying:
$|Pr[S_0] - Pr[S_{(0,b)}]| \leq \mathbf{Adv}_{\mathcal{D}}^{DDH}(1^\lambda)$ (1)
where $\mathbf{Adv}_{\mathcal{D}}^{DDH}(1^\lambda)$ is the DDH-advantage of $\mathcal{D}$ in solving the DDH problem. In the sequel we will used the simplified notation $\mathbf{Adv}_{DDH}$ (respectively $\mathbf{Adv}_{q-DDHI}$) to denote the DDH-advantage (respectively the q-DDHI advantage) of some efficient algorithm in solving the DDH (respectively q-DDHI) problem.

- **Game(0,1-b)**: This is the same game as **Game(0,b)** except that we replace the El Gamal ciphertext E_{1-b} by an encryption of a random value and then simulate the proof Π_{1-b}. Such a simulated proof can easily be done in the random oracle model using standard techniques. Under the DDH assumption, $\mathcal{A}$ cannot detect this change. Indeed, we can easily construct a DDH distinguisher D with DDH-advantage satisfying:
$|Pr[S_{(0,b)}] - Pr[S_{(0,1-b)}]| \leq \mathbf{Adv}_{DDH}$ (2)

- **Game(0,0,b)**: This is the same game as **Game(0,1-b)** except that we replace the serial number B_{k_b} by a random value and then simulate the proof Π_b. Such a simulated proof can easily be done in the random oracle model using standard techniques. Under the q-DDHI assumption, $\mathcal{A}$ cannot detect this change. Indeed, we can easily construct a q-DDHI distinguisher $\mathcal{D}$ with q DDHI-advantage satisfying:
$|Pr[S_{(0,1-b)}] - Pr[S_{(0,0,b)}]| \leq \mathbf{Adv}_{q-DDHI}$ (3)

- **Game(0,0,1-b)**: This is the same game as **Game(0,0,b)** except that we replace the serial number $B_{k_{1-b}}$ by a random value and then simulate the proof Π_{1-b}. Such a simulated proof can easily be done in the random oracle model using standard techniques. Under the q-DDHI assumption, $\mathcal{A}$ cannot detect this change. Indeed, we can easily construct a q-DDHI distinguisher $\mathcal{D}$ with q DDHI-advantage satisfying:
$|Pr[S_{(0,0,b)}] - Pr[S_{(0,0,1-b)}]| \leq \mathbf{Adv}_{q-DDHI}$ (4)

Let **Game 1** be the same game as **Game(0,0,1-b)**

Combining (1), (2), (3) and (4), we obtain:

$$|Pr[S_0] - Pr[S_1]| \leq 2\mathbf{Adv}_{DDH} + 2\mathbf{Adv}_{q-DDHI}$$

We then proceed similarly for each unused m-ticket of i_0 and i_1:

$$\{Tick^i_{k^i_b} = (B^i_{k^i_b}, E^i_{k^i_b}, \Pi^i_{k^i_b})\}_{i=1}^{i=m-1}$$

and

$$\{Tick^i_{k^i_{1-b}} = (B^i_{k^i_{1-b}}, E^i_{k^i_{1-b}}, \Pi^i_{k^i_{1-b}})\}_{i=1}^{i=m-1}.$$

For $i = 1$ to $m-1$, we define the following game.

- **Game(i,b)**: This is the same game as **Game i = Game(i-1,0,1-b)** except that we replace the El Gamal ciphertext E^i_b by an encryption of a random value and then simulate the proof Π^i_b.

- **Game(i,1-b)**: This is the same game as **Game(i,b)** except that we replace the El Gamal ciphertext E^i_{1-b} by an encryption of a random value and then simulate the proof Π^i_{1-b}.
- **Game(i,0,b)**: This is the same game as **Game(i,1-b)** except that we replace the serial number $B^i_{k^i_b}$ by a random value and then simulate the proof Π^i_b.
- **Game(i,0,1-b)**: This is the same game as **Game i,0,b)** except that we replace the serial number $B^i_{k^i_{1-b}}$ by a random value and then simulate the proof Π^i_{1-b}.

Let **Game i + 1** be the same game as **Game(i,0,1-b)**. We obtain as above: $|Pr[S_{i+1}] - Pr[S_i]| \leq 2\mathbf{Adv}_{DDH} + 2\mathbf{Adv}_{q-DDHI}$

Using our notation, **Game m** is the same game as **Game(m-1,0,1-b)**. In the latter game, the unlinkability-challenger gives no information (in a strong information theoretic sense) to $\mathcal{A}$ about the bit b (since all the El Gamal ciphertexts have been replaced by encryptions of random values and all serial numbers have been replaced by random values). Therefore we have: $Pr[S_m] = 1/2$.

We can now give an upper bound for $\mathbf{Adv}^{unlink-b}_{\mathcal{A}}(1^\lambda)$:

$$\mathbf{Adv}^{unlink-b}_{\mathcal{A}}(1^\lambda) = |Pr[\mathbf{Exp}^{unlink-b}_{\mathcal{A}}(1^\lambda) = b] - 1/2| = |Pr[S_0] - Pr[S_m]|$$

We have:

$$|Pr[S_0] - Pr[S_m]| \leq \Sigma_{j=0}^{j=m-1} |Pr[S_j] - Pr[S_{j+1}]| \leq 2m \times (\mathbf{Adv}_{DDH} + \mathbf{Adv}_{q-DDHI})$$

Therefore under the DDH and q-DDHI assumptions, the advantage $\mathbf{Adv}^{unlink-b}_{\mathcal{A}}(1^\lambda)$ of a Type-$(m-1, m-1)$ distinguisher is negligible (and consequently the one of a Type-(i, j) distinguisher, with $i \leq m-1$ and $j \leq m-1$, is also negligible).

We can then conclude that our proposed m-ticketing protocol satisfies the unlinkability* requirement, in the random oracle model, under the DDH and q-DDHI assumptions.

7.6 Implementation

In this section, we give further details about the prototype implementing our m-ticketing system in addition to the performance results of the m-ticket validation phase.

7.6.1 Prototype Details

The m-ticketing development platform is the same as the m-pass development platform. The validator is simulated by a Java swing application connected to an NFC reader using the ISO14443B protocol. The cardlet is embedded within a SIM card. We used a regular Galaxy S3 smartphone running Android 4.1.2. For more details about the validator and SIM card details, the reader can refer to Section 6.6.1 and Section 6.6.2.

We also use the same cryptographic parameters (curve and pairing) like in the m-pass use case. Further details can be found in Section 6.6.3

7.6.2 Validation Measurements

The validation of a m-ticket consists of two sub-phases. The transport cardlet has some pre-compuations in the first sub-phase. This sub-phase occurs in off-line for instance before arriving to station. Then, the second sub-phase comprises real-time computations. Indeed, these computations occur when the user taps his smartphone on the validator in order to enter the transport network.

7.6.2.1 Pre-Computations

The pre-computations consists in computing elements involved in proving that the user knows a valid signature on his secret s. The elements that the card has to store for one signature are 24 $\mathbb{Z}_p$ elements, 10 compressed points and one digest output. The total size of these elements is 1130 *bytes*. In our implementation, we did pre-computations for 10 signatures (A set of 10 m-tickets). Regarding the timings, detailed in Table 7.2, the pre-computations for preparing one signature occurs in 1.7 s.

1 m-ticket	10 m-tickets
1.7 s	17 s

TABLE 7.2: M-Ticketing - Timings of Pre-Computations

7.6.2.2 Real-Time Computations

Tables 7.3, 7.4, 7.5 and 7.6, give timings (average over 50 trials and standard deviation between parentheses) for all real-time computations (the whole validation protocol) which include the validator authentication, the signature generation and an m-ticket verification. The timings of the signature generation include as well the NFC exchanges duration. We denote by "Battery-Off" a powered-off phone either by the user, or because the battery is flat. In this situation, as stated by NFC standards, NFC-access to the SIM card is still possible, but with degraded performances.

Regarding the validator authentication, timings in Table 7.3, we chose to use RSA with a 1984 bits key (this is the greatest size supported by the SIM card.) and a short public verification exponent v ($v = 65537$). The validator asks the card for a challenge (RC_V). Then, he sends back his response to this challenge, his own challenge (ch) (32 bytes) and the current date (TS) (6 bytes).

Battery-On	56.98 ms (0.70)
Battery-Off	76.55 ms (7.46)

TABLE 7.3: M-Ticketing - Timings of the Validator Authentication

Table 7.4 gives the duration of a signature generation (computing an m-ticket B_k, E and Π) and the NFC communication time. The considered operations for generating the

signature are only one hash value and lightweight operations in $\mathbb{Z}_p$. The size of the computed signature is 778 *bytes* (sent by 4 APDUs). Regarding the communication between a SIM card and a reader, it is slow ($\geq 85ms$), but the whole process (Signature+NFC) remains very fast, i.e., 123.01 ms on average.

Battery-On	123.01 ms (3.24)
Battery-Off	185.28 ms (18.68)

TABLE 7.4: M-Ticketing - Timings of the Card Signature and NFC Connections

For the signature verification by the validator, timings in Table 7.5, we distinguish two cases: the validator holds the private signature keys (y, γ), or not. The extra pairing computations performed in the second case do not add an important delay to the verification step. This is because a regular desktop PC can efficiently achieve such computations.

(1) without pairing	(2) with pairing
4.43 ms (1.32)	12.19 ms (3.20)

TABLE 7.5: M-Ticketing - Timings of the Signature Verification

Table 7.6 gives the total duration of a m-ticket validation. In total, it occurs for the first case (without pairing at the verifier side) in $184.25ms$ on average and in $191.80ms$ for the second case (with pairing at the verifier side). When the battery is flat, the validation occurs in at most $272,55ms$ on average.

	(1) without pairing	(2) with pairing
Battery-On	184.25 ms (3.43)	191.80 ms (4.73)
Battery-Off	266.52 ms (17.91)	272.55 ms (25.73)

TABLE 7.6: M-Ticketing - Timings of Real-Time Computations

7.7 Conclusion

In this chapter, we designed a secure m-ticketing protocol that prevents a malicious transport authority from linking users' trips. Moreover, as the entire computations are done within the SIM card, we enhance the user's privacy with regards to a compromised smartphone. Owing to regular reports of unused m-tickets, the user is able to securely post-pay for his m-tickets without disclosing any information about the performed trips. Moreover, we planned countermeasures against m-ticket cloning and multiple usage. Our proposal also satisfies the validation time constraint: an m-ticket validation can be completed in 184.25 ms when the mobile is switched on, and in 266.52 ms when the mobile is switched off or its battery is flat.

As mentioned previously our mobile services, i.e., mobile pass and mobile ticketing, are implemented within a SIM card, but, it can be implemented within a TEE as well. Using the TEE as a development platform have advantages as well as drawbacks. On one hand, it will provide more computational capabilities which will enable more efficiency. On the other hand, a functionality like the ability to validate a transport product when the smartphone is off will no longer be available. Moreover, a new technical issue arises. Indeed, if a user changes his smartphone, he obviously would like to have the same services within the new smartphone. If the application providing the service is running within the SIM card, the service can be easily and securely transfered from a smartphone to another (unplug / plug the SIM card). This is not the same case for the TEE. Therefore, we have worked on the problem of migration of services between two TEE and proposed, in the next chapter, a new secure TEE migration protocol.

Chapter 8

Practical and Privacy-Preserving TEE Profile Migration

Contents

Dans ce chapitre, nous présentons les résultats [114] qui ont été publiés à la conférence WISTP 2015. Nous décrivons notre perception du déploiement et de la mise en œuvre d'un environnement d'exécution de confiance (TEE): nous organisons le TEE en domaines de sécurité avec différents rôles et privilèges. Nous appelons profil TEE les secrets et données privées appartenant à l'utilisateur et manipulés par les applications s'exécutant dans le TEE. En se basant sur le nouveau modèle, nous construisons un protocole de migration de profils TEE assurant la confidentialité et l'intégrité de ce dernier. À cette fin, nous utilisons une clé de re-chiffrement et un jeton d'autorisation par couple de mobile, par fournisseur de service et par transfert. Nous avons également validé notre protocole avec AVISPA, un outil automatisé de validation de protocoles de sécurité.

8.1 Introduction

Like mentioned in Chapter 2, in the last years, a secure mobile operating system running alongside the standard Rich Execution Environment (REE for short), has emerged: the

Trusted Execution Environment (TEE for short). A TEE could have its own CPU and memory, and hosts isolated Trusted Applications (TA for short) that provide secure services to the applications running within the REE. These TAs belong to diverse service providers. Each TA manipulates a profile, constituted of secret credentials and user's private data.

