Abstract of "Efficient Cryptography for Information Privacy" by Foteini Baldimtsi, Ph.D., Brown University, May 2014.

In the modern digital society, individuals, businesses and governments perform numerous everyday tasks such as shopping, banking and commuting using electronic media. Although these electronic transactions provide efficiency and convenience, they usually overlook the privacy of the users. This thesis studies privacy-enhancing technologies in order to get the best of both worlds: all the benefits of electronic transactions but without sacrificing user privacy. Using efficient cryptographic tools such as digital signatures, zero-knowledge proof systems and encryption schemes, we propose secure protocols that protect user privacy and at the same time are practical.

Anonymous credential systems allow users to obtain and demonstrate possession of digital credentials in order to authenticate themselves in a privacy-preserving manner. We first show impossibility results on proving the security of one of the most well-known anonymous credential systems, the one due to Stefan Brands, that is currently being implemented by Microsoft under their credential management project, U-prove. Our impossibility result not only applies to Brands but generalizes to a much broader class of protocols. We then propose *Anonymous Credentials Light*: the first efficient single-use anonymous credential scheme that is provably secure.

Cryptographic e-cash allows secure and private electronic payments and provides similar unforgeability and untraceability as physical cash does. Our Anonymous Credentials Light can be extended to an efficient e-cash scheme that moreover has the nice property of encoding users' attributes in the coins. We provide a smartphone implementation of our proposed scheme and explain how it can be used for efficient and private payments in the public transportation scenario. A limitation of traditional cryptographic e-cash, however, is that it does not allow users to transfer coins to each other. We present the first practical, fully anonymous transferable e-cash scheme that does not

depend on a trusted authority to detect double spending.

Finally, we study how to revoke users' secret keys or credentials when, for example, a user misbehaves or a secret key was compromised. We propose efficient generic revocation mechanisms that can be used as building blocks for various constructions.

Efficient Cryptography for Information Privacy

by

Foteini Baldimtsi

B. Sc., Applied Informatics, University of Macedonia, 2008

M. Sc., Computer Science, Brown University, 2011

A dissertation submitted in partial fulfillment of the

requirements for the Degree of Doctor of Philosophy

in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2014

This dissertation by Foteini Baldimtsi is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____
Anna Lysyanskaya, Director

Recommended to the Graduate Council

Date _____ _____
Roberto Tamassia, Reader

Date _____ _____
Leonid Reyzin, Reader
(Boston University)

Approved by the Graduate Council

Date _____ _____
Peter M. Weber
Dean of the Graduate School

Vita

Foteini Baldimtsi was born and grew in Kolhiko, a small town next to Thessaloniki, Greece. In 2004 she started her undergraduate studies in the Applied Informatics Department at the University of Macedonia, Thessaloniki, Greece. She soon became very interested in theoretical computer science and cryptography and participated in relevant research projects. She obtained her Bachelor's degree in 2008 and then, in 2009 she moved to the US to attend the graduate program of the Computer Science Department at Brown University. In 2011 she received a Masters degree and in 2014 a Doctorate degree both in Computer Science from Brown University. Her research interests are on cryptography, data security and privacy. While at Brown, she spent the summer of 2012 as an intern in IBM Research in Zurich and the fall of 2013 as an intern in Microsoft Research in Redmond, WA. Foteini is a proud recipient of the Paris Kanellakis fellowship and scholarships from Gerondelis Foundation and Bodossaki Foundation.

Acknowledgements

My deepest appreciation goes to my thesis advisor Anna Lysyanskaya. I would have never been able to conclude this thesis if it was not for her support, encouragement and guidance. Anna has shown me the kind of scholar that I want to be as a passionate researcher as well as a tolerant, effective teacher. She taught me to never give up on a proof and that if I cannot prove that something holds, then I should be able to prove that it is impossible. Anna's advice and insightful comments on both academic and personal matters have helped me to develop both as a researcher and as a person. It has been a great privilege to work with her!

I am very grateful to my committee members Roberto Tamassia and Leo Reyzin. Despite his busy schedule as the chair of the CS department, Roberto, would always find the time to meet with me and his academic advice has been invaluable. I am very honoured that Leo has been the external member of my committee. His research questions and feedback in my talks were of great help and I am very happy that I will have the chance to keep working with him in the future.

During the past five years, I had the pleasure of working with a great set of colleagues from whom I learned a great deal. Gesine Hintewälder has been a close collaborator in many parts of this thesis and taught me everything I know about efficient implementation of cryptographic systems. I also greatly enjoyed working with Wayne Burleson, Georg Fuchsbauer, Markulf Kohlweiss, Anja Lehmann, Gregory Neven, Christof Paar, Andy Rupp and Christian Zenger. A special thanks goes to Melissa Chase and Jan Camenisch for giving me the opportunity to collaborate with them while

interning in Microsoft and IBM respectively.

I would have never have started a PhD in cryptography if it was not for my undergraduate advisor George Stephanides. George taught me my first class in cryptography and showed me how fascinating research can be. Along with him, I would also like to thank professors Alexandros Chatzigeorgiou, Ioannis Refanidis and Dimitris Varsakelis, as well as my colleague Konstantinos Chalkias, for all their help, support and encouragement during all these years.

My days in the CIT would not have been the same without my friends in the department. Babis was the student who hosted me when I was visiting Brown back in 2009 and he proved to be one of my closest friends and colleagues. I will always thank him for being there to mentor and cheer me up in all of my breakthroughs and breakdowns during my first years in graduate school. I am truly going to miss my beloved friend Alexandra when I leave Brown. I am very grateful to her for standing by me in sadness and happiness in both personal and academic aspects of my life. A big thanks goes to Evgenios for all of the coffee breaks and laughs we had the last couple of years. I would also like to thank my academic siblings and officemates Feng-Hao and Sasha for making our office a fun place to work and for always being willing to discuss a technical problem, proofread a paper or attend a practice talk. Finally, I owe a debt of gratitude to Lauren Clarke who always had the best answer for everything in the department.

My friends in Providence made sure that the past five years would be unforgettable. Thank you Anastasia S., Anastasia V., Asli, Daniel, Giorgo, Harry, Irina, Jessica, Kosta, Maria, Pari, Sophia, Saki, Tania, Taso and Vasili for all the fun times we had together! A very special thanks goes to my dear friend Katerina, that was always next to me no matter what time of the day (or the night :) it was. Andrea, Dimitri and Yianni although our time together in Providence was short, it has been really fun meeting you all over Europe the years after. Thalia and Antonis, back in Greece, were always ready to support me from afar - thank you guys for all the "online coffees" we had together!

I am very grateful to my parents Kostas and Voula and my brother Christos for their love and everything they have done to get me to this point. They would also always supported my choices no matter how difficult it was for them. Finally, nobody has provided more support, encouragement, and love than my husband Socrates. Words cannot express how much I thank you!

$Ευχαριστώ$!

This thesis was supported by NSF grant 0964379, as well as from a Brown University fellowship, a Paris Kanellakis fellowship and scholarships from Bodossaki and Gerondelis foundations.

Contents

List of Tables

List of Figures

1 Introduction

We live in a digital society. Electronic technologies have radically changed the way individuals, businesses and governments communicate and exchange information. Consider for example how many tasks of your everyday life you now perform online or using electronic means: from reading the news on a website and interacting with friends through social media to shopping, banking or renewing your driver's license. The benefits of these electronic transactions are numerous as they provide ease and convenience, high efficiency, accuracy and reduced operating costs among others. However, this does not come without a price; these technologies usually overlook the privacy of the users and limit the control they have on the dissemination of their personal information. Consequently, as users fulfill various transactions electronically they leave a life-long trail of personal information of which they might not even be aware of.

At the same time handling digital information has become significantly easier given the rapid advances in big data technologies. Storage space is very cheap even for massive amounts of data while collection, aggregation and analysis of information has become very effective and efficient. This, in combination with the immense amount of personal information that is being revealed electronically has imposed various threats to user privacy. A study by Latanya Sweeney in 2000 has shown

that 87% of the US population can be uniquely identified by their gender, date of birth and zip code [135] - information that users comfortably submit in various electronic transactions. As a result, researchers have been combining user data from various sources and using advanced mining techniques have managed to identify individuals or reveal their location [81, 87, 98, 101].

A recent survey performed by the Pew Research Center's Internet Project underwritten by Carnegie Mellon University shows that US citizens are actually more concerned about their privacy online than they were in the past [102]. 59% of internet users said that they should be able to use the internet anonymously and 85% claimed that they have taken at least one step trying to mask their identities online. These concerns have been even more intense in the last year with the revelations about government surveillance programs such as Prism [139] dominating the headlines. A group of cryptography experts signed an open letter [2] against massive surveillance where among other things they highlight the problem of online privacy:

> *"Indiscriminate collection, storage, and processing of unprecedented amounts of personal information chill free speech and invite many types of abuse, ranging from mission creep to identity theft. These are not hypothetical problems; they have occurred many times in the past. Inserting backdoors, sabotaging standards, and tapping commercial data-center links provide bad actors, foreign and domestic, opportunities to exploit the resulting vulnerabilities."*

But what does privacy mean in the online world? One of the most prominent definitions is due to Alan Westin from his book "Privacy and Freedom" [138]: *"Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extend information about them is communicated to others."* Individuals should have the right to determine how much of their personal information is disclosed and to whom, as well as how it is maintained and disseminated. Governments have defined the so-called data minimization principles [100] along these lines: no electronic transaction should require its participants to needlessly reveal private information. These

principles are at the core of European privacy standards [119] and also the National Strategy for Trusted Identities in Cyberspace (NSTIC) published by the U.S. government [69]. The goal is to limit the scope of the data that organizations are allowed to collect; so, to make sure that it is not in violation of this directive, an online bank or vendor interacting with a user has an incentive to learn as little as possible about this user.

This thesis studies privacy-enhancing technologies in order to get the best of both worlds: all the benefits of electronic transactions but without sacrificing user privacy. Privacy in a digital society is a rather challenging problem. One of the main barriers is that usually privacy preserving solutions are significantly more computationally expensive than the non-privacy friendly ones. We propose and use efficient cryptographic tools such as digital signatures, zero-knowledge proof systems and encryption schemes to construct secure protocols that protect user privacy and at the same time are practical. The main contribution of this thesis is on two sub-fields of privacy preserving technologies: (a) privacy preserving authentication methods (anonymous credentials) and (b) private electronic payments (e-cash). The goal of our research is to provide solutions for a digital society where individuals will be able to perform electronic transactions in a secure and intuitive way while retaining control of their personal information throughout their lives.

1.1 Private Authentication - Anonymous Credentials

Anonymous credentials, envisioned by David Chaum [59], and first fully realized by Camenisch and Lysyanskaya [45], allow users to authenticate by proving possession of credentials without revealing any other information about themselves; when such a proof is carried out, it cannot be linked to previous uses of the same credential, or to any other identifying information about the user. Additionally, they give the users the ability to privately obtain credentials. The reason that they have become so popular is that they strictly adhere to data minimization principles mentioned above. Anonymous credentials have proved to be a centrally important building block in privacy-minded

identity management systems currently under development [49, 118, 136].

A variety of anonymous credential schemes has been proposed in the literature [19, 20, 36, 37, 39, 40, 41, 46]. There exist two main categories of anonymous credentials: the *multi-use* credentials, where an issued credential can be used multiple times, and the *single-use* credentials, where each credential is meant to be used only once: in order to unlinkably re-use a credential, a user must get it reissued. Multi-use credentials are useful in certain applications like subscription services where a user might access her subscription multiple times. On the other hand, single-use credentials are essential in applications like electronic cash or electronic voting where each credential (which might be a coin or a ballot) is not supposed to be used more than once. Moreover, for state-of-the-art constructions, both issuance and usage of single-use credentials is considerably more efficient than the multi-use ones. The most well known multi-use scheme is the one due to Camenisch-Lysyanskaya [45] that is being implemented by IBM as part of their Idemix project [136]. On the other side, the scheme due to Stefan Brands [36, 37] which is being implemented by Microsoft under their U-Prove project [118] is the most widely known single-use credential construction.

Efficiency considerations are important when deploying these schemes in practice. While on a PC, the Camenisch-Lysyanskaya credentials [45] and follow-up work [20, 39, 41, 46] take only a fraction of a second of CPU time, in mobile devices, smartcards and RFID cards, they are not yet as practical as we may wish for certain applications. The issue is that they make use of either the RSA group, or groups that admit bilinear pairings, and the security parameter choice needed to provide security under the required assumptions in these groups makes these systems too expensive for smartcards and mobile devices (i.e. original CL credentials require an RSA group with bit-length 2048 or a pairing with 128 bits of security). From the efficiency point of view, therefore, the U-Prove credential system based on Brands' work [36, 37] seems attractive. The U-Prove credential issuing protocol, as well as the algorithm for verifying a credential, are practical enough for the more advanced end of RFID technology, since they can work in any group where the discrete logarithm problem is hard (thus can be based on ECC), and require just a few exponentiations in this group.

Note that software implementations of ECC multiplication are much faster than the equivalent RSA exponentiation [91].

So perhaps one could use these single-use anonymous credentials and just have them reissued as many times as needed. Unfortunately, though, the only well known efficient single-use credential scheme is not provably secure! No proof of security has ever been given for U-Prove under any reasonable assumption. As part of this thesis we study the security of U-Prove and in particular the unforgeability of Brands blind signature [36] which is its main building block. We show that all known approaches for proving unforgeability of Brands blind signature in the random oracle model will fail no matter how strong an assumption one makes [15]. This impossibility result generalizes to a much broader class of blind signatures, as for example the blind Schnorr signature [131] and the blind GQ signature [90].

One might ask what happens if, instead of the Brands blind signature, one tries to construct single-use anonymous credentials from another provably secure blind signature scheme. Unfortunately, blind signatures as traditionally defined [3, 99, 125, 123] do not give any consideration to users' attributes. In a blind signature, a user is free to choose whichever message he wants signed. In a credential scheme, a user should be able to get his particular attribute set signed by the signer. The signers can verify that what they are signing is the correct attribute set, even though they cannot see exactly what they are signing. For example, a signer might want to sign a user's secret key, his age, and his citizenship information, and wants to be able to verify that this is indeed what she is signing. Attributes are, in fact, essential: without them, it is impossible to tie a credential to a particular user's identity, and so users would be able to just pool each others' credentials as desired, instead of tying them to a particular secret key. Moreover, giving the user the ability to choose any message means that the content of the signed messages has not been vetted by the signer at all.

Thus, a natural question is: how do we extend the notion of a blind signature so that it can efficiently accommodate single-use anonymous credentials? In this thesis, we propose a new single-use anonymous credential scheme [14] called *Anonymous Credentials Light* that has comparable

efficiency to U-Prove and it is provably secure (in the random oracle model). In particular, it is unlinkable under the decisional Diffie-Hellman assumption, and unforgeable under the Discrete-Logarithm assumption for sequential composition (the extension to concurrent self-composition is an open problem). Our scheme was inspired by Abe's three-move blind signature [3] which did not support attributes [1]. For the construction, we define a new cryptographic building block, called *blind signatures with attributes*, and discusse how it can be used in combination with a commitment scheme to directly construct a single-use anonymous credential system.

1.2 Electronic Payments and Applications

Payments made with debit or credit cards do not provide any privacy guarantee for the users since the corresponding financial institution can track all their transactions. In contrast, electronic cash (e-cash) schemes allow secure and private electronic payments by providing similar security and anonymity as physical cash. The general e-cash concept describes the interaction between three types of entities: the bank, users, and merchants. Monetary value is represented by electronic coins, which are pieces of data blindly signed by the bank. During withdrawal, the bank issues electronic coins to users where each coin encodes a unique serial number and the identity of its possessing user. The issuance happens in a blind fashion so that the bank cannot link a coin to a specific user. Users can spend their electronic coins to pay a merchant while remaining anonymous. Note that, the bank does not have to be online during the transaction (*off-line* cash). When a merchant deposits the coins that received from users, the bank retrieves the serial numbers and checks whether a deposited coin had been deposited before. If so, the bank can also check whether the merchant deposited the same coin twice, or whether a user double-spent a coin, in which case her identity will be revealed.

Chaum [59] was the first to come up with the idea of an anonymous e-cash system and then

[1] When Abe's blind signature was proposed it came with a proof of security in the RO model for concurrent composition. However, it was later found that the original proof suffered some restrictions since it was only valid for an adversary with overwhelming success probability and a new proof was given but this time in the generic group model [112].

together with Fiat and Naor they defined e-cash secure against double spending in the off-line setting [58]. Following Chaum's et al. paradigm more schemes were proposed [3, 20, 36, 41, 54, 115] which were either improving efficiency or adding extra properties. A very useful property of certain e-cash schemes is the encoding of users' attributes into coins (i.e. user's age or zip code). This is very convenient since at the time of spending allows the user to prove statements on his attributes on a privacy preserving way. For example, a user can prove that he is over 18 at the time of spending a coin if his age attribute is encoded in the coin.

The most efficient e-cash scheme with attributes is the one due to Brands [36] which, similarly to his credential scheme (U-Prove), is based on his blind signature scheme. However, as discussed above, we have showed an impossibility result for proving Brands blind signature unforgeable [15]. For the e-cash setting, this means that there is no guarantee that a malicious attacker cannot forge coins. Luckily, as we show in this thesis, our *Anonymous Credentials Light* scheme can be extended to an efficient e-cash scheme with attributes [97]. Our new e-cash scheme is very efficient: the issuance of a coin takes approximately 300 milliseconds while the spending of a coin takes about 380 milliseconds when implemented in a smartphone device. This makes it very attractive for various applications where private payments are important such as payments for the public transportation scenario.

Public Transit Application. More and more public transportation systems have introduced pre-paid or monthly cards/devices such as MetroCards in NYC, Charlie Cards in Boston or E-ZPass for toll payments to replace their older systems based on actual cash or tickets. These new systems provide many advantages but at the same time introduce concerns as to the privacy of their customers. Essentially, one's MetroCard or Charlie Card is a persistent identifier, and the MTA in New York, or MBTA in Boston, has the ability to locate an individual in a large metropolis, which prompts concerns among privacy advocates [130]. Privacy is an especially challenging problem in this context since it not only spans cryptographic theory and many engineering fields but extends into public policy areas such as environmental justice policy and sociology issues such as "fair access to

all". However, in order to enable a large-scale deployment and broad acceptance of such a payment system, adequate security and privacy mechanisms are an essential requirement.

In theory, cryptographic techniques like e-cash with attributes can be used to allow private payments in the public transit scenario. By using attributes appropriate one can implement additional features such as variable pricing (e.g. reduced fare for senior customers) and privacy-preserving data collection. For example, if the *age* attribute is included in the electronic coins, a senior user can prove that he is over 65 years old and receive a senior discount. The open problem here is whether we can have implementations of cryptographic e-cash that work with the same speed and convenience as non-privacy-preserving MetroCards?

In this thesis we present an implementation of several e-cash schemes on an smartphone (used as a payment device) that supports Near Field Communication (NFC) [1]. NFC allows the smartphone to communicate with other NFC-enabled devices within a range of a few centimeters. The benefit of this type of communication is its simple and hence fast establishment, as the smartphone can connect to the reader by bringing them into proximity of a few inches. For transportation authorities the advantage of relying on users' NFC-enabled smartphones, is that no additional (electronic) tokens will have to be handed out. Instead, only a software application has to be installed to the user's phone. This decreases the revenue collection cost, and further allows the payment system to be updated easily. To make a change to the system, the transportation authority only needs to provide a software update, rather than a hardware rollout.

Transferable E-cash. In traditional cryptographic e-cash, users can only transfer their coins to merchants, who must then deposit the coin at the bank. It would be natural to allow coins to change hands multiple times before they get deposited. Moreover, it would be desirable if these transfers could be done without being connected to the bank, i.e., offline. One of the main advantages of such a transferability property is that it would decrease the communication cost between the bank and the users. Moreover, it would allow to implement more real world scenarios: Consider the example of coins of different denominations. A store, which is offline, wants to give back change to a customer,

using previously received coins. In order to do so, coins need to be transferable multiple times. Transferability of e-cash was proposed in the 1990s [114, 115] and the desired security properties have been analyzed; however, so far no schemes have been proposed that are both practical and satisfy the proposed security and privacy requirements.

As part of this thesis we present the *first efficient and fully anonymous* transferable e-cash scheme [10]. We provide a formal treatment of the security and anonymity properties of transferable e-cash by giving game-based definitions. In transferable e-cash, the owner of a coin should be able to transfer the coin/signature he received from the bank to another user such that the transferred coin is valid, carries all the information necessary to detect double spending and preserves anonymity. Thus, we need a digital signature scheme that allows to compute a "fresh" version of a valid signature (unlinkable to the original one to ensure anonymity) and extends the current signature to include more information (such as an attribute that allows for double spending detection).

A recent proposal of a signature scheme that satisfies the above properties is due to Chase et al. [57]. They propose *malleable* signatures: an extension of digital signatures, where anyone can transform a signature on a message m into a signature on m', as long as $T(m) = m'$ for some allowed transformation T. In our construction, a coin withdrawn by the bank is signed using a malleable signature scheme. Whenever a user wishes to transfer a coin to another user he computes a "mauled" signature on a valid transformation of the coin. A valid transformation guarantees that the transferred coin is valid, it is indeed owned by the sender (i.e. the sender's secret key corresponds to the information encoded in the coin) and the new coin/signature created will encode the right information of the receiver so that a double spending can be successfully detected.

Double spending detection for transferable e-cash is a complex issue, since it needs to ensure that the culprit is identified while the anonymity of honest owners of the coin will be preserved. We propose an efficient double-spending detection mechanism, which is independent of our scheme and could be used by other transferable e-cash constructions.

1.3 Revocation

Anonymous credential schemes and group signatures are very attractive because they allow users to demonstrate possession of credentials or group memberships in a way that does not reveal any additional information. Yet, despite significant advances that allow anonymous authentication and group signatures to have efficiency that is comparable to their non-privacy-preserving counterparts, adoption in practice has been slow. One of the main arguments against using such privacy-preserving mechanisms is that revoking them is rather complicated.

It is impossible to eliminate key compromise and user misbehavior. Thus, the ability to revoke privileges of users whose keys have been compromised, or who have violated their terms of service, is a key requirement of any system. In the non-anonymous setting, revocation of credentials is very easy: it can be done via *white lists* where a user is valid if her public key or identity is on a white list. Alternatively, one can *blacklist* the identities or public keys of the revoked users; anyone not on this blacklist can be presumed to be a valid user. This can be done in conjunction with expiration dates that make it possible to delete old blacklists.

In the non-anonymous setting, a user can demonstrate that she is valid such that (1) the user's computation is independent on the number of valid and revoked users; (2) the verifier needs to look up the list of valid (resp. revoked) users to ensure that the user is on (resp. not on) it; (3) the credential issuer needs to keep the list up to date, and needs to digitally sign the current list, which, if Merkle trees are used to organize a membership list of size n, means a $O(r_i \log n + c)$ amount of work per time period i in which r_i users got revoked (and a constant c amount of work per time period in which no user got revoked). This is because to update a leaf of a Merkle tree one needs to recompute $\log n$ nodes; to tie the resulting tree to the current time period, one needs to sign the root of the Merkle tree.

In the anonymous setting, this seems harder. Neither the blacklist nor the white list approaches work at all, because often even the issuer of a credential does not know the identities of the credential

recipients. Suppose a credential issuer knows his users by pseudonyms, and can form black- or white lists accordingly. Then the first solution that comes to mind — that of using a zero-knowledge proof to show that a user is not on a blacklist — incurs a computation penalty proportional to the size of the blacklist. The first white list solution that comes to mind — one in which every valid user gets a new membership certificate whenever there is a need to update the membership list — requires the group manager to perform work *linear in the number of valid members* upon each update.

Numerous solutions that are much more clever than the above approach have been proposed: solutions based on cryptographic accumulators [42, 44] and zero knowledge proofs [45] or solutions based on the NLL broadcast encryption scheme [103]. However these previous solutions suffer from certain limitations. Most of the proposed methods are tied to a specific group signature or anonymous credential scheme. A natural question would be whether we can construct a generic revocation mechanism that can be used as a plug-in in other constructions. We would like this mechanism to be simple and efficient and work in both the random oracle and standard models under different assumptions depending on the required level of security.

In this thesis we propose a generic revocation mechanism called *"Anonymous Revocation Data Structure"* (ARDS) that can be instantiated in several ways depending on the desired efficiency and security level. This is achieved by building generality into the definitions: a user's membership certificate consists of a signature on her joining order and secret key (the secret key is not revealed in the clear, the signature is the result of a two party protocol). Whenever a user is involved in an algorithm, such as Join, ProveMembership or Verify, a commitment to her secret key is given as input. This allows to use it in combination with group signatures or anonymous credential schemes that require commitments to user secrets and certain types of proofs.

1.4 Thesis Outline

The rest of this dissertation is organized as follows: in Chapter 2 we present some basic cryptographic notation and definitions. In Chapter 3 we show impossibility results on the unforgeability of certain classes of blind signatures (main building block for various privacy preserving applications). Then, in Chapter 4 we explain why U-Prove, one of the most famous anonymous credential schemes, cannot be proven secure based on our impossibility result and present a proposed extension that would satisfy unforgeability. Our new anonymous credential system is presented in Chapter 5, and in Chapter 6 we show how it can be extended to an electronic payment scheme. Moreover, we present an implementation of our new e-cash scheme for private payments in the public transportation scenario. In Chapter 7 we present the first fully anonymous transferable e-cash scheme that does not depend on a trusted authority to detect double spending. In Chapter 8 we study how to revoke users' secret keys or credentials when, for example, a user misbehaves or a secret key was compromised and propose efficient generic revocation mechanisms that can be used as building blocks for various constructions. Finally, Chapter 9 presents the conclusions of the thesis.

2 Preliminaries

In this chapter we set the notation to be used throughout the thesis and we review some of the known cryptographic assumptions, definitions and constructions that will be used in the following chapters.

2.1 Notation and Assumptions

Throughout the thesis k will denote the *security parameter* which determines the desired level of security of the scheme. 1^k denotes the string of 1's of length k. When a probabilistic polynomial time (PPT) algorithm A is given 1^k as input, then, A is allowed to work in time polynomial in k. For a set S, notation $x \leftarrow S$ denotes that the element x is chosen uniformly at random from S. By $A(\cdot, \ldots, \cdot)$ we denote the multiple inputs of an algorithm A and by $y \leftarrow A(x)$ we denote the output of the algorithm on input x. For algorithms A and B, $\langle A, B \rangle$ denotes their interaction. A function $\nu(k)$ is called *negligible* if for all polynomials $p(k)$, and for all sufficiently large k it holds that $\nu(k) < 1/p(k)$.

2.1.1 Cryptographic Assumptions

We now review some basic cryptographic assumptions which will be used in the thesis.

Definition 2.1.1 (Discrete Logarithm Assumption)**.** *Let k be the security parameter. Let $\mathbb{G}$ be a group of order q (k-bit prime) and g be a randomly chosen generator of $\mathbb{G}$. Then, for every polynomial time algorithm $\mathcal{A}$ it holds that:*

$$Pr[h \leftarrow \mathbb{G}; x \leftarrow A(h) : x = \log_g h] \leq \nu(k)$$

where $\nu(k)$ is a negligible function.

Definition 2.1.2 (Decisional Diffie-Hellman (DDH) Assumption)**.** *Let k be the security parameter. Let $\mathbb{G}$ be a group of order q (k-bit prime) and g be a randomly chosen generator of $\mathbb{G}$. Then, for every polynomial time algorithm $\mathcal{A}$ it holds that:*

$$Pr[a, b \leftarrow \mathbb{Z}_q; h_0 \leftarrow g^{ab}; h_1 \leftarrow \mathbb{G}; b \leftarrow \{0,1\}; b' \leftarrow A(g^a, g^b, h_b) : b' = b] \leq \frac{1}{2} + \nu(k)$$

where $\nu(k)$ is a negligible function.

Definition 2.1.3 (RSA Assumption)**.** *Let k be the security parameter. Let $N = pq$ where p and q are k-bit, distinct odd primes. Let e be a randomly chosen positive integer less than and relatively prime to $\phi(N) = (p-1)(q-1)$. Then, for every polynomial time algorithm $\mathcal{A}$ it holds that:*

$$Pr[h \leftarrow \mathbb{Z}_N^*; x \leftarrow A(N, e) : x^e \equiv h \bmod N] \leq \nu(k)$$

where $\nu(k)$ is a negligible function.

2.2 Cryptographic Primitives

2.2.1 Witness Relations and Proofs of Knowledge

A witness relation for a language $L \in \mathcal{NP}$ is defined as:

Definition 2.2.1 (Witness relation [83])**.** *A witness relation for a language $L \in \mathcal{NP}$ is a binary*

relation R_L that is polynomially bounded (i.e., $(h,x) \in R_L$ implies $|x| \leq poly(|h|)$), is polynomial-time-recognizable and characterizes L by: $L = \{h : \exists x \text{ s.t. } (h,x) \in R_L\}$.

For $h \in L$, any x satisfying $(h,x) \in R_L$ is called a *witness* (for the membership $h \in L$). By $R_L(h)$ we denote the set of witnesses for the membership $h \in L$; that is, $R_L(h) = \{x : (h,x) \in R_L\}$. If for each $h \in L$, there exists a unique $x \in R_L(h)$ then we say that R_L is a *unique-witness* relation. The discrete logarithm problem is an example of a unique witness relation where: $R_{G,g,q} = \{h, x \text{ s.t. } x \in \mathbb{Z}_q \text{ and } g^x = h\}$. A relation R_L is said to be "hard", if one can efficiently generate (h,x)'s such that, when given h it is hard to find the witness x.

Definition 2.2.2. *(Interactive Proof System [84, 85]) An interactive proof system (with soundness error $s \in [0,1]$) for a language L with witness relation R_L is a pair of algorithms (P,V), where V is probabilistic polynomial time algorithm, and the following properties hold:*

1. *Completeness. For every $h \in L$ the verifier always accepts:*

$$Pr[\langle P,V\rangle(h) = 1] = 1.$$

2. *s-Soundness. For every $h \notin L$ and every computationally unbounded P^*,*

$$Pr[\langle P^*,V\rangle(h) = 1] \leq s.$$

If soundness error, s, is negligible, then this interactive proof system has *strong soundness.* A useful property in the above setting would be if the verifier V wouldn't learn anything useful from P about the witness x besides the fact that P knows a witness x [85]. This property is called *zero-knowledge.*

Definition 2.2.3. *(Honest Verifier Zero-Knowledge (HVZK)) An interactive proof system (P,V) for a language L is said to be* honest verifier zero-knowledge *if there exists a probabilistic polynomial time algorithm S (the Simulator) such that for all $h \in L$:*

$$view_V[P(h) \leftrightarrow V(h)] \approx S(h),$$

where view_V is the view of the honest verifier V of the interaction between V and P on input h. If

there is a single message transmitted from the prover to the verifier (i.e. one-round protocol), then the proof system is called *non-interactive*.

A *proof of knowledge* [21] is an extension of a zero-knowledge proof in which the prover succeeds to convince a verifier that he knows a certain witness x that $h \in L$ (instead of proving that there exists a witness x).

Definition 2.2.4. *(Proof of Knowledge) Let R_L be a binary realation for a language L and $k \in [0,1]$. We say that V is a knowledge verifier for the relation R with knowledge error k if the following two conditions hold:*

1. *Non-triviality. There exists a prover P such that for all $(h,x) \in R_L$:*

$$Pr[\langle P(h,x), V(h)\rangle = 1] = 1.$$

2. *Validity (with error k). There exists an extractor algorithm, K, such that for all $h \in R_L$, for all computationally unbounded P^*, if*

$$p^*(h) = Pr[\langle P^*, V\rangle(h) = 1] \geq k$$

then, on input h and access to the prover, K computes a value x such that $(h,x) \in R_L$, within an expected number of steps bounded by $\frac{(q(|h|))}{p^*(h)-k}$ for a polynomial $q(\cdot)$.

2.2.2 Σ-Protocols

Σ-protocols are a class of interactive proofs where the Prover and the Verifier (P,V) have a common input h and P proves in zero-knowledge that he knows a value x such that $(h,x) \in R_L$. Their main characteristic is that they have exactly 3 rounds of the following type: (1) P sends a message a to V, (2) V responds with a random challenge c chosen from a domain of size $\Theta(k)$ and (3) P resents a reply r. V decides whether to accept or not given the information he has seen: (h,a,c,r). Formally:

Definition 2.2.5. *(Σ-Protocol) A protocol $\mathcal{P}$ is said to be a Σ-protocol for a relation R_L if:*

1. *$\mathcal{P}$ is of the above three rounds form, and if (P,V) follow the protocol, the verifier always*

accepts.

2. *From any h and any pair of accepting conversations on input h, (a, c, r), (a, c', r') where $c \neq c'$, one can efficiently compute x such that $(h, x) \in R_L$ (special soundness).*
3. *There exists a polynomial-time simulator S, which on input h and a random c outputs an accepting conversation of the form (a, c, r), with the same probability distribution as conversations between the honest P, V on input h (special honest-verifier zero-knowledge).*

A Σ-protocol is said to be unique-witness *Σ-protocol ($UW\Sigma$) if R_L is a unique-witness relation.*

An example of a Σ-protocol is the Schnorr identification scheme [131]. This scheme is essential a proof of knowledge of a discrete logarithm. Let G be a group of prime order q with generator g, and let $\mathbb{Z}_q$ denote the field of integers modulo q. Schnorr's identification scheme works as follows:

Prover$(q, g, h = g^x)$		Verifier(q, g, h)
$y \leftarrow \mathbb{Z}_q$, $a = g^y$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	$c \leftarrow \mathbb{Z}_q$
$r = y + cx \bmod q$	$\xrightarrow{r}$	$g^r \stackrel{?}{=} ah^c$

Σ-protocols are an essential building block for blind signatures and anonymous credentials. For example Brands' [36] and Abe's [3] blind signature schemes are based on a Σ-protocol, while CL anonymous credentials [45] uses ZK proofs which are based on Σ-protocols.

2.2.3 OR-proof Technique

Let $\mathcal{P}$ be a Σ-protocol for a relation R_L, (h_0, h_1) be a common input to (P, V) and P knows a x such that $h_b, x \in R_L$ for $b \in \{0, 1\}$. An OR-proof protocol $\mathcal{P}_{OR}$ is a Σ-protocol for proving that either $(h_0, x) \in R_L$ or $(h_1, x) \in R_L$ [71][1].

The main idea of an OR-proof is that P will complete two Σ protocols Σ_0, Σ_1, one for h_0 and one for h_1 in such a way that the verifier will not be able to tell for which of the two P knows the

[1] This is based on a more general result due to Cramer et. al. [68] where they present a protocol in which the prover demonstrates knowledge of the solution to some subset of n problem instances out of a collection of subsets.

corresponding witness. For h_b the prover can actually follow the real Σ protocol while for h_{1-b} he will have to use a simulator M to create his answers. A $\mathcal{P}_{OR}$ protocol works as follows:

1. P computes the first message of Σ_b, a_b, using (h_b, x) as input. P randomly chooses c_{1-b} and runs the simulator M on input (h_{1-b}, c_{1-b}) and receives $(a_{1-b}, c_{1-b}, r_{1-b})$ as an output. Then, P sends a_0, a_1 to V.

2. V chooses a random string e and sends it to P.

3. P sets $c_b = e \oplus c_{b-1}$ and computes the answer r_b to challenge c_b using h_b, a_b, c_b, x as input. He sends c_0, c_1, r_0, r_1 to V.

4. V checks that $e = c_0 \oplus c_1$ and that both (a_0, c_0, r_0) and (a_1, c_1, r_1) are accepting conversations.

Let $R_{OR} = \{((h_0, h_1), x) | (h_0, x) \in R_L \text{ or } (h_1, x) \in R_L\}$. Then:

Theorem 2.2.6 ([71]). *The $\mathcal{P}_{OR}$ protocol is a Σ-protocol for R_{OR}. Moreover, for any verifier V^*, the probability distribution of conversations between P and V^*, where x is such that $(h_b, x) \in R_L$, is independent of b.*

2.2.4 Composition of Proofs

It is possible to compose proofs of knowledge in order to prove that more than one statement hold simultaneously, *AND of proofs*, or that at least one of the statements holds, *OR of proofs*. As an example, assume that the prover wishes to prove knowledge of integers α, β such that $y = g^\alpha$ and $\tilde{y} = g^\beta$ holds. In order to perform a proof of this type (where the User is proving that two or more statements simultaneously hold) the prover needs to send the first round messages simultaneously to the verifier, who replies with a single challenge (or, in the non-interactive setting, compute a single e), and then execute the rest of the protocol using this single challenge. The verifier accepts if all the verification equations hold [63]. Let's now see how an OR proof is possible. Assume a prover P who proves knowledge of either α or β such that either $y = g^\alpha$ or $\tilde{y} = g^\beta$ holds. Without loss of

generality assume that P knows α; he first picks random r_1, e_2, z_2, sets $a_1 = g^{r_1}$ and $a_2 = g^{z_2}\tilde{y}^{-e_2}$ and sends a_1, a_2 to the verifier. Upon receiving a_1, a_2, the verifier sends a random challenge e to P. Then, the prover computes $e_1 = e \oplus e_2$, $z_1 = r_1 + e\alpha$ and sends e_1, z_1, e_2, z_2 to the verifier who accepts if $e_1 \oplus e_2 = e$, $a_1 y^{e_1} = g^{z_1}$ and $a_2\tilde{y}^{e_2} = g^{z_2}$ [67].

2.2.5 Commitment Schemes

A non-interactive commitment takes as input a message (or set of messages) x and randomness R and outputs a value that, on the one hand, reveals no information about the message but, on the other hand, it is hard to find a (x', r') such that $Commit(x; r) = Commit(x', r')$ but $x \neq x'$ (see [106] for a standard definition and treatment). A commitment scheme is secure secure if it is both *hiding*: the receiver learns nothing about the committed message and *binding*: there is a unique value that opens the commitment and validates during the reveal stage. Two of the most well studied commitment schemes are the one due to Pedersen [121] and the one due to Fujisaki and Okamoto [80] both of which are information theoretically hiding and computationally binding. It is also important to note that there are efficient zero-knowledge proof protocols for proving that a commitment C is to a particular set of values; or to a set of values that satisfy a rich class of relations [36, 47, 70].

The Pedersen commitment is based on the discrete logarithm assumption and works as follows:

1. $PedCommitSetup(1^\lambda)$: On input the security parameter 1^λ pick a group G of prime order $q = \Theta(2^\lambda)$ with generators g, h.

2. $PedCommit(x; r) = g^x h^r$; where $x, r \in \mathbb{Z}_q$.

The Pedersen commitment scheme is information theoretically hiding and computationally binding. It is also important to note that there are efficient zero-knowledge proof protocols for proving that a commitment C is to a particular set of values; or to a set of values that satisfy a rich class of relations [36, 47, 70].

The Fujisaki and Okamoto (FO) scheme [80] is an extension of the Pedersen commitment scheme to the RSA modulus instead of a prime order group (thus, it yields to more efficient range proofs as we will later see). The FO commitment consists of:

1. $FOCommitSetup(1^{\lambda})$: Let p, q be safe primes, $n = pq$ be an RSA modulus, g_1 be a generator of QR_n and $g_2 = g_1^{\alpha}$ for $\alpha \in \mathbb{Z}_n$. Output n, g_1, g_2.

2. $FOCommit(x; r) = g_1^x g_2^r$; where $x \in \mathbb{Z}_n$ and $r \in \mathbb{Z}_{2^{\lambda}n}$.

2.2.6 Range Proofs

Range proofs are a special category of zero-knowledge proofs of knowledge which allow a prover to convince a verifier that he has committed to a value that lies in a specific range. Range proofs are particularly useful for many applications such as anonymous credentials or e-cash and a variety of schemes has been proposed in the literature. Lipmaa [104], based on earlier work due to Boudot [35], gave an efficient construction of a range proof which is based in the observation that any positive number can be represented as a sum of four squares. Informally, in order to show that a committed value x inside a Fujisaki-Okamoto(FO) commitment C lies in the integer interval $[a, b]$, one just needs to show that $x_1 = x - a > 0$ and $x_2 = b - x > 0$. This is possible by performing the following: the prover and the verifier jointly compute commitments to x_1, x_2: $C_{x_1} = C/g^a, C_{x_2} = g^b/C$ and the prover commits to their representation as a sum of four squares. It is straightforward to show that a committed value is a sum of other committed values: let C_1, C_2 be commitments to values c_1, c_2 respectively, to show that C is a commitment to $c = c_1 + c_2$ just show that C is a commitment to $C_1 C_2$. The above range proof technique is quite efficient and it is based on the hardness of the Strong RSA problem.