The TEE has been standardized by GlobalPlatform, however, to the best of our knowledge, there is no specification or research work that models the TEE internal architecture or ecosystem. For instance, comparing to smart cards, the GlobalPlatform Card Specifications [1] have worked on such a model and it is now widely deployed and accepted by all the stakeholders. This is why we propose to study in which extent we can apply it for the TEE context: we identified the limitations of the GlobalPlatform Card model, in the TEE context, when the user wants to migrate its profile from one TEE to another one.

A user, who has many devices or gets a new one, shall be able to securely migrate his TEE profiles from a TEE to another compliant TEE. This problem of migration is currently poorly addressed by TEE implementations, standardization and only few papers have worked on designing TEE migration protocols [115, 116]. Two main solutions can be considered: the straightforward solution, which consists in encrypting the profile (by TEE source), transferring it and decrypting it (by target TEE), or a Trusted Server based solution. These solutions present privacy weaknesses, as discussed in the next sections.

In this chapter, we propose a new approach to transfer the TEE profiles from a TEE to another compliant TEE while preserving its privacy. For this purpose, we propose to organize the TEE into security domains (SD) with different roles and privileges.

This chapter is organized as follows. In Section 8.2, we present the TEE technology and describe the problem of profile migration. Then, in Section 8.3, we describe the previous works and discuss how different are our objectives. We define our assumptions and requirements in Section 8.4. Then, in Section 8.5, we give a detailed description of our transfer protocol. We give the security analysis of our protocol in Section 8.6. Finally, in Section 8.7, we present the validation of our protocol and we conclude in Section 8.8.

8.2 Backgrounds and Problem Statement

Before using a service of the TEE, which is provided by a service provider, several steps should occur.

1. User enrollment: the user registers to the service provided by the SP, using a secure channel. This step allows to associate the user identity to a dedicated TA inside the device.
2. The TA is personalized inside the TEE by the SP. The necessary application in the REE is also installed. After this step, the service is active.
3. The user could acquire a new device and wish to *securely* transfer its TEE profiles from the first device to the new one.

4. The user may want to destroy its profile, also defined as *disabling* credentials [117].

In this chapter we focus on step 3, the migration of a TEE profile. Like step 1, step 3 can be threatened by an external attacker. If we suppose that an attacker may have compromised the rich OS or control the network connection of the smartphone, then the enrollment or migration steps become challenging tasks. Indeed, as shown in Figure 8.1, as the interactions with a TEE crosses the REE, the attacker may succeed to migrate the user's profile from a victim to the attacker's smartphone. This attack may succeed because the service provider has no insurance about the TEE security and the user-to-TEE binding. In the next section, we describe the solutions already proposed in the literature in order to show their limitations and motivate a new way of migrating TEE profiles.

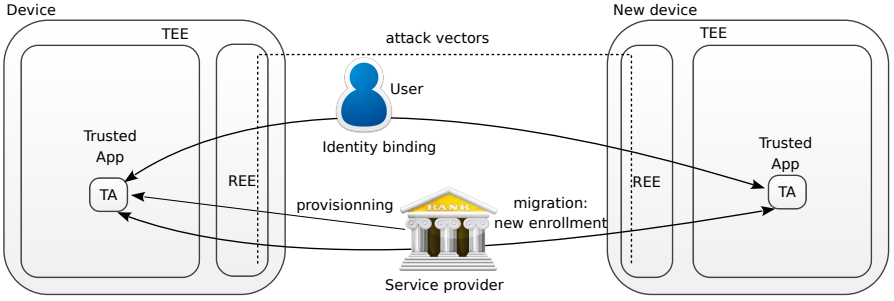


FIGURE 8.1: The Life Cycle of a TEE-based Service

8.3 Related Work

The first papers that studied the security of TEE credentials tried to guarantee its local (within the TEE) confidentiality and integrity. For instance in [118], authors proposed to protect TEE data using a unique PUF (Physical Unclonable Functions) AES encryption key for each device. In [119], authors proposed a TEE key attestation protocol proving that a TEE key has been generated inside the TEE and never left this TEE.

Later, scientists have been interested in the enrollment problem (mainly user-device binding) while assuming that there is no operator responsible for the management of the TEE. For instance, Marforio et al. [115] explained that the user's identity can be bound to the device using a password or a SMS or by collecting the device's IMEI. Unfortunately, an attacker that controls the Rich OS can intercept the protocol exchanges. Thus, Marforio et al. proposed some hardware and software modifications in order to secure the enrollment process. Others, like in [116], assumed that the smartphone is *safe* at the first use. This would enable to store a secret password in the TEE.

The problem of credential migration first appears for Trusted Platform Module (TPM), which is in some way the TEE ancestor. The commands of key migration have been specified in TPM specifications [120] and have undergone many improvements. Later, Sadeghi et al. [121] proposed a property based TPM virtualization in order to have a solution that supports software update and migration. The shortcoming of this solution consists in omitting the service provider during the virtual TPM migration. However, some credential migration requires service provider fresh and explicit agreement.

In the specific context of TEE, the first attempt to address the problem of migration in the TEE has been proposed by authors of [116]. Kari et al. have proposed a credential migration protocol in open credential platforms [116]. They proposed to make the credential migration user-friendly based on delegated-automatic re-provisioning. The credentials are backup in clear in a trusted server. Then, the migration process is a re-provisioning from the backup, protected by a secret password only known by the user. The main weakness of this solution lies in the fact that its security, including to user's privacy, is entirely based on a the trustworthy of the trusted server (TS). This latter, as third party, has full access to TEE credentials and user's private data while it is not its owner or provisioner. This proposal implies privacy issues that we want to address in this thesis.

GlobalPlatform specifications related to smartcards have been interested in credential management in secure elements (smart cards). However, it seems that the credential privacy in some cases has been overlooked. In GlobalPlatform card specifications [1], the smart card is organized into fully isolated areas called Security Domains (SD). There are a root security domain called ISD for Issuer Security Domain and many Supplementary Security Domains (SSDs) for the different Service Providers. Let us call SPSD the security domain for a given SP. For instance, the ISD could be owned by the Mobile Network Operator (MNO) and the SPSD could be owned by a bank. Once, the SPSD created, there are two modes to manage the content of this SPSD: either directly from SP to SPSD, or through ISD. In the first case, the credential migration process would be naturally implemented in the application of the SP within the smart card: encryption with the target public key, transfer and decryption, provided that the MNO initiates the SP space in the target smart card. In the second case, the MNO plays the role of firewall and proxy for the SPSD without having access to the content between SP and its SSD (SPSD). SPSDs do not have any code enabling to perform a credential transfer.

If we adopt the first model for the TEE, the TEE profile migration would be processed like in the smart card: the TEE profile migration process would be naturally implemented in the TA of the SP: encryption with the target public TEE key, transfer and decryption, provided that the TEE admin - MNO initiates the SP space in the target TEE. As a consequence, each service provider would have to implement a migration process for its service.

If we adopt the second model for the TEE, TEE admin will serve as the single entry point to transfer point-to-point credentials: implementing the migration process would imply privacy issues similarly to the Kari et al. [116] solution. TEE admin would have full access to the SP credentials and user's data in order to encrypt and transfer it. In this chapter, we propose a new migration protocol, while adopting the second model, where the TEE Admin plays the role of proxy without having access to SP credentials. We consider the following properties:

- As trusted application profile contains credentials and also personal data (statistics, usage data of the service), during the migration, the profile shall be accessible only by legitimate entities: the SP and the user;
- A special third party, the TEE admin should be responsible of the role of installation, deletion and migration of trusted profiles;

- The trusted application of the SP should not contain any code dedicated to the migration protocol. All the migration software components should be handled by the TEE admin.

8.4 Attacker Model and Requirements

We assume that the enrollment, provisioning and personalization processes are already achieved: the trusted application is provisioned to the TEE and has access to its credentials and the user's personal data. By the profile, we mean the credentials (allowing the access to the service) and private data (related to personalization and the use of application/service).

We consider three different actors: the user (the devices' owner), the Service Provider (e.g. the bank) and a TEE admin (e.g., Mobile Network Operator or smartphone manufacturer). Informally speaking, an attacker will attempt to access the transferred profile. We model it as follows:

A1: Communication control. We consider that we have a Dolev-Yao [122] attacker model: an attacker has full control over communication channels.

A2: TEE control. We consider that TEEs are enough resistant to physical attacks according the Protection Profile proposed by GlobalPlatform [123] which is EAL2+ certified.

A3: REE control. Given the possible vulnerabilities of the rich OS, we assume that an attacker can compromise the REE of a user's device.

A4: Entities control and trustworthy. We assume that (i) an attacker cannot spoof the SP, cannot compromise its dedicated spaces within the TEE and the SP is honest, (ii) an attacker cannot spoof the TEE Admin and cannot compromise its dedicated spaces within the TEE, however the TEE Admin can be honest-but-curious and, (iii) the user is honest.

While keeping in view the above discussions, we define the security requirements that a migration protocol shall meet as follows:

R1: Integrity. During the migration process, the integrity of the TEE profile should be ensured. For a given profile, only the user and the relevant service provider should be able to eventually modify the profile content.

R2: Confidentiality. During the migration process, the confidentiality of the TEE profile should be ensured. For a given profile, only the user and the relevant service provider should be able to eventually read the profile content.

8.5 TEE Migration Protocol

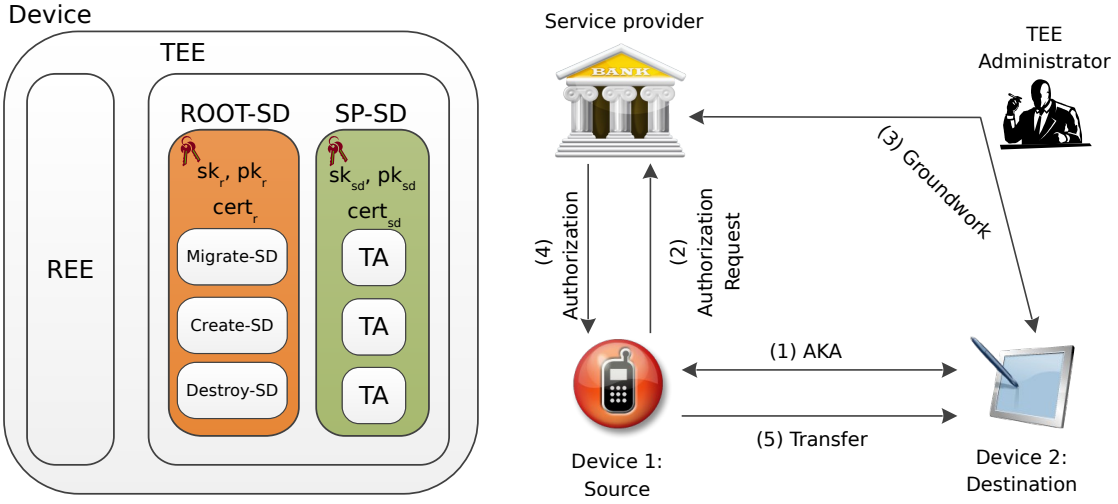
Considering the previous requirements, attacker model, and the GlobalPlatform Card Specifications [1], we introduce a new approach for deploying services in a TEE where:

TAs of a service provider are hosted in a Security Domain (SD) and a new actor, called TEE admin, has a special SD and implements the migration functionalities. With such a new software architecture, we build a protocol that allows to securely transfer a TEE profile from one device to another one.

8.5.1 Architecture Overview

We organize the TEE into Security Domains (SD) [1]. Every SD can contain one or many Trusted Applications (TA) from the same Service Provider. A SD is a fully isolated zone. This functionality could be ensured by memory protection mechanisms, firewall functionalities, data isolation techniques implemented at OS level of the TEE, such as the ones used for Linux systems (SELinux, AppArmor,...). For example, in the commercialized TEE solution of Trustsonic, such an architecture can be implemented using containers. A SD manipulates cryptographic keys which are completely separated from any other SD. These keys enable code execution integrity, credentials and user's private data integrity and confidentiality when using a service. Consequently, a SD must not cipher his credentials or user's private data using any external keys whatever is the case, e.g., transfer. We define two types of SD, represented in Figure 8.2a:

1. SD without management rights (many per TEE): SP-SD (in green). They contain the trusted applications of a service provider.
2. SD with management rights (only one per TEE): ROOT-SD (in orange). It is responsible of creating and deleting SDs, downloading and installing packages in SDs, and also migrating SDs from a TEE to another compliant TEE.



(A) Device architecture (B) Protocol overview

FIGURE 8.2: Architecture and Protocol Overview

8.5.2 Protocol Overview

In order to migrate a TEE profile from a source device to a target one, the user gets the two devices nearby each other in order to establish a wireless communication, such as NFC or bluetooth. Then, the user starts the migration application, noted Migrate-SD in Figure 8.2a, within the ROOT-SD of the TEE source. In order to do this, the user must be authenticated in both source and target TEEs. Owing to the authentication procedure, the TEEs check that only the user enrolled by the TEE admin can start the migration process. This authentication can be done through the "Trusted User Interface" [124], or by using the password or the pin code setup at the enrollment phase, or by using a biometry peripheral if available. The next steps of the protocol that involve the two TEEs are presented in Figure 8.2b and described in the following.

Step 1. An authenticated key agreement takes place between the ROOT-SD of TEE source and the ROOT-SD of target TEE. This prevents the TEE source from disclosing critical information to a malicious environment and prevents the target TEE from accepting malicious data.

Step 2. The TEE source requires a migration authorization from all service providers that have a SD within the TEE source. If a service provider does not agree with the migration of his relevant SD, the migration cannot take place. The migration authorization is temporary and unique per pair of devices, per transfer and per service provider. Indeed, the authorization is related to the date and time of the request that has been initiated. Thus, it is valid only for a given period of time.

Step 3. At that time starts the groundwork for the authorization. First, the service provider checks with the TEE admin whether the target TEE is stated compromised (its corresponding keys have been revoked). Then, the service provider checks that the target TEE is not already a client containing a service provider SD. Finally, the service provider asks the TEE admin to set up a specific SD within target TEE, and updates the SD credentials in order to be the unique master of this SD.

Step 4. Finally, the service provider replies to the TEE source with the authorization and necessary migration credentials. The authorization consists of a service provider signature proving his agreement regarding the migration of his SD. The credentials consist of a re-encryption key [40, 41]. Using this re-encryption key, the Migrate-SD application will be able to perform the transfer without having access to SD profile. In order to send source profile to the target SD, the source SD provides its profile, encrypted with its public key, to the TA, Migrate-SD, that should re-encrypt it with the re-encryption key. In such a case, even if the TEE Admin is honest-but-curious, it cannot eavesdrop the SD profile.

Step 5. The target TEE must check the validity of the received authorization. At this time, the migration can start.

8.5.3 TEE Profile Migration Protocol

In the following, we introduce the notations and cryptographic keys used in our protocol. Later, we detail the phases of our protocol: Authenticated key agreement, Service Provider authorization and TEE Profile migration.

Cryptographic keys and notations We denote (sk_{src}, pk_{src}, $cert_{src}$) (resp. (sk_{tgt}, pk_{tgt}, $cert_{tgt}$)) the TEE private root key and the certified TEE public root key of the source (resp. target) ROOT-SD. These keys are used to authenticate the TEE and set up a key session with an authenticated TEE. A TEE admin is characterized by a private and public key pair (sk_{Admin}, pk_{Admin}). It controls the ROOT-SD and certifies TEE root keys. A Security Domain SP-SD is characterized by a root keys set (sk_{sd}, pk_{sd}, $cert_{sd}$). This is an encryption keys set that enables to securely store SD profile. We denote $SP-SD_{src}$ (resp. $SP-SD_{tgt}$) the service provider SD within TEE source (resp. target TEE). Consequently, the tuple ($sk_{SP-SD_{src}}$, $pk_{SP-SD_{src}}$, $cert_{SP-SD_{src}}$) (resp. the tuple ($sk_{SP-SD_{tgt}}$, $pk_{SP-SD_{tgt}}$, $cert_{SP-SD_{tgt}}$)) is the root keys set of $SP-SD_{src}$ (resp. $SP-SD_{tgt}$). A service provider is characterized by (sk_{sp}, pk_{sp}) and a unique identifier ID_{SP}. It provides the security domains root keys and is responsible of the re-encryption key and transfer authorization generation.

In the following, we denote by $Signature_A$, the signature on the message sent with the signature using the private key of A.

Authenticated Key Agreement The authenticated key agreement (AKA, step 1 in Figure 8.2b) occurs in order to establish a secure session between two TEEs after a mutual authentication. The AKA can be a password based key agreement [125] or a public key authenticated Key agreement [126] and must be a two ways authentication. In the first case, we can use the password or PIN or biometric data introduced by the user during the authentication phase and in the second case, we can use the TEEs root keys. At the end of this phase, the source and target TEEs will share a couple of ephemeral keys (eK_t, eK_m) to secure the migration. eK_t is the private session key, whereas eK_m is used for MAC computations.

Service Provider Authorization The TEE source requires a migration authorization from all service providers having trusted applications within the TEE source (step 2 in Figure 8.2b). This protocol is described in Figure 8.3. For the sake of simplicity, we consider only one service provider.

(1) The migration application within $ROOT-SD_{src}$ sends the request $INIT\_RQ$ with its signature noted $Signature_{ROOT-SD_{src}}$ to the service provider through the TEE admin. The request includes the identity of the service provider (ID_{SP}), the public key of $SP-SD_{src}$ ($pk_{SP-SD_{src}}$) and the certified TEE public root keys of source and target TEE ($cert_{src}, cert_{tgt}$). (2) When receiving the request, TEE admin checks the certificates ($cert_{src}, cert_{tgt}$), the signature and freshness of the request and a timestamp ($Signature_{ROOT-SD_{src}}$)[1]. It should also check whether source or target TEE are compromised[2] for example using the remote attestation protocols of Baiardi et al. [127]. (3) If checks are successful, the TEE admin transmits the request ($INIT$) to the relevant service provider based on the ID_{SP} received within the request.

(4) With the received request, the service provider checks if the TEE source (resp. target TEE) has (resp. has not) an associated SP-SD by checking if $cert_{src}$ (resp. $cert_{tgt}$) is registered in its database. Then, (5) the service provider inquires TEE admin to create a SP-SD within the recipient TEE via the $SD-Create\_RQ(cert_{tgt})$ command. (6) The

[1]TEE implementations like TrustZone offer access to trusted clocks for this usage.
[2]This is already the case for SIM card where MNOs checks if a SIM has been revoked.

TEE admin signs the command (the signature $Signature_{TEE_{Admin}}$ is performed on the command and a timestamp) and forwards it to the trusted application $Create - SD_{tgt}$ in order to create $SP - SD_{tgt}$. (7, 8, 9, 10). The creation is acknowledged by Ack and $Param$ that are returned to the service provider (through the TEE Admin) in order to let him be able to personalize $SP - SD_{tgt}$, as done classically when personalizing TEE security domains. (11) Once the $SP - SD_{tgt}$ installed, the service provider proceeds to the update of $SP - SD_{tgt}$ credentials in order to have the exclusive control over $SP - SD_{tgt}$ [1].

Finally, the service provider generates the migration authorization. It consists of a re-encryption key K_{proxy} and a signature $PERM$. The signature $PERM$ is computed on the SP identifier ID_{SP}, source and target TEE public keys as well as a timestamp: $PERM = \{ID_{SP}, cert_{src}, cert_{tgt}, TimeStamp\}_{sk_{SP}}$. The re-encryption key K_{proxy} is used to re-encrypt the $SD - SP_{src}$ content such that the result will be understandable only by $SP - SD_{tgt}$: $K_{proxy} = rekeygen(pk_{SP-SD_{tgt}}, sk_{SP-SD_{src}})$. In the literature [40, 41], K_{proxy} is called a proxy key. (12, 13) The authorization is sent to the TEE Admin who signs it and transmits it (with its signature) to $ROOT - SD_{src}$.

TEE Profile Migration Using the re-encryption key, the confidentiality and integrity of the migration phase is guaranteed. Any outsider cannot eavesdrop the $SP - SD_{src}$ profile and a honest-but-curious TEE Admin has no visibility about the $SP - SD_{src}$ profile. The migration occurs as follows.

As described in Figure 8.4, $Migrate - SD_{src}$ re-encrypts the protected profile of $SP - SD_{src}$ (P) using the proxy key K_{proxy} to obtain the cipher A. Only $SP - SD_{tgt}$ is able to decrypt A. Afterwards, $Migrate - SD_{src}$ computes B and C. B is the encryption of A and the identifier of the service provider owning $SP - SD_{src}$ (ID_{SP}) using the transfer key eK_t. Regarding C, it is the MAC of A and ID_{SP} using eK_m. At the end of these computations, $Migrate - SD_{src}$ sends A, B, C and $PERM$ to $Migrate - SD_{tgt}$. that proceeds to the verifications of $PERM$ and C. The verification of $PERM$ corresponds to the verification of a signature, its freshness and that its parameters contain the right $cert_{src}$ and $cert_{tgt}$. Next, $Migrate - SD_{tgt}$ decrypts B in order to retrieve A and ID_{SP}. Based on the retrieved ID_{SP}, $Migrate - SD_{tgt}$ transmits A to $SP - SD_{tgt}$.

When the migration finishes, we have two possibilities. On the one hand, the security policy of the service admits to conserve the TEE profile in the source. In such a case, $Migrate - SD_{tgt}$ simply acknowledges that the TEE profile migration is completed successfully (*Signed Ack*). On the other hand, the security policy of the service admits to conserve only one profile. The TEE profile in the source should be destroyed. In order to ensure a fair exchange, exchanges between $Migrate - SD_{src}$ and $Migrate - SD_{tgt}$ must be performed via the service provider. $Migrate - SD_{tgt}$ acknowledges that the TEE profile migration is completed successfully (*Signed Ack*). At this time, $Migrate - SD_{src}$ asks the trusted application $Destroy - SD_{src}$ to destroy the SD corresponding to ID_{SP}. When the operation finishes, $Migrate - SD_{src}$ informs the service provider that the transfer is accomplished. Hence, the service provider will not consider any more TEE source as a client and revoke its corresponding keys.