In case that one doesn't wish to rely on the Strong RSA assumption, the folklore method would be to have the prover commit to every bit of his k-bit secret x, prove that the commitments are to bits of x and then the verifier is convinced that $x \in [0, 2^{k+1} - 1]$ since there are exactly k

commitments. This method results in a proof of size $O(k)$ group elements. Camenisch et al. [38] present a way to reduce this size asymptotically and also use smaller constants which results in a protocol which is a magnitude more efficient than previously known ones for the discrete logarithm based setting. Briefly their idea is to use a u-ary basis instead of a binary one: write the secret value x in base u (optimally chosen) and commit to all ℓ of those u-ary digits in order to show that $x \in [0, u^\ell]$[2]. This immediately reduces the proof size to $O(\ell u)$ for each commitment and requires $O(u \cdot \ell)$ communication. Then, in order to come up with an even more efficient scheme, Camenish et al. [38] suggest reusing part of one u-ary proof in all ℓ proof instances: i.e. the verifier could send one list of u signatures representing u-ary digits and then the prover can use this same list to prove that all ℓ digits are indeed u-ary ones. This reduces the communication complexity to $O(u + \ell)$ and for appropriate choice of u and ℓ their approach yields a proof of size $O\left(\frac{k}{\log k - \log\log k}\right)$. The above construction was proven secure under the $(\log k)$-Strong Diffie Hellman assumption.

In many applications we need non-interactive range proofs. The above proofs can be transformed to non-interactive by instantiating them as $\Sigma-$protocols and then making them non-interactive in the random-oracle model using the Fiat-Shamir heuristic (see below). For non-interactive range proofs without the use of random oracles please refer to [127, 53].

2.2.7 Dynamic Accumulators

An accumulator scheme allows to combine a large set of values into a short one, called the *accumulator*, such that there is a short *witness* that a given value was indeed incorporated into the accumulator. Accumulators were first proposed by Benaloh and de Mare [25] and have many applications including anonymous credentials and group signatures. Camenisch and Lysyanskaya [44] extended the notion of accumulators, by introducing *dynamic accumulators* which allow to dynamically delete and add values from or into the original set.

[2]Their protocol can also handle arbitrary ranges $[a, b]$ by using a trick suggested by Schoenmakers [132].

Definition 2.2.7 (Dynamic Accumulator [3])**.** *An accumulator for a family of inputs* $\{\mathbb{X}_k\}$ *is a family of families of functions* $\mathcal{G} = \{\mathbb{F}_k\}$ *with the following properties:*

Efficient Generation*: There is an efficient probabilistic algorithm* G *that in input* 1^k *produces a random element* f *of* $\{\mathbb{F}_k\}$ *and some auxiliary information about* f *called* aux_f*.*

Efficient evaluation*:* $f \in \{\mathbb{F}_k\}$ *is a polynomial time circuit that, on input* $(u, x) \in \mathcal{U}_f \times \mathbb{X}_k$*, outputs a value* $v \in \mathcal{U}_f$*, where* $\mathcal{U}_f$ *is an efficiently computable input domain for the function* f*; and* $\mathbb{X}_k$ *is the intended input domain whose elements are to be accumulated.*

Quasi-commutative*: For all* k*, for all* $u \in \mathcal{U}_f$*, for all* $x_1, x_2 \in \mathbb{X}_k$*,* $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$*. If* $X = \{x_1, \ldots, x_m\}$ *is a list of elements of* $\mathbb{X}_k$*, possibly with repetitions, then by* $f(u, X)$ *we denote* $f(f(\ldots(u, x_1), \ldots), x_m)$*.*

Witnesses*: Let* $v \in \mathcal{U}_f$ *and* $x \in \mathbb{X}_k$*. A value* $w \in \mathcal{U}_f$ *is called a* witness *for* x *in* v *under* f *if* $u = f(w, x)$*.*

Deletion*: There exist efficient algorithms* $\mathcal{D}, \mathcal{W}$ *such that, if* $v = f(u, X)$*,* $x, x' \in X$*, and* $f(w, x) = v$ *then*

- $\mathcal{D}(aux_f, v, x') = v'$ *such that* $v' = f(u, X/\{x'\})$*;*
- $\mathcal{W}(f, v, v', x, x') = w'$ *such that* $f(w', x) = v'$*.*

In the definition given above the addition operation is already implied by the efficient evaluation and the quasi-commutative properties.

A dynamic accumulator is secure against an adaptive adversary $\mathcal{A}$ who tries to produce a witness for a value x' which is not included in the current accumulator v. $\mathcal{A}$ is given access to an addition oracle: $Add(x)$ oracle which allows the addition of the value x to the accumulator i.e. such that $v = f(u, X \cup x)$ and a $Del(x)$ oracle which checks whether $x \in X$ and, if so, sets $v = \mathcal{D}(aux_f, v, x)$ and $X = X/\{x\}$. In both cases, the value v is returned to A.

[3] This definition is originally due to Baric and Pfitzmann [17] but was modified by Camenisch and Lysyanskaya [44] by adding the quasi-commutative property instead of the generation and the verification algorithms used before.

Definition 2.2.8. *A dynamic accumulator is secure if, for any poly-time adaptive adversary $\mathcal{B}$ involved in the following experiment, we have* $\boldsymbol{Adv}^{AccuSec}(\mathcal{A}) := Pr[\boldsymbol{Expt}^{AccuSec}(k) = 1] \in negl(k)$.

Experiment $\boldsymbol{Expt}^{AccuSec}(k)$

$(f, aux_f, u) \leftarrow G(1^k)$;

Initialize $X := 0, v := u$;

$(x, w, X) \leftarrow \mathcal{A}^{Add(\cdot), Del(\cdot)}(f, v)$:

If $X \subset \mathbb{X}_k, w \in \mathcal{U}'_f, x \in \mathbb{X}'_k, x \notin X, f(w, x) = f(u, X)$ *return 1;*

Return 0;

For the definition above note that only the legitimate accumulated values, $(x_1, \ldots, x_m)$, must belong to $\mathbb{X}_k$; the forged value x can belong to a possibly larger set $\mathbb{X}'_k$.

Let's now briefly describe how the CL accumulator works: let $n = pq$ where p and q are strong primes. In order to add a value $\tilde{x}$ to the accumulator value v, one can do $v' = v^{\tilde{x}} \bmod n$. To delete a value $\tilde{x}$ compute $v' = v^{\tilde{x}^{-1} \bmod (p-1)(q-1)} \bmod n$. To update a witness u after $\tilde{x}$ has been added, simply compute $u' = u^{\tilde{x}}$. In case that a value $\tilde{x} \neq x$ has been deleted, the update of the witness uses the extended GCD algorithm to compute values a, b such that $ax + b\tilde{x} = 1$ and then sets $u' = u^b c v'^a$.

2.3 Digital Signatures

A digital signature signature scheme consists of the following algorithms:

1. $KeyGen(1^k)$: is a probabilistic polynomial time key-generation algorithm which takes as input the security paremeter 1^k and outputs a pair of secret and public key (sk, pk).
2. $Sign(m, sk)$: is a probabilistic algorithm that outputs a signature σ on message m using signing key sk.
3. $Verify(m, \sigma, pk)$: outputs 1 if σ is a valid signature on m under the public key pk.

Definition 2.3.1 (Secure Signature Scheme [134])**.** *We say that a signature scheme is secure (against adaptive chosen message attacks) if it is Correct and Unforgeable.*

1. *Correctness.*

$$\forall m : \Pr[(sk, pk) \leftarrow \mathit{KeyGen}(1^k); \sigma \leftarrow \mathit{Sign}(m, sk) : \mathit{Verify}(m, \sigma, pk) = 1] = 1$$

2. *Unforgeability. Let $\mathcal{A}$ be an adversary with access to a signing oracle $\mathit{Sign}(sk, \cdot)$. Let Q be $\mathcal{A}$'s query tape where $\mathcal{A}$ records all his queries to the oracle. Then,*

$$\Pr[(sk, pk) \leftarrow \mathit{KeyGen}(1^k); (Q, m, \sigma) \leftarrow \mathcal{A}^{\mathit{Sign}(sk,\cdot)}(sk) : (m \notin Q) \wedge \mathit{Verify}(m, \sigma, pk)] \leq \nu(k).$$

Fiat-Shamir Heuristic. Fiat and Shamir [74] proposed a method to transform any three-round interactive proof system (Def. 2.2.2) with negligible soundness error, like Σ-protocols (Def. 2.2.5), into a digital signature scheme using a hash function, modeled as a random oracle.

To transform a three-round proof system into a signature scheme, one could, instead of a random challenge c, compute $c = H(a, m)$, where $H \rightarrow \{0,1\}^*$ is a hash function. Famous digital signatures that have been constructed from Σ-protocols using the Fiat-Shamir heuristic include Schnorr's [131] and GQ signatures [90] and they have been proven secure in the RO model [123].

Going back to our running example, Schnorr's proof of knowledge scheme can be easily turned into a signature scheme using the Fiat-Shamir heuristic. The Schnorr signature scheme is defined as follows:

1. $\mathit{KeyGen}(1^k)$: $(h, x) \leftarrow R_L$, $sk = x$, $pk = (h, H)$.

2. $\mathit{Sign}(m, sk)$: produce a honestly using x, set $c = H(m, a)$, produce r honestly for this c, output $\sigma(m) = (a, r)$.

3. $\mathrm{Ver}(m, (a, r))$: $c = H(a, m)$ and check if (a, c, r) is valid.

The Signer has a secret/public key pair (h, x) and a message m. To sign m the following steps take place: (1) $y \leftarrow \mathbb{Z}_q$, (2) $a = g^y$, (3) $c = H(m, a)$, and (4) $r = y + cx \bmod q$. The signature on the message is $\sigma(m) = (c, r)$ and in order to verify the signature, one should check whether $c = H(m, g^r/h^c)$.

2.3.1 Blind Signatures

In a blind signature scheme, first introduced by Chaum in 1982 [59], a user can have a document signed without revealing the contents of the document to the signer, and in such a way that the signer will not be able to recognize it later, when he sees the signature. Blind signatures have proven to be a very useful building block in applications requiring both anonymity and unforgeability, such as electronic cash (ecash) and anonymous credentials [36, 58, 45, 41, 20, 115].

We present the formal definitions for blind signatures as they were described in [99]. A blind signature scheme is a four-tuple consisting of two interactive Turing machines, the Signer and the User, (S, U) and two algorithms (Gen, $Verify$).

- $Gen(1^k)$: is a probabilistic polynomial time key-generation algorithm which takes as an input a security parameter 1^k and outputs a pair (pk, sk) of public and secret keys.

- $S(pk, sk)$, $U(pk, m)$: are polynomially- bounded probabilistic Interactive Turing machines who have the following (separate) tapes: read-only input tape, write-only output tape, a read/write work tape, a read-only random tape, and two communication tapes, a read-only and a write-only tape. They are both given (on their input tapes) as a common input a pk produced by a key generation algorithm. Additionally, S is given on her input tape a corresponding private key sk and U is given on her input tape a message m, where the length of all inputs must be polynomial in the security parameter 1^k of the key generation algorithm. Both S and U engage in an interactive protocol of some polynomial (in the security parameter) number of rounds. At the end of this protocol S outputs either "completed" or "not-completed" and U outputs either "fail" or $\sigma(m)$.

- $Verify(pk, m, \sigma(m))$: is a deterministic polynomial-time algorithm, which outputs "accept"/"reject" with the requirement that for any message m, and for all random choices of key generation algorithm, if both S and U follow the protocol then S always outputs "completed", and the output of U is always accepted by the verification algorithm.

A blind digital signature scheme is *secure* if for all the probabilistic polynomial time algorithms $\mathcal{A}$ there exists a security parameter $k_{\mathcal{A}}$, such that, for all $k > k_{\mathcal{A}}$ the following properties hold [99]:

- Blindness: the signer is unable to view the messages he signs (protection for the user). Furthermore, a malicious signer cannot link a $(m, \sigma(m))$ pair to any particular execution of the protocol. In order to define blindness formally consider the following experiment. Let $\mathcal{A}$ control the signer but not the user and $b \in \{0, 1\}$ is a randomly chosen bit which is kept secret from $\mathcal{A}$. $\mathcal{A}$ will try to guess the value of b by performing the following steps:

1. $(pk, sk) \leftarrow Gen(1^k)$
2. $\{m_0, m_1\} \leftarrow \mathcal{A}(1^k, pk, sk)$ (i.e. $\mathcal{A}$ produces two documents, polynomial in 1^k, where $\{m_0, m_1\}$ are by convention lexicographically ordered and may even depend on pk and sk).
3. We denote by $\{m_b, m_{1-b}\}$ the same two documents $\{m_0, m_1\}$, ordered according to the value of bit b, where the value of b is hidden from $\mathcal{A}$. $\mathcal{A}(1^k, pk,\ sk, m_0, m_1)$ engages in two parallel (and arbitrarily interleaved) interactive protocols, the first with $U(pk, m_b)$ and the second with $U(pk, m_{1-b})$.
4. If the fist User outputs on her private tape $\sigma(m_b)$ (i.e. does not output fail) and the second user outputs on her private tape $\sigma(m_{1-b})$ (i.e., also does not output fail) then $\mathcal{A}$ is given as an additional input $\{\sigma(m_0), \sigma(m_1)\}$.(We remark that we do not insist that this happens, and either one or both users may output fail).
5. $\mathcal{A}$ outputs a bit b' (given her view of steps 1 through 3, and if conditions are satisfied on step 4 as well).

Then the probability, taken over the choice of b, over coin-flips of the key-generation algorithm, the coin-flips of $\mathcal{A}$, and (private) coin-flips of both users (from step 3), that $b' = b$ is at most $\frac{1}{2} + \nu(k)$, where $\nu(k)$ is a negligible function.

-One-more Unforgeability: a user interacting with a signer S cannot output an additional, valid message/signature pair $(m, \sigma(m))$ no matter how many pairs of messages/ signatures of S he has seen (protection for the signer). To define that formally, consider an adversary $\mathcal{A}$ who controls the user but not the signer and executes the following experiment in order to get "one-more" signature (this is also called "one-more" forgery).

1. $(pk, sk) \leftarrow Gen(1^k)$
2. $\mathcal{A}(pk)$ engages in polynomially many (in k) adaptive, parallel and arbitrarily interleaved interactive protocols with polynomially many copies of $S(pk, sk)$, where $\mathcal{A}$ decides in an adaptive fashion when to stop. Let ℓ be the number of executions, where the Signer outputted "completed" in the end of Step 2.
3. $\mathcal{A}$ outputs a collection $\{(m_1, \sigma(m_1)), \ldots, (m_j, \sigma(m_j))\}$ subject to the constraint that $(m_i, \sigma(m_i))$ for $1 \leq i \leq j$ are all accepted by $Verify(pk, m_i, \sigma(m_i))$, and all m_i's are distinct.

Then the probability, taken over coin-flips of key-generation algorithm, the coin-flips of $\mathcal{A}$, and over the (private) coin-flips of the Signer, that $j > \ell$ is at most $\nu(k)$.

Shnorr Blind Signature. Schnorr's blind signature scheme is the most efficient of all the blind signature schemes proposed in the literature given that it can also be implemented using elliptic curves.

As described in Section 2.3, Schnorr's signature was constructed by applying the Fiat-Shamir heuristic over Shnorr's identification scheme. The blind version of Schnorr's signature would work as follows [66]:

Signer$(q, g, h = g^x)$		User(q, g, h, m)
$y \leftarrow \mathbb{Z}_q$, $a = g^y$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	$\alpha, \beta \leftarrow \mathbb{Z}_q$, $c' = H(m, ag^\alpha h^\beta)$, $c = c' + \beta$
$r = y + cx \bmod q$	$\xrightarrow{r}$	$g^r \stackrel{?}{=} ah^c$, $r' = r + \alpha$, output r', c'

We denote $g^{r'}h^{-c'}$ by a'. The signature is: $\sigma(m) = (a', c', r')$ and the verification checks whether $c' = H(m, a')$.

2.3.2 Malleable Signatures

A malleable (or homomorphic) signature scheme [7, 9, 57] is a special class of digital signatures that allows anyone to compute a signature of a message m' from a signature of m as long as m and m' satisfy some predicate. A mauled signature on m' reveals no extra information about the parent message m.

We adapt the definition by Chase et al. [57], who instead of a predicate consider a set of allowable *transformations*. A malleable signature scheme consists of the algorithms KeyGen, Sign, Verify and SigEval, of which the first three constitute a standard signature scheme. SigEval transforms multiple message/signature pairs into a new signed message: on input the verification key vk, messages $\vec{m} = (m_1, \ldots, m_n)$, signatures $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, and a transformation T on messages, outputs a signature σ' on the message $T(\vec{m})$.

Definition 2.3.2 (Malleability)**.** *A signature scheme* (KeyGen, Sign, Verify) *is* malleable *with respect to a set of transformations $\mathcal{T}$ if there exists an efficient algorithm* SigEval *that on input $(vk, T, \vec{m}, \vec{\sigma})$, where $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, $\mathsf{Verify}(vk, \sigma_i, m_i) = 1$ for all i, and $T \in \mathcal{T}$, outputs a signature σ' for the message $m := T(\vec{m})$ such that* Verify $(vk, \sigma', m) = 1$.

In order to achieve stronger unforgeability and context-hiding notions, Chase et al. [57] provide simulation-based definitions for malleable signatures. *Simulatability* requires the existence of a simulator, which without knowing the secret key can simulate signatures that are indistinguishable from standard ones. Moreover, a simulatable and malleable signature scheme is *context hiding* if a transformed signature is indistinguishable from a simulated signature on the transformed message. A malleable signature scheme is *unforgeable* if an adversary can only derive signatures of messages that are allowed transformations of signed messages. Malleable signatures can be constructed [57]

using controlled-malleable NIZKs [55] instantiated under the Decision Linear assumption [31].

Security Properties of Malleable Signatures. Assume the following expanded notion of a signature scheme in which the key generation process splits into two parts: a trusted algorithm Gen for generating universal parameters `crs`, and an algorithm KeyGen that, given these parameters, generates a keypair specific to a given signer.

Definition 2.3.3 (Simulatability)**.** *A signature scheme* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, where* Gen *outputs common parameters, is* simulatable *if there exists a PPT algorithm* KeyCheck *that on input* (crs, vk, sk) *outputs 1 iff* (vk, sk) *is in the range of* $\mathsf{KeyGen}(crs)$*, and a PPT simulator* $(\mathsf{SimGen}, \mathsf{SimSign})$ *such that the crs in* $(crs, \tau_s) \leftarrow \mathsf{SimGen}(1^\lambda)$ *is indistinguishable from* $crs \leftarrow \mathsf{Gen}(1^\lambda)$ *and the signatures produced by* SimSign *are indistinguishable from honest signatures; i.e., for all PPT adversaries* $\mathcal{A}$*:*

$$Pr\big[crs \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{A}^{S(crs,\cdot,\cdot,\cdot)}(crs) = 1\big] \approx Pr\big[(crs, \tau_s) \leftarrow \mathsf{SimGen}(1^\lambda) : \mathcal{A}^{S'(crs,\tau_s,\cdot,\cdot,\cdot)}(crs) = 1\big] \ ,$$

where S *on input* (vk, sk, m) *outputs* $\perp$ *if* $\mathsf{KeyCheck}(crs, vk, sk) = 0$ *and otherwise* $\mathsf{Sign}(crs, sk, m)$*. Similarly* S' *outputs* $\perp$ *if* $\mathsf{KeyCheck}(crs, vk, sk) = 0$ *and* $\mathsf{SimSign}(crs, \tau_s, vk, m)$ *otherwise.*

Definition 2.3.4 (Simulation context hiding)**.** *For a simulatable malleable signature scheme, a bit* b*, and an algorithm* $\mathcal{A}$*, let* $p_b^{\mathcal{A}}(\lambda)$ *be the probability that* $b' = 0$ *in the following game:*

1. $(crs, \tau_s) \leftarrow \mathsf{SimGen}(1^\lambda)$
2. $(state, vk, \vec{m}, \vec{\sigma}, T) \leftarrow \mathcal{A}(crs, \tau_s)$
3. *If* $\mathsf{Verify}(crs, vk, \sigma_i, m_i) = 0$ *for some* i *or* $T \notin \mathcal{T}$ *then output* $\perp$
4. *If* $b = 0$ *then* $\sigma \leftarrow \mathsf{SimSign}(crs, \tau_s, vk, T(\vec{m}))$

   If $b = 1$ *then* $\sigma \leftarrow \mathsf{SigEval}(crs, vk, T, \vec{m}, \vec{\sigma})$
5. $b' \leftarrow \mathcal{A}(state, \sigma)$

The signature scheme is simulation context hiding *if for all PPT algorithms* $\mathcal{A}$ *it holds that* $|p_0^{\mathcal{A}}(\lambda) - p_1^{\mathcal{A}}(\lambda)|$ *is negligible in* λ.

For *Unforgeability* Chase et al. require the existence of an extractor which given the transformed message and signature as well as the set of signed messages, outputs the transformation used. For this they assume an amplified setup $\mathsf{SimExtGen}$ that outputs (crs, τ_s, τ_e) where τ_e is the extraction trapdoor.

Definition 2.3.5 (Simulation Unforgeability)**.** *For a simulatable malleable signature* (Gen, KeyGen, $\mathsf{Sign}, \mathsf{Verify}$, $\mathsf{SigEval}$) *for a class of transformations* $\mathcal{T}$ *with an associate PPT simulator and extractor* ($\mathsf{SimExtGen}$, $\mathsf{SimSign}, \mathsf{SigExt}$)*, an adversary* $\mathcal{A}$ *and a table* $Q = Q_m \times Q_\sigma$ *that contains messages queried to* $\mathsf{SimSign}$ *and their responses, consider the following game:*

1. $(crs, \tau_s, \tau_e) \leftarrow \mathsf{SimExtGen}(1^\lambda); (vk, sk) \leftarrow \mathsf{KeyGen}(crs)$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SimSign}(crs, \tau_s, vk, \cdot)}(crs, vk, \tau_e)$
3. $(\vec{m}', T) \leftarrow \mathsf{SigExt}(crs, vk, \tau_e, m^*, \sigma^*, Q)$

The signature scheme is simulation unforgeable *if for all PPT algorithms* $\mathcal{A}$ *the probability that* $\mathsf{Verify}(vk, \sigma^*, m^*) = 1$ *and* $(m^*, \sigma^*) \notin Q$ *but either (1)* $\vec{m}' \not\subseteq Q_m$*, (2)* $m^* \neq T(\vec{m}')$*, or (3)* $T \notin \mathcal{T}$ *is negligible in* λ.

3 Security of Blind Signatures

As discussed in the Introduction, blind signatures have proved an essential building block for applications like anonymous credentials or electronic cash. One of the oldest, and most efficient blind signature schemes is the one due to Schnorr that is based on his famous identification scheme and we presented in Section 2.3.1. Although Schnorr's blind signature was proposed over twenty years ago, its unforgeability remains an open problem, even in the random-oracle model. In this Chapter, we show that current techniques for proving unforgeability in the random oracle model do not work for the Schnorr blind signature by providing a meta-reduction [33] which we call "personal nemesis adversary". Our meta-reduction is very interesting since it is the first that does not need to reset the adversary and thus can also rule out reductions to interactive assumptions. Our results generalize to other important blind signatures, such as the one due to Brands (which is at the heart of Microsoft's UProve system).

What are the implications of our results on the security of Schnorr blind signatures and generalizations? We must stress that our results do not in fact constitute an attack, and so for all we know, these schemes might very well be secure. However, we have essentially ruled out all *known* approaches to proving their security. So in order to give any security guarantee on these signature

schemes, the cryptographic community would have to come up with radically new techniques.

3.1 Security of Schnorr Blind Signature

As previously discussed, Schnorr's blind signature is based on his corresponding identification scheme. Thus, if the Schnorr identification scheme is not secure (i.e., after some number of interactions with the prover, the adversary can impersonate him), then the blind Schnorr signature is not one-more unforgeable. It was recently proved that the security of the Schnorr identification scheme cannot be proven under the discrete-logarithm assumption using black-box reductions in the standard model [120], so at the very least, it seems that Schnorr blind signatures require that we assume the security of Schnorr identification (also studied by Bellare and Palacio [23]). Perhaps an even stronger assumption may be reasonable. Can we prove it secure under even this or a stronger assumption?

To make this question more interesting, let us make it more general. Let us consider not just the Schnorr blind signature, but in general the blind variants of all Fiat-Shamir based signature schemes where the signer acts as the prover in an identification protocol. And let us see if they can be proven secure under any reasonable assumption (by reasonable, we mean an assumption that is not obviously false), not just specific ones.

Pointcheval and Stern showed that we can prove the security of blind signature schemes in the RO model when the underlying identification scheme is a witness-indistinguishable proof protocol for proving knowledge of a secret key, such that many secret keys are associated with the same public key [123, 125]. Their result does not apply to the original Schnorr blind signature, in which there is exactly one secret key corresponding to the public key. Other important blind signatures to which it does not apply are the Brands' blind signatures (the ones at the heart of Microsoft's UProve system), and the blind signatures based on the GQ signature [36, 90].

Oracle Replay Reduction. The idea of the Pointcheval-Stern reduction (also called "an oracle replay reduction") is to replay the attack polynomially many times with different random oracles in order to make the attacker successfully forge signatures. More precisely, we first run the attack with random keys, tapes and oracle f. Then, we randomly choose an index j and we replay with same keys and random tapes but with a new, different oracle f' such that the first $j-1$ answers are the same as before. We expect that, with non-negligible probability we will obtain two different signatures, σ, σ' of the same message m and we will be able to use them to solve a hard algorithmic problem (usually the one underlying the blind signature scheme) in polynomial time.

This proof technique works for standard (i.e. not blind) versions of the Schnorr, Brands and GQ signatures. They also showed that it works for a modification of Schnorr blind signature which is less efficient than the original Schnorr's. A very natural question is: can it work for the original Schnorr blind signature and its generalizations, such as the Brands or GQ blind signatures?

Let us take a closer look at oracle replay reductions, as used by Pointcheval and Stern. Their reduction can be modeled as a Turing machine that has a special tape that is used specifically for answering random oracle queries; it always uses the next unused value when answering, afresh, the next random oracle query. We call this type of reductions: *Naive RO replay reductions* and as we will discuss in Section 3.4.1 it can be used to model every known reduction for proving the security of digital signature schemes. Our result is that, in fact, naive RO replay reductions cannot be used to prove security of generalized Schnorr blind signatures, no matter how strong an assumption we make. Our result also holds for interactive assumptions or even if we assume the security of the blind signature scheme itself! Put another way, any such reduction can be used in order to break the underlying assumption.

3.2 Intractability Assumptions

Our meta-reduction rules out reductions for proving security of blind signatures under any *intractability assumption*. We will use the definition given by Pass [120]: an intractability assumption is modeled as an interaction between a probabilistic machine C (the challenger) and an attacker A where they are both given as input 1^k (k is the security parameter). A's running time is measured as a function of k.[1] Once A halts, the challenger outputs 1 or 0. Any challenger C together with a threshold function $t(\cdot)$ intuitively corresponds to the assumption:

> *For every polynomial time adversary A there exists a negligible function ν such that for all k, the probability that C outputs 1 after interacting with A is bounded by $t(k) + \nu(k)$.*

We say that A breaks C with respect to t with advantage p if: $Pr[\langle A, C\rangle(1^k) = 1] \geq t(k) + p$.

As Pass [120] notes, we can easily model all standard cryptographic assumptions as a challenger C and a threshold t. For example, the discrete logarithm assumption (Def. 2.1.1) corresponds to the threshold $t(k) = 0$ and the 2-round challenger C who on input 1^k picks a random x and sends g^x to A. If the attacker responds with $x' = x$ then C outputs 1.

3.3 Generalized Blind Schnorr Signature

Recall the definitions of a *unique witness relation*, Σ*-protocols* and *blind signatures* from Chapter 2. Our impossibility result refers to Schnorr's blind signature and its generalizations, defined as follows:

Definition 3.3.1 (Generalized Blind Schnorr Signature)**.** *A blind signature scheme* $(Gen, S, U, Verify)$ *is called Generalized Blind Schnorr Signature if:*

1. $(pk, sk) \in R_L$ *is a unique witness relation for a language* $L \in \mathcal{NP}$.

[1] Pass also requires that there be a limit to the rounds of interaction between A and C: an r-bounded assumption is one in which there exists some polynomial $r(\cdot)$ such that C on input 1^k communicates with A for at most $r(k)$ rounds; in this paper, however, assumptions that do not bound the number of rounds are still meaningful.

2. *There exists a Σ-protocol (P, V) for R_L such that for every $(pk, sk) \in R_L$ the prover's algorithm, $P(pk, sk)$, is identical to the signer's blind signing algorithm $S(pk, sk)$.*

3. *Let $Sign(pk, sk, m)$ be the signing algorithm implicitly defined by (S, U). Then, there exists a Σ-protocol $P(pk, sk)$, $V(pk)$ such that, in the random oracle (RO) model, a signature $\sigma = (a, c, r)$, where $c = H(m, a)$ is distributed identically to a transcript of the Σ-protocol.*

4. *There exists an efficient algorithm that on input (pk, sk), a "valid tuple" (a, c, r) and a value c', computes r' s.t. (a, c', r') is a valid tuple. (By "valid tuple" we mean a signature for which the verification equation holds.) Note that no additional information about a is required, such as, e.g. its discrete logarithm.*

Schnorr's blind signature falls under the generalized blind Schnorr signature category since: (1) The secret/public key pair is an instance of the DL problem which is a unique witness relation; (2) the signer's side is identical to the prover's side of the Schnorr identification scheme, which is known to be a Σ-protocol; (3) the signature $\sigma(m) = (a', c', r')$ is distributed identically to the transcript of the Schnorr identification protocol since a' comes uniformly at random from G; c' is truly random in the RO model, and r' is determined by α (4) finally, for a tuple (a, c, r) and a value c' one can compute $r' = r - cx + c'x$ so that (a, c', r') is still a valid tuple.

The definition also captures other well-known blind signature schemes, such as the blind GQ [90] and Brands [36] (for Brands also see Chapter 4).

3.4 Security of Generalized Blind Schnorr Signatures

As we mentioned above, one-more unforgeability of generalized blind Schnorr signatures is an open problem. In this section we will first define a general class of RO reductions and we will then prove that generalized blind Schnorr signature schemes cannot be proven unforgeable, and thus secure, using these reductions.

In our proof we make use of the "meta-reduction" method [33]: a separation technique commonly used to show impossibility results in cryptography. Let $\mathcal{A}$ be an adversary who breaks the unforgeability of generalized Schnorr blind signatures with non-negligible probability. We will use a meta-reduction (which we call "personal nemesis adversary") to show that there *cannot* exist a naive RO replay reduction, $\mathcal{B}$, which turns $\mathcal{A}$ into a successful adversary for *any* hard assumption that may be considered. We do that by transforming $\mathcal{B}$ through the meta-reduction into an algorithm that breaks the underlying assumption, without relying on the existence of a successful adversary.

3.4.1 Naive RO Replay Reductions

We first explicitly describe the type of reductions that our result rules out.

Definition 3.4.1 (Naive RO replay reduction)**.** *Let $\mathcal{B}$ be a reduction in the random-oracle model that can run an adversary $\mathcal{A}$, and may also reset $\mathcal{A}$ to a previous state, causing $\mathcal{A}$ to forget $\mathcal{B}'s$ answers to its most recent RO queries. We assume, without loss of generality, that if $\mathcal{A}$ has already queried the RO on some input x, and hasn't been reset to a state that's prior to this query, then $\mathcal{A}$ does not make a repeat query for x.*

We say that $\mathcal{B}$ is a naive RO replay reduction *if: $\mathcal{B}$ has a special random tape for answering the RO queries as follows: when $\mathcal{A}$ queries the RO, $\mathcal{B}$ retrieves the next value v from its RO tape, and replies with $c = f(b, v)$ where b is the input to the reduction, and f is some efficiently computable function.*

Let's now take a closer look at known reductions for proving security of signatures in the RO model and see whether they fall under the naive RO replay reduction category. We first look at the reduction given by Pointcheval and Stern [123] for proving security of blind signatures. Their reduction could be easily modeled as a naive RO replay reduction with f being the identity function. PS reductions are *perfect* since they always create a signature. The same holds for the reduction given by Abe [3]. To convince the reader that our way of modeling reductions in the RO model is

a very natural one, let us also look at the reduction given by Coron [65] proving the security of full domain hash (FDH) RSA signature. Coron's reduction works as follows: the reduction, $\mathcal{B}$, gets as input (N, e, y) where (N, e) is the public key and y is a random element from $\mathbb{Z}_N^*$ and tries to find $x = y^d \bmod n$. $\mathcal{B}$ runs an adversary $\mathcal{A}$, who can break the signature, with input the public key. As usual, $\mathcal{A}$ makes RO and signing queries which $\mathcal{B}$ answers. Whenever $\mathcal{A}$ makes an RO query, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_n^*$ and either returns $h = r^e \bmod N$ with probability p or returns $h = yr^e \bmod N$ with probability $1 - p$. So, it is pretty straightforward that we could model Coron's reduction as a naive RO replay reduction by interpreting the contents of an RO tape as r and the output of a p-biased coin flip (return either r^e or yr^e). Other well-known reductions used in the literature to prove security of digital signatures in the RO model can be modeled as naive RO replay reductions as well [30, 22, 24].

Note that $\mathcal{B}$ may invoke a new run of $\mathcal{A}$ based on an RO query received from an existing run. In that case, we still assume that, when $\mathcal{B}$ is ready to respond to an RO query from $\mathcal{A}$, it will do so with the value that is currently next up in the RO tape.

Programmability. Let us compare naive RO replay reductions with other previously defined types. Non-programmable random-oracle reductions [110] do not give the reduction the power to set the answers to the RO queries; instead these answers are determined by some truly random function. Naive RO replay reductions can be more powerful than that: they can, in fact, answer the adversary's queries in some way they find convenient, by applying the function f to the next value of their RO tape. However, they are not as powerful as the general programmable RO reductions: naive RO replay reductions are not allowed, for example, to compute an answer to an RO query as a function of the contents of the query itself. Fischlin et al. [76] also consider an intermediate notion of programmability, called "random re-programming reductions", which are incomparable to ours (but it would be interesting to extend our results to these reductions as well).

3.4.2 Theorem for Perfect Naive RO Replay Reduction

Our first result is on a simpler class of reductions called "perfect". We will extend it to non-perfect reductions in Section 3.4.3.

Definition 3.4.2 (Perfect-Naive RO replay reduction)**.** *A naive RO replay reduction $\mathcal{B}$ is called* perfect naive RO replay reduction *if $\mathcal{B}$ always gives valid responses to $\mathcal{A}$, i.e. its behavior is identical to that of the honest signer.*

We show that *perfect* naive RO replay reductions cannot be used to prove security of generalized blind Schnorr signature schemes.

Theorem 3.4.3. *Let $(Gen, S, U, Verify)$ be a generalized blind Schnorr signature scheme. Assume that there exists a polynomial-time perfect naive RO replay reduction $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ breaks an interactive intractability assumption C for every $\mathcal{A}$ that breaks the unforgeability of the blind signature (S, U). Then, C can be broken in polynomial time.*

We prove this theorem below. What are the consequences of this theorem for the Schnorr blind signatures, which is our running example? What we have shown is that, even if we assume security of the Schnorr identification scheme, and not just the hardness of the discrete logarithm problem, we still cannot exhibit a perfect naive RO replay reduction that will prove Schnorr blind signatures secure. In fact, somewhat oddly, even if we assume that the Schnorr blind signature scheme is secure, we still cannot find a perfect naive RO replay reduction $\mathcal{B}$ that will break this assumption should $\mathcal{A}$ be able to violate the unforgeability of the scheme. This is because a perfect naive reduction requires that the hash function queries be handled in a very specific way.

Proof of Theorem for Perfect Naive RO Replay Reduction. We start by introducing some terminology. Note that the reduction $\mathcal{B}$ is given black-box access to $\mathcal{A}$ and is allowed to run $\mathcal{A}$ as many times as it wishes, and instead of running $\mathcal{A}$ afresh every time, it may reset $\mathcal{A}$ to some previous state. At the same time, $\mathcal{B}$ is interacting with its own challenger C; we do not restrict C in any way.

Consider how $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{B}$ must give to $\mathcal{A}$ some public key pk for the signature scheme as input. Next, $\mathcal{B}$ runs the blind signing protocol with $\mathcal{A}$; recall that a generalized blind Schnorr signing protocol always begins with a message a from the signer to the user. When $\mathcal{B}$ runs $\mathcal{A}$ again, it can choose to give it the same (pk, a) or different ones. It is helpful for the description of the adversary we give, as well as for the analysis of the interaction, to somehow organize various calls that $\mathcal{B}$ makes to $\mathcal{A}$.

Every time that $\mathcal{B}$ runs $\mathcal{A}$, it either runs it "anew", providing a new public key pk and first message a, or it "resets" it to a previous state, in which some pk and a have already been given to $\mathcal{A}$. In the latter case, we say that $\mathcal{A}$ has been "reincarnated", and so, an *incarnation* of $\mathcal{A}$ is defined by (pk, a). Note that $\mathcal{B}$ may reincarnate $\mathcal{A}$ with the same (pk, a) several times. In this case, we say that this incarnation is *repeated*. Thus, if this is the i^{th} time that $\mathcal{A}$ has been reset to a previous state for this specific (pk, a), then we say that this is the i^{th} repeat of the (pk, a) incarnation. Without loss of generality, $\mathcal{B}$ never runs $\mathcal{A}$ anew with (pk, a) that it has used (i.e., if $\mathcal{B}$ has already created an incarnation for (pk, a), it does not create another one).

Let us consider what happens once $\mathcal{A}$ receives (pk, a). The signing protocol, in which $\mathcal{A}$ is acting as the user, expects $\mathcal{A}$ to send to $\mathcal{B}$ the challenge c. Additionally, $\mathcal{A}$ is free to make any random oracle queries it chooses. Once $\mathcal{B}$ receives c, the signing protocol expects it to send to $\mathcal{A}$ the response r. After that, the security game allows $\mathcal{A}$ to either request another signature, or to output a one-more signature forgery, i.e., a set of signatures (one more than it was issued); also, again, $\mathcal{A}$ can make RO queries. The adversaries that we consider in the sequel will not request any additional signatures, but will, at this point, output two signatures (or will fail).

Note that, if $\mathcal{B}$ is a perfect naive RO replay reduction (as defined above), then it will always provide to $\mathcal{A}$ a valid response r to the challenge c; while if it is not perfect, then it may, instead, provide an invalid response, or stop running $\mathcal{A}$ at this point altogether. Thus, a particular run can be:

- Uncompleted: no valid response, r, was given by $\mathcal{B}$ at the end of the protocol (cannot happen

if $\mathcal{B}$ is perfect).

- Completed but unsuccessful: a valid r was given but $\mathcal{A}$ was not able to output a forgery.
- Completed and successful: a valid r was given and $\mathcal{A}$ did output a forgery.

The technique we follow to prove our theorem is the following. We first define a special adversary which we call the *super adversary*, $s\mathcal{A}$, who exists if it is easy to compute the signing key for this signature scheme from the corresponding verification key. We do not show how to construct such an adversary (because we do not know how to infer the signing key for generalized blind Schnorr, and in fact we generally assume that it is impossible to do so in polynomial time); instead, we construct another adversary, the *personal nemesis adversary*, $p\mathcal{A}$, whose behavior, as far as the reduction $\mathcal{B}$ can tell, will be identical to $s\mathcal{A}$.

Note that, generally, an adversary is modeled as a deterministic circuit, or a deterministic non-uniform Turing machine: this is because, inside a reduction, its randomness can be fixed. Thus, we need $s\mathcal{A}$ to be deterministic. Yet, we need to make certain randomized decisions. Fortunately, we can use a pseudorandom function for that. Thus, $s\mathcal{A}$ is parametrized by s, a seed to a pseudorandom function $F_s : \{0,1\}^* \to \{0,1\}^k$ [2]. Additionally, it is parameterized by two messages m_1, m_2: signatures on these messages will be output in the end.

Consider a *Perfect super adversary*, $s\mathcal{A}_{s,m_1,m_2}$, that interacts with a signer as follows: on input the system parameters:

1. Begin signature issue with the signer and receive (pk, a).
2. Find sk.
3. Use sk to compute the signatures: pick a_1, a_2 and make two RO queries (m_1, a_1) and (m_2, a_2). Produce two forged signatures for m_1, m_2, denote them as σ_1 and σ_2 (remember that $s\mathcal{A}$ is deterministic so if reincarnated he makes the same RO queries).
4. Resume the signature protocol with the signer: send to the signer the value $c = F_s(trans)$ where *trans* is the current transcript between $s\mathcal{A}_{s,m_1,m_2}$, the RO and the signer, and receive

[2] We know that if $\mathcal{B}$ exists then secure signatures exist which imply one way functions existence and pseudorandom functions existence, so this is not an extra assumption.

from the signer the value r in response (which will always be valid for the perfect naive RO reduction $\mathcal{B}$).

5. Output the two message-signature pairs, (m_1, σ_1) and (m_2, σ_2).

Note that when $s\mathcal{A}$ executes the signature issue protocol with the signer it computes c as a pseudorandom function of its current transcript with the RO and the signer. Thus, there is only a very small probability (of about 2^{-k}) for $s\mathcal{A}$ to send the same c in another run.

The following lemma follows directly from the definition of a reduction $\mathcal{B}$:

Lemma 3.4.4. *If a perfect naive RO replay reduction $\mathcal{B}$ exists, then $\mathcal{B}^{s\mathcal{A}(\cdot)}$ $(pk, \textit{system params})$ solves the assumption C.*

Lemma 1 works even if the assumption C is an interactive one. That is why, $s\mathcal{A}$ and $p\mathcal{A}$ are defined in such a way that they do not reset the reduction $\mathcal{B}$.

Next, we define the personal nemesis adversary, $p\mathcal{A}$. Similarly to $s\mathcal{A}$, it is parameterized by (s, m_1, m_2); and so we denote it $p\mathcal{A}_{s,m_1,m_2}$. To the reduction $\mathcal{B}$, $p\mathcal{A}_{s,m_1,m_2}$ will look exactly the same as $s\mathcal{A}_{s,m_1,m_2}$, even though $p\mathcal{A}_{s,m_1,m_2}$ cannot compute sk. Instead, $p\mathcal{A}_{s,m_1,m_2}$ looks inside the reduction $\mathcal{B}$ itself; this is why we call $p\mathcal{A}_{s,m_1,m_2}$ "$\mathcal{B}$'s personal nemesis". The perfect $\mathcal{B}$'s personal nemesis adversary works as follows: on input the system parameters, $p\mathcal{A}_{s,m_1,m_2}$ performs a "one-more" forgery attack, using the following special powers: (1) $p\mathcal{A}_{s,m_1,m_2}$ has full access to $\mathcal{B}$'s random oracle tape; (2) in case $p\mathcal{A}_{s,m_1,m_2}$ is rewound, he remembers his previous state.

$p\mathcal{A}_{s,m_1,m_2}$ performs the one-more forgery for $\ell = 1$. Thus, he runs one signature issuing session with the signer and then outputs two valid signatures. Specifically, in it's ith incarnation, $p\mathcal{A}$ does the following:

1. Begin signature issue with the signer, and receive (pk, a).
2. Do nothing ($p\mathcal{A}$ cannot find sk).
3. - If (pk, a) is the same as in some previous incarnation j then make the same RO queries as the last time this incarnation was run ($s\mathcal{A}$ remembers the previous RO queries; obviously

it will receive different c_1, c_2 than before).

- If (pk, a) is a new tuple, then this is a new incarnation; do the following:
  - If $p\mathcal{A}$ has already computed the sk for this pk, then use this power to forge two signatures on (m_1, m_2); call the resulting signatures σ_1 and σ_2,
  - else (if sk not already known), $p\mathcal{A}$ computes two signatures using its special access to $\mathcal{B}$ by looking in advance what the next c_1, c_2 are going to be, then picking random [3] r_1, r_2 and solving for a_1, a_2 using the third property of generalized blind Schnorr signatures and the simulator from the underlying Σ-protocol. $p\mathcal{A}$ makes two RO queries of the form $(m_1, a_1), (m_2, a_2)$ and gets c_1, c_2 in response. Call the resulting signatures σ_1 and σ_2.

4. Resume the signature issue protocol with the signer: send to the signer the value $c = F_s(trans)$ where *trans* is the current transcript between $p\mathcal{A}$, the RO and the signer, and receive from the signer the value r in response (which will be valid for the perfect naive RO reduction $\mathcal{B}$).

5. - If this is the first time for this incarnation, then output the two message-signature pairs, (m_1, σ_1) and (m_2, σ_2) (completed and successful run).
   - If this is a repeat of some incarnation j, and the value $c = F_s(trans) \neq c_j$, where c_j is the corresponding value from incarnation j, then using r and r_j, property 3 of generalized blind Schnorr signatures and the extractability of the Σ-protocol, compute sk (if you don't already know it for this pk). Next, compute σ_1 and σ_2 consistent with the RO queries from incarnation j, using property 4 of generalized blind Schnorr signatures (completed and successful run).
   - If i is a repeat of j, and the value $c = F_s(trans) = c_j$, then fail (completed and unsuccessful run).

Lemma 3.4.5. *If $\mathcal{B}$ is a perfect naive RO replay reduction, then $\mathcal{B}$'s view in interacting with*

[3] Recall that $p\mathcal{A}$ uses a PRF that takes as input its current state in order to make each random choice.

$p\mathcal{A}_{s,m_1,m_2}$ is indistinguishable from its view when interacting with $s\mathcal{A}_{s,m_1,m_2}$.

Proof. In order to prove this, we will analyze the behavior of $s\mathcal{A}$ and $p\mathcal{A}$ step by step, as they were defined, and we will show that $\mathcal{B}$ receives indistinguishable views when interacting with $s\mathcal{A}_s$ or $p\mathcal{A}_s$ with all but negligible probability (to simplify notation we will omit writing the messages m_1, m_2 to the parameters given to the adversaries). We begin by defining $s\mathcal{A}_{Rand}$ and $p\mathcal{A}_{Rand}$ who behave exactly as $s\mathcal{A}_s$ and $p\mathcal{A}_s$ do but using a truly random source instead of the pseudorandom function F_s. We will use the following hybrid argument:

$$s\mathcal{A}_s \approx s\mathcal{A}_{Rand} \approx p\mathcal{A}_{Rand} \approx p\mathcal{A}_s$$

Let us first argue that $s\mathcal{A}_s \approx s\mathcal{A}_{Rand}$. This follows by a straightforward reduction that contradicts the pseudorandomness of F_s. Similarly, it holds that $p\mathcal{A}_{Rand} \approx p\mathcal{A}_s$.

We prove that $s\mathcal{A}_{Rand} \approx p\mathcal{A}_{Rand}$ by examining step by step the behavior of $s\mathcal{A}_{Rand}$ and $p\mathcal{A}_{Rand}$.

1. In the first step, both $s\mathcal{A}_{Rand}$ and $p\mathcal{A}_{Rand}$ begin the signature issuing with the Signer and wait for him to respond with (pk, a). From the point of view of $\mathcal{B}$ there is no difference whether talking to $s\mathcal{A}_{Rand}$ or $p\mathcal{A}_{Rand}$.

2. In the second step there is no interaction with $\mathcal{B}$.

3. Here we have two different cases on $p\mathcal{A}_{Rand}$'s behavior depending on whether the current incarnation is *repeated* or not. In both cases the interaction between $p\mathcal{A}_{Rand}$ and $\mathcal{B}$ consists of $p\mathcal{A}_{Rand}$ making two RO queries where $p\mathcal{A}_{Rand}$ either makes two RO queries on fresh values that it computed on the current step or makes the same RO queries as in the *repeated* incarnation (so, there is no difference for $\mathcal{B}$). Thus, in Step 3, no matter who $\mathcal{B}$ is talking to, $\mathcal{B}$ receives two RO queries distributed identically.

4. Step 4 is identical for both $s\mathcal{A}_{Rand}$ and $p\mathcal{A}_{Rand}$. Just send $c = R(trans)$, where R is a random function and receive from the signer the value r in response.

5. Since r will always be a valid response (recall that $\mathcal{B}$ is perfect), $s\mathcal{A}_{Rand}$ will always output two message-signature pairs, (m_1, σ_1) and (m_2, σ_2). $p\mathcal{A}_{Rand}$ will also output (m_1, σ_1) and (m_2, σ_2), which are distributed identically to the ones output by $s\mathcal{A}_{Rand}$ unless it is the case that the incarnation is a repeat of j and $c = R(trans) = c_j$. In that case $p\mathcal{A}_{Rand}$ fails. The probability that $c = R(trans) = c_j$ is only $2^{-\Theta(k)}$. Thus, with probability $1 - 2^{-\Theta(k)}$ $\mathcal{B}$'s view is identical no matter whether he is talking to $s\mathcal{A}_{Rand}$ or $p\mathcal{A}_{Rand}$.

So, by the hybrid argument we defined at the beginning of the proof, it holds that $s\mathcal{A}_s \approx p\mathcal{A}_s$. □

3.4.3 Theorem for Non-perfect naive RO replay reductions

Let's apply our result to a broader class of reductions by removing the requirement that our reduction be perfect, i.e. always outputs valid responses. Instead, we will require an upper bound L on the number of times that the reduction can invoke the adversary which is independent of $\mathcal{A}$'s success probability. Note that, of course, $\mathcal{B}$'s success probability needs to depend on $\mathcal{A}$'s success probability. However, the number of times it invokes $\mathcal{A}$ need not; in fact known reductions (such as Coron or Pointcheval and Stern) as a rule only invoke the adversary a constant number of times.

Definition 3.4.6 (L-Naive RO replay reduction)**.** *A naive RO replay reduction $\mathcal{B}$ is called* L-naive RO replay reduction *if there is a polynomial upper bound L on how many time $\mathcal{B}$ resets $\mathcal{A}$; this upper bound is a function of the number of RO queries that $\mathcal{A}$ makes, but otherwise is independent of $\mathcal{A}$, in particular, of $\mathcal{A}$'s success probability.*

Our previous analysis wouldn't work for the *L-naive RO replay reduction.* Think of the scenario where $p\mathcal{A}$ receives a message a from $\mathcal{B}$ for the first time but is not given a valid r at the end. Then in the repeat of this incarnation, $p\mathcal{A}$ will have to make the same two RO queries he did before and output forgeries if given a valid r at the end. But, given the definitions of $\mathcal{B}$ and $p\mathcal{A}$ we gave before, $p\mathcal{A}$ will now get different c_1 and c_2 for his RO queries and thus he will not be able to output the same forgeries he had prepared before.

What changes in our new analysis is that: (a) $p\mathcal{A}$ is also given write access to $\mathcal{B}$'s RO tape, and (b) both $p\mathcal{A}$ and $s\mathcal{A}$ will be successful in producing a forgery with probability only $1/(\binom{L}{2} + L)$.

The following theorem shows that L-naive RO replay reductions cannot be used to prove security of generalized blind Schnorr signature schemes.

Theorem 3.4.7. *Let $(Gen, S, U, Verify)$ be a generalized blind Schnorr signature scheme. Suppose that there exists a polynomial-time L-naive RO replay reduction $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ breaks an intractability assumption C for every $\mathcal{A}$ that breaks the unforgeability of the blind signature (S, U). Then, C can be broken in polynomial time.*

This theorem rules out a broader class of security reductions. If we look back to our running example of Schnorr blind signatures, this theorem shows that under any assumption (DL, security of Schnorr identification, etc.) we cannot find an L-naive RO replay reduction to prove its security.

Proof of theorem for L-naive RO replay reduction Similar to what we did before, we first define the *super adversary* $s\mathcal{A}_{s,m_1,m_2,L}$ who knows L and works as follows: on input the system parameters:

1. Begin signature issue with the signer and receive (pk, a). Decide whether this is going to be a successful incarnation: choose "successful" with probability $1/(\binom{L}{2} + L)$ and "unsuccessful" with probability $1 - 1/(\binom{L}{2} + L)$.
2. Find sk.
3. Use sk to compute the signatures: pick a_1, a_2 and make two RO queries (m_1, a_1) and (m_2, a_2). Produce two forged signatures for m_1, m_2, denote them as σ_1 and σ_2.
4. Resume the signature protocol with the signer: send to the signer the value $c = F_s((trans))$ where *trans* is the current transcript between $s\mathcal{A}$, the RO and the signer, and receive from the signer the value r in response.
5. - If r is not valid, then this was an uncompleted run, then fail.

- If r valid (completed run) and in Step 1 it was decided that this is a successful incarnation, output the two message-signature pairs, (m_1, σ_1) and (m_2, σ_2). Otherwise fail.

The following lemma (similar to Lemma 1) follows from the definition of $\mathcal{B}$:

Lemma 3.4.8. *If an L-naive RO replay reduction $\mathcal{B}$ exists, then $\mathcal{B}^{s\mathcal{A}(\cdot)}$ $(pk, \text{system params})$ solves the assumption C.*

Now we are going to define the personal nemesis adversary, $p\mathcal{A}_{s,m_1,m_2,L}$, which on input the system parameters, $p\mathcal{A}_{s,m_1,m_2,L}$ performs a "one-more" forgery attack, using the following special powers: (1) $p\mathcal{A}_{s,m_1,m_2,L}$ has full read and write access to $\mathcal{B}$'s random oracle tape; (2) in case $p\mathcal{A}_{s,m_1,m_2,L}$ is rewound, it does remember his previous state.

$p\mathcal{A}_{s,m_1,m_2,L}$ performs the one-more forgery for $\ell = 1$. Thus, it runs one signature issuing session with the signer and then outputs two valid signatures with probability $\frac{1}{\binom{L}{2}+L}$. Specifically, in it's i^{th} incarnation[4], $p\mathcal{A}_{s,m_1,m_2,L}$ does the following:

1. Begin signature issue with the signer, and receive (pk, a).
2. Do nothing.
3. - If (pk, a) is received for the first time, then this is a new incarnation; do the following:
     - If $p\mathcal{A}$ has already found sk for this pk, then use this power to forge two signatures on (m_1, m_2) (still required to make two RO queries); call these signatures σ_1 and σ_2,
     - else, $p\mathcal{A}$ guesses (i_1, i_2) where $i_1 (\leq i_2)$ denotes the repeat where c_1 will be given in response to $p\mathcal{A}$'s next RO query; and i_2 is $p\mathcal{A}$'s guess for the first completed repeat of this incarnation. Then, $p\mathcal{A}$ randomly picks v_1, v_2, computes $c_1 = f(v_1), c_2 = f(v_2)$, picks r_1, r_2, solves for a_1, a_2 using the third property of generalized blind Schnorr signatures and the simulator from the underlying Σ-protocol and computes two signatures σ_1 and σ_2.

[4]Recall that the terms "incarnation", "completed" run, "successful run" were defined in Section 3.4.2.

- $p\mathcal{A}$ makes two RO queries of the form $(m_1, a_1), (m_2, a_2)$ (the two RO queries are always the same for a specific incarnation).
- If this is the repeat incarnation i_1, and $\mathcal{B}$ wants a fresh answer to the query (m_1, a_1) then write v_1 on $\mathcal{B}$'s RO tape; else (if this isn't repeat i_1) write a random v_1'.
- If this is the repeat incarnation i_2 then write v_2 on $\mathcal{B}$'s RO tape; else (if this isn't repeat i_2) write a random v_2'.

4. Resume the signature issue protocol with the signer: send to the signer the value $c = F_s(\mathit{trans})$ where F_s is a PRF and *trans* is the current transcript between $p\mathcal{A}$, the RO and the signer, and wait to receive the value r as a response from the signer.
5. - If r is valid (completed run):
     - If already know the secret key, sk, then output (m_1, σ_1) and (m_2, σ_2) with probability $\frac{1}{\binom{L}{2}+2}$ or else fail.
     - If this is the first time for this incarnation, then output the two message-signature pairs, (m_1, σ_1) and (m_2, σ_2).
     - If this is the second successful repeat for this incarnation and the value $c = F_s(\mathit{trans}) \neq c_j$, where c_j is the corresponding value from the j^{th} run of this incarnation, then using r and r_j solve for sk using property 4 of generalized Schnorr signatures. Next, compute σ_1 and σ_2 consistent with the RO queries from this incarnation.
     - If this is the second successful repeat for this incarnation but $c = F_s(\mathit{trans}) = c_j$, then fail (unsuccessful run).
     - If the guess (i_1, i_2) was correct (that is, this is repeat i_2 of this incarnation, it was successful, and $\mathcal{B}$'s answer to (m_1, a_1) was the same as in incarnation i_1; and in incarnation i_1, $\mathcal{B}$ wanted a fresh answer to the (m_1, a_1) RO query) then output the two message-signature pairs, (m_1, σ_1) and (m_2, σ_2).
     - If the guess (i_1, i_2) was wrong then fail (unsuccessful run).

- If r is not valid or r was not received then fail.

Lemma 3.4.9. *If $\mathcal{B}$ is an L-naive RO replay reduction, then $\mathcal{B}$'s view in interacting with $p\mathcal{A}_{s,m_1,m_2}$ is indistinguishable from its view when interacting with $s\mathcal{A}_{s,m_1,m_2}$.*

Proof. Similarly to the proof of Lemma 2, we first consider $p\mathcal{A}$ and $s\mathcal{A}$ that, instead of access to a pseudorandom function F_s have access to a truly random function *Rand*. Just as before, by pseudorandomness of F_s, $p\mathcal{A}_{s,m_1,m_2} \approx p\mathcal{A}_{Rand,m_1,m_2}$ and $s\mathcal{A}_{s,m_1,m_2} \approx s\mathcal{A}_{Rand,m_1,m_2}$; so it is sufficient to show that $p\mathcal{A}_{Rand,m_1,m_2} \approx s\mathcal{A}_{Rand,m_1,m_2}$. (We will omit the subscripts "$Rand, m_1, m_2$" in the rest of the proof.)

Consider $\mathcal{B}$'s view when interacting with $s\mathcal{A}$ for fixed (pk, a), i.e. in a given incarnation. Until $\mathcal{B}$ completes the incarnation by sending a valid response r, $\mathcal{B}$ does not know whether this incarnation is successful or not; thus $\mathcal{B}$'s view with $s\mathcal{A}$ is identical to his view with $s\mathcal{A}'$ defined as follows: $s\mathcal{A}'$ remembers previous times when $\mathcal{B}$ ran it. It is identical to $s\mathcal{A}$, except that it decides (at random) whether or not this incarnation is successful the first time that $\mathcal{B}$ correctly completes this incarnation by sending to $s\mathcal{A}'$ the correct r in Step 4. The way that $s\mathcal{A}'$ will determine whether this is a successful incarnation is by picking (i_1, i_2) the way that $p\mathcal{A}$ does, and then making the incarnation successful if it picked them correctly; note that $s\mathcal{A}'$ makes an incarnation successful if it picks the unique correct (i_1, i_2) out of $\binom{L}{2} + L$ possibilities ($\binom{L}{2}$ ways of picking $i_1 \neq i_2$, L ways to pick $i_1 = i_2$).

Next, let us compare $\mathcal{B}$'s view with $s\mathcal{A}'$ with his view with $p\mathcal{A}$. They make identically distributed queries to the RO; then they successfully produce forgeries whenever they have correctly guessed i_1 and i_2 (except if $p\mathcal{A}$ sends the same query c in both the first and second complete run of this incarnation, which happens with only negligible probability). Therefore, the views that $\mathcal{B}$ receives when talking to $s\mathcal{A}'$ and $p\mathcal{A}$ are statistically indistinguishable, which completes the proof of the lemma. □

3.5 Related work

Security of blind signature schemes. Schnorr and Jakobsson [66] proved security of the Schnorr blind signature in the combined random oracle and generic group model. The generic group model is a very restricted setting in which the only way to sample group elements is by applying group operations; this does not correspond to any intractability assumption.

Fischlin and Schröder [77] show that proving security of a broad class of blind signature schemes (which, in particular, includes what we refer to as generalized Schnorr blind signatures) via black-box reductions in the standard model is as hard as solving the underlying hard problem. Their technique uses the meta-reduction paradigm to show that black-box reductions for this type of blind signatures can be turned into solvers for hard non-interactive assumptions. However, their result does not rule out reductions in the random-oracle model, and in fact is technically very different from ours for that reason.

Rafael Pass studied the assumptions needed for proving security of various cryptographic schemes [120]. In particular, relevant to our work, he considers the Schnorr identification scheme and variants, and a category of blind signatures called "unique blind signatures." Pass considers whether so-called *r-bounded-round* assumptions are strong enough to prove, in a black-box fashion in the standard model, the security of certain schemes when repeated more than r times. His results apply to Schnorr blind signatures (and their generalizations) in the following way: he shows that no so-called bounded-round assumption can imply secure composition of the Schnorr identification scheme using black-box reductions (and therefore the Schnorr blind signature).

Here is how our work goes beyond what was shown by Pass [120] for "unique blind signatures." First of all, we do not limit our consideration to *r-bounded-round* assumptions but we show that our result applies for every possible intractability assumption. Thus, we rule out the existence of a very special type of reduction, the naive RO replay one, that models all the known reductions for proving security of digital signatures, *irrespective of assumption*. As an example, consider the One

More Discrete Logarithm assumption (OMDL) [22] which has been used to prove security of the Schnorr identification scheme against active attacks [23]. Our result directly implies that Schnorr blind signature cannot be proven secure under the OMDL assumption in the RO model. Finally, our result applies even after *just one signature was issued* whereas Pass' result questions the security of schemes when repeated more than r times.

Finally, we study a very specific way of programming the random oracle reductions. Other variants of random oracle reductions have been considered in the literature and their relative strengths have been studied in prior work [76, 110]. In Section 3.4.1, we compare these variants with ours.

The Meta-Reduction Technique. On a relevant note, the meta-reduction technique has been used to analyze security of Schnorr signatures among other cryptographic schemes. Paillier and Vergnaud [117] showed that the security of Schnorr signatures cannot be based on the difficulty of the one more discrete logarithm problem in the standard model. Fischlin and Fleischhacker [75] extended their result by showing that the security of Schnorr signatures cannot be based to the discrete logarithm problem without programming the random oracle. Their work is also relevant to ours since the meta-reduction they define also doesn't need to reset the reduction[5]. However, their result applies to non-programming reductions while our naive RO replay reductions fall somewhere in between the programmable and non-programmable setting (see Section 3.4.1 for a discussion about programmability). Finally, their result holds only for reductions to the discrete logarithm problem and holds for a very limited class of reductions: those that run a single copy of the adversary. Thus, our result is much broader.

3.6 Conclusions

In this Chapter we showed that current techniques for proving security of Schnorr blind signature in the random oracle model do not work. To prove our result we provided a meta-reduction which

[5]This is a result that Fischlin and Fleischhacker [75] obtained after the first version of our manuscript appeared on eprint [13]; our result is in fact the first in which a meta-reduction works without resetting the reduction $\mathcal{B}$.

we call "personal nemesis adversary". What makes our technique particularly interesting is that for the *first time* we introduce a meta-reduction (our personal nemesis adversary) that does not need to reset the reduction $\mathcal{B}$, as it is usually done when using the meta-reduction paradigm. For example, our personal nemesis adversary could reset the reduction $\mathcal{B}$, get an additional signature and return this signature back to $\mathcal{B}$ as his forgery. However, this resetting makes things more complicated since the two executions are correlated. Our technique, instead, is much simpler: the personal nemesis adversary, $p\mathcal{A}$, will simply interact with the reduction $\mathcal{B}$ the way an actual adversary would (but taking advantage of powers not available to an adversarial algorithm, such as remembering its prior state if and when the reduction resets it, and having access to the reduction's random oracle tape), without resetting it at any time. When $\mathcal{B}$ halts, if it succeeded in breaking the assumption (as it should with non-negligible probability, or it wouldn't be a valid security reduction), $p\mathcal{A}$ has succeeded too — but *without* assuming the existence of an actual adversary that breaks the security of the underlying signature scheme.

As we will see in the next Chapter, our results generalize to other important blind signatures, such as the one due to Brands. Brands' blind signature is at the heart of Microsoft's newly implemented UProve system, which makes this work relevant to cryptographic practice as well.

4 Security of U-Prove

In this Chapter we investigate the security of U-Prove which is the most well known single-use credential scheme and is currently implemented by Microsoft. U-Prove is based on the blind signature scheme proposed by Stefan Brands [36]. As we will show, Brands blind signature falls under the category of generalized blind Schnorr signatures (as defined in Def. 3.3.1), and thus, we cannot prove its unforgeability using currently know techniques. We then propose a modification of Brands blind signature that is provably unforgeable. As a blind signature, the resulting signature scheme is inferior, in efficiency, to the provably secure variant of the Schnorr blind signature. As far as its use in an anonymous credentials system is concerned, it is still an open problem since there is no proof that a user who shows the same (single-use) credential more than once, will be identified.

4.1 Brands' Blind Signature

Let us first describe Brands blind signature scheme [36] . G is a group of order q, where q is a k-bit prime, and g is a generator of the group. The signer holds a secret key $x \leftarrow \mathbb{Z}_q$ and the corresponding public key $h = g^x$, while the user knows signer's public key h as well as g, q. $\mathcal{H}$ is a collision resistant

hash function. The signature issuing protocol works as follows:

Signer (g, h, x)		User(g, h)
		$a \in_R \mathbb{Z}_q$
	$\xleftarrow{\quad \alpha \quad}$	$m = g^\alpha$
$w \in_R \mathbb{Z}_q$		
$z \leftarrow m^x$		
$a \leftarrow g^w$		
$b \leftarrow m^w$		
	$\xrightarrow{\quad z, a, b \quad}$	
		$s, t \in_R \mathbb{Z}_q$
		$m' \leftarrow m^s g^t$
		$z' \leftarrow z^s h^t$
		$u, v \in_R \mathbb{Z}_q$
		$a' \leftarrow a^u g^v$
		$b' \leftarrow a^{ut} b^{us} (m')^v$
	$\xleftarrow{\quad c \quad}$	$c' \leftarrow \mathcal{H}(m', z', a', b')$
		$c \leftarrow c'/u \bmod q$
$r \leftarrow w + cx \bmod q$	$\xrightarrow{\quad r \quad}$	$h^c a \stackrel{?}{=} g^r$
		$z^c b \stackrel{?}{=} m^r$
		$r' \leftarrow ur + v \bmod q$

Table 4.1: Brands Blind Signature

A signature on m' is $\sigma(m') = (z', a', b', c', r')$. Anyone can verify a signature by first computing $c' = \mathcal{H}(m', z', a', b')$ and then checking whether the following equations hold: $h^{c'} a' \stackrel{?}{=} g^{r'}$, $(z')^{c'} b' \stackrel{?}{=} (m')^{r'}$.

4.1.1 Security of Brands' Blind Signatures

We now argue that the security of Brands blind signature cannot be proved via a perfect naive or an L-naive RO replay reduction.

Corollary 4.1.1. *If there exists a perfect or an L-naive RO replay reduction $\mathcal{B}$ that solves any intractability assumption C using an adversary $\mathcal{A}$ that breaks the unforgeability of Brands' signature, then assumption C can be solved in polynomial time with non-negligible probability.*

Proof. In order for this corollary to hold we need to show that Brands' blind signature is a generalized blind Schnorr signature. We can show this by inspecting one by one the needed requirements:

1. Brands public/secret key pair is $(h = g^x, x)$, which is a unique witness relation for $L = \{h : g^x = h\} \in \mathcal{NP}$,
2. the signer's side of Brands blind signature is the same as the prover's side in Schnorr's identification scheme, which is known to be a Σ-protocol,
3. Brands blind signature is of the form $\sigma(m') = ((z', a', b'), c', r')$ which has identical distribution to a transcript of a Σ-protocol, as we will explain below
4. given the secret key x and a valid transcript of Brands scheme: $(\hat{a}, c'_1, r'_1)$, where $\hat{a} = (z', a', b')$, then $\forall\ c'_2$ we can compute r'_2 as: $r'_2 = r'_1 - c'_1 x + c'_2 x$ so that $(\hat{a}, c'_2, r'_2)$ is still a valid transcript.

Let's take a closer look at Brands blind signature and see why it is a Σ-protocol. We will do so by inspecting the three properties of Σ-protocols: (a) it's a three-round protocol, (b) for any h and any pair of accepting conversations $(\hat{a}, c'_1, r'_1)$ and $(\hat{a}, c'_2, r'_2)$ where $c'_1 \neq c'_2$ one can efficiently compute x such that $h = g^x$ and (c) there exists a simulator S who on input h and a random c' picks r', m and z, solves for a', b', so he can output an accepting conversation of the form $((z', a', b'), c', r')$.

Thus, by applying Theorems 3.4.3 and 3.4.7, we rule out perfect and L-naive RO replay reductions for Brands' blind signatures.

4.1.2 DLP with Schnorr Prover

It was originally claimed that Brands scheme was secure under the existence of a Schnorr prover [36]. Here, we define the discrete logarithm problem (DLP) under a Schnorr prover and show that it can be modeled as an intractability assumption. Thus, even if we give a perfect or an L- naive RO replay reduction access to the prover side of the Schnorr identification scheme (we will call him Schnorr prover from now on) and an adversary $\mathcal{A}$ that performs a one-more forgery attack on the Brands signature scheme, the reduction cannot solve the discrete logarithm problem unless the DLP is easy in this setting. Let us now define the discrete logarithm problem in the setting with a Schnorr prover.

Definition 4.1.2 (DLP with the Schnorr prover)**.** *Let $\mathcal{A}^{(\cdot)}(\cdot)$ be an oracle Turing machine that takes as input a discrete logarithm instance (G, q, g, h) and has oracle access to the Schnorr prover $Schnorr(G, q, g, x)$ for $h = g^x$. (That is to say, $\mathcal{A}$ may act as the verifier in the Schnorr protocol, as many times as it wishes.) Upon termination, $\mathcal{A}$ outputs a value x'. We say that $\mathcal{A}$* solves *the discrete logarithm problem with the Schnorr prover if $x' = x$.*

Definition 4.1.3 (Security of DLP with Schnorr)**.** *For any probabilistic poly-time family of oracle Turing machines, $\mathcal{A}^{(\cdot)}(\cdot)$, there exists a negligible function $\nu(k)$ such that*

$$Pr[(G, q, g, g^x) \leftarrow S(1^k); x' \leftarrow \mathcal{A}^{Schnorr(g,x)}(g, g^x) : x' = x] = \nu(k).$$

where $S(1^k)$ samples Discrete Logarithm instances of size k.

DLP with Schnorr can easily be modeled as an intractability assumption similar to the standard DLP. It corresponds to threshold $t(k) = 0$ and a 2-round challenger C who on input 1^k picks a random x and sends g^x to the adversary $\mathcal{A}$ which works as defined in Definition 10. $\mathcal{A}$, having oracle access to the Schnorr prover responds with x' to the challenger. If $x' = x$ then C outputs 1. □

4.2 Modifying Brands' Signature

As mentioned in Section 3.1, Pointcheval and Stern [123] showed that we can prove the security of a blind signature scheme in the RO model if the underlying identification scheme is a witness-indistinguishable proof protocol for proving knowledge of a secret key, such that many secret keys are associated with the same public key. One could modify Brands' scheme similarly to how the original Schnorr blind signature was modified to obtain the variant that Pointcheval and Stern proved secure. In this Section we propose such a modification; the public key of the signer will be of the form $H = G_1^{w_1} G_2^{w_2}$ where (H, G_1, G_2) are public and (w_1, w_2) are the secret key.

The setup is as follows: let G be a group of prime order q and three generators $G_1, G_2, g_2 \in G_q$. No

one in the system should be able to compute $\log_{G_1} G_2$. The secret key of the signer is $sk = (w_1, w_2)$, where $w_1, w_2 \in \mathbb{Z}_q$ and the public key is $pk = (H, G_1, G_2)$ where $H = G_1^{w_1} G_2^{w_2}$. The new blind signature is described in Table 4.2.

Signer$((H, G_1, G_2), (w_1, w_2))$		User(H, G_1, G_2)
		$U \in \mathbb{Z}_q$
		$g_1 = G_1^U G_2,\ \pi =$ proof of knowledge of U
	$\xleftarrow{g_1, \pi}$	
verify π		
$w_3 \in \mathbb{Z}_q$		
$h = g_1^{w_1} g_2^{w_3}$		
$v_1, v_2.v_3 \in_R \mathbb{Z}_q$		
$V_1 = G_1^{v_1} G_2^{v_2}$		
$V_2 = g_1^{v_1} g_2^{v_3}$		
	$\xrightarrow{h, V_1, V_2}$	
		$r', z_1', z_2', z_3' \in_R \mathbb{Z}_q$
		$V_1' = G_1^{z_1'} G_2^{z_2'} H^{-r'}$
		$V_2' = g_1^{z_1'} g_2^{z_3'} h^{-r'}$
		$s, k \in_R \mathbb{Z}_q$
		$\tilde{h} = h^s g_2^k$
		$\tilde{g_1} = g_1^s$
		$\tilde{V_1} = V_1 V_1'$
		$\tilde{V_2} = (V_2 V_2')^s$
		$b_1, b_2 \in_R \mathbb{Z}_q$
		$m = G_1^{b_1} G_2^{b_2}$
		$\tilde{r} = \mathcal{H}(m, (H, G_1, G_2), (\tilde{h}, \tilde{g_1}, g_2), (\tilde{V_1}, \tilde{V_2}))$
		$r = \tilde{r} - r'$
	$\xleftarrow{r}$	
$z_i = rw_i + v_i,\ i \in \{1, \ldots, 3\}$		
	$\xrightarrow{z_1, z_2, z_3}$	
		$V_1 H^r \stackrel{?}{=} G_1^{z_1} G_2^{z_2}$
		$V_2 h^r \stackrel{?}{=} g_1^{z_1} g_2^{z_3}$
		$\tilde{z_1} = z_1 + z_1'$
		$\tilde{z_2} = z_2 + z_2'$
		$\tilde{z_3} = (z_3 + z_3')s + k\tilde{r}$

Table 4.2: Brands' blind signature modified

The blind signature on $(m, \tilde{g_1})$ is $\sigma_B(m, \tilde{g_1}) = ((H, G_1, G_2), (\tilde{h}, \tilde{g_1}, g_2), (\tilde{V_1}, \tilde{V_2}), (\tilde{z_1}, \tilde{z_2}, \tilde{z_3}))$.

In order for somebody to verify the signature, he needs to check:

$$\begin{aligned} \tilde{V}_1 H^{\tilde{r}} &\stackrel{?}{=} G_1^{\tilde{z}_1} G_2^{\tilde{z}_2} \\ \tilde{V}_2 \tilde{h}^{\tilde{r}} &\stackrel{?}{=} \tilde{g_1}^{\tilde{z}_1} g_2^{\tilde{z}_3}. \end{aligned}$$

4.2.1 Unforgeability of Modified Brands Blind Signature

By modifying Brands blind signature scheme so that the signer's public key is constructed by using more than one secret keys, we can apply the technique proposed in [123] to prove the unforgeability of the modified withdrawal protocol. We will prove the following Theorem.

Theorem 4.2.1. *Consider the modified Brands' blind signature scheme in the random oracle model. If there exists a probabilistic polynomial time Turing machine which can perform a "one - more" forgery, with non-negligible probability, even under a parallel attack, then the discrete logarithm can be solved in polynomial time.*

Proof. We first describe an outline of the proof, then we will simplify the notations and finally we will complete the proof.

Outline of the proof Let $\mathcal{A}$ be the attacker who can be described as a probabilistic polynomial time Turing machine with random tape ω. Thus, there exists an integer ℓ such that after ℓ interactions with the signer $(v_{1,i}, v_{2,i}, r_i, z_{1,i}, z_{2,i}, z_{3,i})$ for $i \in \{1, \ldots, \ell\}$, and a polynomial number Q of queries asked to the random oracle, $\mathcal{Q}_1, \ldots, \mathcal{Q}_Q$, $\mathcal{A}$ returns $\ell + 1$ valid signatures (coins), $(m_i, \tilde{g}_{1,i}), \tilde{h_i}, (\tilde{V}_{1,i}, \tilde{V}_{2,i}), (\tilde{z}_{1,i}, \tilde{z}_{2,i}, \tilde{z}_{3,i},$ for $i = 1, \ldots, \ell + 1$ (to verify the signature (coin) you would first need to compute $\tilde{r}_i = \mathcal{H}(m_i, (H, G_1, G_2), (\tilde{h}_i, \tilde{g}_{1,i}, g_2), (\tilde{V}_{1,i}, \tilde{V}_{2,i}))$).

The signer possesses a secret key (w_1, w_2) associated to a public key $H = G_1^{w_1} G_2^{w_2}$, and a random tape Ω. The secret key is stored on the knowledge tape of the Turing machine.

Through a collision of the signer and the attacker, we want to compute the discrete logarithm of G_1 relatively to G_2. The technique used is the one described in [124] as technique of "oracle replay".

We first run the attack with random keys, tapes and oracle f (which answers the hash queries). We randomly choose an index j and then replay with same keys and random tapes, but a different oracle f' such that the first $j-1$ answers remain unchanged. We expect that, with non-negligible probability, both executions output a common $\tilde{V}_1$ and $\tilde{V}_2$ coming from the j^{th} query having two distinct representations relatively to G_1 and G_2. Specifically, we expect to get the same $\tilde{V}_1$ for the two different sets $(\tilde{r}, \tilde{z}_1, \tilde{z}_2, \tilde{z}_3)$ and $(\hat{\tilde{r}}, \hat{\tilde{z}}_1, \hat{\tilde{z}}_2, \hat{\tilde{z}}_3)$, where $\hat{}$ denotes the second execution. We could also choose to work with $\tilde{V}_2$ but it wouldn't make any difference. So, we would have:

$$\begin{aligned} \tilde{V}_1 &= H^{-\tilde{r}} G_1^{\tilde{z}_1} G_2^{\tilde{z}_2} \\ \tilde{V}_1 &= H^{-\hat{\tilde{r}}} G_1^{\hat{\tilde{z}}_1} G_2^{\hat{\tilde{z}}_2} \end{aligned}$$

and

$$\log_{G_1} G_2 = \frac{r'_1 - w_1 c' - \hat{r'_1} + w_1 \hat{c'}}{\hat{r'_2} - w_2 \hat{c'} - r'_2 + w_2 c'}$$

where the reduction knows the secret key of the signer, (w_1, w_2).

Cleaning up Notations. Before proceeding to the actual proof we will clean up some notation issues. Without loss of generality, we assume that all the $(m_i, (H, G_1, G_2), (\tilde{h}, \tilde{g}_{1,i}, g_2), (\tilde{V}_{1,i}, \tilde{V}_{2,i}))$ are queries which have been asked during the attack (otherwise, the probability of success would be negligible due to the randomness of the random oracle outputs). Then, we can assume that the indices, $(Ind_1, \ldots, Ind_{\ell+1})$, of $(m_1, (H, G_1, G_2), (\tilde{h}, \tilde{g}_{1,1}, g_2), (\tilde{V}_{1,1}, \tilde{V}_{2,1}), \ldots, (m_{\ell+1}, (H, G_1, G_2), (\tilde{h}, \tilde{g}_{1,\ell+1}, g_2), (\tilde{V}_{1,\ell+1}, \tilde{V}_{2,\ell+1}))$ in the list of queries is constant. As a result, the probability of success decreases from ε to $\rho \approx \varepsilon / Q^{\ell+1}$ (where Q the number of queries asked to the random oracle).

(w_1, w_2) is the secret key used by the signer. The random tape of the signer, Ω, determines the pairs $(v_{1,i}, v_{2,i}, v_{3,i})$ such that $V_{1,i} = G_1^{v_{1,i}} G_2^{v_{2,i}}$ and $V_{2,i} = g_1^{v_{1,i}} g_2^{v_{3,i}}$ for $i = 1, \ldots, \ell$. The distribution of (w_1, w_2, H) where w_1 and w_2 are random and $H = G_1^{w_1} G_2^{w_2}$, is the same as the distribution

(w_1, w_2, H) where w_1 and H are random and w_2 is the unique element such that $H = G_1^{w_1} G_2^{w_2}$. Accordingly, we replace (w_1, w_2) by (w_1, H) and, similarly, each $(v_{1,i} v_{2,i})$ by $(v_{1,i}, V_{1,i})$ and $(v_{1,i} v_{3,i})$ by $(v_{1,i}, V_{2,i})$.

For the rest of the proof, we will group $(\omega, H, (V_{1,1}, V_{1,2}), \ldots, (V_{1,\ell}, V_{1,\ell}))$ under variable ν, and $(v_{1,i}, \ldots, (v_{1,\ell})$ under the variable τ. $\mathcal{S}$ will denote the set of all successful data, i.e. quadruples (ν, w_1, τ, f) such that the attack succeeds. Then,

$$\Pr_{\nu, x_1, \tau, f}[(\nu, w_1, \tau, f) \in \mathcal{S}] \geq \rho.$$

Before continuing with the proof we state a well - known probabilistic lemma:

Lemma 4.2.2. ***(The probabilistic lemma)****. Let A be a subset of $X \times Y$ such that $\Pr[A(x, y)] \geq \varepsilon$, then there exists $\Omega \subset X$ such that*

1. $\Pr[x \in \Omega] \geq \varepsilon/2$
2. *whenever $a \in \Omega$, $\Pr[A(a, y)] \geq \varepsilon/2$.*

The probabilistic lemma is useful to split a set X in two subsets, a non-negligible subset Ω consisting of "good" x's which provide a non-negligible probability of success over y, and its complement, consisting of "bad" x's.