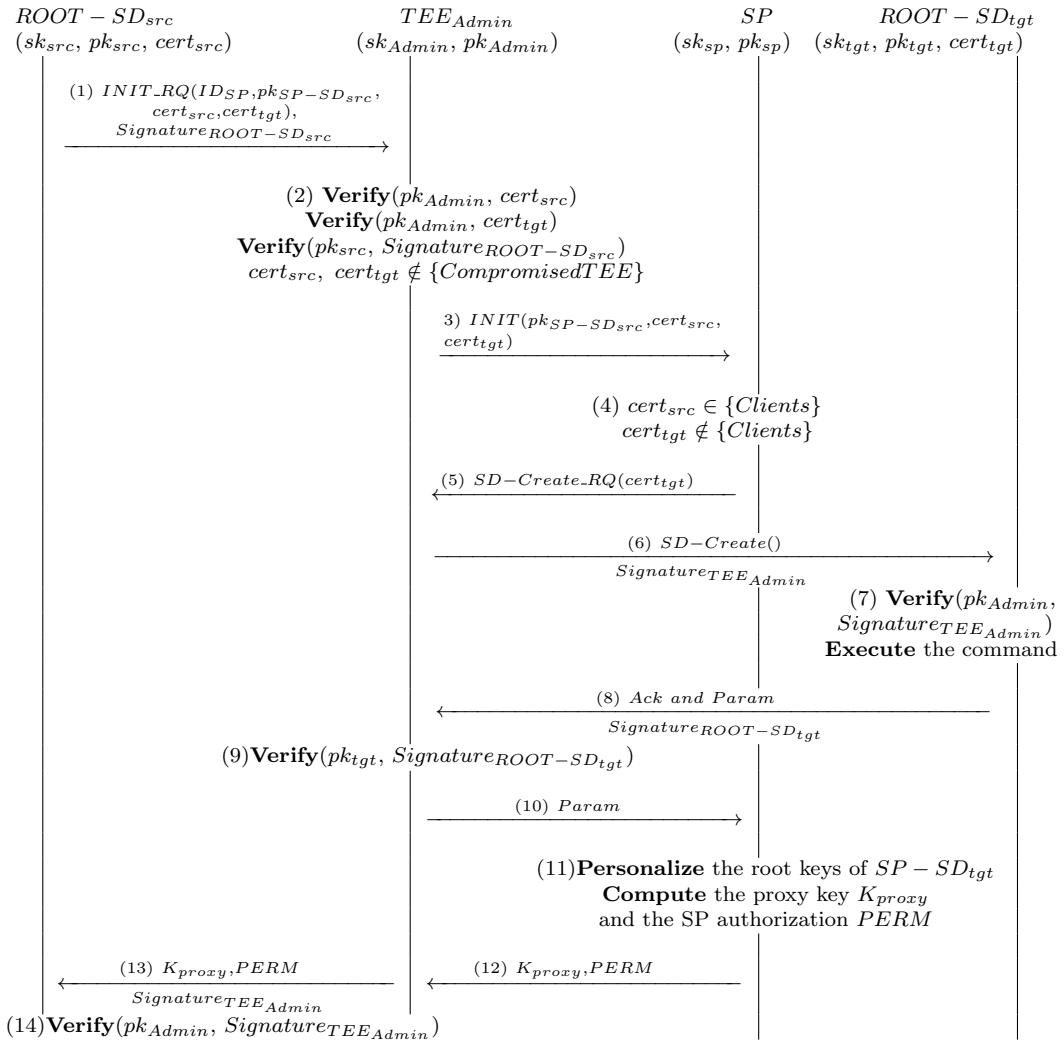


FIGURE 8.3: Service Provider Authorization Protocol

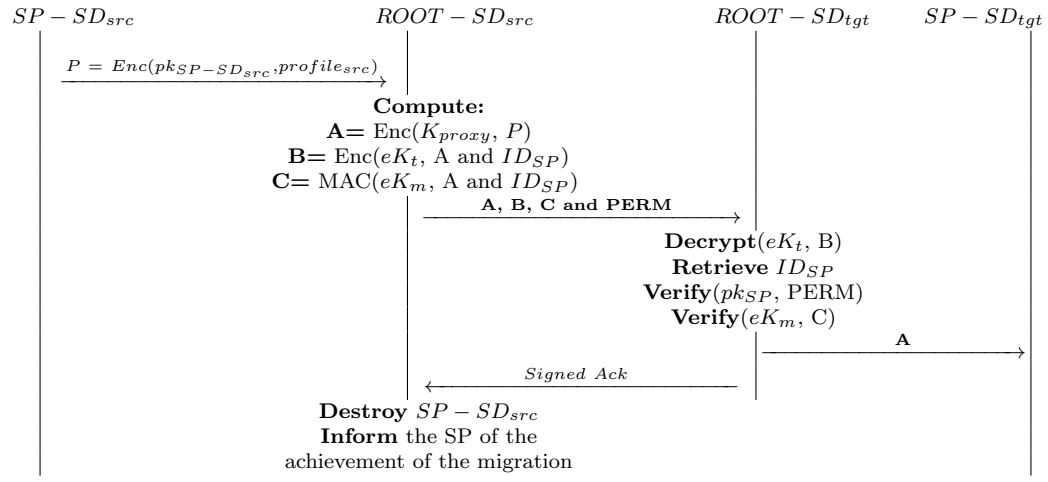


FIGURE 8.4: Profile Migration Protocol

8.5.4 Performance Remarks

As current TEE implementation does not give access to low level cryptographic primitives we cannot implement the whole protocol. To give an idea of performances, the reader should note that TEEs exploit the CPU of the smartphone with an amount of RAM of some MBs. Thus performance are comparable with what can be obtained in the Rich OS. For example, a RSA computation is achieved in 20 ms on a Galaxy SIII smartphone. Our reencryption scheme needs lower than a RSA computation: we measured, on a Galaxy SIII a reencryption time of about 4 ms.

8.6 Security Analysis

User Identification During a TEE profile migration, it is important to ensure that the target TEE (target device) belongs to the owner of the source TEE (source device). In our model, this is guaranteed by the concept of demonstrative identification [128]. Indeed, we proposed to run the migration protocol over a wireless proximity technology (NFC).

Requirements Analysis During the migration, an outsider or a curious TEE Admin must not be able to read or modify the transferred TEE profile (R1, R2). This is ensured by using the cryptographic re-encryption method. Indeed, the migration authorization, delivered by the service provider, consists of two components: K_{proxy} and $PERM$. $PERM$ is a signature computed by the service provider on (ID_{SP}, $cert_{src}$, $cert_{tgt}$, $timeStamp$). An attacker would not be able to replay this authorization because of the timestamp. Moreover, the transfer process would fail if $cert_{src}$ (resp. $cert_{tgt}$) does not correspond to the certified root public key used by source TEE (resp. target) during the AKA phase. Regarding the re-encryption key K_{proxy}, it is computed based on the private key of the source SD and the public key of the target SD. This means that a cipher text of source SD, if encrypted using K_{proxy}, will be converted to a cipher text of target SD. Thus, only source SD, target SD and the service provider have access (read / modify) to the TEE profile. If the re-encryption key K_{proxy} is improperly computed, the attacker cannot get the TEE profile content.

TEE Admin Trustworthy Besides the cryptographic solution, our approach relies on the trustworthy of the TEE Admin. We assume that a TEE Admin can only be honest-but-curious and not malicious (compromised). Indeed, a malicious TEE Admin can get access to service provider credentials and user's private data. However, we estimate that the assumption of a honest-but-curious TEE Admin is reasonable. This is because a malicious TEE Admin (when detected) risks not only huge financial damages but also his reputation. Knowing that this role should be played by the device manufacturer or the mobile network operator. In our opinion, this risk is far from being taken.

8.7 Protocol Validation

We validated our protocol using the AVISPA [9] tool web interface. AVISPA is an automated tool for the validation of security protocols. It takes as input a protocol modelled in High-Level Protocol Specification Language (HLPSL). This latter is translated into Intermediate Format (IF) and forwarded to the back-end analyser tools.

In Appendix A, we show (the core subset of) our migration protocol model written in HLPSL. In this validation model, we mainly focused on Service Provider authorization protocol (Figure 8.3) and Profile transfer protocol (Figure 8.4). Therefore, we assumed that the user authentication and AKA steps have been successfully achieved. Moreover, for the sake of simplicity, we did not consider the $SP - SD_{tgt}$ root key personalization.

We modeled our transfer protocol into six roles in addition to two standard roles (i.e. "`session`" and "`environment`"). First, the role "`sdSrc`" (resp. "`sdTgt`") refers to the $SP - SD_{src}$ (resp. $SP - SD_{tgt}$). Then, the role "`src`" (resp. "`tgt`") represents the Migration TA within TEE source (resp. target TEE). At last, "`teeAdmin`" and "`sp`" respectively correspond to the entities TEE admin and Service Provider. Every role is modelled into a state transition system. A state represents the reception and/or the transmission of a message from our protocol. For instance, "`State` = `0`" in the role "`teeAdmin`" corresponds to the reception of $INIT\_RQ$ by the TEE administrator in Figure 8.3. Regarding the role called "`session`", it represents a single session of our protocol where all the other roles are instantiated.

All the roles communicate over Dolev-Yao [122] channels (`channel(dy)`), i.e., an adversary can fully control the communication channels (A1). The attacker knowledge is defined by the set of constants or variables of the `intruder_knowledge` set in the main role (`environment`) (A2, A3). Then, the intruder actions are modeled by the combination of several sessions where the intruder may take part of the sessions running. On the subject of our protocol, besides the initialization of `intruder_knowledge`, we modeled our attacker by the variable `i` (`i` for intruder). We note that the attacker i did not compromise the SP and the TEE Admin nor their SDs (A4-i / A4-ii) because the roles "sdsrc", "sdtgt" and "spagent" are not played by the attacker in the initialized session (Line 198).

The migration authorization, delivered by the service provider, consists of two components: K_{proxy} and $PERM$. $PERM$ is a signature computed by the SP on (ID_{SP}, $cert_{src}$, $cert_{tgt}$, $timeStamp$). Regarding K_{proxy}, it is not a standard cryptographic tool. Thus, AVISPA does not have its predefined predicate. Our model must manually put up all its features. We designed the proxy re-encryption concept owing to the predicate $\wedge$`equal({EncSD}_KProxy,{SDCred}_PKSDtgt)` at the end of the role "`sdSrc`". This predicate models the equality between "the encryption of `EncSD` (the encryption of `SDCred` using the public key of the source SD) using the proxy key" and "the encryption of `SDCred` using the public key of the target SD". If this equality does not hold, it means that K_{proxy} is a fake key from an attacker which should be assimilated to a denied authorization of the SP.

The HLPSL language provides four predicates to model security requirements. The predicate `secret(E,id,S)` declares the information `E` as secret shared by the agents of set `S`. This security goal will be identified by `id` in the goal section. In addition, `witness`, `request` and `wreuqest` are used to model authentication goals. Regarding

the security requirements R1 (integrity) and R2 (confidentiality with respect to outsiders and a curious TEE Admin), we defined them in one goal owing to the predicate $\texttt{secret}(\texttt{SDCred}, \texttt{sec}_\texttt{S}\texttt{DCred}, \{\texttt{SDsrc}, \texttt{SP}, \texttt{SDtgt}\})$ at the end of the role "sdSrc". This predicate expresses that the content of an SD should remain secret between the SD of TEE source, the service provider and the SD of the target TEE.

We successfully validated our protocol with two AVISPA back-ends (AtSe and OFMC). The AtSe back-end extracts attacks that defeat the security properties by translating the model in constraints on the adversary's knowledge. Using a unification algorithm it integrates at each step of the protocol the new constraints. As our protocol is loop free, the search of possible attacks is complete. Regarding the OFMC back-end, it builds the infinite tree defined by the protocol analysis problem in a demand-driven way.

8.8 Conclusion

In this chapter, we have introduced a TEE architecture based on security domains. The root security domain is controlled by the TEE admin and the other security domains isolate the service providers trusted applications. With such an architecture, we have proposed a practical and privacy-preserving TEE profile migration protocol. This protocol requires the dynamic interaction of the service provider and the TEE admin. Owing to the security and functional characteristics of the used re-encryption method, the integrity and the confidentiality of the TEE profile, with respect to external attackers and TEE Admin, are guaranteed. Finally, we successfully validated our protocol using the AVISPA tool.

Chapter 9

Conclusion and Perspectives

(The french version is below)

In this thesis, we presented contributions related to contactless mobile services. In addition to a secure TEE migration protocol, we presented three privacy-preserving contributions, i.e., a new set-membership proof scheme and two new privacy-preserving protocols for the public transport service.

We first introduced a new set-membership proof enabling a prover to prove in a zero-knowledge way that he holds a secret belonging to a public set. The main idea is to prove that the user knows a public signature on this secret without revealing neither the secret nor the signature. Unlike the previous constructions, our set-membership proof does not require pairing computations at the prover side and in certain use cases at the verifier side too. This is very advantageous for resource constrained environments like the SIM cards as we will show in the implementation of our prototypes.

Then, we proposed a privacy-preserving mobile pass protocol based on Direct Anonymous Attestation and Camenisch-Lysanskaya signature schemes. This protocol enables a user to use the transport service without being tracked by the transport authority. We adapted the existing schemes in order to provide a revocation option. Owing to this property, the anonymity of a user can be lifted in extreme circumstances such as a judge request or a fraud. We also provided other security features like the anti-pass back or the black list management.

Afterwards, we provided a new privacy-preserving mobile ticketing protocol. Our protocol enables a user to use a book of m-tickets while ensuring the unlinkability of his trips. To this end, we used our new set-membership proof and provided several optimizations of Boneh-Boyen signatures in order to make it more efficient. Our protocol ensures a strict anonymity: even the index of a m-ticket must not be revealed during a m-ticket validation. We also defined a new feature: a secure post-payment mode where a user can post-pay his usage without questioning his privacy. Moreover, we proposed countermeasures against m-tickets cloning and double spending.

After building these protocols, for every one, we specified a framework, defined the security and privacy requirements in a game-based model, and proved their security in the random oracle model.

In addition to security and privacy requirements, that we specified in the security models, the transport service imposes a stringent time constraint, i.e., the validation time of a transport product must be less than 300 *ms*. We therefore implemented our protocols on a NFC-enabled SIM card in order to verify that our proposals satisfy this functional requirement. Indeed, we respected the time constraint regarding the validation time of the m-pass and the m-ticketing protocols. Our prototypes also have a major asset: they work, with a limited degradation of performance, even when the smartphone is off or the battery is flat.

Finally, the protocols previously presented can be implemented on TEE. Therefore, we focused on the security of TEE data and software. We presented a new TEE ecosystem and internal architecture in addition to a new approach to securely transfer TEE credentials and private data from a TEE to another one. Indeed, we organized the TEE into isolated security domains and introduced a new role, i.e., a TEE Admin who manages the TEE security by handling the creation and deletion of security domains, the download and installation of packages in SDs, and also the migration of SDs from a TEE to another compliant TEE. The main idea of the migration protocol is the use of a proxy re-encryption scheme. We validated our protocol using an automated security protocol validation tool.