Lemma 4.2.3. ***(The forking lemma.)*** *Randomly choose an index j, the keys and the random tapes. Run the attack twice with the same random tapes and two different random oracles, f and f', providing identical answers to the $j-1$ first queries. With non-negligible probability, the different outputs reveal two different representations of some $\tilde{V}_{1,i}$, relatively to G_1 and G_2.*

Proof. By proving this lemma we basically prove Theorem 4.2.1. What we want to show is, that

after a replay, we can obtain a common $\tilde{V}_{1,i}$ such that:

$$\begin{aligned}\tilde{V}_{1,i} &= G_1^{\tilde{z}_{1,i}} G_2^{\tilde{z}_{2,i}} H^{-\tilde{r}_i} = G_1^{\tilde{z}_{1,i} - w_1\tilde{r}_i} G_2^{\tilde{z}_{2,i} - w_2\tilde{r}_i} \\ &= G_1^{\hat{\tilde{z}}_{1,i}} G_2^{\hat{\tilde{z}}_{2,i}} H^{-\hat{\tilde{r}}_i} = G_1^{\hat{\tilde{z}}_{1,i} - w_1\hat{\tilde{r}}_i} G_2^{\hat{\tilde{z}}_{2,i} - w_2\hat{\tilde{r}}_i}\end{aligned}$$

where, $\tilde{z}_{1,i} - w_2\tilde{r}_i \neq \hat{\tilde{z}}_{1,i} - w_1\hat{\tilde{r}}_i$. We can remark that, for each i, $\tilde{V}_{1,i}$ only depends on (ν, w_1, τ) and the first $Ind_i - 1$ answers of f. What is left to study is whether or not the random variable $\chi_i = r'_{1i} - x_1 c'_i$ is sensitive to queries asked at steps Ind_i, $Ind_i + 1$, etc. We expect the answer to be yes. We can consider the most likely value taken by χ_i when (ν, w_1, τ) and the $Ind_i - 1$ first answers of f are fixed. Then, we are led to consider a function $e_i(\nu, w_1, \tau, f_i)$, where f_i ranges over the set of answers to the first $Ind_i - 1$ possible queries. Set

$$\lambda_i(\nu, w_1, \tau, f_i, e) = \Pr_f \left[(\chi_i(\nu, w_1, \tau, f) = e) \ \& \ ((\nu, w_1, \tau, f) \in \mathcal{S}) \,|\, f \text{ extends } f_i \right].$$

We define $e_i(\nu, w_1, \tau, f_i)$ as any value e such that $\lambda_i(\nu, w_1, \tau, f_i, r)$ is maximal. We then define the "good" subset $\mathcal{G}$ of $\mathcal{S}$ whose elements satisfy, for all i, $\chi_i(\nu, w_1, \tau, f) = e_i(\nu, w_1, \tau, f_i)$, where f_i denotes the restriction of f to queries of index strictly less than Ind_i, and the "bad" $\mathcal{B}$ its compliments in $\mathcal{S}$.

Definition 4.2.4. *We denote by Φ the transformation which maps any quadruple (ν, w_1, τ, f) to $(\nu, w_1 + 1, \tau - c, f)$, where $\tau - r = (v_{1,1} - r_1, \ldots, v_{1,\ell} - r_\ell)$.*

Lemma 4.2.5. *Both executions corresponding to (ν, w_1, τ, f) and $\Phi(\nu, w_1, \tau, f)$ are totally identical with respect to the view of the attacker. Especially, outputs are the same.*

Proof. Let (ν, w_1, τ, f) be an input for the collusion. Replay with $\hat{w}_1 = w_1 + 1$ and $\hat{\tau} = \tau - r$, the same ν and the same oracle f. The answers of the oracle are unchanged and the interactions with

the signer become

$$\hat{z_{1,i}}(\hat{w_1}, \hat{v}_{1,i}, r_i) = \hat{v}_{1,i} + \hat{w_1} r_i = (v_{1,i} - r_i) + r_i(w_1 + 1) = v_{1,i} + r_i w_1 = z_{1,i}(w_1, v_{1,i}, r_i).$$

Thus, everything remains the same. □

Corollary 4.2.6. *Φ is a one-to-one mapping from $\mathcal{S}$ onto $\mathcal{S}$.*

Lemma 4.2.7. *For fixed (ν, w_1, τ), the probability*

$$\Pr_f[((\nu, w_1, \tau, f) \in \mathcal{G}) \ \& \ (\Phi(\nu, w_1, \tau, f) \in \mathcal{G}) \leq 1/q.$$

Which means that Φ sends the set $\mathcal{G}$ into $\mathcal{B}$, except for a negligible part.

Proof. We will prove the above lemma by contradiction. Assume that $\Pr_f[(\nu, w_1, \tau, f) \in \bigcup_{r_1,\ldots,r_\ell} Y(r_1, \ldots, r_\ell)] > 1/q$, where the set $Y(r_1, \ldots, r_\ell)$ is defined by the conditions $(\nu, w_1, \tau, f) \in \mathcal{G}$, $\Phi(\nu, w_1, \tau, f) \in \mathcal{G}$ and $(r_1, \ldots, r_\ell)$ are the successive questions asked to the authority. Then, there exists a ℓ-tuple $(r_1, \ldots, r_\ell)$ such that $\Pr_f[Y(r_1, \ldots, r_\ell)] > 1/(q^{\ell+1})$. Thus, there exist two oracles f and f' in $Y(r_1, \ldots, r_\ell)$ which provide distinct answers for some queries $\mathcal{Q}_{Ind_j} = (m_j, (H, G_1, G_2), (\tilde{h}_j, \tilde{g}_{1,j}, g_2), (\tilde{V}_{1,j}, \tilde{V}_{2,j}))$ to the oracle, for some $j \in 1, \ldots, \ell + 1$, and are such that answers to queries not of the form of $\mathcal{Q}_{Ind_j}$ are similar. We will denote by i the smallest such index j. Then $f_i = f'_i$ and $\tilde{r}_i \neq \hat{\tilde{r}_i}$. Also, we have $(\nu, w_1, \tau, f) \in \mathcal{G}$, $\Phi(\nu, w_1, \tau, f) \in \mathcal{G}$ and similarly $(\nu, w_1, \tau, f') \in \mathcal{G}$,

$\Phi(\nu, w_1, \tau, f') \in \mathcal{G}$. Because of the property of Φ, and by definition of $\mathcal{G}$,

$$\begin{aligned}
e_i(\nu, w_1, \tau, f_i) &= z_{1,i}(\nu, w_1, \tau, f) - w_1\tilde{r}_i \\
&= z_{1,i}(\Phi(\nu, w_1, \tau, f)) - w_1\tilde{r}_i \\
&= e_i(\nu, w_1 + 1, \tau - r, f_i) + ((w_1 + 1) - w_1)\tilde{r}_i \\
e_i(\nu, w_1, \tau, f'_i) &= z_{1,i}(\nu, w_1, \tau, f') - xw1\,\hat{\tilde{r}}_i \\
&= z_{1,i}(\Phi(\nu, w_1, \tau, f')) - w_1\,\hat{\tilde{r}}_i \\
&= e_i(\nu, w_1 + 1, \tau - \hat{r}, f'_i) + ((w_1 + 1) - w_1)\,\hat{\tilde{r}}_i
\end{aligned}$$

The equality $f_i = f'_i$ implies $e_i(\nu, w_1, \tau, f_i) = e_i(\nu, w_1, \tau, f'_i)$. Since we have assumed $(r_1, \ldots, r_\ell) = (\hat{r_1}, \ldots, \hat{r_\ell})$, then $e_i(\nu, w_1 + 1, \tau - r, f_i) = e_i(\nu, w_1 + 1, \tau - \hat{r}, f'_i)$. Thus, $\tilde{r}_i = \hat{\tilde{r}}_i$ which contradicts the hypothesis. □

Lemma 4.2.7 says that for any (ν, w_1, τ),

$$\Pr_f[((\nu, w_1, \tau, f) \in \mathcal{G}) \;\&\; (\Phi(\nu, w_1, \tau, f) \in \mathcal{G})] \leq 1/q.$$

By making the sum over all the triplets (ν, w_1, τ), and using the bijectivity of Φ (Corollary 4.2.6), we obtain

$$\begin{aligned}
\Pr[\mathcal{G}] &= \Pr_{\nu,w_1,\tau,f}[((\nu, w_1, \tau, f) \in \mathcal{G}) \;\&\; (\Phi(\nu, w_1, \tau, f) \in \mathcal{G})] \\
&\quad + \Pr_{\nu,w_1,\tau,f}[((\nu, w_1, \tau, f) \in \mathcal{G}) \;\&\; (\Phi(\nu, w_1, \tau, f) \in \mathcal{B})] \\
&\leq \frac{1}{q} + \Pr_{\nu,w_1,\tau,f}[\Phi(\nu, w_1, \tau, f) \in \mathcal{B}] \leq \frac{1}{q} + \Pr[\mathcal{B}]
\end{aligned}$$

Then, $\Pr[\mathcal{B}] \geq (\Pr[\mathcal{S}] - 1/q)/2$. Since $1/q$ is negligible w.r.t $\Pr[\mathcal{S}]$, for enough large keys, we have, $\Pr[\mathcal{B}] \geq \Pr[\mathcal{S}]/3 \geq \rho/3$. □

Conclusion. We will use this probability to show the success of forking.

$$\begin{aligned}\frac{\rho}{3} \leq \Pr[\mathcal{B}] &= \Pr_{\nu,w_1,\tau,f}[\mathcal{S} \ \& \ ((\exists i)\chi_i(\nu,w_1,\tau,f) \neq e_i(\nu,w_1,\tau,f_i))] \\ &\leq \sum_{i=1}^{\ell+1} \Pr_{\nu,w_1,\tau,f}[\mathcal{S} \ \& \ (\chi_i(\nu,w_1,\tau,f) \neq e_i(\nu,w_1,\tau,f_i))].\end{aligned}$$

There exists k such that $\Pr[\mathcal{S} \ \& \ (\chi_k(\nu,w_1,\tau,f) \neq e_i(\nu,w_1,\tau,f_k))] \geq \rho/3\ell+1$. Let us randomly choose the forking index i. With probability greater than $1/(\ell+1)$, we have guessed $i=k$. The probabilistic lemma ensures that there exists a set X such that

1. $\Pr_{\nu,w_1,\tau,f}[(\nu,w_1,\tau,f_i) \in X] \geq \rho/6(\ell+1)$
2. for all $(\nu,w_1,\tau,f_i) \in X$, $\Pr_f[(\nu,w_1,\tau,f) \in \mathcal{S} \ \& \ (\chi_i \neq e_i)|$ extends $f_i] \geq \rho/6(\ell+1)$.

Let us choose a random quadruple (ν,w_1,τ,f). With probability greater than $(\rho/6(\ell+1))^2$, $(\nu,w_1,\tau,f) \in \mathcal{S}$, $(\nu,w_1,\tau,f_i) \in X$ and $\chi_i(\nu,w_1,\tau,f) \neq e_i(\nu,w_1,\tau,f_i)$. We will denote by α the value $\chi_i(\nu,w_1,\tau,f)$ and by β the value $e_i(\nu,w_1,\tau,f_i)$. Then, two cases appear relatively to $\lambda_i(\nu,w_1,\tau,f_i,\alpha)$:

- if $\lambda_i(\nu,w_1,\tau,f_i,\alpha) \geq \rho/12(\ell+1)$, then, by definition of e_i, we know that $\lambda_i(\nu,w_1,\tau,f_i,\beta) \leq \rho/12(\ell+1)$.
- otherwise,

$$\begin{aligned}&\lambda_i(\nu,w_1,\tau,f_i,\alpha) + \Pr_{f'}[\mathcal{S} \ \& \ (\chi_i(\nu,w_1,\tau,f_i,\alpha') \neq \alpha)|f' \text{ extends } f_i] \\ &= \Pr_{f'}[S|f' \text{ extends } f_i] \\ &\geq \Pr_{f'}[\mathcal{S} \ \& \ (\chi_i(\nu,w_1,\tau,f') \neq \beta)|f' \text{ extends } f_i] \geq \rho/6(\ell+1).\end{aligned}$$

Both cases lead to $\Pr_{f'}[\mathcal{S} \ \& \ (\chi_i(\nu,w_1,\tau,f') \neq \alpha)|f'$ extends $f_i] \geq \rho/12(\ell+1)$. Thus, if we replay with the same keys and random tapes but another random oracle f' such that $f'_i = f_i$, we obtain, with probability at least $\rho/12(\ell+1)$, a new success with $\chi_i(\nu,w_1,\tau,f') \neq \alpha$. Then, both executions provide two different representations of a_i with respect to G_1 and G_2.

Global Complexity of the Reduction By using a replay oracle technique with a random forking index, the probability of success is greater than

$$\frac{1}{\ell+1} \times \left(\frac{\rho}{6(\ell+1)}\right)^2 \times \frac{\rho}{12(\ell+1)} \times \left(\frac{1}{6(\ell+1)} \times \frac{\varepsilon}{\mathcal{Q}^{\ell+1}}\right)^3$$

where ε is the probability of success of an $\ell, \ell+1$-forgery and Q the number of queries asked to the random oracle. □

4.3 Conclusions

In this Chapter we proved that Brands' blind signature scheme can be described as a Generalized Schnorr Blind Signature and thus its security cannot be proved via a perfect naive or an L-naive RO replay reduction. This implies that proving Brands' blind signature unforgeable would require radically new techniques. We then propose a modification of Brand's blind signature in which the public key is constructed using two pieces of a secret. This allows the Pointcheval and Stern reduction go through and we prove the new scheme unforgeable. However, whether the new scheme can be used for a secure single-use anonymous credential scheme (or an e-cash scheme) is still an open problem since there is no guarantee that one can detect a malicious user who uses the same credential (or coin) more than once[1].

The fact that Brands' blind signature is in the heart of Microsoft's anonymous credentials scheme, U-Prove, makes this work relevant to cryptographic practice as well. In the next Chapter we will describe a new anonymous credential scheme that will have comparable efficiency to U-Prove but will be provable secure.

[1] Note that this is an open problem for U-Prove as well.

5 Anonymous Credentials Light

As already discussed in the previous Chapter, U-Prove, the most efficient single-use anonymous credential scheme, is unlikely to be proven secure under currently known techniques. In this Chapter, we propose a new anonymous credential scheme called *Anonymous Credentials Light* [12, 14] that has comparable efficiency to the one due to Brands and it is provably secure (in the random oracle model). In particular, it is unlinkable under the decisional Diffie-Hellman assumption, and unforgeable under the Discrete-Logarithm assumption for sequential composition (the extension to concurrent self-composition is an open problem). For the construction, we defined a new cryptographic building block, called *blind signatures with attributes*, and discussed how it can be used in combination with a commitment scheme to directly get an anonymous credential system.

5.1 More on Cryptographic Commitments

In Section 2.2.5 we defined cryptographic commitments and we presented the Pedersen commitment scheme. Here we give a generalized version that allows a commitment to a set of messages $(L_1, \ldots, L_n)$. The scheme is defined as follows:

1. *Setup*: On input the security parameter 1^k and the maximum number of messages n, pick a group G of prime order $q = \Theta(2^k)$ with generators $h, h_1, \ldots, h_n$.
2. $Commit(L_1, \ldots, L_n; R) = h^R \prod_{i=1}^{n} h_i^{L_i}$; and $L_i \in \mathbb{Z}_q$.

The Pedersen commitment scheme (constructed from a corresponding Σ-protocol) is information theoretically hiding and computationally binding. It is also important that there are efficient zero-knowledge proof protocols for proving that a commitment C is to a particular set of values; or to a set of values that satisfy a rich class of relations [36, 47, 70].

5.1.1 Combined Commitment Schemes

Let $Commit_1$ be a commitment scheme that takes as input its parameters $params_1$, n messages $(L_1, \ldots, L_n)$ to which we will refer to as *attributes* from now on and randomness R_1 and outputs a commitment C_1. Let $Commit_2$ be a commitment scheme that takes as input its parameters $params_2$, an attribute L_0 and randomness R_2 and outputs a commitment C_2. Suppose that another commitment scheme, $Commit$, is a *combination* of these two commitments; i.e., on input C_1 and C_2 it produces a commitment C to the combined attributes $(L_0, L_1, \ldots, L_n)$ with combined randomness $R = R_1 + R_2$.

For example, this can be instantiated by a generalized Pedersen commitment scheme: the parameters for the combined scheme are generators $(h, h_0, \ldots, h_n)$, $Commit_1(L_1, \ldots, L_n; R_1) = (\prod_{i=1}^{n} h_i^{L_i}) h^{R_1}$, $Commit_2(L_0; R_2) = h_0^{L_0} h^{R_2}$, and a combined commitment can be obtained either by multiplying together the two component commitments, or by computing it from scratch as $Commit(L_0, L_1, \ldots, L_n; R) = (\prod_{i=0}^{n} h_i^{L_i}) h^R$.

5.1.2 Blinded Pedersen Commitment Scheme

Here, we note that the Pedersen commitment scheme can be further extended. Let $(h, h_1, \ldots, h_n)$ be the parameters of the Pedersen commitment. Consider an additional parameter $z \in G$, where $z \neq 1$. Let $C = Commit(L_1, \ldots, L_n; R)$. Then the values (z^γ, C^γ) can also be viewed as a

commitment to the same $(L_1, \ldots, L_n)$ with randomness (R, γ). Let us define a new commitment scheme, which we will call the *blinded* Pedersen commitment scheme: $Commit^B(L_1, \ldots, L_n; R, \gamma) = (z^\gamma, Commit(L_1, \ldots, L_n; R)^\gamma)$, where *Commit* is the Pedersen commitment. It is easy to see that this commitment is unconditionally hiding, same as Pedersen. It is also easy to see that it is binding: given (z^γ, C^γ), γ is uniquely defined, and therefore so is C, which is binding. Finally, using well-known Σ-protocols, it is easy to see that the same set of relations that can be proven about values inside a Pedersen commitment can be proven about values inside a blinded Pedersen commitment.

5.2 Defining Blind Signatures with Attributes

In this section, for the first time, we define what a blind signature scheme with attributes is and its security properties. For definitions of standard (without attributes) blind signatures please refer to Section 2.3.1. In a blind signature scheme with attributes, the signer and the user both get as input a cryptographic commitment C to the user's attributes; this way, the user can prove that the commitment contains the correct attributes via a separate zero-knowledge proof. As output, the user obtains another, unlinkable, commitment $\tilde{C}$ to the same attributes, and a signature on this commitment and a message of the user's choice. *Blindness* ensures that, even upon seeing two signatures obtained this way on commitments of his own choice, the signer cannot link a signature to its issuing. *Unforgeability* ensures that a user cannot produce more signatures than he was issued, and also that the multiset of openings to the input commitments is the same as the multiset of openings to the output commitments. Formally:

Definition 5.2.1 (Blind Signature With Attributes)**.** *Let $Commit(x; r)$ be a non-interactive commitment scheme. A blind signature scheme with n attributes, for this commitment scheme, consists of three algorithms: $KeyGen, BlindSign, Verify$ where $BlindSign$ is a protocol between the Signer(S) and the User(U).*

- *KeyGen(1^k): is a probabilistic polynomial time key generation algorithm. It takes a security parameter k as input and outputs a pair (pk, sk) of public and secret keys for the system and the maximum number of attributes n that can be embedded during BlindSign.*

- *BlindSign: is an interactive, probabilistic polynomial time protocol between S and U. The public parameters are the Signer's public key pk, the parameters of the commitment scheme and $C = Commit(L_1, \ldots, L_n; R)$ where $(L_1, \ldots, L_n)$ is the set of attributes and R is some randomness. The Signer's private input is sk and User's private input is $(L_1, \ldots, L_n, R)$ and the message, m, that he wishes to have signed. The User's output in the protocol is a pair $(\tilde{R}, \sigma)$, where $\sigma = \sigma(m, \tilde{C})$ is the Signer's signature on $(m, \tilde{C})$, and $\tilde{C} = Commit(L_1, \ldots, L_n; \tilde{R})$; the Signer's output is "completed".*

- *Verify($PK, m, \tilde{C}, \sigma$): is the signature verification algorithm; i.e. a deterministic polynomial time algorithm that gets as input the public key, the message, a commitment to the attributes and the blind signature on the message and attributes and checks the validity of the signature σ. If it is valid the algorithm outputs "1", otherwise outputs "0".*

A blind signature scheme with attributes is secure if it is both blind and unforgeable. Blindness is defined in a similar way as in blind signature schemes without attributes: the Signer is unable to view the messages and the attributes he signs (protection for the User). A malicious Signer, $\mathcal{A}$, cannot link a $(m, \tilde{C}, \sigma)$ tuple to any particular execution of the protocol, even if $\mathcal{A}$ chooses $m, L_1, \ldots, L_n$.

Definition 5.2.2 (Blindness for Blind Signatures with Attributes)**.** *Let $\mathcal{A}$ be a malicious Signer and $b \in \{0, 1\}$ be a randomly chosen bit which is kept secret from $\mathcal{A}$. $\mathcal{A}$ will try to guess the value b by performing the following steps:*

1. $(pk, sk) \leftarrow KeyGen(1^k)$

2. $\{m_0, m_1, \overrightarrow{L_0}, \overrightarrow{L_1}, R_0, R_1\} \leftarrow \mathcal{A}(1^k, pk, sk)$ *(i.e. $\mathcal{A}$ produces two messages $\{m_0, m_1\}$, polynomial in 1^k, and two attribute vectors $\overrightarrow{L_0}, \overrightarrow{L_1}$ with the corresponding randomness).*

3. $\mathcal{A}(1^k, pk,\ sk, m_0, m_1, \overrightarrow{L_0}, \overrightarrow{L_1}, R_0, R_1)$ *engages in two parallel (and arbitrarily interleaved as desired by* $\mathcal{A}$*) interactive protocols, the first with* $U(pk, \{m_b, \overrightarrow{L_0}, R_0\})$ *and the second with* $U(pk,$ $\{m_{1-b}, \overrightarrow{L_1}, R_1\})$.

4. *If neither of the User instances failed, then* $\mathcal{A}$ *gets two signatures and the corresponding blinded commitments:* $\sigma(m_0, \tilde{C}_b), \tilde{C}_b$ *and* $\sigma(m_1, \tilde{C}_{1-b}), \tilde{C}_{1-b}$.

5. $\mathcal{A}$ *outputs a bit* b'.

Then the probability, taken over the choice of b*, over coin-flips of the key-generation algorithm, the coin-flips of* $\mathcal{A}$*, and (private) coin-flips of both users (from step 3), that* $b' = b$ *is at most* $\frac{1}{2} + \nu(k)$*, where* $\nu(k)$ *is a negligible function.*

We give the definition of one-more unforgeability for the sequential composition case.

Definition 5.2.3 (Sequential One-More Unforgeability for Blind Signatures with Attributes)**.** *(KeyGen, BlindSign, Verify) is a one-more unforgeable blind signature scheme with respect to Commit if* $\forall$ *ppt* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins in the following game is negligible:*

1. $(pk, sk) \leftarrow KeyGen(1^k)$

2. $\mathcal{A}(pk, C_i, m_i)$ *engages in polynomially many (in* k*) adaptive, sequential interactive protocols with polynomially many copies of* $S(pk, sk)$*, where* $\mathcal{A}$ *decides in an adaptive fashion when to stop. Let* ℓ *be the number of executions, where the Signer output "completed" in the end of the protocol.*

3. $\mathcal{A}$ *outputs a collection* $\{(\tilde{C}_1, m_1, \sigma_1), \ldots, (\tilde{C}_j, m_j, \sigma_j)\}$ *where* $(\tilde{C}_i, m_i, \sigma_i)$ *for* $1 \leq i \leq j$ *are all accepted by* $Verify(pk, \tilde{C}_i, m_i,\ \sigma_i)$*, and all* $(\tilde{C}_i, m_i)$*'s are distinct.*

We say that $\mathcal{A}$ *wins the game if either:*

1. $j > \ell$ *(i.e.* $\mathcal{A}$ *outputs more* $(\tilde{C}, m, \sigma)$ *tuples than he received).*

2. $\mathcal{A}$ *opens the sets of commitments* $\{C_i\}$ *and* $\{\tilde{C}_i\}$ *and the resulting multisets do not match.*

5.3 From Blind Signatures with Attributes to Single-Use Anonymous Credentials

It is easy to see that blind signatures with attributes is the right building block for single-use anonymous credentials: a user with a particular set of attributes can form a commitment C to these attributes, prove in zero-knowledge that he has committed to the correct attributes, and then obtain a credential by running a blind signature with attributes on input this commitment. He can then prove that he has a credential with the desired attributes by revealing his signature and the output commitment $\tilde{C}$, and proving in zero knowledge that $\tilde{C}$ corresponds to the needed attributes.

For example, suppose that we allow users to obtain and anonymously show age credentials. Then Alice will form a commitment C to her secret key sk and her age age, prove to the signer, who serves as an identity validator, that she has committed to sk that corresponds to her pk and to the correct age, and run a blind signature with attributes protocol to obtain a fresh commitment $\tilde{C}$ on the same sk and age, and a signature σ. Then, when she needs to prove her age, she can reveal $(\tilde{C}, \sigma)$ and prove that the age inside commitment $\tilde{C}$ allows her entry to some age-restricted resource, for example a video store that requires viewers to be over 13. If she wants to do it again, she needs to run the blind signature with attributes protocol with the signer again. She can further anonymously obtain credentials that are connected to her identity: let's say that the video store wants to give her a credential for a free movie; she can obtain it by running a blind signature with attributes with the video store. She will form her input commitment C' by committing to the same sk and age as in $\tilde{C}$ and proving to the video store that she did so (by proving equality of committed values); once she runs the protocol with the video store, she receives an output $(\tilde{C}', \sigma)$, which is a credential from the video store (and not from the original signer) on her (sk, age) — even though the video store never saw the public key at all. More interestingly, we can require the free movie to be a single-use credential, by additional clever use of attributes (see Section 5.3), so that Alice can be traced if she tries to get two free movies using the same single-use credential more than once. Thus we see that

blind signatures with attributes are the right building block for single-use anonymous credentials.

Recall our definition of generalized commitment schemes $(Commit_1, Commit_2, Commit)$ from Section 5.1.1. Consider the following construction of a single-use credential with n attributes from a blind signature scheme with $n+1$ attributes. The common inputs to the User and Signer are the Signer's public key for verifying signatures, and a commitment C_1 to the User's attributes $(L_1, \ldots, L_n)$. We assume that the attribute L_1 contains the user's identity, so that learning that attribute will allow one to trace the user. The Signer's private input is its signing key, while the User's private input is the message m that the user wants signed (m will serve as the serial number for this credential, so the user needs to have chosen it at random during the signing protocol) and the opening to C_1, $(L_1, \ldots, L_n; R_1)$. The commitment C_1 corresponds to this user's identity and never changes; each time this user obtains a signature, they input the same C_1.

First, the User forms a commitment $C_2 = Commit_2(L_0; R_2)$ for random (L_0, R_2); this commitment is specific to this credential, and the value L_0 will be used in the double-spending equation, below. Next, the user submits C_2 to the signer and provides a zero-knowledge proof of knowledge of its opening.

Next, the User and the Signer carry out the blind signature protocol relative to the combined commitment scheme $Commit$, whereby the User's output is a fresh commitment $\tilde{C}$ to the values $(L_0, \ldots, L_n)$ and randomness $\tilde{R}$, and the signature σ on $(m, \tilde{C})$.

To use this credential to a verifier V, the User reveals $(m, \tilde{C}, \sigma)$, obtains from V a challenge c, and reveals the double-spending equation $cL_1 + L_0$ (the multiplication and addition is in whatever ring the committed values belong to) and whatever other attributes are needed for the transaction; the User then provides a zero-knowledge proof that these values were revealed correctly, relative to commitment $\tilde{C}$.

To trace a user who spent a single-use credential twice, it is sufficient to examine two double-spending equations for the same credential. If double spending occurred, then the attribute L_1, which encodes the User's identity, will be revealed.

The following theorem is stated informally because we omitted a formal definition of single-use anonymous credentials.

Theorem 5.3.1. *(Informal) Let KeyGen, BlindSign, Verify be a blind signature scheme with $n+1$ attributes, associated with the commitment scheme Commit which is a combination of $Commit_1$ and $Commit_2$ as explained above. Then the above construction is a single-use credential scheme with n attributes.*

5.4 Our Construction: ACL

Let us now describe our proposed construction of a blind signature scheme with attributes, called "Anonymous Credentials Light" (ACL). ACL is based on Abe's blind signature scheme [3] which has been modified appropriately so that it will also allow the users to encode attributes in the signatures they get in a provably secure way. Abe's scheme is a witness indistinguishable variant of the Schnorr signature scheme where the Signer owns a *real public key* $y = g^x$ and a *tag public key* $z = g^w$. The idea is that the signature can be issued only with the real secret key x but no one can distinguish which of the two (real or tag) secret keys was used (remember the OR-proof technique presented above).

Although the modification seems straightforward, proving the security of the resulting scheme turns out to be challenging as we will see later in the next section. Also note that Abe's construction, as it was first presented, had a faulty proof of security as it was later found by Abe and Ohkubo [112]. The original proof was in the RO model for concurrent composition but it turned out to only be valid for an adversary with overwhelming success probability so they ended up presenting a new proof in the generic model.

Setup. Let G be a group of order q and g is a generator of this group. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. We assume the existence of a trusted party, TP, which chooses G, g and outputs

$params = (q, G, g, z, h, h_0, \ldots, h_n)$ where $z, h, h_0, \ldots, h_n \in G$ and n is the maximum number of attributes a User can possibly have. We will use $(h_0, \ldots, h_n)$ as parameters of the generalized Pedersen commitment scheme $Commit$; we will use $(z, h_0, \ldots, h_n)$ as parameters for the blinded Pedersen commitment scheme $Commit^B$.

The Signer picks his secret key $x \in_R \mathbb{Z}_q$ and computes his *real public key* to be $y = g^x \bmod q$. The TP given (p, q, g, h, y) outputs z which is the *tag public key* of the Signer. The public key of the Signer is (p, q, g, h, y, z), and the private key is x.

Signature/ Credential Issuing. The signature issuing protocol will be described in three phases: registration, preparation and validation. The registration phase actually only needs to happen once for each user/set of attributes and also, preparation and validation phases can happen simultaneously (we choose to present them separately in order to describe the construction in a more modular way).

The signature issuing protocol is basically a $\mathcal{P}_{OR}$-protocol for proving knowledge of one of the following:

- y-side: proof of knowledge x of $y = g^x$
- z-side: proof of knowledge (w_1, w_2) of $z_1 = g^{w_1}$, $z_2 = h^{w_2}$ (where z_1, z_2 are the so called "one-time" tag keys that the signer creates).

Notice that the z-side witness in not known to the Signer, so the z-side proof will be done by simulation following the OR-proof technique paradigm described in Section 2.2.3.

Registration. The User's input includes the system parameters *params*, the signer's real public key y, the message m to be signed and $(L_1, \ldots, L_n; R)$ where $L_1, \ldots, L_n$ is a set of attributes and R is some randomness. The Signer also gets as input the system parameters *params*, a commitment $C = Commit(L_1, \ldots, L_n; R)$ and his secret key x. During registration the User and the Signer carry out a standard interactive zero-knowledge proof of knowledge protocol where the User creates a proof π_1 to convince the Signer that he knows an opening of the commitment C.

Preparation. The Signer prepares z_1 and z_2 (the z-side proof). He first picks $rnd \in \mathbb{Z}_q$ and creates the "one-time" tag keys: $z_1 = Cg^{rnd}$ and $z_2 = z/z_1$. The Signer sends rnd to the User in order to convince him that $\log_g z_1$ is not known to him. The User computes himself $z_1 = Cg^{rnd}$ and then picks $\gamma \in \mathbb{Z}_q^*$ and blinds z, z_1, z_2 into $\zeta = z^\gamma$, $\zeta_1 = z_1^\gamma, \zeta_2 = \zeta/\zeta_1$ so that $\log_z z_1 = \log_\zeta \zeta_1$ holds.

Finally, the User picks $\tau \in \mathbb{Z}_q$ and computes $\eta = z^\tau$ (this will serve as an element of an additional Schnorr signature that proves knowledge of γ such that $\zeta = z^\gamma$).

Validation. In this phase two Σ protocols are going to take place and combined according to the OR-proof. By (a, c, r) we are going to denote the transcript of Σ_y and by (a', c', r') we will denote the transcript of Σ_z. The Signer will compute Σ_z himself and Σ_y in interaction with the user.

1. The Signer begins by creating a for Σ_y by picking a random $u \in \mathbb{Z}_q$ and computing $a = g^u$. Then, for Σ_z, following the OR-proof he picks random $c' \in \mathbb{Z}_q$ and $r' = \{r'_1, r'_2 \in \mathbb{Z}_q\}$. and sets $a'_1 = g^{r'_1} z_1^{c'}$ and $a'_2 = h^{r'_2} z_2^{c'}$. Finally he sends to the User $a, a' = \{a'_1, a'_2\}$.
2. The User checks whether $a, a'_1, a'_2 \in G$ and then picks blinding factors $t_1, t_2, t_3, t_4, t_5 \in_R \mathbb{Z}_q$ and blinds a into $\alpha = ag^{t_1} y^{t_2}$ and a'_1, a'_2 into $\alpha'_1 = a_1^{'\gamma} g^{t_3} \zeta_1^{t_4}$ and $\alpha'_2 = a_2^{'\gamma} h^{t_5} \zeta_2^{t_4}$. We denote $\alpha' = \{\alpha'_1, \alpha'_2\}$. Then, computes $\varepsilon = \mathcal{H}(\zeta, \zeta_1, \alpha, \alpha', \eta, m)$ where m is the message to be signed. The User sends to the Signer $e = (\varepsilon - t_2 - t_4) \bmod q$.
3. The Signer, according to the OR-proof technique, computes $c = e - c' \bmod q$ and $r = u - cx \bmod q$. Then, sends (c, r, c', r') to the User.
4. Finally, the User "unblinds" the received values and gets: $\rho = r + t_1 \bmod q$, $\omega = c + t_2 \bmod q$, $\rho'_1 = \gamma r'_1 + t_3 \bmod q$, $\rho'_2 = \gamma r'_2 + t_5 \bmod q$, $\omega' = c' + t_4 \bmod q$, $\mu = \tau - \omega' \gamma \bmod q$.

A signature is a 8-tuple $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho' = \{\rho'_1, \rho'_2\}, \omega', \mu))$ where ζ_1 encodes the attributes of the User ((ζ, ζ_1) corresponds to $\tilde{C}$, which is a blinded Pedersen commitment to $(L_1, \ldots, L_n)$ with randomness (R, γ)).

The complete signature issuing protocol is also described in Figure 5.1.

Signer$(params, x, C)$		User$(params, y, m, (L_0, \ldots, L_n; R))$
Registration		
		$C = h_0^R h_1^{L_1} h_2^{L_2} \ldots h_n^{L_n}$
		π_1
	$\xleftrightarrow{\pi_1}$	
Preparation		
$rnd \in_R \mathbb{Z}_q$		
$z_1 = Cg^{rnd}$		
$z_2 = z/z_1$		
	$\xrightarrow{rnd}$	
		$z_1 = Cg^{rnd}$
		$\gamma \in_R \mathbb{Z}_q^*$
		$\zeta = z^\gamma$
		$\zeta_1 = z_1^\gamma$
		$\zeta_2 = \zeta/\zeta_1$
		$\tau \in_R \mathbb{Z}_q$
		$\eta = z^\tau$
Validation		
$u, r_1', r_2', c' \in_R \mathbb{Z}_q$		
y-side $a = g^u$		
$a_1' = g^{r_1'} z_1^{c'}$		
z-side $a_2' = h^{r_2'} z_2^{c'}$		
	$\xrightarrow{a, a' = \{a_1', a_2'\}}$	
		check if $a, a_1', a_2' \in G$
		$t_1, t_2, t_3, t_4, t_5 \in_R \mathbb{Z}_q$
		$\alpha = a g^{t_1} y^{t_2}$ y-side
		$\alpha_1' = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4}$
		$\alpha_2' = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4}$ z-side
		$\varepsilon = \mathcal{H}_2(\zeta, \zeta_1, \alpha, \alpha_1', \alpha_2', \eta, m)$
	$\xleftarrow{e}$	$e = (\varepsilon - t_2 - t_4) \bmod q$
$c = e - c' \bmod q$		
$r = u - cx \bmod q$		
	$\xrightarrow{c, r, c', r' = \{r_1', r_2'\}}$	$\rho = r + t_1 \bmod q$
		$\omega = c + t_2 \bmod q$
		$\rho_1' = \gamma r_1' + t_3 \bmod q$
		$\rho_2' = \gamma r_2' + t_5 \bmod q$
		$\omega' = c' + t_4 \bmod q$
		$\mu = \tau - \omega' \gamma \bmod q$

Figure 5.1: Proposed ACL Construction

Verification. A signature tuple (m, ζ_1, σ) (where ζ_1 corresponds to $\tilde{C}$) verifies if $\zeta \neq 1$ and

$$\omega + \omega' = \mathcal{H}(\zeta, \zeta_1, g^{\rho} y^{\omega}, g^{\rho'_1} \zeta_1^{\omega'}, h^{\rho'_2} \zeta_2^{\omega'}, z^{\mu} \zeta^{\omega'}, m) \bmod q.$$

Discussion. As briefly mentioned in the description of our construction there are certain "tricks" we can do to improve its efficiency. First of all notice that the *Registration* phase only needs to happen once for each User and a corresponding set of attributes. The Signer stores the attribute commitment C together with some identification information asked from the User (e.g. passport, ID).Then, it is sufficient if every time in the beginning of the *Signature Issuing* protocol the User identifies himself to the Signer in order to prove ownership of his account. Moreover, notice that *Preparation* and *Validation* phases can be combined and executed simultaneously. As a result, the whole signing protocol consists of three rounds.

Let's now discuss the differences between our ACL construction and the blind signature protocol described by Abe [3]. The basic advantage of ACL is that allows for the encoding of users' attributes in the signatures they receive from the Signer. In order for this to happen we need the *Registration* phase during which the User commits to his set of attributes. Those attributes are then encoded in the signature in ζ_1. The crucial difference from Abe's approach is that z_1 can no longer be computed as the result of some hash function given a random input rnd. In ACL, z_1 needs to include the attribute commitment C so $z_1 = Cg^{rnd}$ (we need the g^{rnd} factor so that two different signature issuings with the same user cannot be linked to each other). This is the step where the blinded signature inherits some structure from the values the signer can see in the signing step, and allows the encoding of attributes. The fact that z_1 is not the result of a hash function $\mathcal{H}'$ makes our security analysis different from Abe's. We can no longer define $\mathcal{H}'$ so that it will return the output we need, instead we will have to make use of ZK extractors. Unfortunately, we cannot take advantage of Abe's analysis because it only applied to special types of adversaries [112]. Of independent interest, our analysis indicates that, for sequential composition, Abe's blind signature may still be provably

secure under DDH in the RO model. The formal analysis follows in the next section.

5.4.1 Proof of Security

The correctness of the scheme is straightforward:

$$\begin{aligned}
\omega + \omega' &= c + t_2 + c' + t_4 = e + t_2 + t_4 = \varepsilon \bmod q \\
g^{\rho} y^{\omega} &= g^{r+t_1} y^{c+t_2} = g^{r+cx} g^{t_1} y^{t_2} = \alpha \\
g^{\rho_1'} \zeta_1^{\omega'} &= g^{\gamma r_1' + t_3} \zeta_1^{c'+t_4} = (a_1' z_1^{-\omega'})^{\gamma} g^{t_3} \zeta_1^{c'+t_4} = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4} = \alpha_1' \\
g^{\rho_2'} \zeta_2^{\omega'} &= h^{\gamma r_2' + t_5} \zeta_2^{c'+t_4} = (a_2' z_2^{-\omega'})^{\gamma} h^{t_5} \zeta_2^{c'+t_4} = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4} = \alpha_2' \\
z^{\mu} \zeta^{\omega'} &= z^{\tau - \omega' \gamma} \zeta^{\omega'} = \zeta^{\gamma} = \eta.
\end{aligned}$$

Theorem 5.4.1 (Blindness)**.** *The proposed scheme satisfies blindness under the Decisional Diffie-Hellman assumption in the random oracle model.*

Proof. We wish to show that, when interacting with a challenger as described in the definition of blindness, the adversary cannot guess which signature is which (i.e. the adversary should not be able to tell which signature corresponds to each instance and each attribute set). We distinguish between two types of signatures that a challenger might output: a signature $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho_1', \rho_2', \omega', \mu))$ is *correct* for a particular interaction with the signer if there exists a value γ such that $\zeta = z^{\gamma}$ and $\zeta_1 = z_1^{\gamma}$ where z_1 is the value corresponding to this interaction with the signer, and the signature verifies. A signature is *fake* if no such γ exists. Note that it is easy for a challenger to generate both kinds of signatures: a correct signature can simply be output by correctly following the user's side of the protocol. With control of the random oracle, a fake signature is computed as follows: first, the challenger picks ζ and ζ_1 at random from G, and let $\zeta_2 = \zeta/\zeta_1$. Next, it picks random $\rho, \omega, \rho_1', \rho_2', \omega', \mu$ from $\mathbb{Z}_q$ and sets the random oracle such that $\omega + \omega' = \mathcal{H}(\zeta, \zeta_1, g^{\rho} y^{\omega}, g^{\rho_1'} \zeta_1^{\omega'}, h^{\rho_2'} \zeta_2^{\omega'}, z^{\mu} \zeta^{\omega'}, m)$, which ensures that the fake signature verifies.