Other interesting issues related to TEE security or privacy in mobile contactless services remain to be addressed.

We think that it is important to look at issues related to TEE credential attestation (external / internal). Indeed, as mentioned previously trusted applications within the TEE offer secure services to applications within the standard mobile OS. In order to perform their functions, the trusted applications handle secrets and credentials. If the credentials of a trusted application have been compromised, the offered service is not any more secure and trustworthy. Therefore, it is important to be able to prove that the used credentials have been generated and kept within the TEE, have not been revoked, and that the trusted application using these credentials is authorised to use them, i.e., the used credentials do not belong to another trusted application.

As regards to users' privacy, many privacy enhancing cryptographic tools exist, like the group signature schemes used in this thesis. It would be interesting to apply the optimizations introduced in this thesis in the context of other services like e-voting. Such application bears security requirements and constraints similar to the public transport use case. Indeed, in an e-voting system, voters must be authenticated (to verify that they are entitled to vote) while keeping their votes secret. This is quite alike to what is required in a transport service where subscribers should be authenticated whilst ensuring the anonymity of their transport product. Besides to these basic requirements, in e-voting system, two additional properties should be satisfied: voters should be able to verify that not only their own votes have been taken into account in the final tabulation but also those of other voters. Moreover, they should not be able to prove how they voted in order to render useless any attempt of bribery or coercion. Although these properties do not have their counterparts in the transport context, the last one (coercion-resistance) is usually obtained by using "anonymous credentials", which are generally based on group signatures. We believe that our optimizations on group signatures could significantly enhance the efficiency of coercion-resistant voting schemes.

Conclusion et perspectives

Dans cette thèse, nous avons présenté plusieurs contributions liées aux services mobiles sans contact, à savoir, une nouvelle preuve d'appartenance à un ensemble, deux nouveaux protocoles permettant de préserver la vie privée d'usagers des transports publics ainsi qu'un protocole sécurisé de transfert de secrets d'un TEE à un autre.

Notre première contribution consiste en une nouvelle preuve, à divulgation nulle de connaissance, permettant à un prouveur de convaincre un vérifieur que son secret, dont il a révélé une mise en gage, appartient à un ensemble public. L'idée principale est de prouver que l'utilisateur connaît une signature publique sur ce secret sans révéler ni le secret ni la signature. Contrairement aux constructions précédentes, notre preuve d'appartenance à un ensemble ne nécessite pas de calculs de couplages du côté du prouveur voire dans certains cas du côté du vérifieur. Ceci est très avantageux pour les environnements à ressources limitées comme les cartes SIM, comme cela a été illustré à travers l'implémentation de nos différents prototypes.

Nous avons ensuite proposé une carte d'abonnement aux transports publics, similaire au passe Navigo de la RATP mais offrant de meilleures garanties en termes de sécurité et de respect de la vie privée. Le détenteur d'une telle carte peut aussi prouver à une borne de transport qu'il dispose d'un titre de transport valide sans être tracé par l'autorité de transport. Nous nous somme appuyés pour cela sur des accréditations anonymes et des schémas de signature de type Camenisch-Lysanskaya. L'anonymat d'un usager peut toutefois être levée dans des circonstances exceptionnelles à la demande d'un juge par exemple ou bien en cas de fraude.

Nous avons également proposé un système de billetterie électronique pour les transports publics. Ce système est fondé sur l'utilisation de carnet de tickets anonymes et intraçables. Pour ce faire, nous avons utilisé notre nouvelle preuve d'appartenance à un ensemble et fourni plusieurs optimisations des signatures Boneh-Boyen. Notre système garantit un niveau d'anonymat élevé, même l'indice d'un m-ticket n'est pas révélé lors de sa validation. Nous offrons également une nouvelle fonctionnalité inédite pour ce type de systèmes: le post-paiement. Un utilisateur peut ainsi payer à postériori l'utilisation de ses m-tickets sans que cela ne remette en cause sa vie privée. En outre, nous avons proposé des contre-mesures contre le clonage de billets et les dépenses multiples.

Pour chacun de ces systèmes (carte d'abonnement et m-ticketing), nous avons spécifié le modèle de sécurité et formellement prouvé leur sécurité dans le modèle de l'oracle aléatoire.

Les services de transport imposent un cahier des charges très contraignant, en particulier, le temps de validation d'un titre de transport doit être inférieur à 300 ms. Nous avons donc implémenté nos protocoles sur une carte SIM NFC afin de vérifier que nos

différentes propositions satisfont bien cette exigence fonctionnelle. Pour les deux cas, nos prototypes confirment que nous respectons bien cette contrainte de temps. Nos prototypes ont aussi un atout majeur: ils fonctionnent, avec une certaine dégradation limitée des performances, même lorsque le mobile est éteint ou que sa batterie est déchargée.

Enfin, les protocoles présentés précédemment peuvent être aussi implémentés sur le TEE. Par conséquent, nous nous sommes concentrés sur la sécurité des secrets dans le TEE. Nous avons présenté un nouvel écosystème du TEE et une nouvelle architecture interne en plus d'une nouvelle approche pour le transfert sécurisé de secrets d'un TEE à un autre. Nous avons organisé à cet effet le TEE en domaines de sécurité isolés et introduit un nouveau rôle, à savoir, un "TEE Admin" qui gère la sécurité du TEE notamment la création et la suppression des domaines de la sécurité, le téléchargement et l'installation de paquets dans un domaine de sécurité, et aussi la migration des domaines de sécurité d'un TEE à un autre TEE. L'idée principale du protocole de migration réside dans l'utilisation d'un système de proxy de re-chiffrement. Nous avons validé la sécurité de notre protocole en utilisant un outil automatique de vérification de protocole de sécurité.

D'autres questions intéressantes liées à la sécurité du TEE ou bien au respect de la vie privée dans les services mobiles sans contact se posent.

Nous pensons qu'il est important d'examiner les questions liées à l'attestation des secrets du TEE (interne / externe). En effet, comme mentionné précédemment les applications de confiance du TEE offrent des services sécurisés aux applications de l'environnement mobile standard. Afin d'assurer leurs fonctions, les applications de confiance manipulent des secrets et des données privées. Si les secrets d'une application de confiance ont été compromis, le service offert n'est plus sûr. Par conséquent, il est important d'être en mesure de prouver que les secrets utilisés ont été générés et conservés dans le TEE, n'ont pas été révoqués, et que l'application de confiance qui les utilise est autorisée à le faire, autrement dit, les secrets utilisés n'appartiennent pas à une autre application de confiance.

En ce qui concerne le respect de la vie privée des utilisateurs, beaucoup d'outils cryptographiques existent, comme les schémas de signature de groupe utilisés dans cette thèse. Il serait intéressant d'appliquer les optimisations introduites dans cette thèse dans le contexte d'autres services comme le vote électronique (e-vote). Cette application a des exigences de sécurité et des contraintes similaires à celles d'un service de transport en commun. En effet, dans un système de vote électronique, les électeurs doivent être authentifiés (pour vérifier qu'ils sont autorisés à voter) mais leur vote doit rester secret. Ceci est tout à fait semblable à ce qui est requis dans un service de transport, où les abonnés doivent être authentifiés, sans compromettre l'anonymat de leurs titres de transport. Outre ces exigences de base, dans un système de vote électronique, deux autres propriétés de sécurité doivent être satisfaites: les électeurs devraient être en mesure de vérifier que non seulement leurs propres votes ont été pris en compte dans le dépouillement final, mais aussi ceux des autres électeurs. En outre, ils ne devraient pas être en mesure de prouver comment ils ont voté afin de déjouer toute tentative de coercition ou de corruption. Bien que ces propriétés n'aient pas d'équivalent dans le contexte du transport, la dernière (la propriété de résistance à la coercition) est généralement obtenue en utilisant des " accréditations anonymes ", qui sont souvent basées sur les signatures de groupe. Nous pensons donc que nos optimisations sur les signatures de groupe pourraient améliorer considérablement l'efficacité des systèmes de vote proposant des contre-mesures aux les tentatives de coercition ou de corruption.

Thesis Publications

International Journal

- **Ghada Arfaoui**, Guillaume Dabosville, Sébastien Gambs, Patrick Lacharme, and Jean-François Lalande. A Privacy-Preserving NFC Mobile Pass for Transport Systems. EAI Endorsed Transactions on Mobile Communications Applications, 14(5), 2014. doi:10.4108/mca.2.5.e4.
- **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 of Privacy Enhancing Technologies (PoPETs), 2015.2 : 25-45, June 2015. doi:10.1515/popets-2015-0019

International Conferences

- **Ghada Arfaoui**, Sébastien Gambs, Patrick Lacharme, Jean-François Lalande, Roch Lescuyer, and Jean Claude Paillès. A Privacy-Preserving Contactless Transport Service for NFC Smartphones. In Mobile Computing, Applications, and Services - 5th International Conference, MobiCASE 2013, Paris, France, November 7-8, 2013, Revised Selected Papers, pages 282-285, 2013. doi:10.1007/978-3-319-05452-0_24.
- **Ghada Arfaoui**, Saïd Gharout, and Jacques Traoré. Trusted Execution Environment. In The International Workshop on Trusted Platforms for Mobile and Cloud Computing (TPMCC) held at Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on, pages 259-266, Oxford, UK, April 2014. doi:10.1109/MobileCloud.2014.47.
- **Ghada Arfaoui**, Saïd Gharout, Jean-François Lalande, and Jacques Traoré. In Information Security Theory and Practice - 9th IFIP WG 11.2 International Conference, WISTP 2015 Heraklion, Crete, Greece, August 24-25, 2015 Proceedings, pages 153-168, 2015. doi:10.1007/978-3-319-24018-3_10.

National Conferences

- **Ghada Arfaoui** and Jean-François Lalande. A Privacy Preserving Post-Payment Mobile Ticketing Protocol for Transports Systems. In Atelier sur la Protection de la Vie Privée, APVP 2014, Cabourg, France, June 2014.

- Ghada Arfaoui, Guillaume Dabosville, Sébastien Gambs, Patrick Lacharme, and Jean-François Lalande. Un pass de transport anonyme et intraçable pour mobile NFC. In Atelier sur la Protection de la Vie Privée 2014, APVP 2014, Cabourg, France, June 2014.