We will prove the theorem by a hybrid argument. Consider the following three games in which the adversarial signer engages in two signature issuing instances with a challenger who outputs two signatures:

1. `Real`: The challenger outputs two correct signatures, σ_1, σ_2, in random order by honestly following the protocol. A correct signature is one that is honestly generated by interacting with the signer. Thus, in a correct signature the user blinds z, z_1 by the same γ so that $\log_z(z_1) = \log_\zeta(\zeta_1)$.

2. `Hybrid`: The challenger's output consists of a randomly picked correct signature and a fake signature, in random order. In a fake signature the challenger doesn't use the same random γ to blind z and z_1 so that $\log_z(z_1) \neq \log_\zeta(\zeta_1)$ except for negligible probability. Above, we explained how a fake signature can be generated, assuming that the challenger controls the random oracle.

3. `Fake`: The challenger outputs two fake signatures.

What we need to prove is that `Real`$\approx$`Hybrid` and then that `Hybrid`$\approx$`Fake`. Then, it holds that `Real` $\approx$ `Fake`. This proves the blindness of the ACL scheme since the view of the adversary is indistinguishable no matter if he receives two correct or two fake signatures.

We will first show that `Real`$\approx$`Hybrid` and then that `Hybrid`$\approx$`Fake`.

Case 1: `Real`$\approx$`Hybrid`

Suppose that there exists a poly-time adversary $\mathcal{A}$ who is successful in distinguishing between `Real` and `Hybrid` with probability $1/2 + \epsilon$, where the advantage ϵ is not negligible. We then show that there exists a poly-time algorithm $\mathcal{B}$ that solves the DDH problem with non-negligible advantage.

The reduction $\mathcal{B}$ gets as input an instance (g, A, B, D) where (g, A, B, D) is a DDH instance. $\mathcal{B}$ first fixes $z = A$ and then sets $h_0 = g^{e_0}$, and $h_1, \ldots, h_n$ to $A^{e_0}, \ldots, A^{e_n}$ for randomly chosen $e_i \in \mathbb{Z}_q$. $\mathcal{B}$ sends the parameters to $\mathcal{A}$. The adversary $\mathcal{A}$ creates his public key y and sends y to $\mathcal{B}$ along with messages m_0, m_1, commitments C_0, C_1 and openings $(\overrightarrow{L_0}, \overrightarrow{L_1})$ which are two attributes vectors

of size n. $\mathcal{A}$ and $\mathcal{B}$ engage in two interleaving registration and signature issuing instances. During the registration phase we have: $C_{(j)} = A^{k_1^{(j)}}$, $(k_1^{(j)} = e_0 R + e_1 L_{j,1} + \cdots + e_n L_{j,n})$ honestly created, for $j \in \{0,1\}$. Note that the value $k_1^{(j)}$ is known to the reduction $\mathcal{B}$.

Then, during the two signature issuing instances, $\mathcal{A}$ computes $z_1^{(j)} = A^{k_1^{(j)}} g^{rnd^{(j)}}$ where $rnd^{(j)}$ was sent to $\mathcal{B}$. $\mathcal{B}$ responds with random $e^{(j)}$ and then computes two signatures σ_1, σ_2 in the following way:

$\mathcal{B}$ flips two random coins: $coin, b \in \{0,1\}$. Then, lets σ_{2-coin} be the correct signature on $(m_0, \tilde{C}_b)$ (where $\tilde{C}$ is denoted as ζ_1 in our ACL scheme) for the corresponding instance, and σ_{coin+1} be a signature on $(m_1, \tilde{C}_{1-b})$ generated from the input to $\mathcal{B}$, as follows: set $\zeta^{(j)} = D$ and $\zeta_1^{(j)} = D^{k_1^{(j)}} B^{rnd^{(j)}}$; let $\zeta_2^{(j)} = \zeta^{(j)} / \zeta_1^{(j)}$, randomly choose $\rho, \omega, \rho_1', \rho_2', \omega', \mu$, and then define $\mathcal{H}$ so that the signature verifies.

After receiving the signatures, $\mathcal{A}$ outputs `Real` or `Hybrid`. If $\mathcal{A}$ outputs `Real` then $\mathcal{B}$ outputs "DH" or $\mathcal{B}$ outputs "random" otherwise.

Let us analyze the success of $\mathcal{B}$ in distinguishing Diffie-Hellman tuples from random tuples. If the input (g, A, B, D) is a DH tuple, then: $\zeta^{(j)} = D = g^{ab} = A^b = z^b$ and $\zeta_1^{(j)} = D^{k_1^{(j)}} B^{rnd^{(j)}} = g^{abk_1^{(j)}} g^{brnd^{(j)}} = (g^{ak_1^{(j)}} g^{rnd^{(j)}})^b = (A^{k_1^{(j)}} g^{rnd^{(j)}})^b = z_1^b$ so, the signature σ_{coin+1} is distributed identically to a correct signature, and this is precisely the `Real` game. Similarly, if the input is not a DH tuple then it is the `Hybrid` game. Therefore, $\mathcal{B}$ will be correct exactly when $\mathcal{A}$ is and will distinguish a DH tuple with probability $1/2 + \epsilon$, which contradicts to the DDH assumption.

Case 2: `Hybrid`$\approx$`Fake`

Similarly, we build a reduction $\mathcal{B}$ which given an adversary $\mathcal{A}$, who distinguishes between `Hybrid` and `Fake` with probability $1/2 + \epsilon$, solves the DDH problem.

Working in a similar way we did before, $\mathcal{B}$ will now create a fake signature and one using the input tuple. If $\mathcal{A}$ outputs `Hybrid` then $\mathcal{B}$ will output "DH" or "not DH" otherwise. If the input is a DH instance then the output will correspond to the `Hybrid` game and to the `Fake` otherwise. So, $\mathcal{B}$

will be successful whenever $\mathcal{A}$ is and will break the DDH assumption with non-negligible advantage ϵ which is a contradiction.

Finally, since `Real`$\approx$`Hybrid` and `Hybrid`$\approx$`Fake` it follows that `Real`$\approx$`Fake`. □

Theorem 5.4.2 (One-More Unforgeability)**.** *The signature issuing protocol is $(\ell, \ell+1)-$unforgeable for polynomially bounded ℓ if the discrete logarithm problem is intractable and $\mathcal{H}$ is a random oracle.*

Outline of the proof We first discuss the witness indistinguishability of the protocol (Lemma 5.4.4), which allows us to to simulate the Signer with either y-side or z-side witness(es) to extract the witness of the other side. Then, in Lemma 5.4.5 we prove that in order for the User to get a valid signature, he has to blind (z, z_1) into (ζ, ζ_1) only in such a way that $\log_z \zeta = \log_{z_1} \zeta_1$. In Lemma 5.4.6 we prove that is infeasible to create a valid signature without engaging in the issuing protocol with the legitimate Signer. From Lemmas 5.4.5 and 5.4.6 we see that if a User engages in the signature issuing protocol ℓ times and outputs $\ell + 1$ signatures, then, there exist at least two valid signatures linked to a particular run of the issuing protocol. Finally, it needs to be proven that a forger who manages to produce two signatures from a single protocol run can be used to solve the discrete logarithm problem.

We will denote the i-th execution of the issuing protocol by run_i. Recall that a transcript of run_i of the issuing protocol contains values $(z_{1,i}, z_{2,i})$; by the z-side witness of run_i we denote $(w_{1,i}, w_{2,i})$ such that $z_{1,i} = g^{w_{1,i}}$ and $z_{2,i} = h^{w_{2,i}}$.

Consider an alternative signing algorithm that, instead of using the y-side witness in the issuing protocol, uses the z-side witness. Let $params = (q, G, h, z, \{h_j\})$ be the public parameters, and y be the public key of the Signer. Suppose that the registration and the preparation phases of the signing protocol for run_i resulted in the signer and the user setting up the values $z_{1,i} = g^{w_{1,i}}$, $z_{2,i} = h^{w_{2,i}}$. The alternative signing algorithm takes as input the public parameters $params$, the public key y, the preparation phase output (z_1, z_2) and the values $(w_{1,i}, w_{2,i})$ instead of the secret key x, and works

as follows:

1. Generate $c_i, r_i \in_U \mathbb{Z}_q$ and set $a_i := g^{r_i} y^{c_i}$.
2. Compute $a'_{1,i} := g^{u_{1,i}}$ and $a'_{2,i} := h^{u_{2,i}}$ with $u_{1,i}, u_{2,i} \in_U \mathbb{Z}_q$.
3. Sends $a_i, a'_{1,i}, a'_{2,i}$ to the user.
4. Given e_i from user, compute $c'_i := e_i - c_i \bmod q$, $r'_{1,i} := u_{1,i} - c'_i w_{1,i} \bmod q$ and $r'_{2,i} := u_{2,i} - c'_i w_{2,i} \bmod q$.
5. Send $r_i, c_i, r'_{1,i}, r'_{2,i}, c'_i$ to the user.

Definition 5.4.3. *The signing protocol described above is called* the z-side signer.

Lemma 5.4.4. *The signer's output is perfectly indistinguishable from the output of a z-side signer.*

Proof. Lemma 5.4.4 follows from the result of Cramer [68]. □

Note that, in order to run the z-side signer, it is necessary to somehow get the corresponding witness. We will see later that, with black-box access to an adversarial user in the registration phase, a reduction can use standard Σ-protocol extraction techniques in order to learn the representations of C; if the reduction is, in addition, given the discrete logarithms of all the public parameters to the base g, it will be able to compute the z-side witness, and so, inside a reduction, the z-side signer can be invoked.

Lemma 5.4.5. *(Restrictive Blinding) Let $\mathcal{A}$ be a User that engages in the signature issuing protocol ℓ times, and outputs a valid signature $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu))$. Let run_i denote the i-th execution of the protocol and $z_{1,i}$ denote z_1 used by the signer $\mathcal{S}$ in run_i. For polynomially bounded ℓ and for all polynomial-time $\mathcal{A}$, the probability that $\log_z \zeta \neq \log_{z_{1,i}} \zeta_1$ holds for all i, is negligible if the discrete logarithm problem is intractable and $\mathcal{H}$ is a random oracle.*

Proof. Let $\mathcal{A}$ have at most q_h accesses to $\mathcal{H}$ and ask at most ℓ signatures to $\mathcal{B}$. Let $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu))$ be a signature that $\mathcal{A}$ outputs and satisfies $\log_z \zeta \neq \log_{z_{1,i}} \zeta_1$ for all i with probability ϵ_0 which is not negligible in n (q_h and ℓ are bounded by a polynomial in the security parameter

n). We randomly fix an index $I \in \{1, \ldots, \ell\}$ and regard $\mathcal{A}$ as successful only if the RO query for the resulting signature was during the I-th run. (If the resulting signature does not correspond to any query, then $\mathcal{A}$ is successful only with negligible probability due to the randomness of $\mathcal{H}$.) Let $(\mathbf{p}; \mathbf{q}; \mathbf{g}; \mathbf{Y})$ be an instance of the DL problem.

Reduction Algorithm: $\mathcal{B}$ first sets $(p, q, g) := (\mathbf{p}, \mathbf{q}, \mathbf{g})$. Then, it flips a coin $\chi \in_U \{0, 1\}$ to either select $y := \mathbf{Y}$ (case $\chi = 0$), or $h := \mathbf{Y}$ (case $\chi = 1$) and also guesses a random index $I \in \{1, \ldots, \ell\}$.

Case $y = \mathbf{Y}$: (extracting y-side witness)

1. *Key Generation*: $\mathcal{B}$ selects $k, k', \{k_j\} \in_U \mathbb{Z}_q$, for $0 \leq j \leq n$, and sets $h := g^k$ and $h_j := g^{k_j}$, $z = g^{k'}$. As a result of setting the parameters this way, $\mathcal{B}$ will always be able to compute z-side witnesses, as long as $\mathcal{B}$ can successfully extract the user's attributes and randomness in the registration phase.
2. *Registration:* $\mathcal{B}$ extracts values $(L_1, \ldots, L_n; R)$ using a knowledge extractor for proof π_1 (since we only worry about sequential composition the extractor may rewind) and defines $K = k_0 R + \sum_{ij1}^{n} k_j L_j$ (so that $C = g^K$).
3. *Preparation:* $\mathcal{B}$ selects $w_{1,i} \in \mathbb{Z}_q$ and sets $rnd = w_{1,i} - K$. Then, computes $w_{2,i} = (k' - w_{1,i})/k \bmod q$ (so we have $z_{1,i} = g^{w_{1,i}}, z_{2,i} = h^{w_{2,i}}$). $\mathcal{B}$ sends rnd to $\mathcal{A}$.
4. *Validation* $\mathcal{B}$ runs $\mathcal{A}$ using the z-side signer described above. At the end of each run_i the z-side signer sends: $r_i, c_i, r'_{1,i}, r'_{2,i}, c'_i$ to $\mathcal{A}$. $\mathcal{B}$ simulates $\mathcal{H}$ by returning $\varepsilon \in_U \mathbb{Z}_q$ to the random oracle queries issued by $\mathcal{A}$. $\mathcal{A}$ outputs a signature $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu)$, that corresponds to some ε.
5. If the RO query for the resulting signature σ happened in run_I then move to the next step, otherwise the reduction fails.
6. *Rewinding*: Reset and restart $\mathcal{A}$ with the same setting. For run_I and after, $\mathcal{B}$ simulates $\mathcal{H}$ with $\tilde{\varepsilon} \in_U \mathbb{Z}_q$. $\mathcal{A}$ outputs a signature, say $\tilde{\sigma} = (\zeta, \zeta_1, \tilde{\rho}, \tilde{\omega}, \tilde{\rho'_1}, \tilde{\rho'_2}, \tilde{\omega'}, \tilde{\mu})$.

   If $\omega \neq \tilde{\omega}$, $\mathcal{B}$ outputs $x := (\rho - \tilde{\rho})/(\tilde{\omega} - \omega) \bmod q$. The simulation fails, otherwise.

Case $h = \mathbf{Y}$: (extracting z-side witness)

1. *Key generation*: $\mathcal{B}$ selects $x, \{k_j\} \in_U \mathbb{Z}_q$ and sets $y := g^x$ and $h_j := g^{k_j}$. It also selects $w_1, w_2 \in_U \mathbb{Z}_q$ and sets $z := g^{w_1} h^{w_2}$.
2. *Signature Issuing*: $\mathcal{B}$ runs $\mathcal{A}$ simulating the signer as follows.
   (a) For $i \neq I$, $\mathcal{B}$ follows the protocol with y-side witness, x.
   (b) For $i = I$, $\mathcal{B}$ engages in the issuing protocol using the z-side witness (w_1, w_2) as follows.
      i. *Registration* Extract values $(L_1, \ldots, L_n; R)$ using a ZK extractor and define $K = k_0 R + \sum_{i=1}^{n} k_i L_i$ (so that $C = g^K$).
      ii. *Preparation* Set $rnd = w_1 - K$ (so we have $z_{1,J} = g^{w_1}, z_{2,J} = h^{w_2}$). $\mathcal{B}$ sends rnd to $\mathcal{A}$.
      iii. *Validation* $\mathcal{B}$ runs $\mathcal{A}$ using the z-side signer and at the end of run_I sends $(r_I, c_I, r'_{1I}, r'_{2I}, c'_I)$ to $\mathcal{A}$.

   $\mathcal{B}$ simulates $\mathcal{H}$ by returning $\varepsilon \in_U \mathbb{Z}_q$ to the RO queries made by $\mathcal{A}$.
3. $\mathcal{A}$ outputs a signature, say $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu)$, that corresponds to some ε. If the RO query for the forged signature didn't happen in run_I then fail, else move to the next step.
4. *Rewinding*: Rewind and restart $\mathcal{A}^*$ with the same setting. For run_I and forward, $\mathcal{B}$ simulates $\mathcal{H}$ by returning $\tilde{\varepsilon} \in_U \mathbb{Z}_q$.
5. $\mathcal{A}$ outputs a signature, say $\tilde{\sigma} = (\zeta, \zeta_1, \tilde{\rho}, \tilde{\omega}, \tilde{\rho'_1}, \tilde{\rho'_2}, \tilde{\omega'}, \tilde{\mu})$, that corresponds to some $\tilde{\varepsilon}$. If $\omega' \neq \tilde{\omega'}$, $\mathcal{B}$ computes $w'_1 = (\rho'_1 - \tilde{\rho'}_1)/(\mu - \tilde{\mu}) \bmod q$, $w'_2 = (\rho'_2 - \tilde{\rho'_2})/(\mu - \tilde{\mu}) \bmod q$, and outputs $w = (w_1 - w'_1)/(w'_2 - w_2) \bmod q$ (where $w = \log_g h$). Simulation fails otherwise.

Evaluation of Success Probability. The reduction is successful when it gets $\omega \neq \tilde{\omega}$ for $\chi = 0$ or when it gets $\omega' \neq \tilde{\omega}'$ for $\chi = 1$. What we need to argue about is that, independently of the choice of χ, the above will happen with non negligible probability. Note that we construct the reduction by giving different responses after the Ith run (for a randomly chosen $I \in \{1, \ldots, \ell\}$) when rewinding and we let the reduction fail if the forgery didn't happen in run_I.

Let us now fix the adversary's view and his RO tape. Once everything is fixed, we notice that $\mathcal{A}$'s forgery, is uniquely determined by c' and ε. We now consider two cases: (1) $\omega' = \omega$ with non negligible probability even if ε and c' change (after rewinding). In this case $\omega' = \tilde{\omega'}$ and thus $\omega \neq \tilde{\omega}$: $\omega + \omega' = \varepsilon \neq \tilde{\varepsilon} = \tilde{\omega} + \tilde{\omega'}$, so the y-side witness can be extracted. (2) $\omega' \neq \omega$ with non negligible probability if ε and c' change. So, when we run the adversary again with access to z-side witness and with different $\tilde{\varepsilon}$ we get two signatures from him where $\omega = \tilde{\omega}$ and thus $\omega' \neq \tilde{\omega'}$ and thus we can extract the z-side witness. Since we fail if the forgery didn't happen in run_I we do not need to consider what happens when we run $\mathcal{A}$ with y-side witness for $\chi = 1$ and $i \neq I$.

What is left to show is that for $\chi = 1$, the z-side witness, (w'_1, w'_2), extracted is not the one that B already knows Remember that the reduction in the beginning, selects $w_1, w_2 \in_U \mathbb{Z}_q$ and sets $z := g^{w_1} h^{w_2}$) and in run_I sets rnd so that $z_{1,I} = g^{w_1}, z_{2,I} = h^{w_2}$. The signature that $\mathcal{A}$ produces contains proofs that $\zeta = z^{\gamma}, \zeta_1 = g^{w'_1}$ and $\zeta_2 = h^{w'_2}$, so $w'_1 = \gamma w_1$ and $w'_2 = \gamma w_2$ and we can extract those w'_1, w'_2 by rewinding. For the forgery that $\mathcal{A}$ outputs it holds $\log_z \zeta \neq \log_{z_{1,I}} \zeta_1$ (by definition of Lemma 2) which guarantees that we will obtain two different representations of z. □

Lemma 5.4.6. *Any poly-time adversary $\mathcal{A}$ outputs a valid signature without interacting with the Signer only with negligible probability if the discrete logarithm problem is intractable and $\mathcal{H}$ is a random oracle.*

Proof. Suppose that there exists adversary $\mathcal{A}$ that outputs $\ell + 1$ valid signatures with probability ϵ not negligible after interacting with the Signer at most ℓ times. The case $\ell = 0$ has been ruled out by Lemma 5.4.6. We consider $\ell \geq 1$.

By Lemmas 5.4.5 and 5.4.6, among the $\ell+1$ signatures, there exist at least two signature-message pairs which contain (ζ, ζ_1) and $(\bar{\zeta}, \bar{\zeta_1})$ such that $\log_{\zeta} \zeta_1 = \log_{\bar{\zeta}} \bar{\zeta_1} = \log_z z_{1I}$ holds for z_{1I} used in run_I for some I in $\{1, \ldots, \ell\}$. Now, there exist two queries to $\mathcal{H}$ that correspond to those signatures. In a similar way as in the proof of Lemma 5.4.5, we guess the indices of these queries and regard $\mathcal{A}$ as being successful only when the guess is correct. We consider an equivalent adversary $\mathcal{A}^*$ that asks

$\mathcal{H}$ only twice and succeeds with probability $\epsilon^* = \epsilon/\binom{q_h}{2}$ in producing two signatures in the expected relation.

We construct a reduction $\mathcal{B}$ that, given $(\mathbf{p}, \mathbf{q}, \mathbf{g}, \mathbf{Y})$, solves $\log_g \mathbf{Y}$ in $\mathbb{Z}_q$ by using $\mathcal{A}^*$.

Reduction Algorithm: $\mathcal{B}$ sets $(p, q, g) := (\mathbf{p}, \mathbf{q}, \mathbf{g})$. It then flips a coin, $\chi \in_U \{0, 1\}$, to select either $y := \mathbf{Y}$ (case $\chi = 0$), or $y := g^x$ with randomly chosen x (case $\chi = 1$).

1. *Setup* If $\chi = 0$, set up and use the z-side signer. Set h, $h_0, \ldots, h_n$ such that their discrete logarithm base g is known. Else, set up the y-side signer by letting $y = g^x$ for a randomly chosen x; set up h, $h_0, \ldots, h_n$ such that their discrete logarithm base Y is known.
2. *Registration* If $\chi = 0$, extract $(R, L_0, \ldots, L_n)$. Else follow the protocol.
3. *Preparation* Follow the protocol.
4. *Validation* If $\chi = 0$, use the z-side signer, else follow the protocol.
5. *Responding to the RO queries* $\mathcal{B}$ simulates $\mathcal{H}$ by returning random values, say ε_1 and ε_2, to the two RO queries.
6. *Rewinding* When $\mathcal{A}^*$ outputs two signatures $(\sigma^{(1)} = (\zeta^{(1)}, \zeta_1^{(1)}, \rho^{(1)}, \omega^{(1)}, \rho'^{(1)}_1, \rho'^{(1)}_2, \omega'^{(1)}, \mu^{(1)}))$ and $(\sigma^{(2)} = (\zeta^{(2)}, \zeta_1^{(2)}, \rho^{(2)}, \omega^{(2)}, \rho'^{(2)}_1, \rho'^{(2)}_2, \omega'^{(2)}, \mu^{(2)}))$ corresponding to ε_1 and ε_2, $\mathcal{B}$ resets the adversary and this time responds to the two RO queries with $\tilde{\varepsilon}_1$ and $\tilde{\varepsilon}_2$.
7. *Interacting with $\mathcal{A}^*$ after rewinding* is the same as before, according to the value of χ.
8. *Extracting* When $\mathcal{A}^*$ again outputs two signatures $(\tilde{\sigma}^{(1)} = (\zeta^{(1)}, \zeta_1^{(1)}, \tilde{\rho}^{(1)}, \tilde{\omega}^{(1)}, \tilde{\rho'}^{(1)}_1, \tilde{\rho'}^{(1)}_2, \tilde{\omega'}^{(1)}, \tilde{\mu}^{(1)})$ and $(\tilde{\sigma}^{(2)} = (\zeta^{(2)}, \zeta_1^{(2)}, \tilde{\rho}^{(2)}, \tilde{\omega}^{(2)}, \tilde{\rho'}^{(2)}_1, \tilde{\rho'}^{(2)}_2, \tilde{\omega'}^{(2)}, \tilde{\mu}^{(2)})$ corresponding to $\tilde{\varepsilon}_1$ and $\tilde{\varepsilon}_2$, do: if $\chi = 0$ and $\omega^{(1)} \neq \tilde{\omega}^{(1)}$ or $\omega^{(2)} \neq \tilde{\omega}^{(2)}$, then extract the y-side witness as described in the proof of Lemma 5.4.5; else if $\chi = 1$ and $\omega'^{(1)} \neq \tilde{\omega'}^{(1)}$ or $\omega'^{(2)} \neq \tilde{\omega'}^{(2)}$, then extract the z-side witness as described in Lemma 5.4.5.

Evaluation of Success Probability. Here we need to show that $\mathcal{B}$ will extract the witness that it doesn't already know with non-negligible probability (otherwise, χ is revealed). Consider all the q^2 possible $(\varepsilon_1, \varepsilon_2)$ pairs that can be given as responses to $\mathcal{A}^*$'s two RO queries. Let *succ* be the set

of pairs $(\varepsilon_1, \varepsilon_2)$ for which $\mathcal{A}^*$ succeeds: $succ = \{(\varepsilon_1, \varepsilon_2) | \mathcal{A}$ breaks the seq. one-more unforgeability$\}$. What we want to show is that for a random pair $(\varepsilon_1, \varepsilon_2) \in succ$ there exists $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2}) \in succ$ such that $\mathcal{A}^*$ produces the same e with non-negligible probability in the run where the first RO query happened. Notice that the transcript (i.e. the messages exchanged between the Signer and $\mathcal{A}^*$) of these two runs has to be the same. We call such a pair $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$ *good* for $(\varepsilon_1, \varepsilon_2)$.

Consider $\mathcal{A}^*$ is run twice: once with $(\varepsilon_1, \varepsilon_2)$ and once with $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$, both times producing two message-signature pairs corresponding to z_{1I}. Taken together, $(\varepsilon_1, \varepsilon_2)$, $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$, the transcript of the runs and the four message-signature pairs allow the reduction to compute either a y side or a z side witness. However, we need to argue that, with non-negligible probability, this will be a witness the reduction doesn't already know. Since the transcripts are identical, the joint distribution of the two views ($\mathcal{A}^*$'s view in the first run and its view, after the rewinding, in the second run) is independent of χ.

First, note that there are only q possible e's but q^2 possible $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$, and so there can only be a negligible fraction of $(\varepsilon_1, \varepsilon_2)$ for which no *good* $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$ exists. Suppose that $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$ is *good* for $(\varepsilon_1, \varepsilon_2)$ but $(\varepsilon_1^*, \varepsilon_2^*)$ is not; moreover, the reduction that gives $(\varepsilon_1, \varepsilon_2)$ before rewinding, and $(\varepsilon_1^*, \varepsilon_2^*)$ after rewinding, can only compute the witness it already knows (i.e. if $\chi = 0$, it can compute the z-side witness, and for $\chi = 1$ the y-side). Then for the same choice of χ, the reduction would be computing the witness it does *not* already know if, instead, it used $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$ and $(\varepsilon_1^*, \varepsilon_2^*)$. For example, if $\chi = 0$ we would have $\omega \neq \tilde{\omega}$ and $\omega = \tilde{\omega^*}$ respectively, which implies that $\tilde{\omega} \neq \tilde{\omega^*}$ and thus by choosing $(\varepsilon_1, \varepsilon_2)$ and $(\varepsilon_1^*, \varepsilon_2^*)$ the y side witness could be extracted. Since it is just as likely to pick $(\varepsilon_1, \varepsilon_2)$ or $(\tilde{\varepsilon_1}, \tilde{\varepsilon_2})$ for the values before rewinding, it is just as likely to succeed as to fail, independently of χ (this analysis was first given by Abe and Okamoto [6]).

Finally, the only thing left to prove is that an adversary cannot open the sets of commitments $\{C^{(i)}\}$ and $\{\zeta^{(i)}, \zeta_1^{(i)}\}$ to two different multisets. Note that $\zeta^{(i)}$ uniquely determines $\gamma^{(i)}$, and by Lemma 5.4.5, it follows that for every i, there exists some $z_1^{(i')}$ such that $\zeta_1^{(i)} = (z_1^{(i')})^{\gamma^{(i)}}$. Thus, an alternative opening of the commitment $\{\zeta^{(i)}, \zeta_1^{(i)}\}$ gives rise to an alternative representation of

$z_1^{(i')}$ in bases $h, h_1, \ldots, h_n, g$. In turn, by standard techniques, this breaks the discrete logarithm assumption. (The reduction will just honestly emulate the signer, and embed the DL instance into the parameters $h, h_1, \ldots, h_n, g$.) □

Setting Parameters. Note that our reduction is not tight: an adversary $\mathcal{A}$ whose success probability is ϵ gives rise to the reduction $\mathcal{B}$ with the same runtime but success probability $\epsilon/\binom{q_h}{2} \approx \epsilon/q_h^2$. This, in other words means that if $\mathcal{A}$ takes time t to break unforgeability with probability ϵ, then our reduction, $\mathcal{B}$, in order to break the discrete logarithm assumption with the same success probability would need time $t \cdot q_h^2$. So, if $\mathcal{A}$ breaks unforgeability faster than in time 2^k, then $\mathcal{B}$ breaks DL in time faster than $2^k \cdot q_h^2 = 2^{k+2\log q_h}$. Thus, we need to set the parameters to be sufficiently large so that is reasonable to assume that the reduction would take time $2^{k+2\log q_h}$ to break DL directly.

As suggested by NIST [111] in order to tolerate an adversary whose computational power is 2^k, one needs an elliptic curve group of size 2^{2k} which for our reduction should be $2^{2(k+2\log q_h)}$. Now, let's be very conservative and assume that $q_h = 2^{80}$, thus the factor $\binom{q_h}{2}$ we lose is of 160-bits. Thus, if we want to achieve 128-bit security we would need an elliptic curve group where the bit size of an element is $2(128 + 2 \cdot 80) = 576$-bits. Note that the equivalent security level for RSA requires 3072-bit keys.

5.5 Related Work and Comparisons.

Let's now take a closer look in our ACL construction and compare it with other well known blind signature and credential schemes. For our comparisons we focus on the efficiency of signing (which is analogous to the credential issuing), the cost of verification of the signature/credential, the size of the signature/credential, the security of the schemes i.e. blindness and unforgeability and whether the schemes support attributes which is obviously not going to be the case for a "traditional" blind signature scheme.

	Brands [36]		Abe [3]		ACL Scheme		CL credentials [45] Rsa		CL credentials [45] Pairings	
Efficiency	S^1	U^2	S	U	S	U	S	U	S	U
Signing[3]	2	12	6	12	7	13	10	8	15	14p
Verification	7	0	11	1	11	1	NC[4]		5+6p	9+6p
Sign. size	6 elem.		9 elem.		9 elem.		~ 0[5]		~ 0[5]	
Blindness	✓		✓		✓		✓		✓	
Prov. Unforg.	✗		GM[6]		✓		✓		✓	
Attributes	✓		✗		✓		✓		✓	

[1] Signer, [2] User, [3] In number of exponentiations
[4] Non-comparable, [5] It can be used multiple times
[6] Generic group model

Table 5.1: Comparison of anonymous credentials

The ACL construction consists of three phases: registration, preparation and validation. Registration is the most expensive phase, but it need not be repeated for the same user/signer pair should the user need his credential reissued (as long as his attribute set is the same). Preparation and validation require the signer to perform 7 exponentiations, while the user performs 13; verification requires 8 exponentiations. These numbers are essentially the same as Abe's blind signature; Brands' blind signature at the heart of UProve requires essentially the same number of exponentiations for signature issue, and 4 exponentiations fewer for verification. Thus, we get comparable efficiency to the most efficient examples of protocols that either lack essential features (such as previously provable secure blind signatures) or provable security (such as UProve). We also compare ourselves to the Camenisch-Lysyanskaya [45] which is significantly less efficient during issuing due to the use of pairings or RSA group but is a multi-use credential, thus we denote the signature/credential size to be close to zero since it can be used multiple times. See Table 5.1 for the comparisons.

A related work is due to Guajardo et al. [89] where they proposed "encrypted anonymous credentials". The idea is that an issuer can certify encrypted attributes in such a way that none of the involved parties, including the user, can learn the value of the attributes. This is an interesting application but cannot be used as a building block so it is incomparable to our work. Moreover, their construction [89] is based on Brands scheme which as mentioned above cannot be proven one-more unforgeable [13].

Recommended parameter setting. For 128 bit security we recommend using a group of 576 bits[1]. In order to achieve an equivalent security level with CL credentials keys of 3072 bits would be required.

5.6 Conclusions

We defined and proposed an efficient and provably secure construction of blind signatures with attributes. Prior notions of blind signatures did not yield themselves to the construction of anonymous credential systems, not even if we drop the unlinkability requirement of anonymous credentials. Our new notion in contrast is a convenient building block for anonymous credential systems. In addition to the definition, we solve the open problem of constructing anonymous credentials based on the DDH assumption. Our new construction is efficient: it requires just a few exponentiations in a prime-order group in which the decisional Diffie-Hellman problem is hard. Our construction is inspired by the Abe blind signature [3] in which, unlike other provably secure blind signatures [30, 43, 113, 123, 133], blinding still preserves some structural elements into which attributes can be embedded. In the random-oracle model, our construction is unlinkable under the decisional Diffie-Hellman assumption, and unforgeable under the discrete-logarithm assumption for sequential composition (we leave extending it to concurrent self-composition as an open problem).

The fact that our new construction works in elliptic curve groups makes it very appealing for applications in mobile devices, RFIDs and smartcards. In the next Chapter we present an application of our Anonymous Credentials for private payments in the public transportation scenario.

[1] Please refer to the proof of Theorem 5.4.2 for a justification of this number.

6
Efficient Payments for Public Transportation Systems

Electronic cash (e-cash) can be viewed as a subcategory of anonymous credentials. It is a *single-use* credential scheme with an extra mechanism that allows double spending detection. You think of the electronic coins as being single-use credentials issued by the Bank to a user. The *Anonymous Credentials Light* construction we just presented can be extended to an efficient e-cash scheme that moreover has the nice property that allows the encoding of users attributes in the coins (i.e. user age, address etc.).

In this Chapter we are going to discuss how to use cryptographic e-cash in the public transportation scenario in order to achieve secure and private electronic payments. One of the main challenges of such an attempt is how to implement e-cash constructions efficiently on lightweight devices like smartphones or RFID cards. We present NFC-smartphone implementations of: Brands' e-cash scheme [36] (with and without attributes), Abe's e-cash scheme [3] (which is without attributes) and the e-cash constructed by Anonymous Credentials Light [14] and we compare and evaluate the performance of those four different schemes in Section 6.4. Due to their efficiency during the spending phase, when compared to other schemes, and the fact that payments can be verified offline,

these schemes are especially suited for, but not limited to, use in public transport. Additionally, the encoding of validated attributes (e.g. a user's age range, zip code etc.) is possible in the coins being withdrawn, which allows for additional features such as variable pricing (e.g. reduced fare for senior customers) and privacy-preserving data collection. We present a subtle technique to make use of the ECDHKeyAgreement class that is available in the BlackBerry API (and in the API of other systems) and show how the schemes can be implemented efficiently to satisfy the tight timing imposed by the transportation setting.

Electronic Cash. Electronic payments have been becoming more and more prevalent and the two major required properties are security and privacy. Payments made with debit or credit cards do not provide any privacy guarantee for the users since the corresponding financial institution can track all their transactions. Starting with Chaum [59], the cryptographic community worked on electronic analogues to physical money (e-cash) that guarantee secure and private payments [59, 58, 36, 41, 20]. As discussed in the Introduction, typical e-cash system consists of three types of entities: the Bank, the Users and the Merchants. Users withdraw electronic coins from the Bank, spend them to merchants who finally deposit them back to the Bank. E-cash systems should satisfy two main properties (1) *unforgeability*: an adversarial user cannot spend more e-coins than he withdrew; and (2) *anonymity*: nobody (including the Bank) can link spending transactions to each other or to specific withdrawal instances. As opposed to physical cash, it is easy to duplicate electronic coins, so mechanisms ensuring a user cannot spend one coin multiple times are needed. Two solutions have been proposed in the literature: the first is *online* e-cash [60], in which the merchants are constantly connected to the Bank and can therefore check whether a coin has already been deposited before accepting it. In order to overcome the strong requirement of an online Bank, a second solution is to use a *double-spending* mechanism [58]. The idea is that as long as a user is honest, his anonymity is preserved, but once he tries to cheat the system by spending the same e-coin multiple times his identity is revealed. In this thesis we focus on off-line e-cash since we believe it is much more realistic

to not assume the bank always being online.

Privacy in Public Transportation Payments. Electronic payments in transit systems are very convenient, but at the same time, they raise concerns about the security and the privacy of their customers. Currently employed electronic public transportation payment systems have suffered security attacks [82, 8], and they do not incorporate means to protect the user's (locational) privacy. For example it is reported that "in the period from August 2004 to March 2006 alone, the Oyster system (electronic payment system for public transport in greater London) was queried 409 times" [126], which shows that location data about customers is collected and stored and later used by other agencies. "Anonymous" cards are offered in some systems, but even with those cards user privacy can be sacrificed. The reason is that there is a unique identifier assigned in each card which makes payments made using the same card linkable.

Privacy is an especially challenging problem in this context since it not only spans cryptographic theory and many engineering fields but extends into public policy areas such as environmental justice policy and sociology issues such as "fair access to all". However, in order to enable a large-scale deployment and broad acceptance of such a payment system, adequate security and privacy mechanisms are an essential requirement. Indeed, current users of FasTrak, the electronic toll collection system of California, rank "more secure technology to prevent security and privacy issues" in the top three recommendations of a recent study [128].

One may argue that giving up one's privacy is a small price to pay for such important benefits as ease and convenience, not to mention the fact that the information collected can facilitate advanced traveler information dissemination, traffic management, travel time estimation, emergency management, congestion pricing and carbon emissions control, and environmental justice assessments. However, by using e-cash with attributes one can get all the benefits of electronic payments without sacrificing privacy.

NFC. In order to satisfy the tight timing requirements imposed by the transportation setting, we choose a payment device that can communicate with an access point in a contactless fashion. A standard for contactless communication integrated in modern smartphones is Near Field Communication (NFC) [1]. It allows a smartphone to communicate with other NFC-enabled devices within a range of a few centimeters. While the throughput is moderate, the benefit of this type of communication is its simple and hence fast establishment, as electronic devices can be connected with a simple touch. There are predictions that in the long run NFC devices will replace the multitude of smart cards that many users carry around currently. For transportation authorities the advantage of relying on users' NFC-enabled smartphones, is that no additional (electronic) tokens will have to be handed out. Instead only a software-app has to be provided that the user can download to his phone. This contributes to decreasing the revenue collection cost, and further allows the payment system to be updated easily. If a change to the system is made, the transportation authority only needs to provide a software update, rather than a hardware roll-out.

6.1 Payment System Requirements of the Transportation Setting

Several features of the considered e-cash schemes are especially useful for the design of a payment system that fulfills the requirements of transportation payments. These will be presented in the following.

Verifying a Payment Offline We envision a scenario that is based on currently employed transportation payment systems, where a user buys fares at a vending machine and pays for a fare at an entrance point of the transportation system. An access granting device grants the user access after validating the payment. For example in the case of buses it cannot be assumed that those access granting devices are permanently connected to the back-end system of the transportation authority. Yet, we assume temporary connection to transfer the data of collected payments. Consequently,

verifying a payment needs to be possible in an offline fashion. This holds for the chosen schemes, where a verification of the payment does not require access to the database. However, in this case fraud cannot be detected at the time of the payment, as it requires comparing the received data with the database. Alternatively the chosen e-cash schemes reveal a crime after the fact, and allow a user to be penalized, when misusing the system.

Modular Payment System In case of multiple transportation authorities the e-cash concept offers great convenience advantages for users, as it allows for multiple banks and shops. A user does not need multiple payment devices to use different transportation systems. Rather he can withdraw coins at one transportation authority TA_1 and use them to pay for a trip in the transportation system of another transportation authority TA_2. Thus TA_1 would act as the bank and TA_2 as a shop. TA_2 can later deposit the received coins to its account at TA_1. This is achieved, as no trust between shops and banks is assumed in the e-cash concept.

Different Denominations A transportation authority needs to offer different fare prices. This can be accomplished by assigning a low monetary value to coins and letting the user spend many coins to pay for a fare. Yet, spending many coins increases the execution time of a payment. It is desirable to keep the amount of coins that have to be spent low. Allowing for different denominations of coins reduces the number of coins that need to be spent. This can be achieved, in the e-cash setting. One way to accomplish this is to have the bank possess multiple public keys, one key for each possible denomination of coins.