Appendix A

TEE Migration Protocol in HLPSL

```
role sdSrc(SDsrc,SrcTEE,SDtgt,SP: agent,
           PKSDsrc, Kproxy, PKSDtgt: public_key,
           SND, RCV: channel(dy))

played_by SDsrc
def=

  local
        State : nat,
        SDCred: text
init State:=0

transition
        1. State = 0   /\ RCV(start)
     =|>
     State' := 1
     /\ SDCred':=new()
     /\ SND({SDCred'}_PKSDsrc)
     /\ secret(SDCred, sec_SDCred,{SDsrc,SDtgt})
     /\ equal({{SDCred}_PKSDsrc}_Kproxy, {SDCred}_PKSDtgt)
end role

role src(SrcTEE, SDsrc, TgtTEE, TEEAdmin: agent,
           PKSDsrc, PKSrcTEE, PKTgtTEE, PKTEEAdmin: public_key,
           SK: symmetric_key,
           SND, RCV: channel (dy))

played_by SrcTEE
def=

  local
        State : nat,
        TimeStamp, Ts, SDCred: text,
        Ack, PERM: message,
        Kproxy: public_key
  init
      State:=0

  transition
   1. State = 0 /\ RCV({SDCred'}_PKSDsrc)
      =|>
```

```
42        State' := 1
43        /\ TimeStamp':= new()
44        /\ SND(PKSDsrc.PKSrcTEE.PKTgtTEE.{PKSDsrc.PKSrcTEE.PKTgtTEE.TimeStamp
      '}_inv(PKSrcTEE))
45
46     2. State = 1 /\ RCV(Kproxy'.PERM'.{Kproxy'.PERM'.Ts'}_inv(PKTEEAdmin))
47        =|>
48        State' := 2
49        /\ SND({{{SDCred}_PKSDsrc}_Kproxy}_SK)
50
51     3. State = 2  /\ RCV({Ack'}_SK)
52        =|>
53        State' := 3
54 end role
55
56 role teeAdmin(TEEAdmin, SrcTEE, TgtTEE, SP: agent,
57               PKSDsrc, PKTEEAdmin, PKSrcTEE, PKTgtTEE: public_key,
58               SK: symmetric_key,
59               SND, RCV: channel (dy))
60
61 played_by TEEAdmin
62 def=
63
64   local
65         State : nat,
66         TimeStamp, Param: text,
67         PERM, SDCreate, Ack: message,
68         Kproxy: public_key
69   init
70        State:=0
71
72   transition
73     1. State = 0   /\ RCV(PKSDsrc.PKSrcTEE.PKTgtTEE.{PKSDsrc.PKSrcTEE.
      PKTgtTEE.TimeStamp'}_inv(PKSrcTEE))
74        =|>
75        State' := 1
76       /\ SND(PKSDsrc.PKSrcTEE.PKTgtTEE)
77
78     2. State = 1 /\ RCV(SDCreate')
79        =|>
80        State' := 2
81        /\ SND(SDCreate'.{SDCreate'}_inv(PKTEEAdmin))
82
83     3. State = 2 /\ RCV(Ack'.Param'.{Ack'.Param'}_inv(PKTgtTEE))
84        =|>
85        State' := 3
86        /\ SND(Param')
87
88     4. State = 3 /\ RCV(Kproxy'.PERM')
89        =|>
90        State' := 4
91        /\ TimeStamp':= new()
92        /\ SND(Kproxy.PERM.{Kproxy.PERM.TimeStamp'}_inv(PKTEEAdmin))
93 end role
94
95 role sp(SP, TEEAdmin: agent,
96         PKSDsrc, PKSrcTEE, PKTgtTEE, Kproxy: public_key,
97         SND, RCV: channel (dy))
98
99 played_by SP
100 def=
101
```

```
102   local
103         State : nat,
104         Param: text,
105         PERM, SDCreate: message
106
107   init
108       State:=0
109
110   transition
111    1. State = 0  /\ RCV(PKSDsrc.PKSrcTEE.PKTgtTEE)
112       =|>
113       State' := 1
114      /\ SND(SDCreate.PKTgtTEE)
115
116    2. State = 1  /\ RCV(Param')
117       =|>
118       State' := 2
119      /\ SND(Kproxy.PERM)
120 end role
121
122 role tgt(TgtTEE, SrcTEE, TEEAdmin: agent,
123          PKSDsrc, PKTgtTEE, PKTEEAdmin: public_key,
124          SK: symmetric_key,
125          SND, RCV: channel (dy))
126
127 played_by TgtTEE
128 def=
129
130   local
131         State : nat,
132         SDCred, Param: text,
133         Ack, SDCreate: message,
134         Kproxy: public_key
135   init
136       State:=0
137   transition
138    1. State = 0 /\ RCV(SDCreate'.{SDCreate'}_inv(PKTEEAdmin))
139           =|>
140           State':=1
141           /\ Param':= new()
142           /\ SND(Ack.Param'.{Ack.Param'}_inv(PKTgtTEE))
143    2. State = 1 /\ RCV({{{SDCred'}_PKSDsrc}_Kproxy'}_SK)
144       =|>
145       State' := 2
146       /\ Ack':= new()
147       /\ SND({Ack'}_SK)
148       /\ SND({{SDCred'}_PKSDsrc}_Kproxy)
149 end role
150
151 role sdTgt(SDtgt,TgtTEE: agent,
152            PKSDsrc: public_key,
153            SND, RCV: channel(dy))
154
155 played_by SDtgt
156 def=
157   local
158         State : nat,
159         SDCred: text,
160         Kproxy: public_key
161   init
162      State:=0
163   transition
```

```
164    1. State = 0    /\ RCV({{SDCred'}_PKSDsrc}_Kproxy')
165       =|>
166       State' := 1
167 end role
168
169 role session(SDsrc, SrcTEE, TgtTEE, TEEAdmin, SP, SDtgt: agent,
170              PKSDsrc, PKSrcTEE, PKTgtTEE, PKTEEAdmin, Kproxy,PKSDtgt:
     public_key,
171              SK: symmetric_key)
172
173 def=
174
175   local
176         SND0, SND1, SND2, SND3, SND4, SND5, RCV0, RCV1, RCV2, RCV3, RCV4,
     RCV5: channel (dy)
177
178   composition
179        sdSrc(SDsrc,SrcTEE,SDtgt,SP,PKSDsrc,Kproxy,PKSDtgt,SND0,RCV0)
180     /\ src(SrcTEE,SDsrc,TgtTEE,TEEAdmin,PKSDsrc,PKSrcTEE,PKTgtTEE,
     PKTEEAdmin,SK,SND1,RCV1)
181     /\ teeAdmin(TEEAdmin,SrcTEE,TgtTEE,SP,PKSDsrc,PKTEEAdmin,PKSrcTEE,
     PKTgtTEE,SK,SND3,RCV3)
182     /\ sp(SP,TEEAdmin,PKSDsrc,PKSrcTEE,PKTgtTEE,Kproxy,SND4,RCV4)
183     /\ tgt(TgtTEE,SrcTEE,TEEAdmin,PKSDsrc,PKTgtTEE,PKTEEAdmin,SK,SND2,RCV2)
184     /\ sdTgt(SDtgt,TgtTEE,PKSDsrc,SND5,RCV5)
185 end role
186
187 role environment()
188 def=
189  const sdsrc, srctee, tgttee, teeadmin, spagent, sdtgt : agent,
190        sec_SDCred : protocol_id,
191        pksdsrc, kproxy, pksrctee, pktgttee, pkteeadmin, pksdtgt: public_key
     ,
192        sk: symmetric_key
193
194  intruder_knowledge = {sdsrc,srctee,tgttee,teeadmin,spagent,sdtgt,kproxy,
     pksdsrc,pksrctee,pktgttee,pkteeadmin, pksdtgt}
195
196  composition
197
198         session(sdsrc,srctee,tgttee,teeadmin,spagent,sdtgt,pksdsrc,pksrctee
     ,pktgttee,pkteeadmin,kproxy,pksdtgt,sk)
199 end role
200
201 goal
202   secrecy_of sec_SDCred
203 end goal
204
205 environment()
```

Bibliography

[1] GlobalPlatform. *GlobalPlatform Card Specification - v2.2.1*, January 2011.

[2] Andy Rupp, Gesine Hinterwälder, Foteini Baldimtsi, and Christof Paar. P4R: Privacy-Preserving Pre-Payments with Refunds for Transportation Systems. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 205–212. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39883-4. doi:10.1007/978-3-642-39884-1_17. URL http://dx.doi.org/10.1007/978-3-642-39884-1_17.

[3] Ramon T. Llamas and William Stofega. Worldwide smartphone 2013-2017 forecast and analysis, http://www.idc.com/getdoc.jsp?containerId=239847, March 2013. URL http://www.idc.com/getdoc.jsp?containerId=239847.

[4] John Jackson. Worldwide and U.S. Mobile Applications Download and Revenue 2013-2017 Forecast: The App as the Emerging Face of the Internet, http://www.idc.com /getdoc.jsp? containerId =241295, December 2013. URL http://www.idc.com/getdoc.jsp?containerId=241295.

[5] BIG Brother Awards. Les gagnants Big Brother Awards 12. URL http://www.bigbrotherawards.be/index.php/fr/.

[6] Christina Hager. Divorce Lawyers Using Fast Lane To Track Cheaters. URL http://msl1.mit.edu/furdlog/docs/2007-08-10_wbz_fastlane_tracking.pdf.

[7] GSMA Mobile NFC. White Paper: Mobile NFC in Transport. http://www.uitp.org/public-transport/technology/Mobile-NFC-in-Transport.pdf, September 2012.

[8] Christian Funk and Maria Garnaeva. Kaspersky Security Bulletin 2013. Overall statistics for 2013, http://media.kaspersky.com/pdf/KSB_2013_EN.pdf, December 2013.

[9] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P.Hankes Drielsma, P.C. Heam, O. Kouchnarenko, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigano, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In Kousha Etessami and SriramK. Rajamani, editors, *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-27231-1. doi:10.1007/11513988_27. URL http://dx.doi.org/10.1007/11513988_27.

[10] Asia Pacific Smart Card Association (APSCA), Korea Smart Card Co. Ltd (KSCC), and The Seoul Metropolitan Government. 4th Asian Transport Revenue Collection Forum. URL http://www.apsca.org/events/info.php?event=121.

[11] Octopus Company. Octopus History. URL http://www.octopus.com.hk/about-us/corporate-profile/our-history/en/index.html.

[12] Press Center: Aruna Sharma, Vanessa Clarke, and Marta Bordonada. Gemplus Launches World's First Contactless Combi Card for Mobile Payment. URL http://www.gemalto.com/press-site/gemplus/2003/banking/contactlesscombicardformobilepayment06022003.htm.

[13] Judith Vanderkay NFC Forum. Nokia, Philips And Sony Establish The Near Field Communication (NFC) Forum, . URL http://nfc-forum.org/newsroom/nokia-philips-and-sony-establish-the-near-field-communication-nfc-forum/.

[14] NFC Forum. NFC and Contactless Technologies, . URL http://nfc-forum.org/what-is-nfc/about-the-technology/.

[15] Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 17 and Cards and personal identification. *ISO/IEC 14443:2008*, June 2008. URL https://www.iso.org/obp/ui/#iso:std:iso-iec:14443:-1:ed-2:v1:en.

[16] NFC Forum. NFC Forum Technical Specifications, . URL http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/nfc-forum-technical-specifications/.

[17] NFC. History of Near Field Communication. URL http://www.nearfieldcommunication.org/history-nfc.html/.

[18] IHS Pressroom. NFC-Enabled Cellphone Shipments to Soar Fourfold in Next Five Years. URL http://press.ihs.com/press-release/design-supply-chain/nfc-enabled-cellphone-shipments-soar-fourfold-next-five-years.

[19] Gemalto security to be free. Why the future of ticketing is mobile. URL http://review.gemalto.com/post/why-the-future-of-ticketing-is-mobile.

[20] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *ACM Conference on Computer and Communications Security*, pages 1087–1098, Berlin, Germany, 2013. doi:10.1145/2508859.2516687.

[21] Macià Mut-Puigserver, M. Magdalena Payeras-Capellà, Josep-Lluís Ferrer-Gomila, Arnau Vives-Guasch, and Jordi Castellà-Roca. A survey of electronic ticketing applied to transport. *Computers & Security*, 31(8):925–939, November 2012. ISSN 01674048. doi:10.1016/j.cose.2012.07.004.

[22] NFC Forum. What It Does, . URL http://nfc-forum.org/what-is-nfc/what-it-does/.

[23] ETSI. *TR 102 216, V3.0.0, Smart cards; Vocabulary for Smart Card Platform specifications*, 2003-09.

[24] (U)SIM Java Card Platform Protection Profile Basic and SCWS Configurations-Evolutive Certification Scheme for (U)SIM cards, Version 2.0.2. http://www.ssi.gouv.fr/IMG/certificat/ANSSI-CC-cible_PP-2010-04en.pdf, June 2010.

[25] Samuel A. Bailey, Don Felton, Virginie Galindo, Franz Hauswirth, Janne Hirvimies, Milas Fokle, Fredric Morenius, Christophe Colas, Jean-Philippe Galvan,

Gil Bernabeu, and Kevin Gillick. White paper: The trusted execution environment: Delivering enhanced security at a lower cost to the mobile market, February 2011.

[26] Ghada Arfaoui, Saïd Gharout, and Jacques Traoré. Trusted Execution Environments: A Look under the Hood. In *The International Workshop on Trusted Platforms for Mobile and Cloud Computing (TPMCC) held at Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, pages 259–266, Oxford, UK, April 2014. doi:10.1109/MobileCloud.2014.47. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6834973&isnumber=6823830.

[27] N. Asokan, Jan Erik Ekberg, and Kari Kostiainen. The untapped potential of trusted execution environments on mobile devices. *IEEE Security And Privacy*, 12(4):293–294, August 2013. ISSN 03029743. doi:10.1007/978-3-642-39884-1_24.

[28] 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. ISBN 978-3-540-16463-0. doi:10.1007/3-540-39799-X_31. URL http://dx.doi.org/10.1007/3-540-39799-X_31.

[29] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177): 203–209, 1987.

[30] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. 106. Springer-Verlag, 1986. ISBN 978-1-4757-1920-8.

[31] 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. doi:10.1145/103418.103434. URL http://doi.acm.org/10.1145/103418.103434.

[32] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993. doi:10.1109/18.259647. URL http://dx.doi.org/10.1109/18.259647.

[33] 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. doi:10.1007/10722028_23. URL http://dx.doi.org/10.1007/10722028_23.

[34] André Weil. Sur les fonctions algébriques de constantes finies. In *Comptes rendu de l'Académie des sciences*, volume 210, pages 592–594. 1940.

[35] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. In *Mathematics of Computation*, volume 62 of *206*, pages 865–874. American Mathematical Society, 1994.

[36] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. doi:10.1112/plms/s2-42.1.230. URL http://plms.oxfordjournals.org/content/s2-42/1/230.short.

[37] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 33–43, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8. doi:10.1145/73007.73011. URL http://doi.acm.org/10.1145/73007.73011.

[38] Ivan Bjerre Damgård. Collision free hash functions and public key signature schemes. In David Chaum and WynL. Price, editors, *Advances in Cryptology - EUROCRYPT' 87*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216. Springer Berlin Heidelberg, 1988. ISBN 978-3-540-19102-5. doi:10.1007/3-540-39118-5_19. URL http://dx.doi.org/10.1007/3-540-39118-5_19.

[39] W. Diffie and M.E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Nov 1976. ISSN 0018-9448. doi:10.1109/TIT.1976.1055638.

[40] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144, Helsinki, Finland, may 1998. Springer.

[41] Sébastien Canard, Julien Devigne, and Fabien Laguillaumie. Improving the Security of an Efficient Unidirectional Proxy Re-Encryption Scheme. *Journal of Internet Services and Information Security (JISIS)*, 1(2/3):140–160, August 2011.

[42] David Chaum and Eugene van Heyst. Group signatures. In DonaldW. Davies, editor, *Advances in Cryptology - EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Berlin Heidelberg, 1991. ISBN 978-3-540-54620-7. doi:10.1007/3-540-46416-6_22. URL http://dx.doi.org/10.1007/3-540-46416-6_22.

[43] Sébastien Canard, Berry Schoenmakers, Martijn Stam, and Jacques Traoré. List signature schemes. *Discrete Applied Mathematics*, 154(2):189–201, 2006. doi:10.1016/j.dam.2005.08.003.

[44] 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, New York, NY, USA, 2004. ACM. ISBN 1-58113-961-6. doi:10.1145/1030083.1030103.

[45] 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. ISBN 978-3-540-18047-0. doi:10.1007/3-540-47721-7_12.

[46] Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient fair payment system. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS '96, pages 88–94, New York, NY, USA, 1996. ACM. ISBN 0-89791-829-0. doi:10.1145/238168.238193. URL http://doi.acm.org/10.1145/238168.238193.

[47] Andrew M. Odlyzko. Discrete logarithm and smooth polynomials. In Gary L. Mullen and Peter Jau-Shyong Shiue, editors, *Finite Fields: Theory, Applications and Algorithms*, volume 168 of *Contemporary Mathematics*, pages 269–278. American Mathematical Society, 1994.

[48] Kevin McCurley. The discrete logarithm problem. In Gary L. Mullen and Peter Jau-Shyong Shiue, editors, *Cryptology and computational number theory*, volume 42 of *Proceedings of Symposia in Applied Mathematics*, pages 49–74. American Mathematical Society, 1990.

[49] Patrik Bichsel, Jan Camenisch, Gregory Neven, NigelP. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In JuanA. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15316-7. doi:10.1007/978-3-642-15317-4_24. URL http://dx.doi.org/10.1007/978-3-642-15317-4_24.

[50] Bellare, Namprempre, Pointcheval, and Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003. ISSN 0933-2790. doi:10.1007/s00145-002-0120-1. URL http://dx.doi.org/10.1007/s00145-002-0120-1.

[51] Stefan A. Brands. An efficient off-line electronic cash system based on the representation problem. Technical report, Amsterdam, The Netherlands, The Netherlands, 1993.

[52] David Chaum and Hans van Antwerpen. Undeniable signatures. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO' 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer New York, 1990. ISBN 978-0-387-97317-3. doi:10.1007/0-387-34805-0_20. URL http://dx.doi.org/10.1007/0-387-34805-0_20.

[53] David Chaum. Zero-knowledge undeniable signatures (extended abstract). In IvanBjerre Damgard, editor, *Advances in Cryptology - EUROCRYPT'90*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer Berlin Heidelberg, 1991. ISBN 978-3-540-53587-4. doi:10.1007/3-540-46877-3_41. URL http://dx.doi.org/10.1007/3-540-46877-3_41.

[54] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In Matt Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22668-0. doi:10.1007/978-3-540-28628-8_3. URL http://dx.doi.org/10.1007/978-3-540-28628-8_3.

[55] 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. ISBN 978-3-540-21935-4. doi:10.1007/978-3-540-24676-3_4. URL http://dx.doi.org/10.1007/978-3-540-24676-3_4.

[56] 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. doi:10.1007/3-540-49162-7_18. URL http://dx.doi.org/10.1007/3-540-49162-7_18.

[57] Anna Lysyanskaya, RonaldL. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67185-5. doi:10.1007/3-540-46513-8_14. URL http://dx.doi.org/10.1007/3-540-46513-8_14.

[58] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984. ISSN 0022-0000. doi:http://dx.doi.org/10.1016/0022-0000(84)90070-9. URL http://www.sciencedirect.com/science/article/pii/0022000084900709.

[59] S. Goldwasser, S. Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 441–448, Oct 1984. doi:10.1109/SFCS.1984.715946.

[60] 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. ISSN 0097-5397. doi:10.1137/0217017. URL http://dx.doi.org/10.1137/0217017.

[61] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'96, pages 399–416, Berlin, Heidelberg, 1996. Springer-Verlag. ISBN 3-540-61186-X. URL http://dl.acm.org/citation.cfm?id=1754495.1754541.

[62] 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. ISBN 978-3-540-64892-5. doi:10.1007/BFb0055741. URL http://dx.doi.org/10.1007/BFb0055741.

[63] 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. ISBN 978-3-540-43319-4. doi:10.1007/3-540-45861-1_1. URL http://dx.doi.org/10.1007/3-540-45861-1_1.

[64] 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. ISBN 0-89791-629-8. doi:10.1145/168588.168596. URL http://doi.acm.org/10.1145/168588.168596.

[65] 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. ISBN 0-89791-962-9. doi:10.1145/276698.276741. URL http://doi.acm.org/10.1145/276698.276741.

[66] 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. ISBN 978-3-540-21000-9. doi:10.1007/978-3-540-24638-1_3. URL http://dx.doi.org/10.1007/978-3-540-24638-1_3.

[67] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-21935-4. doi:10.1007/978-3-540-24676-3_11. URL http://dx.doi.org/10.1007/978-3-540-24676-3_11.

[68] 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. ISBN 978-3-540-24454-7. doi:10.1007/978-3-540-30580-4_28. URL http://dx.doi.org/10.1007/978-3-540-30580-4_28.

[69] Yevgeniy Dodis. Efficient Construction of (Distributed) Verifiable Random Functions. In YvoG. Desmedt, editor, *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00324-3. doi:10.1007/3-540-36288-6_1. URL http://dx.doi.org/10.1007/3-540-36288-6_1.

[70] TorbenPryds Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology-CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Berlin Heidelberg, 1992. ISBN 978-3-540-55188-1. doi:10.1007/3-540-46766-1_9. URL http://dx.doi.org/10.1007/3-540-46766-1_9.

[71] 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. ISBN 0-387-15658-5. URL http://dl.acm.org/citation.cfm?id=19478.19480.

[72] 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. ISBN 0-387-97317-6. URL http://dl.acm.org/citation.cfm?id=118209.118237.

[73] Pierre-Alain Fouque and Jacques Stern. Fully distributed threshold RSA under standard assumptions. In *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*, pages 310–330, 2001. doi:10.1007/3-540-45682-1_19. URL http://dx.doi.org/10.1007/3-540-45682-1_19.

[74] 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. ISSN 0001-0782. doi:10.1145/359340.359342. URL http://doi.acm.org/10.1145/359340.359342.

[75] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21(2):149–177, 2008. ISSN 0933-2790. doi:10.1007/s00145-007-9005-7. URL http://dx.doi.org/10.1007/s00145-007-9005-7.

[76] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *The 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 89–105, London, UK, UK, 1993. Springer-Verlag. ISBN 3-540-57340-2. URL http://dl.acm.org/citation.cfm?id=646757.705670.

[77] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer Berlin Heidelberg, Santa Barbara, CA, USA, 2004. ISBN 978-3-540-22668-0. doi:10.1007/978-3-540-28628-8_4.

[78] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A Non-interactive Range Proof with Constant Communication. In AngelosD. Keromytis, editor, *Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 179–199. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32945-6. doi:10.1007/978-3-642-32946-3_14. URL http://dx.doi.org/10.1007/978-3-642-32946-3_14.

[79] 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*, volume 8341 of *Lecture Notes in Computer Science*, pages 47–64. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-53996-1. doi:10.1007/978-3-642-53997-8_4. URL http://dx.doi.org/10.1007/978-3-642-53997-8_4.

[80] 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 (PoPETs)*, 2015.2:25–45, June 2015. doi:10.1515/popets-2015-0019.

[81] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient Protocols for Set Membership and Range Proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-89254-0. doi:10.1007/978-3-540-89255-7_15. URL http://dx.doi.org/10.1007/978-3-540-89255-7_15.

[82] Piotr Szczechowiak, LeonardoB. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks. In Roberto Verdone, editor, *Wireless Sensor Networks*, volume 4913 of *Lecture Notes in Computer Science*, pages 305–320. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77689-5. doi:10.1007/978-3-540-77690-1_19. URL http://dx.doi.org/10.1007/978-3-540-77690-1_19.

[83] AugustoJun Devegili, Michael Scott, and Ricardo Dahab. Implementing Cryptographic Pairings over Barreto-Naehrig Curves. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages

197–207. Springer Berlin Heidelberg, Tokyo, Japan, July 2007. ISBN 978-3-540-73488-8. doi:10.1007/978-3-540-73489-5_10. URL http://dx.doi.org/10.1007/978-3-540-73489-5_10.

[84] Conrado Porto Lopes Gouvêa, Leonardo B. Oliveira, and Julio López. Efficient software implementation of public-key cryptography on sensor networks using the MSP430X microcontroller. *J. Cryptographic Engineering*, 2(1):19–29, 2012. doi:10.1007/s13389-012-0029-z. URL http://dx.doi.org/10.1007/s13389-012-0029-z.

[85] Calypso Networks Association. Calypso handbook v1.1, 2010. URL http://www.calypsostandard.net/index.php/documents/specifications/public-documents/79-100324-calypso-handbook.

[86] Calypso Networks Association. Functional card application v1.4, 2010. URL http://www.calypsostandard.net/index.php/documents/specifications/public-documents/78-010608-functional-card-application.

[87] The project Touch and Travel, 2008. URL http://www.touchandtravel.de/.

[88] Sony Corp. Info. Mobile payment services using nfc sims equipped with sony felica^{TM} technology to begin in hong kong, October 2013. URL http://www.sony.net/SonyInfo/News/Press/201310/13-137E/.

[89] Gematlo. Gemalto enables commercial mobile nfc transport and payment roll-out in hong kong, October 2013. URL http://www.gemalto.com/php/pr_view.php?id=1685.

[90] Andrew Lee, Timothy Lui, and Bryon Leung. Security analysis of the octopus system, visited the 31th January 2013. URL http://www.proxmark.org/files/Documents/13.56%20MHz%20-%20Felica/security_analysis_of_octopus_smart_card_system.pdf.

[91] Jan-Erik Ekberg and Sandeep Tamrakar. Mass transit ticketing with NFC mobile phones. In Liqun Chen, Moti Yung, and Liehuang Zhu, editors, *Trusted Systems*, volume 7222 of *Lecture Notes in Computer Science*, pages 48–65. Springer Berlin Heidelberg, Beijing, China, 2012. ISBN 978-3-642-32297-6. doi:10.1007/978-3-642-32298-3_4.

[92] Sandeep Tamrakar and Jan-Erik Ekberg. Tapping and tripping with nfc. In Michael Huth, N. Asokan, Srdjan Čapkun, Ivan Flechais, and Lizzie Coles-Kemp, editors, *Trust and Trustworthy Computing*, volume 7904 of *Lecture Notes in Computer Science*, pages 115–132. Springer Berlin Heidelberg, London, United Kingdom, 2013. ISBN 978-3-642-38907-8. doi:10.1007/978-3-642-38908-5_9.

[93] Thomas S. Heydt-Benjamin, Hee-Jin Chae, Benessa Defend, and Kevin Fu. Privacy for public transportation. In *Proceedings of the 6th International Conference on Privacy Enhancing Technologies*, PET'06, pages 1–19, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-68790-4, 978-3-540-68790-0. doi:10.1007/11957454_1. URL http://dx.doi.org/10.1007/11957454_1.