Encoding Attributes As mentioned above the collection of user data is important to improve the system. A system is desirable, where data that is needed to analyze the system can be collected from users, without sacrificing their privacy. With the use of e-cash this can be achieved by encoding attributes into coins. Those attributes allow the user to reveal some information, as for example his zip code, while keeping further information hidden, and hence hiding his identity. Apart from private data collection, encoding attributes into coins allows for private variable pricing. The system could require the users to encode specific information like their age and then, when the coin is spent,

compute the right fare according to the presented attributes.

6.2 E-cash with Attributes

In this section we describe how e-cash with attributes works and we provide instantiations based on Brands' e-cash scheme [36] and ACL [14]. The descriptions are tailored to match the transportation setting but can be easily adopted for other settings as well. To our knowledge this is the first time that an explicit description of how Brands and ACL schemes can be used as e-cash schemes with attributes is given. Although at the end we will compare the implementation results of Brands with attributes, Brands without attributes, ACL and Abe's e-cash (which is essentially ACL without attributes), here we will only provide the detailed construction of the attribute supporting schemes. In our descriptions we will point out the differences between the attribute and the corresponding non-attribute version.

A transportation payment system based on e-cash is described as an interaction between three kinds of players: the transportation authority TA, with vending machines that are connected to its database, the users of the system $\mathcal{U}$ and the payment machines $\mathcal{M}$ that are placed at the entrance points and, after receiving a valid payment, grant a user access to the transportation system.

We now give an informal, generic description of the protocols that constitute an e-cash scheme with attributes.

Setup. During the setup phase the transportation authority (or another trusted authority) generates the public parameters for the system together with the TA's secret key.

Account opening. Initially, users need to register with the TA and open an account. To do so, a user $\mathcal{U}$, would have to present some form of identification (e.g. a passport) to the TA and provide a cryptographic commitment C for a set of attributes $(L_1, \ldots, L_n)$ that are required for the system and for his public key $pk_U = g^{sk_U}$. The attribute types and their order are determined by the TA during the setup of the system. A possible setting is to use attribute L_1 for the user's secret key

($sk_U = L_1$), L_2 for his age, L_3 for his zip code etc. Note that for efficiency reasons we may require all attributes that are revealed during the spending phase to be binary attributes i.e. L_2 equals 0 if the user is more than 18 and less than 65 years old and 1 otherwise (used for age discounts). This way we avoid *range proofs* which are rather costly. Besides, for the transportation setting binary attributes would work rather well, given that we need the attributes mainly for variable pricing or private data collection. For both settings it is sufficient to just have binary attributes. Also note that we may require the users to update their accounts say once a year in order to update attributes that change over the year. After the account opening phase, the TA stores for each user his commitment C and a copy of his identification.

Withdrawal. Whenever $\mathcal{U}$ wants to withdraw coins from the TA, he first needs to prove ownership of his account and then he runs the withdrawal protocol. In order to prove ownership of his account, we require $\mathcal{U}$ to form digital signature (i.e. a Schnorr signature [131]) on a message that describes the number of coins he wishes to withdraw and includes some kind of timestamp. This is useful for two reasons: (1) the TA can easily identify the user by checking whether the signature is valid under the user's public key and (2) it provides an extra level of security against a man in the middle who may intercept the communication and try to withdraw more coins. Note that depending on the protocol the adversary may not be able to actually spend these coins (since he needs the user's secret key to execute the payment protocol). However, he can still hurt $\mathcal{U}$ by reducing his account balance. Alternatively, we could require the execution of the identification phase for every single coin withdrawn but this is obviously less efficient than identifying $\mathcal{U}$ once for all the coins he wishes to withdraw.

Spending. In order for a user $\mathcal{U}$ to spend a coin at a payment machine, $\mathcal{U}$ runs the spending protocol as described in the corresponding e-cash system. During this spending phase he may choose to reveal some of his attributes depending on the setting of the system. When a User reveals an attribute L_j he can either send the attribute value in clear or provide a proof of knowledge of that attribute. Note that in each case he needs to prove that the attribute he reveals is the same one he committed

to during the account opening phase.

Deposit. The payment machines present the spending transcript to the TA. Here, an extra mechanism is required to protect the TA against cheating users, this mechanism is called: *double spending detection*. The double spending detection is executed off-line in order to detect and penalize users, who spent the same coin more than once. In order for double spending detection to be possible, the TA needs to preserve a special database where all the deposited coins are stored. Obviously we cannot assume that all the coins are stored there forever, thus we need to introduce some kind of *coin expiration date* after which the coin will be deleted from the database. This could be done either by having the expiration date encoded as an attribute in the coins (so the user would have to prove that the coin is still valid during spending) or by changing the TA's public key (i.e. once a year) and thus invalidating the coins issued under the old key.

6.2.1 Brands' E-cash with Attributes

We describe how Brands' e-cash scheme [36] can be modified to support the encoding of users' attributes. The main differences between the attribute and the non-attribute version can be found in the account opening and the spending phases (when attributes are revealed). The actual withdrawal protocol is essentially the same.

Setup. The TA (or another trusted authority) picks a group G of prime order q, generators $h, g, g_1, \ldots, g_n$ of this group, where n is the maximum number of attributes needed for the system, and a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The public key of the TA is $y = g^x$, where $x \in \mathbb{Z}_q$ is its secret key.

Account Opening. The account opening procedure of Brands with attributes is presented in Table 6.1.

When a user $\mathcal{U}$ wants to open an account with the TA, he first presents an identification document to the TA and then encodes his attributes (L_1, L_2, ..., $L_n \in \mathbb{Z}_q$) into a commitment I [1]. The value

[1] How the user obtains these attributes is out of the scope of this paper. Similar to a credential system the user

$\mathsf{TA}(y = g^x)$		$\mathcal{U}(pk_U = g_1^{L_1})$
		$I = g_1^{L_1} \dots g_n^{L_n}$ $\pi = PK\{(\Lambda_1, \dots, \Lambda_n) :$ $I/pk_U = g_2^{\Lambda_2} \dots g_n^{\Lambda_n} \wedge pk_U = g_1^{\Lambda_1}\}$
Verify π	$\xleftarrow{I, pk_U, \pi}$	If $Ih \neq 1$
Store identifying information of $\mathcal{U}$ together with I		
$z = (Ih)^x$	$\xrightarrow{z}$	Store z

Table 6.1: Brands' with Attributes Account Opening Protocol

L_1 (which is not revealed to the TA) serves as the user's secret key. His public key is $pk_U = g_1^{L_1}$. Then the user needs to provide a proof of knowledge π that he knows the opening of the commitment I and that the same value L_1 has been used to generate I and pk_U. The proof π can be computed as a standard Schnorr *AND proof* of knowledge of several discrete logarithms [131, 105]. Upon receiving π, I, pk_U, the TA checks the validity of the proof and then stores the user's information and computes $z = (Ih)^x$ which it sends to the user. The corresponding protocol for Brands without attributes only required the user to proof knowledge of his secret key.

Withdrawal. To withdraw k coins, $\mathcal{U}$ first has to identify himself to the TA. In this identification phase the user proves knowledge of his secret key and claims how many coins he wants to withdraw. These two actions can be achieved simultaneously by computing a Schnorr signature [131] $\sigma(m)$ on a message m of the form: $m =$"# of coins + time/date". The identification phase is presented in Table 6.2 and it is the same for the non-attribute version of Brands as well.

$\mathsf{TA}(y = g^x)$		$\mathcal{U}(pk_U = g_1^{L_1})$
		$m =$"# of coins + time/date"
Check σ_{pk_U}	$\xleftarrow{m, \sigma_{pk_U}(m)}$	Compute signature $\sigma_{pk_U}(m)$

Table 6.2: User Identification Phase

The TA will authorize the user to perform the withdrawal, if the signature validates under the user's public key and there are sufficient funds in his account. Then the user runs the withdrawal

could either reveal these attributes to the TA when committing to them or obtain those attributes from some other trusted authority and then just prove knowledge of them to the TA.

protocol k times, once for each coin he withdraws. Brands' withdrawal protocol, when supporting the encoding of attributes, is shown in Table 6.3.

$\mathsf{TA}(y = g^x)$		$\mathcal{U}(pk_U = g_1^{L_1})$
$w \in_{\mathcal{R}} \mathbb{Z}_q$		
$a = g^w,\ b = (Ih)^w$	$\xrightarrow{a,b}$	$s \in_{\mathcal{R}} \mathbb{Z}_q^*,\ A = (Ih)^s$
		$z' = z^s$
		$x_0, x_1, \ldots, x_n, u, v \in_{\mathcal{R}} \mathbb{Z}_q$
		$B = A^{x_0} g_1^{x_1} \ldots g_n^{x_n}$
		$a' = a^u g^v,\ \ b' = b^{su} A^v$
		$c' = \mathcal{H}(A, B, z', a', b')$
	$\xleftarrow{c}$	$c = c'/u \mod q$
$r = cx + w \mod q$	$\xrightarrow{r}$	$g^r \stackrel{?}{=} y^c a,\ (Ih)^r \stackrel{?}{=} z^c b$
		$r' = ru + v \mod q$

Table 6.3: Brands' with Attributes Withdrawal Protocol

For each coin, $\mathcal{U}$ needs to store the values: $A, B, \text{sign}(A, B) \hat{=} A, B, z', a', b', r'$ together with s and the values $x_0, x_1, \ldots, x_n$, where $\text{sign}(A, b)$ is essentially a blind Chaum-Pedersen signature [63]. In order to verify the signature one needs to check whether: $g^{r'} = y^{c'} a'$ and $A^{r'} = z'^{c'} b'$. Note that the basic difference of Brands withdrawal when using attributes is found in the computation of B since you need to pick as many random values x_i as the number of attributes in the scheme and compute $B = A^{x_0} g_1^{x_1} \ldots g_n^{x_n}$. Those random values x_i will be used later, during the spending phase in order for the user to prove knowledge of his attributes. For Brands withdrawal without attributes the value B is computed as $B = A^{x_0} g_1^{x_1}$: only one random value x_1 is required and it will be used for proving knowledge of user's secret key in the spending phase.

Spending. When $\mathcal{U}$ spends a coin to $\mathcal{M}$ (with identifying information $I_{\mathcal{M}}$) the spending protocol is executed. In Table 6.4 we present Brands spending protocol when supporting the encoding of but not revealing any attributes. During this phase the user presents the coin $A, B, \text{sign}(A, B)$ to $\mathcal{M}$ and also proves knowledge of the representation of A (i.e. $\mathcal{U}$ proves knowledge of his attributes). After the transaction and if the signature verifies (i.e. the coin is valid), $\mathcal{M}$ saves the payment transcript consisting of $A, B, \text{sign}(A, B)$, $(R, r_1, \ldots, r_n)$ and the time stamp date/time.

This protocol is significantly less efficient compared to Brands e-cash without attributes, since

the user needs to prove knowledge of the representation of A, which now includes $n+1$ exponents. For Brands without attributes the user only needs to compute r_1 to prove knowledge of his secret key.

$\mathcal{U}(pk_U = g_1^{L_1})$		$\mathcal{M}$
	$\xrightarrow{A,B,\text{sign}(A,B)}$	$A \stackrel{?}{\neq} 1$
$r_1 = -dL_1 + x_1 \mod q$	$\xleftarrow{d}$	$d = \mathcal{H}_0(A, B, I_{\mathcal{M}}, \text{date/time})$
$\dots$		
$r_n = -dL_n + x_n \mod q$		
$R = d/s + x_0 \mod q$	$\xrightarrow{(R,r_1,\dots,r_n)}$	$g_1^{r_1} \dots g_n^{r_n} h^{-d} \stackrel{?}{=} A^{-R} B$
		Verify $\text{sign}(A, B)$

Table 6.4: Brands' with Attributes Spending Protocol when not Revealing Attributes

What if $\mathcal{U}$ additionally wants to reveal an attribute, say L_j, during the spending protocol? For the transportation setting we assume that revealing attributes in clear is good enough and doesn't violate user's privacy given that for efficiency reasons we are only using binary attributes anyway. So in order to reveal L_j, $\mathcal{U}$ does not need to compute r_j, when receiving the value d from $\mathcal{M}$. Instead, he sends the attribute value L_j together with its blinding value x_j in his response to $\mathcal{M}$, which is shown in Table 6.5. Thus, revealing a larger number of attributes reduces the user-side's computation, but increases the amount of data that $\mathcal{U}$ sends to the payment machine $\mathcal{M}$.

$\mathcal{U}(pk_U = g_1^{L_1})$		$\mathcal{M}$
	$\xrightarrow{A,B,\text{sign}(A,B)}$	$A \stackrel{?}{\neq} 1$
	$\xleftarrow{d}$	$d = \mathcal{H}_0(A, B, I_{\mathcal{M}}, \text{date/time})$
$r_1 = -dL_1 + x_1 \mod q$		
$\dots$		
$r_{j-1} = -dL_{j-1} + x_{j-1} \mod q$		
$r_{j+1} = -dL_{j+1} + x_{j+1} \mod q$		
$\dots$		
$r_n = -dL_n + x_n \mod q$		
$R = d/s + x_0 \mod q$	$\xrightarrow{(R,r_1,\dots r_{j-1},r_{j+1},\dots,r_n)(L_j,x_j)}$	$g_1^{r_1} \dots g_j^{-dL_j+x_j} \dots g_n^{r_n} h^{-d}$
		$\stackrel{?}{=} A^{-R} B$
		Verify $\text{sign}(A, B)$

Table 6.5: Brands' with Attributes Spending Protocol when Revealing the Attribute L_j

Deposit. In order to deposit a coin, $\mathcal{M}$ submits the payment transcript to the TA. The TA first

checks the validity of the coin (i.e. it verifies sign(A, B) and checks whether $g_1^{r_1} \dots g_n^{r_n} h^{-d} \stackrel{?}{=} A^{-R} B$) and then queries the database, where all deposited coins are recorded, to check whether this coin had been deposited before. This double spending check does not need to happen during the deposit phase. The TA could run it at specific time intervals for all the coins in the database. If the deposited coin had not been recorded in the database before, the TA will store $(A, \text{date/time}, R, r_1, \dots, r_n)$ in her database. However, if the coin had been recorded, it means that it was spent twice by $\mathcal{U}$ (we assume that the payment machines in our system are trusted and will not try to submit the same coin twice: yet, this could easily be checked by storing the payment machine's identification $I_{\mathcal{M}}$ together with each coin that is stored). If a coin had been double spent the identity of the cheating user can be revealed by computing: $I = g_1^{(r_1 - r_1')/(R-R')} \dots g_n^{(r_n - r_n')/(R-R')}$ which was stored together with some identifying information of $\mathcal{U}$ during the account registration phase.

6.2.2 ACL E-cash with Attributes

In this section we explicitly describe how our ACL scheme can be used as e-cash with attributes. Note that Abe's blind signature scheme itself is immediately an e-cash scheme (without attributes though) and has been described in the past [3]. Thus, as mentioned above, we will not provide a detailed description of Abe's instead we will just point out the differences while describing ACL.

Setup. The setup phase of ACL e-cash is similar to the original ACL scheme. The TA chooses a group G of order q, a generator g and a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$. It also picks $z, h, h_0, h_1, h_2, \dots h_n \in_{\mathcal{R}} G$, where n is the maximum number of attributes. The secret key of the transportation authority is $x \in_{\mathcal{R}} \mathbb{Z}_q$ and the public key is: $y = g^x$. y will serve as the real public key, whereas z is the tag public key.

Account opening. When a user $\mathcal{U}$ with attributes $(L_2, \dots, L_n)$, secret key $L_1 \in_{\mathcal{R}} \mathbb{Z}_q$ and public key $pk_U = h_1^{L_1}$ wants to open an account at the TA he presents a valid identification document and commits to his attributes and public key, as shown in Table 6.6. For each User the TA stores: $pk_U, C/pk_U$ and a copy of his identification document. $\mathcal{U}$ also needs to store the randomness R that

corresponds to his commitment C.

$\mathsf{TA}(y = g^x, z)$		$\mathcal{U}(pk_U = h_1^{L_1})$
		$R \in_{\mathcal{R}} \mathbb{Z}_q$ $C = h_0^R h_1^{L_1} \prod_{i=2}^{n} h_i^{L_i}$ $\pi = PK\{(P, \Lambda_1, \ldots, \Lambda_n) :$
Check π Store identifying information of $\mathcal{U}$ together with C	$\xleftarrow{C/pk_U, pk_U, \pi}$	$C/pk_U = h_0^P h_2^{\Lambda_2} \ldots h_n^{\Lambda_n} \wedge pk_U = h_1^{\Lambda_1}\}$

Table 6.6: ACL with Attributes Account Opening Protocol

Withdrawal. To withdraw k coins from his account $\mathcal{U}$ first identifies himself to the transportation authority. This identification phase can be executed similar to the one in Brands' scheme (Table 6.2). Then $\mathcal{U}$ runs the ACL blind signature protocol k times (Table 6.7) once for every coin.

$\mathsf{TA}(y = g^x, z)$		$\mathcal{U}(pk_U = h_1^{L_1})$
$rnd \in_{\mathcal{R}} \mathbb{Z}_q$ $z_1 = Cg^{rnd}$, $z_2 = z/z_1$ $u, c', r_1', r_2' \in_{\mathcal{R}} \mathbb{Z}_q$ $a = g^u$, $a_1' = g^{r_1'} z_1^{c'}$, $a_2' = h^{r_2'} z_2^{c'}$	$\xrightarrow{rnd, a, a_1', a_2'}$	$z_1 = Cg^{rnd}$ $\gamma \in_{\mathcal{R}} \mathbb{Z}_q^*$, $\zeta = z^\gamma$ $\zeta_1 = z_1^\gamma$, $\zeta_2 = \zeta/\zeta_1$ $\tau \in_{\mathcal{R}} \mathbb{Z}_q$, $\eta = z^\tau$ Check whether $a, a_1', a_2' \in G$ $t_1, t_2, t_3, t_4, t_5 \in_{\mathcal{R}} \mathbb{Z}_q$ $\alpha = a g^{t_1} y^{t_2}$, $\alpha_1' = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4}$ $\alpha_2' = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4}$ $\varepsilon = \mathcal{H}(\zeta, \zeta_1, \alpha, \alpha_1', \alpha_2', \eta)$
$c = e - c' \mod q$	$\xleftarrow{e}$	$e = (\varepsilon - t_2 - t_4) \mod q$
$r = u - cx \mod q$	$\xrightarrow{c, r, c', r_1', r_2'}$	$\rho = r + t_1 \mod q$ $\omega = c + t_2 \mod q$ $\rho_1' = \gamma r_1' + t_3 \mod q$ $\rho_2' = \gamma r_2' + t_5 \mod q$ $\omega' = c' + t_4 \mod q$

Table 6.7: ACL with Attributes Withdrawal Protocol

After each execution of the withdrawal protocol $\mathcal{U}$ obtains a $coin = (\zeta, \zeta_1, \rho, \omega, \rho_1', \rho_2', \omega')$, which he stores together with rnd, τ and γ. Note that he omits to compute and store the value μ of the original ACL scheme. Instead, during the spending phase (Table 6.8), he computes μ_p to "bind" the coin to a specific transaction. For each withdrawn coin, the TA stores z_1 together with the public

key of $\mathcal{U}$ that this coin corresponds to (z_1 is going to be used to reveal the identity of the user in case of double-spending). The withdrawal of Abe's e-cash is essentially the same.

Spending. During the spending phase a user $\mathcal{U}$ simply releases the coin to the merchant together with ϵ_p and μ_p. The spending protocol for ACL e-cash and Abe's e-cash is identical and presented in Table 6.8.

$\mathcal{U}(pk_U = h_1^{L_1})$		$\mathcal{M}$
$\epsilon_p = \mathcal{H}(z^\tau, \text{coin}, \text{desc})$	$\xleftarrow{\text{desc}}$	$desc = \mathcal{H}(I_\mathcal{M}, \text{date/time})$
$\mu_p = \tau - \epsilon_p \gamma \mod q$	$\xrightarrow{\varepsilon_p, \mu_p, \text{coin}}$	$\zeta \stackrel{?}{\neq} 1$
		$\epsilon_p \stackrel{?}{=} \mathcal{H}_4(z^{\mu_p}\zeta^{\epsilon_p}, \text{coin}, \text{desc}) \mod q$
		$\omega + \omega'$
		$\stackrel{?}{=} \mathcal{H}(\zeta, \zeta_1, g^\rho y^\omega, g^{\rho'_1}\zeta_1^{\omega'}, h^{\rho'_2}\zeta_2^{\omega'}, z^{\mu_p}\zeta^{\epsilon_p}) \mod q$

Table 6.8: ACL with Attributes Spending Protocol

In order for a user $\mathcal{U}$ to also reveal an attribute L_j during spending, he will have to execute the Revealing Attribute L_j protocol (Table 6.9) additionally to the Spending protocol. In this protocol he needs to prove that the attributes on his initial commitment correspond to the attributes encoded in the withdrawn coin (i.e. in value ζ_1). In other words, the user needs to provide a proof of equality of committed values under different commitment keys and bases [105], for a new commitment C' that the user computes for the attribute L_j and ζ_1 in which the attributes are encoded.

From a theoretical point of view the user side of the spending protocol of ACL (Table 6.8), when supporting the encoding but not revealing attributes, is more efficient than the one of Brands' scheme, since, following the trick that was first suggested by Abe [3], the user only needs to compute the updated μ_p value for the coin verification instead of proving knowledge of all his attributes. In practice this depends on the relative cost for executing modular arithmetic in $\mathbb{Z}_q$ compared to the hash function and the communication cost.

Deposit. During deposit, the payment machine $\mathcal{M}$ sends the coin as well as ε_p, μ_p, $desc$ and the date and time of the transaction to the TA which verifies that both the coin is valid and $desc$ correctly encodes date/time and I_M. Double spending can be checked later in an off-line fashion. In order to

$\mathcal{U}(pk_U = h_1^{L_1})$		$\mathcal{M}$
recall $\zeta_1 = h_0^{R\gamma} h_1^{L_1\gamma} \dots h_n^{L_n\gamma} g^{rnd\gamma}$		
$rnd' \in_R \mathbb{Z}_q$		
$C' = h_j^{L_j\gamma} g^{rnd'\gamma}$		
$= h_j^{L_j\gamma} 1^{R\gamma} 1^{L_1\gamma} \dots 1^{L_{j-1}\gamma} 1^{L_{j+1}\gamma} \dots 1^{L_n\gamma} g^{rnd'\gamma}$		
$r, r', r_0, \dots, r_n \in_R \mathbb{Z}_q$		
$\widetilde{\zeta_1} = h_0^{r_0} \dots h_n^{r_n} g^r$		
$\widetilde{C'} = 1^{r_0} 1^{r_1} \dots h_j^{r_j} \dots 1^{r_n} g^{r'}$		
$c = \mathcal{H}(\zeta_1, \widetilde{\zeta_1}, C', \widetilde{C'}, \text{date/time})$		
$s_0 = r_0 + cR\gamma$		
$s_1 = r_1 + cL_1\gamma$		
$\dots$		
$s_n = r_n + cL_n\gamma$		
$s = r + c\ rnd\ \gamma$		
$s' = r' + c\ rnd'\ \gamma$	$\xrightarrow{L_j, \zeta_1, \widetilde{\zeta_1}, C', \widetilde{C'}, s_0, \dots, s_n, s, s'}$	$c = \mathcal{H}(\zeta_1, \widetilde{\zeta_1}, C', \widetilde{C'}, \text{date/time})$
		$\widetilde{\zeta_1}\zeta_1^c \stackrel{?}{=} h_0^{s_0} \dots h_n^{s_n} g^s$
		$\widetilde{C'} C'^c \stackrel{?}{=} 1^{s_0} 1^{s_1} \dots h_j^{s_j} \dots 1^{s_n} g^{s'}$

Table 6.9: ACL with Attributes Revealing the Attribute L_j Protocol

do that the TA needs to check whether a coin has been deposited with two different *desc* and *desc*$'$. In this case we will have (ε_p, μ_p) and (ε'_p, μ'_p) and the TA can calculate: $\gamma = (\mu'_p - \mu_p)/(\epsilon_p - \epsilon'_p)$ and $z_1 = \zeta_1^{1/\gamma}$, the TA can find the user to whom z_1 was given and "punish" him. A useful observation is that in the ACL scheme, when a user is found cheating, his identity can be revealed without the TA learning his secret key and attribute values. From a privacy point of view this is much better, since the user does not need to form a new secret key and commit to his attributes all over again, after getting "punished" by the TA .

6.3 Framework Implementation

In our measurement setup the terminal, which represents the vending machines as well as the turnstiles, is composed of a personal computer and an OMNIKEY smart card reader from HID Global that is connected to the computer via USB. The payment device of the user is represented by a BlackBerry Bold 9900, featuring NFC-capabilities. The BlackBerry Bold 9900 is programmed

using the BlackBerry Java SDK API 7.1.0[2] provided by RIM.

We base the schemes on Elliptic Curve Cryptography. Thus, both the smartphone and the terminal had to be provided with the ECC and the NFC functionality. Implementation aspects of these frameworks will be described in the following.

6.3.1 Near Field Communication (NFC) Framework

All aspects of NFC are specified in ISO/IEC standards. We use the so-called card-emulation mode provided by the BlackBerry API, in which the NFC-smartphone emulates a standard-conform smart card. Building the payment system on standard appliances, makes it conform to already installed payment infrastructure and hence facilitates deployment. The underlying standard of the card emulation mode is ISO/IEC 14443-A. This standard describes the communication signal interface of contactless smart cards, operating at 13.56 MHz with a bandwidth of 106 kbit/s. Both the Java SDK API 7.1.0 of the BlackBerry device, and the JRE 6 System Library of the terminal support this standard.

Data is exchanged between the terminal and the smart card using so called Application Protocol Data Units (APDUs). The reader initializes the communication by sending a command APDU to the smart card. The smart card executes this command and replies with a response APDU. This communication procedure is specified in standard ISO/IEC 7816-4. Note that the size of an APDU is limited to 256 bytes, which impacts the execution time of the protocols, as will be discussed further in Section 6.4.

6.3.2 Cryptographic Framework

We deduce from [34] that a 160-bit elliptic curve presents sufficient security for a micro-payment system. We chose the standardized curve *secp160r1* from [52]. This curve is based on a 160-bit prime field $\mathbb{Z}_p$. The modulus of $\mathbb{Z}_p$ is a generalized Mersenne prime, namely $p = 2^{160} - 2^{31} - 1$, which

[2] http://www.blackberry.com/developers/docs/7.1.0api/

allows for an efficient implementation of the reduction, and thus leads to an efficient implementation of the curve arithmetic. On the terminal side we use the Bouncy Castle Crypto Library version 1.5[3]. This library provides a general elliptic curve framework that allows the use of many different curves. A dedicated implementation of the elliptic curve functionality for the terminal's hardware could lead to a better performance of the execution of the payment schemes and is realistic in the transportation setting. However, our investigations focus on the execution of the protocols on the user device and the communication of the protocols, which is why we chose to use a standard library for the terminal side's implementation.

The data formats required for transmission as well as the two different finite algebraic structures involved in Elliptic Curve Cryptography force us to use different data types. Since the size of the byte-array representation of the integer values can be shorter than the designated 20 or 21 bytes, we have to pad with leading `0x00`. It is important to consider the property of signed/unsigned for the classes CryptoInteger (on the BlackBerry) and BigInteger (on the terminal). The conversion of a CryptoInteger to byte-array to a BigInteger variable has to be bijective. We had to take care of this in detail, because BigInteger is a signed variable and CryptoInteger an unsigned. If the source is the CryptoInteger class and the value has a leading 1 in the binary representation, then the BigInteger constructor will interpret it as a negative number that makes it $\mod p$ or $\mod n$ to a different element. The solution is to always put a leading `0x00` in front of the array before constructing a BigInteger.

The BlackBerry API 7.1.0. supports the chosen curve. As such, an implementation of the ECDH key agreement based on this curve is provided by the API. Yet, the BlackBerry API does not implement all functionality necessary for the implementation of the proposed e-cash schemes, and hence had to be extended. Point addition and doubling were implemented in Java making use of the modular arithmetic functionality provided in the BlackBerry API. The implementation method to execute the scalar multiplication efficiently is described in detail in the following. Note,

[3]http://www.bouncycastle.org/

the description is ajar on curve *secp160r1*, but could easily be adapt to the following Certicom SEC2 [52] curves which are supported by BlackBerry API since version 3.6.0: *SECP192R1*, *SECP224R1*, *SECP256R1*, *SECP384R1*, *SECP521R1*, *SECT163K1*, *SECT163R1*, *SECT163R2*, *SECT233K1*, *SECT233R1*, *SECT239K1*, *SECT283K1*, *SECT283R1*, *SECT409K1*, *SECT409R1*, *SECT571K1*, and *SECT571R1*.

6.3.3 Efficient Execution of EC Scalar Multiplication Using the ECDH Key Agreement

An implementation of the scalar multiplication $\boldsymbol{Q_k} = k \cdot \boldsymbol{P}$ in Java, making use of the modular arithmetic functionality provided in the BlackBerry API, leads to an execution time for the scalar multiplication of about 141 ms. Fortunately, the API contains the ECDHKeyAgreement class. This class offers the method *generateSharedSecret*, which executes a scalar multiplication of the input point $\boldsymbol{P} = (x_P, y_P)$ with the input scalar k. This method executes in 1 ms, but only returns the x-coordinate x_{Q_k} of the resulting point $\boldsymbol{Q_k}$. In the protocols of the considered payment schemes multiple point multiplications have to be executed. Hence, knowledge of the y-coordinate y_{Q_k} of $\boldsymbol{Q_k}$ is essential for further computations.

This drawback can be overcome. Going from the short Weierstrass equation ($y^2 = x^3 + ax + b$), on which the chosen elliptic curve is based, the magnitude of y_{Q_k} can be calculated from x_{Q_k} using Equation 6.1. Algorithm 3.36 in [106] describes how to calculate the square root in $\mathbb{Z}_p$, if $p \equiv 3 \mod 4$, which holds for the chosen curve.

$$\pm\, y_{Q_k} = \sqrt{x_{Q_k}^3 + a x_{Q_k} + b} \mod p \tag{6.1}$$

This results in two options for the resulting point $\boldsymbol{Q_k}$:

$$\boldsymbol{Q_k} = (x_{Q_k}, \pm y_{Q_k}) \begin{cases} \boldsymbol{Q_k^{(+)}} & = (x_{Q_k}, +y_{Q_k}) \\ \boldsymbol{Q_k^{(-)}} & = (x_{Q_k}, -y_{Q_k}) \end{cases} \tag{6.2}$$

To choose the correct option ($\boldsymbol{Q_k^{(+)}}$ or $\boldsymbol{Q_k^{(-)}}$) for $\boldsymbol{Q_k}$ we verify y_{Q_k} over the coherence

$$\boldsymbol{Q_{k+1}} = \boldsymbol{Q_k} + \boldsymbol{P} = (k+1) \cdot \boldsymbol{P}.$$

We pick the positive result $+y_{Q_k}$ and calculate the x-coordinate of $\boldsymbol{Q_{k+1}^{(+)}}$ by adding $\boldsymbol{Q_k^{(+)}}$ and $\boldsymbol{P}$, using the group law for point addition on an elliptic curve [92]

$$x_{Q_{k+1}}^{(+)} = \left(\frac{+y_{Q_k} - y_P}{x_{Q_k} - x_P}\right)^2 - x_P - x_{Q_k} \bmod p. \tag{6.3}$$

Then we check whether the result is equal to the x-coordinate of $\boldsymbol{Q_{k+1}}$ returned when executing the *generateSharedSecret* function on $(k+1)$ and $\boldsymbol{P}$. While this algorithm, which is summarized as Algorithm 4.1, executes the generateSharedSecret method twice and calculates a square root in the prime field $\mathbb{Z}_p$, it still achieves a major speed-up in execution time, when compared to the Java implementation based on the arithmetic functionality that is provided in the BlackBerry API, i.e. yields an execution time of around 4 ms.

6.4 Implementation Results

The time critical phases of the e-cash schemes are the withdrawal and especially the spending phase, as those have to be executed frequently, whereas the account opening only happens once for each user. We limit the discussion of our results to those time critical parts.

The results for the execution of the withdrawal phase for all schemes are presented in Table 6.10, and the results for the spending phase in Table 6.11. We present two cases: I) Brands' e-cash

Algorithm 6.3.1: Recovering the y-coordinate, when using the ECDH key agreement class for point multiplication

Data: input point P, input scalar k
Result: x- coordinate and y-coordinate of resulting point $Q = (x_Q, y_Q)$

begin
  $x_{Q_k} \leftarrow \text{generateSharedSecret}(k, P)$
  $x_{Q_{k+1}} \leftarrow \text{generateSharedSecret}((k+1), P)$
  $\pm y_{Q_k} = \sqrt{x_{Q_k}^3 + a \cdot x_{Q_k} + b} \mod p$
  $x_{Q_{k+1}}^{(+)} = (\frac{+y_{Q_k} - y_P}{x_{Q_k} - x_P})^2 - x_P - x_{Q_k} \mod p$
  if $x_{Q_{k+1}} == x_{Q_{k+1}}^{(+)}$ **then**
    return $(x_{Q_k}, +y_{Q_k})$;
  else
    return $(x_{Q_k}, -y_{Q_k})$;

scheme, when not allowing the encoding of any attributes, and Abe's scheme, which also does not allow the encoding of attributes, and II) Brands' scheme and ACL when allowing the encoding of two attributes and revealing both of them. Note that the private key of the user is not counted as an attribute. The user encodes two attributes, L_2 and L_3, additionally to his private key L_1. Of course, our implementation could support a bigger number of attributes if the transportation system requires so, but keep in mind that there is a trade-off between the number of attributes and the spending time.

Scheme		Execution time in milliseconds			
		Terminal	Communication	Smartphone	Total
I) Without attributes	Brands	66.1	45.1	123.8	235
	Abe	93.6	69.6	137.5	301
II) With attributes	Brands	73.2	44.1	128.7	246
	ACL	93.6	69.9	137.5	301

Table 6.10: Execution time of withdrawal per coin for I) Brands and Abe not supporting attributes and II) Brands and ACL supporting the encoding of and revealing 2 attributes.

Scheme		Execution time in milliseconds			
		Terminal	Communication	Smartphone	Total
I) Without attributes	Brands	58.8	96.8	1.4	157
	Abe	79.3	81.0	10.7	171
II) With attributes	Brands	87.3	114.8	2.0	204
	ACL	151.2	221.4	11.4	384

Table 6.11: Execution time of spending per coin for I) Brands and Abe not supporting attributes and II) Brands and ACL supporting the encoding of and revealing 2 attributes.

Figures 6.1 and 6.2 illustrate those results, where the execution times of the different protocols have been summarized to: *Terminal* all computation executed on the terminal side, *Communication* execution time of the entire communication, and *Smartphone* all computation executed on the BlackBerry smartphone. In our implementation all steps are executed serially, i.e. while waiting for the terminal the execution on the smartphone is suspended. This resembles the execution on a standard smart card. Due to the extended capabilities of the smartphone computations on the smartphone and the terminal could be parallelized, which would lower the total execution time. Thus what is summarized as the total time is an upper limit of the execution time of the protocols.

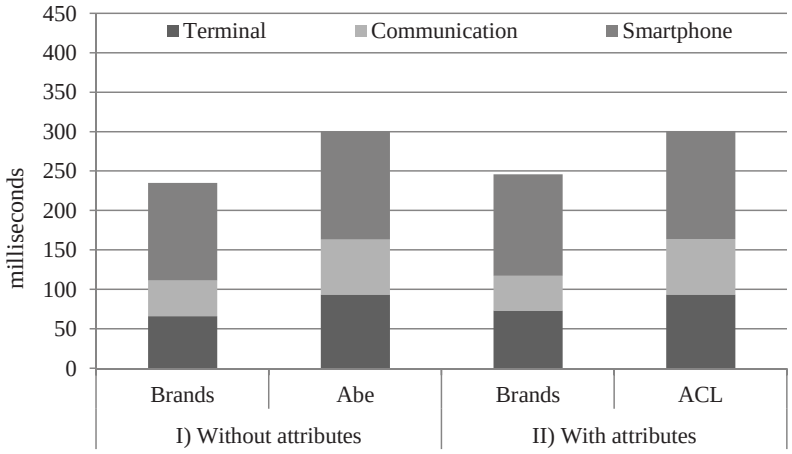

Figure 6.1: Execution times of withdrawal protocols for the cases I) not supporting attributes and II) supporting the encoding and revealing 2 attributes.

An advantage of the ACL scheme is that for the revealing attributes phase (Table 6.9) the values $C', \widetilde{\zeta_1}, \widetilde{C'}$ can be precomputed, which has been realized for the implementation at hand. The computation time for those precomputed values is 39 ms and is not included in the results. By doing so the total execution time for spending a coin of all schemes does not exceed 400 ms, which is the acceptance threshold for spendings in the transportation domain.

While the terminal side is represented by a powerful computer the execution of a scalar multiplication on the terminal takes longer than on the BlackBerry device; on the BlackBerry an execution of the point multiplication takes 4 ms, whereas on the computer it takes 6 ms. As mentioned in Section 6.3 we focus on the execution time on the payment device and of the communication.

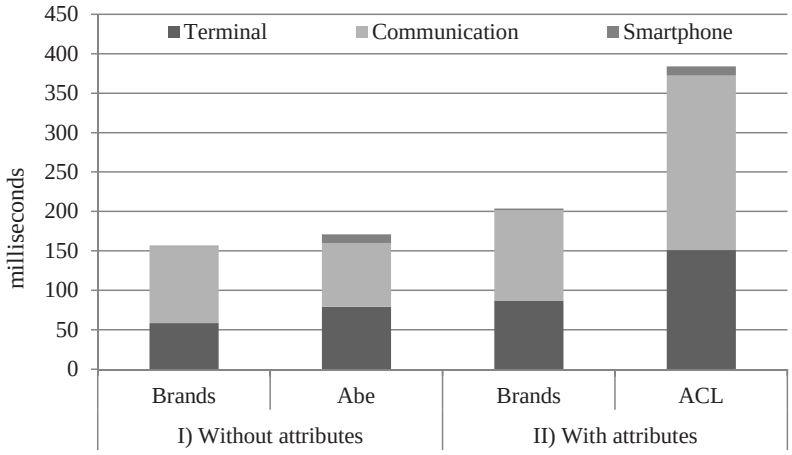

Figure 6.2: Execution times of spending protocols for the cases I) not supporting attributes and II) supporting the encoding and revealing 2 attributes.

The implementation results could be improved when not relying on a Java implementation for the terminal side, which is a realistic scenario, since the TA has full control over those devices.

Surprisingly, a limiting factor for the execution of the different protocols is the card emulation mode supported by the BlackBerry device. The communication bandwidth is limited to 106 kbits/sec, while the maximum bandwidth supported by the NFC standard is 424 kbits/sec. An additional deceleration limits the practical bandwidth on the application layer to 62.5 kbit/s. Since the communication plays an integral part in the execution of the spending protocol, a faster communication could significantly improve the execution timings, as easily can be verified by looking at Figures 6.1 and 6.2. Moreover the length of an APDU is limited to 256 bytes. For some protocol steps data had to be sent by two APDUs. Hence, the overall execution time could be improved when allowing longer APDUs.

A further observation is that the computational complexity of Brands' spending protocol, when allowing attributes, decreases the more of them are revealed. Yet, at the same time the data to be communicated increases. For our implementation the increase in data that needs to be communicated, dominates the change in execution time. This could be different for other platforms, where the communication plays a less important role in the overall execution time of the protocol.

Table 6.12 shows the coin size for each of the schemes, i.e. the data that needs to be stored on the

user device for each coin. Since for the ACL scheme $C', \widetilde{\zeta_1}, \widetilde{C'}$ have been precomputed, they need to be stored on the device as part of the coin. If storage space would be more critical in comparison to the communication, those values could be computed on-the-fly when spending a coin, which would lead to the same storage amount for a coin as in Abe's scheme, but longer execution times.

Scheme		Coin Elements	Coin Size in bytes
I) Without attributes	Brands	$A, B, z', a', b', r', s, x_0, x_1$	289
	Abe	$\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \tau, \gamma$	229
II) With attributes	Brands	$A, B, z', a', b', r', s, x_0, x_1, x_2, x_3$	331
	ACL	$\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \tau, \gamma, C', \widetilde{\zeta_1}, \widetilde{C'}$	352

Table 6.12: Coin size in byte for the cases I) Brands and Abe not supporting attributes and II) Brands and ACL supporting the encoding of 2 attributes.

The implementation shows that it is feasible to spend a coin meeting the extreme time constraints of the transportation setting. Spending several coins serially exceeds these time constraints. Yet, the execution time could be further reduced by batching the executions that are required for spending a coin.