[94] David Derler, Klaus Potzmader, Johannes Winter, and Kurt Dietrich. Anonymous ticketing for nfc-enabled mobile phones. In Liqun Chen, Moti Yung, and

Liehuang Zhu, editors, *Trusted Systems*, volume 7222 of *Lecture Notes in Computer Science*, pages 66–83. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32297-6. doi:10.1007/978-3-642-32298-3_5. URL http://dx.doi.org/10.1007/978-3-642-32298-3_5.

[95] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, 2000. ISBN 9780262024914.

[96] Glenn, A., Goldberg, I., Légaré, F., Stiglic, A. A description of protocols for private credentials, 2001. URL http://eprint.iacr.org/2001/082.

[97] A. P. Isern-Deya, A. Vives-Guasch, M. Mut-Puigserver, M. Payeras-Capella, and J. Castella-Roca. A secure automatic fare collection system for time-based or distance-based services with revocable anonymity for users. *The Computer Journal*, 56(10):1198–1215, April 2012. ISSN 0010-4620. doi:10.1093/comjnl/bxs033.

[98] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004. ISSN 0933-2790. doi:10.1007/s00145-004-0314-9. URL http://dx.doi.org/10.1007/s00145-004-0314-9.

[99] Megan Geuss. Japanese railway company plans to sell data from e-ticket records. Ars Technica. http://arstechnica.com/business/2013/07/japanese-railway-company-plans-to-sell-data-from-e-ticket-records/, July 2013.

[100] Ghada Arfaoui, Guillaume Dabosville, Sébastien Gambs, Patrick Lacharme, and Jean-François Lalande. A Privacy-Preserving NFC Mobile Pass for Transport Systems. *ICST Trans. Mobile Communications Applications*, 5:e4, 2014. doi:10.4108/mca.2.5.e4. URL http://dx.doi.org/10.4108/mca.2.5.e4.

[101] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000. doi:10.1007/s001450010003. URL http://dx.doi.org/10.1007/s001450010003.

[102] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004. URL http://eprint.iacr.org/2004/332.

[103] Seek for Android, 2013. http://code.google.com/p/seek-for-android/.

[104] PauloS.L.M. Barreto and Michael Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33108-7. doi:10.1007/11693383_22. URL http://dx.doi.org/10.1007/11693383_22.

[105] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *The Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 80–89, New Orleans, Louisiana, USA, 1991. ACM. ISBN 0-89791-397-3. doi:10.1145/103418.103434. URL http://doi.acm.org/10.1145/103418.103434.

[106] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers . *Discrete Applied Mathematics*, 156(16):3113 – 3121, 2008. ISSN 0166-218X. doi:http://dx.doi.org/10.1016/j.dam.2007.12.010. URL http://www.sciencedirect.com/science/article/pii/S0166218X08000449. Applications of Algebra to Cryptography.

[107] The Paris Convention and Visitors Bureau. Public transport in paris. /http://en.parisinfo.com/practical-paris/how-to-get-to-and-around-paris/public-transport-paris. [Online; accessed 26-October-2014].

[108] Berlin.de. Billets, tarifs et reseaux des lignes. http://www.berlin.de/fr/transports-en-commun/1772016-3000866-billets-tarifs-reseaux-des-lignes.fr.html. [Online; accessed 26-October-2014].

[109] Moscow. http://moscow.ru/fr/guide/trip_planning/inner_transport/transport/metro/. [Online; accessed 26-October-2014].

[110] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure Applications of Pedersen Distributed Key Generation Protocol. In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 373–390. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00847-7. doi:10.1007/3-540-36563-X_26. URL http://dx.doi.org/10.1007/3-540-36563-X_26.

[111] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. ISSN 0933-2790. doi:10.1007/BF00196725. URL http://dx.doi.org/10.1007/BF00196725.

[112] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In JuanA. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security*, volume 5888 of *Lecture Notes in Computer Science*, pages 226–247. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10432-9. doi:10.1007/978-3-642-10433-6_15. URL http://dx.doi.org/10.1007/978-3-642-10433-6_15.

[113] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42987-6. doi:10.1007/3-540-45682-1_21. URL http://dx.doi.org/10.1007/3-540-45682-1_21.

[114] Ghada Arfaoui, Saïd Gharout, Jean-François Lalande, and Jacques Traoré. Practical and privacy-preserving TEE migration. In *Information Security Theory and Practice - 9th IFIP WG 11.2 International Conference, WISTP 2015 Heraklion, Crete, Greece, August 24-25, 2015 Proceedings*, pages 153–168, 2015. doi:10.1007/978-3-319-24018-3_10. URL http://dx.doi.org/10.1007/978-3-319-24018-3_10.

[115] Claudio Marforio, Nikolaos Karapanos, Claudio Soriente, Kari Kostiainen, and Srdjan Capkun. Secure enrollment and practical migration for mobile trusted execution environments. *Third ACM workshop on Security and privacy in smartphones & mobile devices*, pages 93–98, 2013. ISSN 15437221. doi:10.1145/2516760.2516764.

[116] Kari Kostiainen, N. Asokan, and Alexandra Afanasyeva. Towards user-friendly credential transfer on open credential platforms. In Javier Lopez and Gene Tsudik, editors, *Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 395–412. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21553-7. doi:10.1007/978-3-642-21554-4_23. URL http://dx.doi.org/10.1007/978-3-642-21554-4_23.

[117] Kari Kostiainen, N Asokan, and Jan-erik Ekberg. Credential Disabling from Trusted Execution Environments. In *15th Nordic Conference on Secure IT Systems*, number 2, pages 171–186, Espoo, Finland, October 2012. Springer Berlin Heidelberg.

[118] Matthew Areno and Jim Plusquellic. Securing Trusted Execution Environments with PUF Generated Secret Keys. In *11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1188–1193, Liverpool, England, UK, June 2012. IEEE Computer Society. doi:10.1109/TrustCom.2012.255.

[119] Kari Kostiainen, Alexandra Dmitrienko, Jan-Erik Ekberg, Ahmad-Reza Sadeghi, and N. Asokan. Key Attestation from Trusted Execution Environments. In Alessandro Acquisti, SeanW. Smith, and Ahmad-Reza Sadeghi, editors, *Trust and Trustworthy Computing*, volume 6101 of *Lecture Notes in Computer Science*, pages 30–46. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13868-3. doi:10.1007/978-3-642-13869-0_3. URL http://dx.doi.org/10.1007/978-3-642-13869-0_3.

[120] Trusted Computing Group. TPM Main Specification. http://www.trustedcomputinggroup.org/resources/tpm_main_specification, 2015.

[121] Ahmad-Reza Sadeghi, Christian Stüble, and Marcel Winandy. Property-based tpm virtualization. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85884-3. doi:10.1007/978-3-540-85886-7_1. URL http://dx.doi.org/10.1007/978-3-540-85886-7_1.

[122] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society. doi:10.1109/SFCS.1981.32. URL http://dx.doi.org/10.1109/SFCS.1981.32.

[123] GlobalPlatform Device Committee. *TEE Protection Profile Version 1.2, Public Release, GPD_SPE_021*, November 2014.

[124] GlobalPlatform Device technology. *Trusted User Interface API, version 1.0*, June 2013.

[125] Jean-Sébastien Coron and Aline Gouget and Thomas Icart and Pascal Paillier. Supplemental Access Control (PACEv2): Security Analysis of PACE Integrated Mapping. In David Naccache, editor, *Cryptography and Security: From Theory to Applications*, volume 6805 of *Lecture Notes in Computer Science*, pages 207–232. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28367-3. doi:10.1007/978-3-642-28368-0_15. URL http://dx.doi.org/10.1007/978-3-642-28368-0_15.

[126] Jean-Sébastien Coron, Aline Gouget, Pascal Paillier, and Karine Villegas. SPAKE: A Single-Party Public-Key Authenticated Key Exchange Protocol for Contact-Less Applications. In Radu Sion, Reza Curtmola, Sven Dietrich, Aggelos Kiayias, JosepM. Miret, Kazue Sako, and Francesc Seb, editors, *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 107–122. Springer Berlin Heidelberg, Tenerife, Canary Islands, Spain, 2010. ISBN 978-3-642-14991-7. doi:10.1007/978-3-642-14992-4_11. URL http://dx.doi.org/10.1007/978-3-642-14992-4_11.

[127] Fabrizio Baiardi, Diego Cilea, Daniele Sgandurra, and Francesco Ceccarelli. Measuring Semantic Integrity for Remote Attestation. In Liqun Chen, ChrisJ. Mitchell, and Andrew Martin, editors, *2nd International Conference on Trusted Computing*, volume 5471 of *Lecture Notes in Computer Science*, pages 81–100. Springer Berlin Heidelberg, Oxford, UK, 2009. ISBN 978-3-642-00586-2. doi:10.1007/978-3-642-00587-9_6. URL http://dx.doi.org/10.1007/978-3-642-00587-9_6.

[128] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2002, San Diego, California, USA*, 2002. URL http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/balfan.pdf.

Ghada ARFAOUI

Conception de protocoles cryptographiques préservant la vie privée pour les services mobiles sans contact

Résumé :
Avec l'émergence de nouvelles technologies telles que le NFC (Communication à champ proche) et l'accroissement du nombre de plates-formes mobiles, les téléphones mobiles vont devenir de plus en plus indispensables dans notre vie quotidienne. Ce contexte introduit de nouveaux défis en termes de sécurité et de respect de la vie privée. Dans cette thèse, nous nous focalisons sur les problématiques liées au respect de la vie privée dans les services NFC ainsi qu'à la protection des données privées et secrets des applications mobiles dans les environnements d'exécution de confiance (TEE).

Nous fournissons deux solutions pour le transport public: une solution utilisant des cartes d'abonnement (m-pass) et une autre à base de tickets électroniques (m-ticketing). Nos solutions préservent la vie privée des utilisateurs tout en respectant les exigences fonctionnelles établies par les opérateurs de transport. À cette fin, nous proposons de nouvelles variantes de signatures de groupe ainsi que la première preuve pratique d'appartenance à un ensemble, à apport nul de connaissance, et qui ne nécessite pas de calculs de couplages du côté du prouveur. Ces améliorations permettent de réduire considérablement le temps d'exécution de ces schémas lorsqu'ils sont implémentés dans des environnements contraints par exemple sur carte à puce. Nous avons développé les protocoles de m-passe et de m-ticketing dans une carte SIM standard : la validation d'un ticket ou d'un m-pass s'effectue en moins de 300ms et ce tout en utilisant des tailles de clés adéquates. Nos solutions fonctionnent également lorsque le mobile est éteint ou lorsque sa batterie est déchargée. Si les applications s'exécutent dans un TEE, nous introduisons un nouveau protocole de migration de données privées, d'un TEE à un autre, qui assure la confidentialité et l'intégrité de ces données. Notre protocole est fondé sur l'utilisation d'un schéma de proxy de rechiffrement ainsi que sur un nouveau modèle d'architecture du TEE. Enfin, nous prouvons formellement la sécurité de nos protocoles soit dans le modèle calculatoire pour les protocoles de m-pass et de ticketing soit dans le modèle symbolique pour le protocole de migration de données entre TEE.

Mots clés : services mobiles, respect de la vie privée, signature de groupe, proxy de rechiffrement, TEE, migration de secrets

Design of privacy preserving cryptographic protocols for mobile contactless services

Summary:
The increasing number of worldwide mobile platforms and the emergence of new technologies such as the NFC (Near Field Communication) lead to a growing tendency to build a user's life depending on mobile phones. This context brings also new security and privacy challenges. In this thesis, we pay further attention to privacy issues in NFC services as well as the security of the mobile applications private data and credentials namely in Trusted Execution Environments (TEE).

We first provide two solutions for public transport use case: an m-pass (transport subscription card) and a m-ticketing validation protocols. Our solutions ensure users' privacy while respecting functional requirements of transport operators. To this end, we propose new variants of group signatures and the first practical set-membership proof that do not require pairing computations at the prover's side. These novelties significantly reduce the execution time of such schemes when implemented in resource constrained environments. We implemented the m-pass and m-ticketing protocols in a standard SIM card: the validation phase occurs in less than 300ms whilst using strong security parameters. Our solutions also work even when the mobile is switched off or the battery is flat. When these applications are implemented in TEE, we introduce a new TEE migration protocol that ensures the privacy and integrity of the TEE credentials and user's private data. We construct our protocol based on a proxy re-encryption scheme and a new TEE model. Finally, we formally prove the security of our protocols using either game-based experiments in the random oracle model or automated model checker of security protocols.

Keywords: mobile services, privacy, group signature, proxy re-encryption, TEE, credential migration

Laboratoire d'Informatique Fondamentale d'Orléans