6.5 Related Work

Privacy Preserving Payment Schemes. Sadeghi et al. [129] and Blass et al. [28] proposed RFID-based, privacy-preserving e-ticket schemes for transit applications which are limited to only protect the users' privacy against outsiders and not against the transportation authority. Pirker et al. described how to make use of certain hardware capabilities of some mobile phones to build a secure, NFC-based, privacy-preserving, prepaid payment system [122], but the system requires the devices accepting a payment to be online at all times. Heydt-Benjamin et al. [95] proposed the use of recent advances in anonymous credentials and e-cash to design offline privacy-preserving public transportation payment systems. They consider a hybrid system with two kinds of tickets: passive RFID transponders and embedded systems such as cell phones.

E-Cash Implementations. Implementations of anonymous credentials on Java cards have been presented in [26] and [18]. Hinterwälder et al. presented an implementation of Brands' e-cash scheme for a computational RFID tag [96]. It is shown that it is feasible to execute the spending part efficiently on the tag, while the execution of the withdrawal part is still problematic. Derler et al. [72] implemented an NFC-based mobile ticketing system, which is based on Brands' private credential scheme. Execution times of several seconds are achieved, which can be a limiting factor in some application settings. Similarly, [64] presented a PDA implementation of an offline e-cash scheme that is based on Brands', which achieves an execution time of several seconds for the withdrawal phase.

6.6 Conclusions

In this Chapter we discussed the benefits of using e-cash with attributes in the public transportation setting. We presented an explicit description of e-cash with attributes for both Brands' [36] and ACL [14] schemes. Further we showed a full implementation of Brands' with and without attributes, Abe's and ACL on a BlackBerry Bold 9900. We proposed a method that allows the use of the ECDHKeyAgreement class of the BlackBerry API to calculate the point multiplication, by recovering the y-coordinate of the resulting point, which led to transaction times that meet real-world requirements of transportation payment systems for all considered schemes. Our results are very promising: first of all we provide the *first efficient and practical implementation of e-cash in smartphones*, and moreover, we show that a provable secure e-cash scheme like ACL is actually practical and has comparable running time to those of schemes without rigorous security proofs.

7 Transferable E-cash

As discussed in Chapter 6, in traditional e-cash users can only transfer their coins to merchants, who must then deposit the coin at the Bank. It would be natural to allow users to transfer coins to other users (which might be merchants or not), who should be able to transfer the received coins, and so on. Moreover, it would be desirable if these transfers could be done without being connected to the bank, i.e., offline. One of the main advantages of such a transferability property is that it would decrease the communication cost between the Bank and the users. Moreover, it would allow to implement more real world scenarios: Consider the example of coins of different denominations. A store, which is offline, wants to give back change to a customer, using coins previously received. In order to do so, coins need to be transferable multiple times. Transferability of e-cash was proposed in the 1990s and the desired security properties have been analyzed; however, so far no schemes have been proposed that are both practical and satisfy the proposed security and privacy requirements. In this Chapter we present the *first efficient and fully anonymous* transferable e-cash scheme [10]. We first provide a formal treatment of the security and anonymity properties of transferable e-cash by giving game-based definitions. We then proposed a practical construction based on malleable signatures that does not assume any trusted party. Finally, we present an efficient double-spending

detection mechanism, which is independent of our scheme and could be used by other transferable e-cash constructions.

7.1 Defining Transferable E-Cash

In this Section we define anonymous and secure transferable e-cash. The first definitions for transferable e-cash were given in [50] and then modified in [29] for the scenario that there is a trusted judge. We strengthen those older definitions in several aspects and in particular, we introduce a stronger anonymity notion.

In a transferable e-cash scheme there are two types of parties: the bank $\mathcal{B}$ and users $\mathcal{U}_i$. Coins are denoted by c and each coin is uniquely identifiable via a serial number $\mathtt{SN}$, which will be retrieved by the bank during deposit to check for double spending. We let $\mathcal{CL}$ denote the list of deposited coins; if multiple coins with the same serial number were deposited, we keep all of them in $\mathcal{CL}$. Note that in contrast to previous versions of these definitions we add a protocol for user registration and we also merge the `Deposit` and `Identify` protocols.

A transferable e-cash scheme consists of the following algorithms (which are probabilistic unless otherwise stated):

`ParamGen`(1^λ) on input the security parameter λ outputs the system parameters par (we assume that λ can be deduced from par, which is a default input to the remaining algorithms).

`BKeyGen`() and `UKeyGen`() are executed by $\mathcal{B}$ and a user $\mathcal{U}$ respectively and output $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ and $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$. The bank's key $sk_{\mathcal{B}}$ might be divided into two parts: $sk_{\mathcal{W}}$ for the registration and withdrawal phase and $sk_{\mathcal{D}}$ for the deposit phase. Thus, we define two extra algorithms `WKeyGen`() and `DKeyGen`() for the bank's key generation. We also assume that during the bank key generation the list, $\mathcal{CL}$, is initialized to be empty.

`Registration`$(\mathcal{B}[sk_{\mathcal{W}}, pk_{\mathcal{U}}], \mathcal{U}[sk_{\mathcal{U}}, pk_{\mathcal{B}}])$ is an interactive protocol between $\mathcal{B}$ and a user. At the end the user receives a certificate $cert_{\mathcal{U}}$; both parties output either ok or $\perp$ in case of error.

`Withdraw`($\mathcal{B}[sk_{\mathcal{W}}, pk_{\mathcal{U}}]$,$\mathcal{U}[sk_{\mathcal{U}}, pk_B]$) is a protocol between $\mathcal{B}$ and a user. At the end the user either receives a coin c and outputs ok or outputs $\perp$. $\mathcal{B}$'s output is its view $\mathcal{V}_{\mathcal{B}}^{W}$ of the protocol or $\perp$.

`Spend`($\mathcal{U}_1[c, sk_{\mathcal{U}_1}, cert_{\mathcal{U}_1}, pk_{\mathcal{B}}]$,$\mathcal{U}_2[sk_{\mathcal{U}_2}, pk_{\mathcal{B}}]$) is a protocol in which $\mathcal{U}_1$ spends/ transfers the coin c to $\mathcal{U}_2$. At the end, $\mathcal{U}_2$ either outputs a coin c' and ok or it outputs $\perp$; $\mathcal{U}_1$ either marks the coin c as spent and outputs ok, or it outputs $\perp$ in case of error.

`Deposit`($\mathcal{U}[c, sk_{\mathcal{U}}, cert_{\mathcal{U}}, pk_{\mathcal{B}}]$, $\mathcal{B}[sk_{\mathcal{D}}, pk_{\mathcal{U}}, \mathcal{CL}]$) is a protocol where a user $\mathcal{U}$ deposits a coin c at the bank. We split the deposit protocol into two subroutines. First `CheckCoin` checks whether the coin c is consistent, and if not outputs $\perp$. Else, the subroutine `CheckDS` is run by $\mathcal{B}$, which outputs the serial number `SN` of the deposited coin. $\mathcal{B}$ checks whether $\mathcal{CL}$ already contains an entry for `SN`. If not, $\mathcal{B}$ includes `SN` in $\mathcal{CL}$, credits $\mathcal{U}$'s account and returns "success" and $\mathcal{CL}$. Otherwise, the coin was double-spent: the subroutine `DetectDS` is run, which outputs $(pk_{\mathcal{U}}, \Pi)$, where $pk_{\mathcal{U}}$ is the public key of the user being accused, and Π is a proof that $pk_{\mathcal{U}}$ is the registered key of a user that double spent the coin. Note that Π reveals nothing about the coin itself.

`VerifyGuilt`($pk_{\mathcal{U}}, \Pi$) is a deterministic algorithm that can be executed by anyone. It outputs 1 if the proof verifies and 0 otherwise.

`VerifyDSCounter`($pk_{\mathcal{U}}, t, \Pi'$) is a deterministic algorithm only required in order to achieve stronger flavors of exculpability. On input a user's public key, the number t of double spends the user is accused of, and a proof Π', it outputs 0 or 1. We can assume that the output proof Π of `Deposit` contains Π' and t.

Notice that in our definition a transferable e-cash scheme is *stateless* since there is no common state information shared between the algorithms. This means that a coin withdrawn will be the same, whether it was the first or the n^{th} coin the bank issues to a specific user. Moreover, when a user receives a coin from another user then the transferred coin will only depend on the original coin (not on other coins received by $\mathcal{U}_2$ or coins transferred by $\mathcal{U}_1$). Thus, the bank and the users do not need

to remember anything about past transactions—for transfer the information already stored in the coin must be sufficient.

7.1.1 Global Variables and Oracles

In order to formally define the security properties of transferable e-cash, we first define some global variables and oracles.

Global variables. In the user list, UL, we store all the information about users, keys and certificates. Its entries are of the form $(i, pk_i, sk_i, cert_i, uds_i)$, where uds indicates how many times this user double-spent (this counter will be useful for the exculpability definition). If user i is corrupted then $sk_i = \perp$; if it has not been registered then $cert_i = \perp$. We keep a counter n of the total number of generated/registered users which is initialized to 0.

In the coin list, $\mathcal{CL}$, we keep information about the coins created in the system. For each *original* coin withdrawn we store a tuple $(j, owner, c, fc, fd, cds, origin)$, where j is its index in $\mathcal{CL}$, *owner* stores the index i of the user who withdrew the coin[1] and c is the coin itself. The flag fc indicates whether the coin has been corrupted[2] and the flag fd indicates whether the coin has been deposited. We also keep a counter, cds, of how many times this *specific instance* of the coin has been spent, which is initialized as $cds = 0$. In *origin* we write "$\mathcal{B}$" if the coin was issued by the honest bank and "$\mathcal{A}$" if the adversary issued it.

After a coin was transferred, we add a new entry to $\mathcal{CL}$ of the following format: $(j, owner, c, cds, pointer)$, where position in $\mathcal{CL}$, *owner* shows the current owner, c is the new, transferred coin and cds indicates how many times the coin has been spent. In *pointer* we store a pointer j' indicating which *original* coin this *transferred* coin corresponds to. Once a *transferred* coin is deposited or corrupted, we mark the *original* coin's flags fc, fd appropriately.

We keep a counter $\mathcal{AC}$ that shows the total number of *corrupted* coins (which should be equal

[1] We do not store the coins withdrawn by the adversary.

[2] A *corrupted coin* is defined as a coin that was under the adversary's control at some point. Once a coin is flagged as *corrupted*, it cannot be "un-flagged", even if it is later under the control of an honest user.

to the number of original coins that are marked as corrupted in $\mathcal{CL}$). We also define the list $\mathcal{DCL}$, where we store the set of deposited coins.

Finally, note that for the description of the oracles below we assume that if the corresponding algorithm fails (outputs $\perp$), then the oracle also stops. Otherwise the call to the oracle is considered to be *successful* (except for the oracles used for deposit where a successful call is one that also didn't detect any double spending).

Creation, registration and corruption of users. The adversary can instruct the creation of honest users, can play the role of the Bank during registration, passively observe registration or corrupt users:

`Create`() sets $n = n + 1$, executes $(sk_n, pk_n) \leftarrow$ `UKeyGen`() and sets $UL[n] = (n, pk_n, sk_n, \perp, 0)$ and outputs pk_n.

`BRegister`(pk) plays the bank side of the `Register` protocol and interacts with $\mathcal{A}$. Let *cert* be the generated certificate. If $pk \notin UL$ then set $n = n + 1$ and $UL[n] = (n, pk, \perp, cert, 0)$, else abort.

`URegister`(i), for $i \leq n$, plays the user side of the `Register` protocol and adds *cert* to the corresponding field of UL.

`Register`(i), for $i \leq n$, simulates both sides of the `Register` protocol. If user i was not registered then add *cert* to the corresponding field of UL.

`Corrupt`(i, S), for $i \leq n$, allows the adversary to corrupt a subset, S, of user i's coins. If $sk_i = \perp$ (i.e. this user is already corrupted) then abort. The set S consists of coin indices in $\mathcal{CL}$. For every $j \in S$ look in $\mathcal{CL}$ and if $owner \neq i$ then ignore this coin and remove it from S. The oracle first outputs sk_i and then updates UL by setting $sk_i = \perp$ to mark this user as corrupted. Then, the *non-corrupted* coins in the set S are given to the adversary $\mathcal{A}$ and the $\mathcal{AC}$ counter increases accordingly. The coins that are given to $\mathcal{A}$ are marked as corrupted i.e. we set the fc flag of the corresponding *original* coin to $fc = 1$. Note that if $\mathcal{A}$ tries to corrupt unregistered

users, this doesn't give him any extra power. Also, once a user is corrupted he is considered to be an *adversarial* user and thus $\mathcal{A}$ will be running instead of him[3].

Withdrawal oracles. The adversary can either withdraw a coin from a trusted Bank, play the role of the Bank or passively observe a withdrawal.

`BWith`() plays the bank side of the `Withdraw` protocol. It increases the counter of adversarial coins $\mathcal{AC} = \mathcal{AC} + 1$. Note that coins belonging to $\mathcal{A}$ are not added to the list $\mathcal{CL}$.

`UWith`(i) plays user i in a `Withdraw` protocol. It adds the entry $(j, owner = i, c, fc = 0, fd = 0, cds = 0, origin = \mathcal{A})$ to the coin list $\mathcal{CL}$.

`With`(i) simulates a complete `Withdraw` protocol playing both $\mathcal{B}$ and user i. It adds $(j, owner = i, c, 0, 0, 0, \mathcal{B})$ to $\mathcal{CL}$ and outputs the transcript.

Spend and deposit oracles.

`Rcv`(i) allows $\mathcal{A}$ to spend a coin to user i. The oracle plays the role of $\mathcal{U}_2$ with the secret key of user i in the `Spend` protocol. A new entry $(j, owner = i, c, fc = 1, fd = 0, cds = 0, origin = \mathcal{A})$ is added to $\mathcal{CL}$.

`Spd`(j) enables $\mathcal{A}$ to receive coin number j in $\mathcal{CL}$. If the coin belongs to a corrupted user it aborts. Otherwise, it plays the role of user $\mathcal{U}_1$ in the `Spend` protocol with the secret key of the owner i of the coin j in $\mathcal{CL}$. It increases the coin spend counter cds of entry j in $\mathcal{CL}$ by 1. If cds was already greater than zero (i.e., this specific user has already spent this coin) then the double spending counter, uds, of the owner of coin j is increased by one.

- If the coin has been deposited, i.e. $fd = 1$, or
- if the coin is marked as corrupted, i.e. $fc = 1$, then do not increase $\mathcal{AC}$;
- else set $\mathcal{AC} = \mathcal{AC} + 1$.

[3] This means that $\mathcal{A}$ cannot run honest user oracles on corrupted users: i.e. cannot run oracles `With, UWith, Rcv,S&R`.

Finally, whenever a coin is received by $\mathcal{A}$, we mark the *original* instance of this coin as corrupted, i.e., we set $fc = 1$. This ensures that in case an honest user double-spent a coin and the coin reached the adversary more than once, it will not be added to $\mathcal{AC}$ multiple times.

S&R(i, j) is the Spend-and-Receive oracle that allows $\mathcal{A}$ to passively observe spending of coin j by its owner to user i (who must not be corrupted). It increases n by 1 and adds $(n, owner = i, c, cds = 0, pointer)$ to $\mathcal{CL}$, where $pointer = pointer_j$ if $pointer_j \notin \{\mathcal{A}, \mathcal{B}\}$, else $pointer = j$. It also increases the coin spend counter cds_j in entry j by 1. If cds_j was already greater than zero then the double spending counter uds of user i_1 is also increased by 1.

BDepo() interacts with $\mathcal{A}$ playing the role of the bank during Deposit. It updates $\mathcal{DCL}$ accordingly.

UDepo(j) simulates the role of the owner (who must not be corrupted) of coin j in the Deposit protocol, interacting with the adversary playing the bank. It increases the spend counter cds_j in entry j in $\mathcal{CL}$ by 1. If cds_j was already greater than zero then the double spending counter uds of the owner of coin j is increased by 1. It also marks $fd = 1$ for the original coin.

Depo(j) simulates a Deposit of coin j between the bank and the owner of j (who must not be corrupted). It increases cds_j in entry j of $\mathcal{CL}$ by 1. If cds_j was already greater than zero then uds of the owner of coin j is increased by one. It also marks $fd = 1$ in the original coin and adds the coin to $\mathcal{DCL}$.

Let size(c) be a function that outputs the size of a coin. A coin just withdrawn from the bank has size 1 and after a transfer the size increases by 1. We say that coins c_1 and c_2 are *compatible*, comp$(c_1, c_2) = 1$, if size(c_1) = size(c_2). We need this property, since transferred coins necessarily grow in size [62] and thus an adversary may break anonymity by distinguishing coins of different sizes.

7.2 Security Properties

We define the security properties of transferable e-cash by refining previous definitions by [50, 29]. In the beginning of security games the challenger typically runs $par \leftarrow \texttt{ParamGen}(1^\lambda)$ and $(sk_{\mathcal{B}}, pk_B) \leftarrow \texttt{BKeyGen}()$, which we merge into one algorithm `AllGen`.

Unforgeability. This notion protects the bank in that an adversary should not be able to spend more coins than the number of coins he withdrew. In [29] an adversary can interact with honest users and wins the unforgeability game if the number of coins he withdrew is smaller than the number of coins he successfully deposited (were "successfully" means that no double spending was detected). We simplify the definition noticing that is not necessary for the adversary to create, corrupt or instruct *honest* users to withdraw, spend, receive and deposit, since the adversary could simulate these users itself. An unforgeability definition *without* honest user oracles implies thus the definition *with* these oracles given in [29]. (We also avoid the *While* loop in the definitions from [29].) Consider the following experiment:

Experiment $\mathbf{Expt}_{\mathcal{A}}^{\texttt{unforg}}(\lambda)$;

$(par, sk_{\mathcal{B}}, pk_{\mathcal{B}}) \leftarrow \texttt{AllGen}(1^\lambda)$;

$\mathcal{A}^{\texttt{BRegister},\texttt{BWith},\texttt{BDepo}}(par, pk_{\mathcal{B}})$;

Let q_W, q_D be the number of successful calls to `BWith`, `BDepo` respectively;

If $q_W < q_D$ then return 1;

Return $\perp$.

Definition 7.2.1 (Unforgeability)**.** *A transferable e-cash system is* unforgeable *if for any PPT adversary $\mathcal{A}$, we have* $\mathbf{Adv}_{\mathcal{A}}^{\texttt{unforg}}(\lambda) := \Pr[\mathbf{Expt}_{\mathcal{A}}^{\texttt{unforg}}(\lambda) = 1]$ *is negligible in λ.*

Identification of Double Spenders. No collection of users can spend a coin twice (double-spend) without revealing one of their identities. Consider the following experiment:

Experiment $\mathbf{Expt}_{\mathcal{A}}^{\texttt{ident}}(\lambda)$

$(par, sk_{\mathcal{B}}, pk_{\mathcal{B}}) \leftarrow \texttt{AllGen}(1^{\lambda})$;

$\mathcal{A}^{\texttt{Create,BRegister,Register,Corrupt,BWith,With,Rcv,Spd,S\&R,BDepo,Depo}}(par, pk_{\mathcal{B}})$;

Let (pk_{i^*}, Π_G) be the output of the last call to `BDepo`;

Return 1 if any of the following hold:

- $\texttt{VerifyGuilt}(pk_{i^*}, \Pi_G) = 0$;
- $pk_{i^*} \notin UL$;
- $pk_{i^*} \in UL$ and $cert_i = \perp$;

Return $\perp$.

Definition 7.2.2 (Double-Spender Identification)**.** *A transferable e-cash system is* secure against double-spending *if for any PPT adversary $\mathcal{A}$, we have* $\mathbf{Adv}^{\texttt{ident}}_{\mathcal{A}}(\lambda) := \Pr[\mathbf{Expt}^{\texttt{ident}}_{\mathcal{A}}(\lambda) = 1]$ *is negligible in λ.*

Note that we cannot detect double-spending unless both of the double-spent coins are deposited.

Exculpability. Exculpability ensures that the bank, even when colluding with a collection of malicious users, cannot falsely accuse honest users of double-spending. We present two flavors of exculpability: *Weak exculpability* guarantees that an adversarial bank cannot output a double spending proof $(pk_{\mathcal{U}}, t, \Pi')$ that verifies if the (honest) user $\mathcal{U}$ has not double-spent. This notion is similar to the definition of exculpability in [29]. However, we allow the adversary to generate the bank's keys himself, he just needs to output the index of the user being accused of double spending and the corresponding proof.

Our stronger version of exculpability guarantees that a user cannot be accused of double spending *more* coins than the ones he did double-spend. It is for this notion that we introduced the algorithm `VerifyDSCounter`, which verifies the number of double spendings a user is accused of.

Experiment $\mathbf{Expt}^{\texttt{weak-excul}}_{\mathcal{A}}(\lambda)$

$par \leftarrow \texttt{ParamGen}(1^\lambda)$;

$(pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(par)$;

$(i^*, d, \Pi^*) \leftarrow \mathcal{A}^{\substack{\texttt{Create,URegister,Corrupt,}\\ \texttt{UWith,Rcv,Spd,S\&R,UDepo}}}(st)$;

If $\texttt{VerifyGuilt}(pk_{i^*}, \Pi^*) = 1$

& $sk_{i^*} \neq \perp$ & $uds_{i^*} = 0$ then return 1;

Return $\perp$.

Experiment $\mathbf{Expt}^{\texttt{excul}}_{\mathcal{A}}(\lambda)$

$par \leftarrow \texttt{ParamGen}(1^\lambda)$;

$(pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(par)$;

$(i^*, t, \Pi^*) \leftarrow \mathcal{A}^{\substack{\texttt{Create,URegister,Corrupt,}\\ \texttt{UWith,Rcv,Spd,S\&R,UDepo}}}(st)$;

If $\texttt{VerifyDSCounter}(pk_{i^*}, t, \Pi^*) = 1$

& $sk_{i^*} \neq \perp$ & $uds_{i^*} < t$ then return 1;

Return $\perp$.

Definition 7.2.3 (Exculpability)**.** *A transferable e-cash system is* (weakly) exculpable *if for any PPT adversary $\mathcal{A}$, we have* $\mathbf{Adv}^{\texttt{excul}}_{\mathcal{A}}(\lambda) := \Pr[\mathbf{Expt}^{\texttt{excul}}_{\mathcal{A}}(\lambda) = 1]$ *is negligible in λ (and analogously for weak exculpability).*

7.3 Anonymity Properties

We first consider the three anonymity notions given in [29]:

Observe-then-Receive Full Anonymity (OtR-FA). The adversary, impersonating the bank, cannot link a coin he receives as an adversarial user or as the bank to a previously (passively) observed transfer between honest users. This covers both the case where the adversary receives a coin during a transfer or receives a coin when impersonating the bank during deposit.

Spend-then-Observe Full Anonymity (StO-FA). The adversary, impersonating the bank, cannot link a (passively) observed coin transferred between two honest users to a coin he has already owned as a "legitimate" user.

Spend-then-Receive Full Anonymity (StR-FA). When the bank is honest, the adversary cannot recognize a coin he previously owned when he receives it again.

These three notions are incomparable as proved in [50]. Their formal definitions are given below. Note that we allow $\mathcal{A}$ to pick the secret keys himself, in particular that of the bank when impersonating it (in contrast to [50, 29], where the bank's keys are created by experiment).

In the OtR-game the adversary outputs two indices of coins owned by honest users and receives one of them, either as a `Spend` (by setting $v = 0$) or as a `Deposit` (setting $v = 1$). The adversary must not receive the coin a second time (impersonating the bank, he could otherwise distinguish them), which the game ensures by resetting the flags to 0 and checking whether they remain that way.

Experiment $\mathbf{Expt}^{\texttt{OtR-fa}}_{\mathcal{A},b}(\lambda)$

$par \leftarrow \texttt{ParamGen}(1^\lambda)$; $(pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(par)$;

$(j_0, j_1, st, v) \leftarrow \mathcal{A}^{\texttt{Create,URegister,Corrupt,UWith,Rcv,Spd,S\&R,UDepo}}(st)$;

If $\texttt{comp}(j_0, j_1) = 0$ or $fc_{j_0} = 1$ or $fc_{j_1} = 1$ or $fd_{j_0} = 1$ or $fd_{j_1} = 1$ then return $\perp$;

If $v = 0$ simulate $\texttt{Spd}(j_b)$ to $\mathcal{A}$, else if $v = 1$ simulate $\texttt{UDepo}(j_b)$, else return $\perp$;

Reset the flags to $fd_{j_0} = 0$, $fd_{j_1} = 0$, $fc_{j_0} = 0$, $fc_{j_1} = 0$;

$b^* \leftarrow \mathcal{A}^{\texttt{Create,URegister,Corrupt,With,Rcv,Spd,S\&R,UDepo}}(st)$;

If $fd_{j_0} = 1$ or $fd_{j_1} = 1$ or $fc_{j_0} = 1$ or $fc_{j_1} = 1$ then abort;

Return b^*.

For the StO game we use a modified Spend&Receive oracle $\texttt{S\&R}^*$: for the coin c being transfreed, it creates a new entry in $\mathcal{CL}$ in the form of an *original* coin whose origin is marked to be *Challenger* while $owner = i$ and $fd = 0, fc = 0$. If the adversary tries to corrupt, receive or deposit this coin (or a transferred coin whose "parent" in $\mathcal{CL}$ is the *Challenger*) then we abort.

Experiment $\mathbf{Expt}^{\texttt{StO-fa}}_{\mathcal{A},b}(\lambda)$

$par \leftarrow \texttt{ParamGen}(1^\lambda)$; $(pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(par)$;

$(j_0, j_1, i, st) \leftarrow \mathcal{A}^{\texttt{Create,URegister,Corrupt,UWith,Rcv,Spd,S\&R,UDepo}}(st)$;

For $\beta = 0, 1$, let u_β be index of the owner of coin j_β (i.e., $owner_{j_\beta} = u_\beta$);

If $\texttt{comp}(j_0, j_1) = 0$ or $sk_{U_{j_0}} = \perp$ or $sk_{U_{j_1}} = \perp$ or $sk_i = \perp$ then return $\perp$;

Run $out \leftarrow \texttt{S\&R}^*(j_b, i)$;

$b^* \leftarrow \mathcal{A}^{\texttt{Create,URegister,Corrupt,UWith,Rcv,Spd,S\&R,UDepo}}(out,\ st)$;

If for the coin with origin *Challenger* it is $fd = 1$ or $fc = 1$ then abort;

Return b^*.

In the Spend-then-Receive game we assume that the bank is honest. The adversary picks two coins of the same size, with indices j_0, j_1, whose owners are uncorrupted. We then transfer the coin j_b to $\mathcal{A}$ for a randomly selected coin b and his goal is to guess b. When he runs again we have to make sure that he will not try to deposit any of the coins: j_0, j_1 or else he could trivially win by also depositing his coin and checking whether a double spending happened. To avoid this scenario, whenever he deposits a coin we check if it collides with one of the j_0, j_1 and if it does we deposit all j_0, j_1 and his coin so that he will not gain any knowledge about which was the coin.

Experiment $\mathbf{Expt}^{\texttt{StR-fa}}_{\mathcal{A},b}(\lambda)$

$(par, sk_{\mathcal{B}} = (sk_{\mathcal{W}}, sk_{\mathcal{D}}), pk_{\mathcal{B}}) \leftarrow \texttt{AllGen}(1^\lambda)$;

$(j_0, j_1, st) \leftarrow \mathcal{A}^{\texttt{Create,Register,Corrupt,With,Rcv,Spd,S\&R,BDepo,Depo}}(par, sk_{\mathcal{W}}, pk_{\mathcal{B}})$;

For $\beta = 0, 1$, let u_β be index of the owner of coin j_β (i.e., $owner_{j_\beta} = u_\beta$);

If $\texttt{comp}(j_0, j_1) = 0$ or $sk_{u_0} = \perp$ or $sk_{u_1} = \perp$ then return $\perp$;

Simulate $\texttt{Spd}(j_b)$ to $\mathcal{A}$;

$b^* \leftarrow \mathcal{A}^{\texttt{Create,Register,Corrupt,With,Rcv,Spd,S\&R,BDepo,Depo}}(st)$;

Whenever $\mathcal{A}$ calls `BDepo` or `Depo` first run the `CheckCoin` subroutine of `Deposit`.

If OK, initialize $\mathcal{DCL}' = \emptyset$ and run $\texttt{Deposit}(j_0^*)$, $\texttt{Deposit}(j_1^*)$ and then `CheckDS`

for the coin $\mathcal{A}$ deposits but using $\mathcal{DCL}'$ instead. If double spending detected then

run `Deposit` for coins j_0, j_1 and $\mathcal{A}$'s coin and add the three coins to $\mathcal{DCL}$;

Else run `CheckDS` with $\mathcal{DCL}$ and add $\mathcal{A}$'s coin to $\mathcal{DCL}$;

Return b^*.

Finally, we introduce a new, stronger notion of anonymity we call *Spend-then-Receive**: although the adversary, when impersonating the bank, can tell whenever he receives a coin he owned before, he should not be able to learn anything about the identities of the users that owned the coin in between. We define this as an indistinguishability game in which the adversary picks a pair of users, one of which (according to a random bit b) the coins are transferred to. The goal is to guess this bit

b.

Experiment $\mathbf{Expt}^{\texttt{StR*-fa}}_{\mathcal{A},b}(\lambda)$

$par \leftarrow \texttt{AllGen}(1^\lambda)$; $(pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(par)$;

$(i_0^*, i_1^*, 1^k) \leftarrow \mathcal{A}^{\texttt{Create,URegister,Corrupt,UWith,Rcv,Spd,S\&R,UDepo}}(st)$;

If $sk_{i_0^*} = \perp$ or $sk_{i_1^*} = \perp$ then return $\perp$;

Run $\texttt{Rcv}(i_b^*)$ and let c_1^* be the received coin;

$(i_0, i_1) \leftarrow \mathcal{A}(st)$; If $sk_{i_0} = \perp$ or $sk_{i_1} = \perp$ then return $\perp$;

For $\alpha = 1, \ldots, k$, let j_α be index of the coin $c*_\alpha$ in $\mathcal{CL}$;

Run $\texttt{S\&R}(i_b, j_1)$; Let c_2^* be the received coin; $i_b^* \leftarrow i_b$;

Repeat the last two steps k times;

Run $\texttt{Spd}(j_k)$ to $\mathcal{A}$;

$b^* \leftarrow \mathcal{A}^{\texttt{Create,URegister,Corrupt,UWith,Rcv,Spd,S\&R,UDepo}}(st)$;

If for any of coins $c_1^*, \ldots c_k^*$ we have $cds > 1$ then output $\perp$;

If any of the owners of $c_1^*, \ldots c_k^*$ is corrupted then output $\perp$;

Return b^*.

Definition 7.3.1. *(Anonymity) A transferable e-cash scheme is fully anonymous if for any PPT adversary $\mathcal{A}$ we have* $\mathbf{Adv}^{\texttt{OtR-fa}}_{\mathcal{A}}(\lambda) := \Pr[(\mathbf{Expt}^{\texttt{OtR-fa}}_{\mathcal{A},1}(\lambda) = 1] - \Pr[(\mathbf{Expt}^{\texttt{OtR-fa}}_{\mathcal{A},0}(\lambda) = 1]$ *is negligible in λ (and analogously for* $\mathbf{Expt}^{\texttt{StO-fa}}_{\mathcal{A},b}$, $\mathbf{Expt}^{\texttt{StR-fa}}_{\mathcal{A},b}$, *and* $\mathbf{Expt}^{\texttt{StR*-fa}}_{\mathcal{A},b}$*).*

7.4 Double-Spending Detection

We start by describing our independent double-spending detection mechanism. In our construction, every coin in the system contains a serial number $\texttt{SN} = \texttt{SN}_1 \| \ldots \| \texttt{SN}_k$ where $\texttt{SN}_1$ was jointly generated by the bank and first user who received the coin, $\texttt{SN}_2$ was generated by the second user who received the coin and so on, and a set of doublespending tags $\texttt{DS} = \texttt{DS}_1 \| \ldots \| \texttt{DS}_{k-1}$ which allows the bank to identify which user doublespent whenever a coin is deposited twice. (These values will be encrypted so that only the bank can see them, to satisfy our anonymity requirements.)

Here we first describe abstractly the properties we need from the serial number and doublespending tag. Section 7.4.2 will describe concrete instantiations satisfying these properties, and Section 7.5 will use these properties in our transferable e-cash construction.

7.4.1 Properties of Serial Numbers and Doublespending Tags

As we will see in Section 7.4.2, for transferable e-cash it seems essential that generation of each $\mathtt{SN}_i$ use both randomness chosen by the user who receives the ith coin, and the secret key of that user. We thus define a *Serial Number Function*, $f_{\mathtt{SN}}$, which on input a nonce and a secret key (n_i, sk_i) or just a nonce (n_i) outputs the serial number of the coin. We require a form of collision resistance, which essentially guarantees that different (n_i, sk_i) generate different $\mathtt{SN}$. Formally:

Definition 7.4.1 (Serial Number Function)**.** *A serial number function $f_{\mathtt{SN}}$ for parameters Gen_{SN} takes as input parameters $par_{SN} \leftarrow \mathsf{Gen}_{SN}$, a nonce and a secret key (n_i, sk_i), and outputs a serial number $\mathtt{SN}_i$. (We omit par_{SN} when it is clear from context.)*

- *$f_{\mathtt{SN}}$ is called* collision resistant *if given $par_{SN} \leftarrow \mathsf{Gen}_{SN}$, it is hard to find $(sk_i, n_i) \neq (sk'_i, n'_i)$ such that $f_{\mathtt{SN}}(par_{SN}, n_i, sk_i) = f_{\mathtt{SN}}(par_{SN}, n'_i, sk'_i)$.*

We also define a *Double Spending Tag Function*, $f_{\mathtt{DS}}$, that takes as input the nonce n_i that the coin owner $\mathcal{U}_i$ had picked when receiving the coin, $\mathcal{U}_i$'s secret key sk_i and the serial number, $\mathtt{SN}_{i+1}$ that is computed by the receiver of the coin. We also allow it to take as input some additional identifying information, ID_i, about $\mathcal{U}_i$. The output is a double-spending tag that reveals nothing about the owner, $\mathcal{U}_i$, unless she transfers that same coin to more than one users (i.e. double spends). In that case, the bank can, given a database of public keys of all the users (and associated info ID for each) identify which user doublespent and produce a proof accusing that user. More formally:

Definition 7.4.2 (Double Spending Tag Function)**.** *A double spending tag function $f_{\mathtt{DS}}$ for parameters Gen_{SN} and Key Generation algorithm $\mathtt{KeyGen}$ takes as input $par_{SN} \leftarrow \mathsf{Gen}_{SN}$, $(ID_i, n_i, sk_i, \mathtt{SN}_{i+1})$ and outputs the double spending tag $\mathtt{DS}_i$.*

- $f_{\mathtt{DS}}$ *is* 2-show extractable *if whenever we compute* $\mathtt{DS}_i$ *and* $\mathtt{DS}'_i$ *for the same* $(ID_i, n_i, sk_i, par_{SN})$ *but different* $\mathtt{SN}_{i+1} \neq \mathtt{SN}'_{i+1}$*, there exists an efficient function* $f_{\mathtt{DetectDS}}$ *that on input* $\mathtt{DS}_i$ *and* $\mathtt{DS}'_i$ *and a list of identifiers* $\mathcal{I}$ *such that* $(ID_i, pk_i) \in \mathcal{I}$ *for a* pk_i *corresponding to* sk_i *(according to* KeyGen*), efficiently extracts* (pk_i, Π) *where* Π *is an accepting proof for* pk_i*.*

- $f_{\mathtt{DS}}$ *is* exculpable *if, given* pk_i *produced by* KeyGen *and* $par_{SN} \leftarrow \mathsf{Gen}_{SN}$*, it is computationally difficult to compute an accepting proof for* pk_i*.*

Finally, we want to be able to guarantee some anonymity even against a malicious bank who gets to see the serial numbers and doublespending tags for deposited coins. Thus, we require that as long as the nonce n_i is fresh and random, these values reveal nothing about the other values used to generate them.

Definition 7.4.3 (Anonymous Double Spending Tag and Serial Number Functions)**.** *We say that a doublespending tag function* $f_{\mathtt{DS}}$ *and a serial number function* $f_{\mathtt{SN}}$ *are* anonymous *if, for all* $ID_i, sk_i, \mathtt{SN}_{i+1}$*,* $ID'_i, sk'_i, \mathtt{SN}'_{i+1}$*, if* $par_{SN} \leftarrow \mathsf{Gen}_{SN}$ *and* n_i *is chosen at random, then* $(par_{SN}, f_{\mathtt{SN}}(par_{SN}, n_i, sk_i)$*,* $f_{\mathtt{DS}}(par_{SN}, ID_i, n_i, sk_i, \mathtt{SN}_{i+1}))$ *is computationally indistinguishable from* $(par_{SN}, f_{\mathtt{SN}}(par_{SN}, n_i, sk'_i)$*,* $f_{\mathtt{DS}}(par_{SN}, ID'_i, n_i, sk'_i, \mathtt{SN}'_{i+1}))$*.*

7.4.2 A Double Spending Detection Mechanism

Here we propose a concrete instantiation for the functions $f_{\mathtt{SN}}, f_{\mathtt{DS}}$ used to generate the serial numbers and double spending tags. To give some intuition, we first consider the natural translation of traditional (non-transferable) ecash double spending techniques [61], and show why it is not sufficient in the transferable setting. Assume that $\mathcal{U}_i$,$\mathcal{U}_{i+1}$ execute the Spend protocol where the first user transfers a coin to the second one. Let $\mathtt{SN}_{i+1}$ be the nonce that the second user randomly picks and sends to $\mathcal{U}_i$ in the clear. Then, $\mathcal{U}_i$ could compute the double spending tag as follows: $\mathtt{DS}_i = pk_i^{n_{i+1}} h^{n_i}$ with $\mathtt{SN}_{i+1} = n_{i+1}$. Assume now that $\mathcal{U}_i$ double spends the coin (i.e. transfers it to users $\mathcal{U}_{i+1}$ and $\mathcal{U}'_{i+1}$) and the two coins eventually get deposited at the bank. If the bank detects a double spending,

it looks for the first difference in the sequences of the nonces in $\mathtt{SN}, \mathtt{SN}'$ and will find out that $\mathtt{SN}_{i+1} \neq \mathtt{SN}'_{i+1}$ and thus $\mathcal{U}_i$ double spent. Combining the double-spending tags $\mathtt{DS}_i$ and $\mathtt{DS}'_i$, the bank can compute the public key of the double spender: $pk_i = (\mathtt{DS}_i(\mathtt{DS}'_i)^{-1})^{(n_{i+1}-n'_{i+1})^{-1}}$ But what if a coin was double spent and the two different receivers picked the same nonce n_{i+1}? We consider two cases:

Case 1: $\mathcal{U}_i$ transfers the same coin (double spends) to the *same* user twice and user $\mathcal{U}_{i+1}$ picks the same nonce n_{i+1} in both transactions. When the coins is deposited and the bank compares the serial numbers, the first difference occurs at position $i+2$ (assuming that $\mathcal{U}_{i+1}$ spent the two instances to users that pick different nonces n_{i+2} and n'_{i+2}). From the double-spending tags, the bank computes pk_{i+1} and will accuse U_{i+1} of double spending.

Case 2: $\mathcal{U}_i$ transfers the same coin to two *different* users with pk_{i+1} and pk'_{i+1} who pick the same nonce n_{i+1} when receiving the coin. As before, the bank's serial numbers will diverge at position $i+2$. However, in this case computation of a public key will fail, since $\mathtt{DS}_{i+1}$ and $\mathtt{DS}'_{i+1}$ contain different public keys.

The first case seems unavoidable. However, a user who picks a fresh nonce each time when receiving a coin will not be falsely accused of double spending, and a malicious user who does choose the same nonce twice could be seen to be cooperating in the doublespending. To address the second case, we need to ensure that different users cannot use the same nonce when receiving a coin. A possible idea is to ask the users to use some fixed value, unique to each user, (for instance the user's secret key) together with the nonce. In the above construction this means we'd need the exponent to be some combination of sk_i, n_i, in such a way that it would be hard for two users to find sk_i, n_i, sk'_i, n'_i that produce the same value (otherwise we haven't solved the problem) and in such a way that we can still generate efficient proofs that $\mathtt{DS}$ is well formed. Unfortunately t is not clear how to do this using existing proof systems. Briefly, we overcome this by duplicating the double-spending tag: $\mathtt{DS}_i = (A_i, B_i)$. The A_i tags use only the nonce value and the B_i tags use a combination of the secret key and the nonce.

Our construction In our solution, par_{SN} will define an asymmetric pairing group of prime order q (G_1, G_2, G_T, e, q), and five random generators of G_1, $(g, h, h', \tilde{h}, \tilde{h}')$. We assume that secret keys and the info ID are elements of Z_q. We define the serial number function, $f_{\mathtt{SN}}$, as follows: when a user $\mathcal{U}_{i+1}$ receives a coin, he picks a random $n_{i+1} \in \mathbb{Z}_q$ and computes

$$f_{\mathtt{SN}}(n_{i+1}, sk_{i+1}) = \{N_{i+1} = g^{n_{i+1}},\ M_{i+1} = g^{sk_{i+1} \cdot n_{i+1}}\}\ .$$

The serial number of the coin is $\mathtt{SN}_{i+1} = (N_{i+1}, M_{i+1})$ which is sent to $\mathcal{U}_i$. Now, the user $\mathcal{U}_i$ forms the double-spending tags to be:

$$f_{\mathtt{DS}}(ID_i, n_i, sk_i, (N_{i+1}, M_{i+1})) = \{A_i = N_{i+1}^{ID_i} h^{n_i},\ B_i = M_{i+1}^{ID_i} {h'}^{n_i},$$
$$\tilde{A}_i = N_{i+1}^{sk_i} \tilde{h}^{n_i},\ \tilde{B}_i = M_{i+1}^{sk_i} {\tilde{h}'}^{n_i}\}.$$

Our Construction Satisfies the Properties Defined in 7.4.1 . It is easy to see that our $f_{\mathtt{SN}}$ function is *collision resistant*: the only way to get $N_{i+1} = N'_{i+1}$ is for the adversary to pick $n_{i+1} = n'_{i+1}$ but then the only way to get $M_{i+1} = M'_{i+1}$ is to pick $sk_{i+1} = sk'_{i+1}$.

Next we consider doublespending. We assume that the bank stores a database of pairs (pk, ID) for all registered user, and that the ID's are unique. When a coin is deposited, the bank retrieves the serial number of the coin which is $\mathtt{SN} = \mathtt{SN}_1 \| \ldots \| \mathtt{SN}_k$. In case a double spending is detected, i.e., there is another coin deposited with $\mathtt{SN} \neq \mathtt{SN}'$ but $\mathtt{SN}_1 = \mathtt{SN}'_1$, the bank looks for the first pair such that $\mathtt{SN}_i = (N_{i+1}, M_{i+1}) \neq \mathtt{SN}'_i = (N'_{i+1}, M'_{i+1})$ in order to detect where the double spending happened. If both N_{i+1} and M_{i+1} are different then the bank can use either $((A_i, A'_i), (N_{i+1}, N'_{i+1}))$ or $((B_i, B'_i), (M_{i+1}, M'_{i+1}))$ to reveal the ID of the user who double spent by checking for all $ID \in \mathcal{DB}_B$:

$$(A_i (A'_i)^{-1}) \stackrel{?}{=} (N_{i+1} (N'_{i+1})^{-1})^{ID} \quad or \quad (B_i (B'_i)^{-1}) \stackrel{?}{=} (M_{i+1} (M'_{i+1})^{-1})^{ID}$$

This is a relatively cheap operation that can be implemented efficiently. (In our ecash construction

in Section 7.5, *ID* will be the user's position in the registered user list. Then all these values will be fairly small, so these should be very efficient operations.) If we have `KeyGen` that outputs $pk_i = \hat{g}^{sk_i}$ for a fixed generator $\hat{g}$ of G_2, then when the bank detects the *ID* that satisfies the equation above, it can look up in its database the public key of the user associated with *ID* and check whether the following pairing was also satisfied:

$$e(\tilde{A}_i(\tilde{A}'_i)^{-1}, \hat{g}) = e(N_{i+1}(N'_{i+1})^{-1}, pk_i) \tag{7.1}$$

or similar for $\tilde{B}_i, \tilde{B}'_i, M_{i+1}, M'_{i+1}$ if these equations where used. Obviously, if a double spending was detected and $(N_{i+1}, M_{i+1}) \neq (N'_{i+1}, M'_{i+1})$ but $(N_{i+1} = N'_{i+1})$ or $(M_{i+1} = M'_{i+1})$ then the bank will have to use A_i, A'_i or B_i, B'_i respectively to detect the *ID* of the double spender. (If none of these produces a match with a pk, ID in the database, the bank outputs $(\perp, \perp)$, but this should never happen.) Thus, the function $f_{\texttt{DetectDS}}$ on input $\texttt{DS}_i, \texttt{DS}'_i, \mathcal{DB}_B$ outputs pk and $\Pi = (\texttt{DS}_i, \texttt{DS}'_i)$. The verification for this proof just checks equation 7.1. (Thus, our $f_{\texttt{DS}}$ function is 2-show extractable.)

Informally, the double spending function is exculpable by SXDH: If $\mathcal{A}$ can take $pk_i = \hat{g}^x$ and produce $A_i, \tilde{A}'_i, N_{i+1}, N'_{i+1}$ satisfying 7.1, then we could use these values to distinguish e.g. $\hat{g}^{xy}$ from random and break SXDH (DDH in G_2).

Finally the anonymity property of $f_{\texttt{SN}}, f_{\texttt{DS}}$ follows from SXDH (DDH in G_1).

Discussion. Note that we could just use the equations: $(\tilde{A}_i, \tilde{B}_i)$ to detect double spending. However, this ends up to a less efficient double spending detection since it requires the bank to check a pairing equation for every public key in its database. Also, if exculpability is not required (i.e. the bank is honest and will not try to falsely accuse honest users of double spending) then we could only use the A_i, B_i double spending tags.

7.5 Transferable E-Cash Based on Malleable Signatures

We now describe a generic construction of a transferable e-cash scheme using malleable signatures. Assume the existence of: a malleable signature scheme ($\mathsf{MSGen}, \mathsf{MSKeyGen}, \mathsf{MSign}, \mathsf{MSVerify}, \mathsf{MSigEval}$) with allowed transformation class $\mathcal{T}$ as defined below, a signature scheme ($\mathsf{SKeyGen}$, Sign, Verify), a randomizable public key encryption scheme ($\mathsf{EKeyGen}$, $\mathsf{Enc}, \mathsf{REnc}$, Dec), a commitment scheme ($\mathsf{ComSetup}, \mathsf{Com}$), a zero knowledge proof system $\langle P, V \rangle$ and a hard relation R[4]. We also assume the existence of the functions $f_{\mathtt{SN}}, f_{\mathtt{DS}}, f_{\mathtt{DetectDS}}$ for Gen_{SN} as defined above.

A *re-randomizable encryption scheme* allows to re-randomize a ciphertext c to a new ciphertext c' such that are both encryptions of the same plaintext but are statistically independent.

Definition 7.5.1 (Re-Randomizable Encryption Scheme [93])**.** *A public key cryptosystem* ($\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec}$) *is statistically re-randomizable if:*

- ($\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec}$) *is semantically-secure (IND-CPA)*
- *There exists an efficient algorithm* REnc *such that if* r' *is chosen uniformly at random from coins(*REnc*) and* r_0 *is chosen from coins(*Enc*), where coins() is a randomness source for the system, then the distributions*

$$\{\mathsf{Enc}(pk, m, r_0)\} \approx_s \{\mathsf{REnc}(\mathsf{Enc}(pk, m, r_1), r')\}$$

for all public keys pk*, messages* m*, and randomness* r_1*.*

If ($\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec}$) is homomorphic, then re-randomization is possible by $\mathsf{REnc}(pk, c, r') = c \cdot \mathsf{Enc}(pk, 0, r')$. This holds for all known homomorphic cryptosystems such as ElGamal [73], Paillier [116] and Goldwasser-Micali [86].

Back to the construction, the bank's withdrawal key consists of $(vk_B^{(MS)}, sk_B^{(MS)}) \leftarrow \mathsf{MSKeyGen}(1^\lambda)$ and $(vk_B^{(S)}, sk_B^{(S)}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$ while the deposit key is $(pk_{\mathcal{D}}, sk_{\mathcal{D}}) \leftarrow \mathsf{EKeyGen}(1^\lambda)$. Users have

[4]Informally, a relation R is said to be hard if for $(x, w) \in R$, a PPT adversary $\mathcal{A}$ given x will output w_A s.t. $(x, w_A) \in R$ with only negligible probability

keys $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$ and when registering with the bank they receive a certificate $cert_{\mathcal{U}} = \mathsf{Sign}_{sk_B^{(S)}}(pk_U, I_{\mathcal{U}})$, where $I_{\mathcal{U}}$ is their joining order.

7.5.1 Allowed Transformations

In a malleable signature scheme we define a class of allowed transformations, and then unforgeability must guarantee that all valid signatures are generated either by the signer or by applying one of the allowed transformations to another valid signature. In the following we will define two different types of transformations: one to be used when a coin is transferred from a user to another, $\mathcal{T}_{CSpend}$, and a second one that will be used when a user withdraws a coin from the bank $\mathcal{T}_{CWith}$.

Coin Spend Transformation. A coin that has been transferred i times (counting withdrawal as the first transfer) will have the following format: $c = (par, (C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}}), (n_i, R_{\mathtt{SN}_i}), \sigma)$ where par denotes the parameters of the transferable e-cash scheme and $C_{\overline{\mathtt{SN}_i}} = C_{\mathtt{SN}_1} \parallel \cdots \parallel C_{\mathtt{SN}_i}$, $C_{\overline{\mathtt{DS}_{i-1}}} = C_{\mathtt{DS}_1} \parallel \cdots \parallel C_{\mathtt{DS}_{i-1}}$, for $C_{\mathtt{SN}_j} = \mathsf{Enc}(\mathtt{SN}_j)$ and $C_{\mathtt{DS}_j} = \mathsf{Enc}(\mathtt{DS}_j)$ respectively (all the encryptions are done under $pk_{\mathcal{D}}$). By $\mathtt{DS}_{i-1}$ we denote the double spending tag that was computed by the user $\mathcal{U}_{i-1}$ when he transferred the coin to user $\mathcal{U}_i$, n_i is a nonce picked by $\mathcal{U}_i$ when he received the coin[5], and by $R_{\mathtt{SN}_i}$ the randomness used to compute the encryption of $\mathtt{SN}_i$, i.e., $C_{\mathtt{SN}_i} = \mathsf{Enc}(\mathtt{SN}_i; R_{\mathtt{SN}_i})$. Finally, σ is a malleable signature on $(C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}})$.

Assume now that the user $\mathcal{U}_i$ wishes to transfer the coin c to $\mathcal{U}_{i+1}$. First, $\mathcal{U}_{i+1}$ will pick a nonce n_{i+1} and will send $\mathtt{SN}_{i+1} = f_{\mathtt{SN}}(n_{i+1}, sk_{i+1})$ to $\mathcal{U}_i$. Then, $\mathcal{U}_i$ will compute the new signature to be (with T defined below):

$$\sigma' = \mathsf{MSigEval}(par, T, (C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}}), \sigma).$$

The transferred coin that $\mathcal{U}_{i+1}$ will eventually obtain has the form:

$$c' = (par, (C_{\overline{\mathtt{SN}_{i+1}}}, C_{\overline{\mathtt{DS}_i}}), (n_{i+1}, R_{\mathtt{SN}_{i+1}}), \sigma').$$

[5]Depending on the instantiation, the nonce, n_i might be computed as the output of a function on a random number the user picks.

Note that the value n_{i+1} is only known to $\mathcal{U}_{i+1}$ and he will have to use it when he wants to further transfer the coin, while the randomness $R_{\mathtt{SN}_{i+1}}$, used to encrypt $\mathtt{SN}_{i+1}$, was sent by $\mathcal{U}_i$. What is left is to define the transformation T which will take as input $m = (C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}})$ and will output $T(m) = (C_{\overline{\mathtt{SN}_{i+1}}}, C_{\overline{\mathtt{DS}_i}})$. A transformation of this type is described by the following values: (i.e. this is the information that one must "know" in order to apply the transformation)

$$\langle T \rangle = ((sk_i, I_i, cert_i), (n_i, R_{\mathtt{SN}_i}, R_{\mathtt{SN}_{i+1}}, R_{\mathtt{DS}_i}, R), \mathtt{SN}_{i+1}).$$

The output of T as defined by these values on input $m = (C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}})$ is then computed as follows:

1. If $\mathsf{Enc}(\mathtt{SN}_i; R_{\mathtt{SN}_i}) \neq C_{\mathtt{SN}_i}$ or $\mathtt{SN}_i \neq f_{\mathtt{SN}}(n_i, sk_i)$, then output $\perp$.
2. The new part of the serial number is encoded using randomness $R_{\mathtt{SN}_{i+1}}$: $C_{\mathtt{SN}_{i+1}} = \mathsf{Enc}(\mathtt{SN}_{i+1}; R_{\mathtt{SN}_{i+1}})$.
3. The new part of the double spending tag is first computed using $f_{\mathtt{DS}}$ and then encrypted: $\mathtt{DS}_i = f_{\mathtt{DS}}(I_i, n_i, sk_i, \mathtt{SN}_{i+1})$ and $C_{\mathtt{DS}_i} = \mathsf{Enc}(\mathtt{DS}_i; R_{\mathtt{DS}_i})$.
4. These encryptions are appended to the re-randomizations of $C_{\overline{\mathtt{SN}_i}}$ and $C_{\overline{\mathtt{DS}_{i-1}}}$:

$$C_{\overline{\mathtt{SN}_{i+1}}} = \mathsf{REnc}(C_{\mathtt{SN}_1}; R_1) \parallel \ldots \parallel \mathsf{REnc}(C_{\mathtt{SN}_i}; R_i) \parallel C_{\mathtt{SN}_{i+1}}$$

$$C_{\overline{\mathtt{DS}_i}} = \mathsf{REnc}(D_{\mathtt{DS}_1}; R'_1) \parallel \ldots \parallel \mathsf{REnc}(C_{\mathtt{DS}_{i-1}}; R'_{i-1}) \parallel C_{\mathtt{DS}_i}$$

   where $R_1, \ldots, R_i, R'_1, \ldots, R'_{i-1}$ are all parts of the randomness R included in the description of the transformation.

We define $\mathcal{T}_{CSpend}$ to be the set of all transformations of this form such that:

1. The certificate $cert_i$ is be valid (verifiable under the bank's verification key) and correspond to the secret key sk_i and some additional info I_i.
2. The random values $R_{\mathtt{SN}_i}$,$R_{\mathtt{SN}_{i+1}}$, $R_{\mathtt{DS}_i}$, R picked from $\mathcal{U}_i$ belong to the correct randomness space as defined by the encryption scheme.

Coin Withdrawal Transformation. A coin that was just withdrawn has a format different from a coin that has already been transferred, as there is no need to include double spending tags (we ensure that each coin withdrawn is a different coin) by the bank and the user. When a user $\mathcal{U}_i$ withdraws a coin from the bank, she picks a nonce n_1, computes a commitment $com = \mathsf{Com}(n_1, sk_i; open)$ on n_1 and her secret key and sends it to the bank. (For the user to remain anonymous it is important that the bank does not learn n_1.) The bank computes $\sigma = \mathsf{MSign}(sk_{\mathcal{W}}, com)$ and sends it to the user. The latter computes $\mathtt{SN}_1 = f_{\mathtt{SN}}(n_1, sk_i)$, chooses randomness $R_{\mathtt{SN}_1}$ and sets $C_{\mathtt{SN}_1} = \mathsf{Enc}(\mathtt{SN}_1; R_{\mathtt{SN}_1})$ and computes a new signature $\sigma' = \mathsf{MSigEval}(par, T, com, \sigma)$, which yields the coin defined as $c = (par, C_{\mathtt{SN}_1}, (n_1, R_{\mathtt{SN}_1}), \sigma')$. A transformation $T \in \mathcal{T}_{CWith}$, which takes as input $m = com$ and outputs $T(m) = C_{\mathtt{SN}_1}$ is described by $\langle T \rangle = ((sk_i, I_i, cert_i), (n_1, open), R_{\mathtt{SN}_1}, \mathtt{SN}_1)$. We define

$$T(com_{n_1}) = \begin{cases} C_{\mathtt{SN}_1} = \mathsf{Enc}(\mathtt{SN}_1; R_1) & \text{if } \mathsf{Com}(n_1, sk_i; open) = com \ \& \ \mathtt{SN}_1 = f_{\mathtt{SN}}(sk_i, n_1) \\ \perp & \text{otherwise.} \end{cases}$$

We define $\mathcal{T}_{CWith}$ to be the set of all transformations of this form such that:

1. The certificate $cert_i$ is valid (verifiable under the bank's verification key) and correspond to the secret key sk_i and I_i.

2. Randomness $R_{\mathtt{SN}_1}$ belongs to the appropriate randomness space.

The class of allowed transformations We will allow users to apply a transformation in $\mathcal{T}_{CWith}$ followed by any number of transformations in $\mathcal{T}_{CSpend}$. Thus, we define the allowed class of transformations $\mathcal{T}$ for our malleable signature to be the closure of $\mathcal{T}_{CWith} \cup \mathcal{T}_{CSpend}$.

7.5.2 A Transferable E-Cash Construction

Below we describe a transferable e-cash scheme based on malleable signatures. For our construction we assume *secure channels* for all the communications, thus an adversary cannot overhear or tamper

with the transferred messages.

$\texttt{ParamGen}(1^\lambda)$: Compute $crs \leftarrow \mathsf{MSGen}(1^\lambda)$, $par_{SN} \leftarrow \mathsf{Gen}_{SN}(1^\lambda)$, and $par' \leftarrow \mathsf{ComSetup}(1^\lambda)$. Output $par := (1^\lambda, crs, par', par_{SN})$.

$\texttt{UKeyGen}(par)$: Output a random pair $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$ sampled from R.

$\texttt{BKeyGen}(par)$: Compute the withdrawal keys of the bank as $(vk_B^{(MS)}, sk_B^{(MS)}) \leftarrow \mathsf{MSKeyGen}(1^\lambda)$ and $(vk_B^{(S)}, sk_B^{(S)}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$ and the deposit keys as $(pk_{\mathcal{D}}, sk_{\mathcal{D}}) \leftarrow \mathsf{EKeyGen}(1^\lambda)$. Define $pk_{\mathcal{W}} = (vk_B^{(MS)}, vk_B^{(S)})$ and $sk_{\mathcal{W}} = (sk_B^{(MS)}, sk_B^{(S)})$ and output $((pk_{\mathcal{W}}, sk_{\mathcal{W}}), (pk_{\mathcal{D}}, sk_D))$. The bank maintains a list UL of all registered users and a list $\mathcal{DCL}$ of deposited coins.

$\texttt{Registration}(\mathcal{B}[sk_{\mathcal{W}}, pk_{\mathcal{U}}], \mathcal{U}[sk_{\mathcal{U}}, pk_{\mathcal{W}}])$: if $pk_{\mathcal{U}} \notin UL$, the bank computes $cert_{\mathcal{U}} = \mathsf{Sign}_{sk_B^{(S)}}(pk_U, I_{\mathcal{U}})$, where $I_{\mathcal{U}} = |UL| + 1$. Add $pk_{\mathcal{U}}, cert, I_U$ to the user list UL and output $(cert_{\mathcal{U}}, I_{\mathcal{U}})$ or $\perp$.

$\texttt{Withdraw}(\mathcal{B}[sk_{\mathcal{W}}, pk_{\mathcal{U}}], \mathcal{U}[sk_{\mathcal{U}}, pk_{\mathcal{W}}])$: The user picks a nonce n_1 and sends $com \mathsf{Com}(n_1, sk_{\mathcal{U}}; open)$. The bank computes $\sigma \leftarrow \mathsf{MSign}(sk_B^{(MS)}, com)$ and sends it to the user. If $\mathsf{MSVerify}(crs, pk_B^{(MS)}, \sigma, com) = 0$, the user aborts; otherwise she sets $\mathtt{SN}_1 = f_{\mathtt{SN}}(n_1, sk_{\mathcal{U}})$, chooses randomness $R_{\mathtt{SN}_1}$ and computes $C_{\mathtt{SN}_1} = \mathsf{Enc}(\mathtt{SN}_1; R_{\mathtt{SN}_1})$. Then she sets $\langle T \rangle = ((sk_i, cert_i), (n_1, open), R_{\mathtt{SN}_1}, \mathtt{SN}_1)$ and computes the new signature as $\sigma' = \mathsf{MSigEval}(par, T, com, \sigma)$. The output is the coin $c = (par, C_{\mathtt{SN}_1}, (n_1, R_{\mathtt{SN}_1}), \sigma')$.

$\texttt{Spend}(\mathcal{U}_1[c, sk_{\mathcal{U}_1}, cert_{\mathcal{U}_1}, pk_{\mathcal{W}}], \mathcal{U}_2[sk_{\mathcal{U}_2}, pk_{\mathcal{W}}])$ Parse the coin as

$$c = (par, C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}}), (n_i, R_{\mathtt{SN}_i}), \sigma) \ .$$

$\mathcal{U}_2$ picks a nonce n_{i+1}, computes $\mathtt{SN}_{i+1} = f_{\mathtt{SN}}(n_{i+1}, sk_{\mathcal{U}_2})$ and sends it to $\mathcal{U}_1$. $\mathcal{U}_1$ computes the double spending tag $\mathtt{DS}_i = f_{\mathtt{DS}}(sk_{\mathcal{U}_1}, n_i, \mathtt{SN}_{i+1})$ and defines the transformation $\langle T \rangle = ((sk_{\mathcal{U}_1}, cert_{\mathcal{U}_1}), (n_i, R_{\mathtt{SN}_i}, R_{\mathtt{SN}_{i+1}}, R_{\mathtt{DS}_i}, R), \mathtt{SN}_{i+1})$. Next, he computes $C_{\mathtt{SN}_{i+1}} = \mathsf{Enc}(\mathtt{SN}_{i+1}; R_{\mathtt{SN}_{i+1}})$

and $C_{\mathtt{DS}_i} = \mathsf{Enc}(\mathtt{DS}_i; R_{\mathtt{DS}_i})$, which he appends to the randomized ciphertext contained in c:

$$C_{\overline{\mathtt{SN}_{i+1}}} = \mathsf{REnc}(C_{\mathtt{SN}_1}; R_1) \parallel \ldots \parallel \mathsf{REnc}(C_{\mathtt{SN}_i}; R_i) \parallel C_{\mathtt{SN}_{i+1}}$$

$$C_{\overline{\mathtt{DS}_i}} = \mathsf{REnc}(D_{\mathtt{DS}_1}; R'_1) \parallel \ldots \parallel \mathsf{REnc}(C_{\mathtt{DS}_{i-1}}; R'_{i-1}) \parallel C_{\mathtt{DS}_i}$$

$\mathcal{U}_1$ computes $\sigma' = \mathsf{MSigEval}(par, T, (C_{\overline{\mathtt{SN}_{i+1}}}, C_{\overline{\mathtt{DS}_i}}), \sigma)$ and sends $(\sigma', R_{i+1}, (C_{\overline{\mathtt{SN}_{i+1}}}, C_{\overline{\mathtt{DS}_i}}))$ to $\mathcal{U}_2$. If $\mathsf{MSVerify}(crs, pk_B^{(MS)}, \sigma', (C_{\overline{\mathtt{SN}_{i+1}}}, C_{\overline{\mathtt{DS}_i}})) = 0$ then $\mathcal{U}_2$ aborts. Otherwise, $\mathcal{U}_2$ outputs $c' = (par, (C_{\overline{\mathtt{SN}_{i+1}}}, C_{\overline{\mathtt{DS}_i}}), (n_{i+1}, R_{\mathtt{SN}_{i+1}}), \sigma')$.

`Deposit`$(\mathcal{U}[c, sk_{\mathcal{U}}, cert_{\mathcal{U}}, pk_{\mathcal{B}}], \mathcal{B}[sk_{\mathcal{D}}, pk_{\mathcal{U}}, \mathcal{CL}])$: First, $\mathcal{U}$ runs a `Spend` protocol with the bank playing the receiver: `Spend`$(\mathcal{U}[c, sk_{\mathcal{U}}, cert_{\mathcal{U}_1}, pk_{\mathcal{W}}], \mathcal{B}[\perp, pk_{\mathcal{W}}])$ (the bank can set the secret key to $\perp$, as it will not transfer this coin). If the protocol did not abort, $\mathcal{B}$ holds a valid coin $c = (par, (C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}}), (n_i,\ R_{\mathtt{SN}_i}), \sigma)$. Next, using $sk_{\mathcal{D}}$, $\mathcal{B}$ decrypts the serial number $\overline{\mathtt{SN}_i} = \mathtt{SN}_1 \parallel \cdots \parallel \mathtt{SN}_i$ and the double spending tags $\overline{\mathtt{DS}_{i-1}} = \mathtt{DS}_1 \parallel \cdots \parallel \mathtt{DS}_{i-1}$. It checks if in $\mathcal{CL}$ there exists another coin c' with $\mathtt{SN}'_1 = \mathtt{SN}_1$; of not then adds coin to $\mathcal{CL}$.

Otherwise, a double spending must have happened and the bank looks for the first position d, where $\mathtt{SN}'_d \neq \mathtt{SN}_d$. (Except with negligible probability such a position exists, since $\mathtt{SN}_i$ was chosen by the bank.) It applies the double-spending detection function $f_{\mathtt{DetectDS}}$ on the corresponding double spending tags $\mathtt{DS}_{d-1}$ and $\mathtt{DS}'_{d-1}$. If $f_{\mathtt{DetectDS}}$ outputs $\perp$ then $\mathcal{B}$ aborts. Otherwise, it outputs $(pk_U, \Pi)) = f_{\mathtt{DetectDS}}(\mathtt{DS}_{d-1}, \mathtt{DS}'_{d-1}, UL)$.

`VerifyGuilt`$(pk_{\mathcal{U}}, \Pi)$: it outputs 1 if the proof Π verifies and 0 otherwise.

`Withdraw` and `Spend` are depicted in Figures 7.1 and 7.2.

7.6 Security and Privacy of the New Construction

Theorem 7.6.1. *If the malleable signature scheme* (MSGen, MSKeyGen, MSign, MSVerify, MSigEval) *is simulatable, simulation unforgeable and simulation hiding with respect to $\mathcal{T}$, the signature scheme*

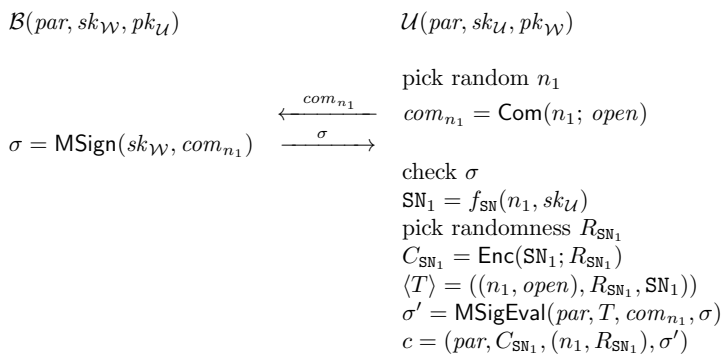

Figure 7.1: Withdrawal

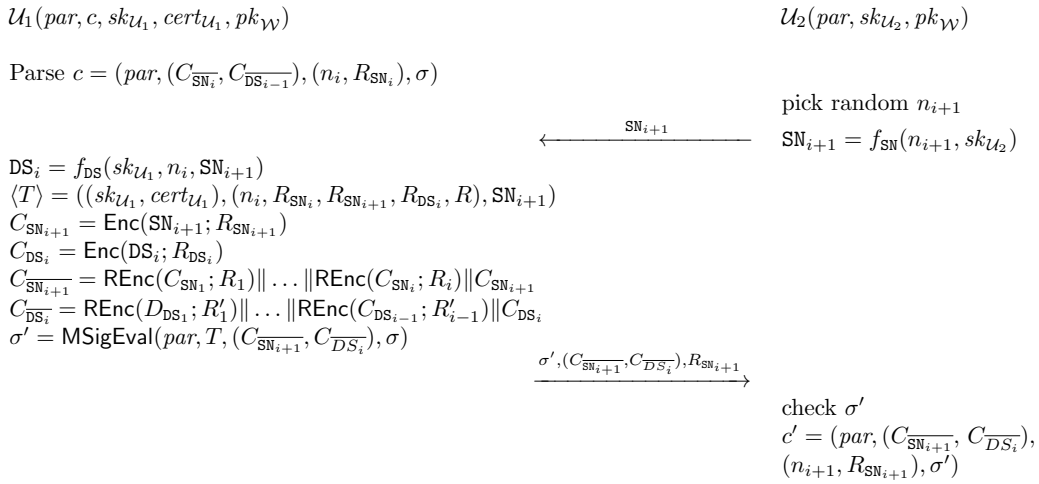

Figure 7.2: Transfer

$(\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is existentially unforgeable, the randomizable public-key encryption scheme* $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{REnc}, \mathsf{Dec})$ *is semantically secure and statistically re-randomizable, and the commitment scheme* $(\mathsf{ComSetup}, \mathsf{Com})$ *is computationally hiding and perfectly binding, then the construction in Section 7.5.2 describes a secure and anonymous transferable e-cash scheme as defined in Section 7.1.*

Lemma 7.6.2 (Unforgeability)**.** *If* $(\mathsf{MSGen}, \mathsf{MSKeyGen}, \mathsf{MSign}, \mathsf{MSVerify}, \mathsf{MSigEval})$ *is simulatable and simulation unforgeable as defined in Definition 2.3.5 and* $(\mathsf{ComSetup}, \mathsf{Com})$ *is perfectly binding, then the construction in Section 7.5.2 describes an unforgeable transferable e-cash scheme.*

Proof. Assume that there exists an adversary $\mathcal{A}$ that breaks unforgeability as defined in Definition 7.2.1 with non-negligible probability. We can construct a reduction $\mathcal{A}'$ that either breaks the unforgeability of the underlying malleable signature scheme or the binding property of the commitment scheme with non-negligible probability. The input that $\mathcal{A}'$ receives is $(1^\lambda, crs, vk^{(MS)}, \tau_e)$. Then, the following steps take place:

- $\mathcal{A}'$ runs $par' \leftarrow \mathsf{ComSetup}(1^\lambda)$ and $par_{SN} \leftarrow \mathsf{Gen}_{SN}(1^\lambda)$ to generate the parameters of the transferable e-cash scheme and sends $par := (1^\lambda, crs, par', par_{SN})$ to $\mathcal{A}$. It also initializes $q_W = 0, q_D = 0$, $\mathcal{DCL} = \emptyset$, $\mathcal{CL} = \emptyset$ and a table Q with all the queries and answers submitted to $\mathsf{SimMSign}$.

- The reduction, $\mathcal{A}'$, will act as the bank when answering $\mathcal{A}$'s queries. $\mathcal{A}'$ generates the bank's keys: $(vk^{(S)}, sk^{(S)}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$, and $(pk_{\mathcal{D}}, sk_{\mathcal{D}}) \leftarrow \mathsf{EKeyGen}(1^\lambda)$. It sets $pk_{\mathcal{W}} = (vk^{(MS)}, vk^{(S)})$ and sends $(pk_{\mathcal{W}}, pk_{\mathcal{D}})$ to $\mathcal{A}$. Note that $\mathcal{A}'$ does not know $sk^{(MS)}$.

- The adversary $\mathcal{A}$, has access to the following oracles: `BRegister, BWith, BDepo Depo`. Whenever $\mathcal{A}$ queries `BRegister` the reduction just follows the real protocol.

- When $\mathcal{A}$ queries `BWith`, again the real withdrawal protocol is invoked, except that $\mathcal{A}'$ cannot run MSign since it doesn't know $sk^{(MS)}$. Thus, when a malleable signature is needed, $\mathcal{A}'$ calls its own $\mathsf{SimMSign}$ oracle on input com_{n_1} and receives a signature σ which it forwards to $\mathcal{A}$ as part of the withdrawal. $\mathcal{A}'$ stores all the signature queries and corresponding responses in Q. $\mathcal{A}'$ then adds the corresponding coin to $\mathcal{CL}$ and sets $q_W = q_W + 1$.

- When $\mathcal{A}$ queries `BDepo`, $\mathcal{A}'$ just simulates the real protocol. If a successful deposit happens, $\mathcal{A}'$ increases the counter q_D and adds the coin to $\mathcal{DCL}$.

Note that $\mathcal{A}$ cannot distinguish between interacting with $\mathcal{A}'$ or the unforgeability game since $\mathcal{A}'$ follows the real protocols and MSign is indistinguishable from $\mathsf{SimMSign}$ (by simulatability of

the malleable signatures). Thus, the above game will still produce $q_D > q_W$ with non-negligible probability.

Next, let SigExt be the extractor defined by simulation-unforgeability of the malleable signature scheme. When $\mathcal{A}$ concludes the game (and is successful, i.e. $q_W < q_D$), we apply SigExt to all q_D successfully deposited coins, and extract all the messages and corresponding transformations. Note that all messages signed were of the form com_{n_1} and the signed messages in deposited coins must have the form $(C_{\overline{\mathtt{SN}_{(1,t_1)}}}, C_{\overline{\mathtt{DS}_{(1,t_1-1)}}})$, so by the definition of our allowed transformations, SigExt should produce for each deposited coin an initial message $com_{n_1,t}$, for $t = 1, \ldots, q_D$, and a transformation including the opening $((n_{1,t}, sk_{\mathcal{U}_t}), open_t)$, and the resulting serial number $\mathtt{SN}_{1,t}$, as well as the randomness $R_{1,t}$ used to encrypt it. Let $L_E = \{com_{n_{1,t}}, ((n_{1,t}, sk_{\mathcal{U}_t}), open_t)\}_{t=1,\ldots,q_D}$

Note that the `Deposit` algorithm will accept a deposit as successful if the first component of the serial number, $\mathtt{SN}_1$, is different from all previously deposited coins. Now we argue that for every $\mathtt{SN}_1$ that is revealed during a deposit there will be a corresponding distinct $com_{n_1} \in Q$, so we must have $q_D \leq q_W$. We will show that if this does not hold we will be able to break either the binding property of the commitment or the simulation-unforgeability of the malleable signature.

- If there exists a commitment $com_{n_{1,t}}$ in L_E that doesn't correspond to any of the commitments queried to $\mathsf{SimMSign}$ and stored in Q, then the extractor has failed to produce a valid initial message. In this case $\mathcal{A}'$ outputs the corresponding signature $(\sigma^*, m^* = (C_{\overline{\mathtt{SN}_{(1,t_1)}}}, C_{\overline{\mathtt{DS}_{(1,t_1-1)}}}))$ and thus breaks the unforgeability of the malleable signature scheme.

- If the extractor fails to produce values of the appropriate form, or for some t it produces an invalid opening $(n_{1,t}, sk_{\mathcal{U}_t}), open_t$ for $com_{1,t}$, or fails to produce $\mathtt{SN}_{1,t} = f_{\mathtt{SN}}(n_{1,t}, sk_{\mathcal{U}_t})$ or randomness R_t such that $C_{\mathtt{SN}_{1,t}} = \mathsf{Enc}(\mathtt{SN}_{1,t}, R_t)$, then the extractor failed to extract an appropriate allowed transformation $T \in \mathcal{T}$, and $\mathcal{A}'$ wins simulation unforgeability again by outputting $(\sigma, (C_{\overline{\mathtt{SN}_{(1,t_1)}}}, C_{\overline{\mathtt{DS}_{(1,t_1-1)}}}))$. (Note that if $\mathcal{A}'$ does not win in this way, then it means that the $\mathtt{SN}_1$ decrypted by the `Deposit` algorithm on the t-th successful deposit is equal to

$f_{\mathtt{SN}}(n_{1,t}, sk_{\mathcal{U}_t})$.

- Check if L_E contains two entries with the same commitment $com_{n_{(1,t)}}$ but different messages $(n_{1,t}, sk_{\mathcal{U}_t})$. If so then we could construct an algorithm which runs like $\mathcal{A}'$, and outputs $com^* = com_{n_{1,i}} = com_{n_{1,j}}$ and the two openings and thereby breaks the binding property of the commitment scheme.

If note of these cases hold, each deposit yielding a different $\mathtt{SN}_1$ corresponds to a different pair (n_1, sk_U) and thus to a different commitment com_{n_1}, which must have been signed at withdrawal. The number of successful deposits must therefore be lower than the number of withdrawals. □

Lemma 7.6.3 (Double-Spender Identification)**.** *If* (MSGen, MSKeyGen, MSign, MSVerify, MSigEval) *is simulatable and simulation unforgeable, as defined in Definition 2.3.5, and* (SKeyGen, Sign, Verify) *is existential unforgeable and* $f_{\mathtt{DS}}$ *is 2-show extractable, then the construction in Section 7.5.2 describes a transferable e-cash scheme secure against double spending.*

Proof. Let $\mathcal{A}$ be an adversary who breaks the double spending security of the transferable e-cash scheme. Then, we can construct an adversary $\mathcal{A}'$ who breaks the unforgeability of the underlying malleable signature scheme with the same probability. $\mathcal{A}'$ receives as input $(1^\lambda, crs, vk^{(MS)}, \tau_e)$.

- $\mathcal{A}'$ generates the parameters of the transferable e-cash scheme by running $par' \leftarrow \mathsf{ComSetup}(1^\lambda)$ and $par_{SN} \leftarrow \mathsf{Gen}_{SN}(1^\lambda)$ and sends $par := (1^\lambda, crs, par', par_{SN})$ to $\mathcal{A}$. It also initializes $UL = \emptyset$, $\mathcal{DCL} = \emptyset$, $\mathcal{CL} = \emptyset$, $\mathcal{AC} = 0$ and a list of all the malleable signing queries that it asks and their responses: $Q = \emptyset$.

- The reduction, $\mathcal{A}'$, will act as the bank when answering $\mathcal{A}$'s queries. $\mathcal{A}'$ generates the bank's keys: $(vk^{(S)}, sk^{(S)}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$, and $(pk_{\mathcal{D}}, sk_{\mathcal{D}}) \leftarrow \mathsf{EKeyGen}(1^\lambda)$. It sets $pk_{\mathcal{W}} = (vk^{(MS)}, vk^{(S)})$ and sends $(pk_{\mathcal{W}}, pk_{\mathcal{D}})$ to $\mathcal{A}$. Note that $\mathcal{A}'$ does not know $sk^{(MS)}$.

- When $\mathcal{A}$ queries any oracle but `BWIth` and `With`, $\mathcal{A}'$ follows the real protocol and adds the generated users to UL or updates $\mathcal{CL}$, $\mathcal{DCL}$, $\mathcal{AC}$ according to the oracles.

- When $\mathcal{A}$ queries $\texttt{BWith}$ or $\texttt{With}$, $\mathcal{A}'$ invokes the withdrawal protocol, except, not knowing $sk^{(MS)}$, it cannot compute malleable signatures. Instead $\mathcal{A}'$ will call its oracle $\mathsf{SimMSign}$ on input com_{n_1} and will forward the received signature σ to $\mathcal{A}$. $\mathcal{A}'$ will also keep all the commitments in a list L_{com}. If withdrawal was successful, $\mathcal{A}'$ adds the corresponding coin to $\mathcal{CL}$.

- $\mathcal{A}'$ simulates the honest user oracles by running the honest algorithms.

$\mathcal{A}$ cannot distinguish between interacting with $\mathcal{A}'$ or the double-spending game. The only difference is whether signatures are created by MSign (in the real game) or by $\mathsf{SimMSign}$ (via $\mathcal{A}'$'s oracle). By simulatability of the malleable signatures (Definition 2.3.3), this is indistinguishable. Thus, $\mathcal{A}$ still wins the game with non-negligible probability.

Say that the last call $\texttt{BDepo}$ outputs (pk_{i^*}, Π_G) and $\mathcal{A}$ wins the double-spending identification game. Similarly to the proof of unforgeability above, we use SigExt to extract all the messages and corresponding transformations for all q_D successfully deposited coins. Here we also need to store the $(sk_1, I_1, cert_1), \ldots, (sk_\ell, I_\ell, cert_\ell)$ from each of the extracted transformations.

If a double spending was detected for user pk_i^*, by definition of $\texttt{Deposit}$, this means that there are two coins that, when decrypted, have both the same $\texttt{SN}_1, \ldots, \texttt{SN}_j$, but have different $\texttt{SN}_{j+1}$ (for some j), pk_i^* is extracted from the two double-spending equations $\texttt{DS}_j$ and $\texttt{DS}'_j$.

1. If the extractor fails on either of these two coins, i.e. if for either coin it fails to produce valid $(sk_j, I_j, cert_j)$,or it fails to produce $\texttt{SN}_1, \ldots, \texttt{SN}_\ell, \texttt{DS}_1, \ldots, \texttt{DS}_\ell$ and valid randomness explaining the encryptions $C_{\overline{\texttt{SN}}}, C_{\overline{\texttt{DS}}}$, or it fails to produce n_j such that $\texttt{DS}_j = f_{\texttt{DS}}(I_j, n_j, sk_j, \texttt{SN}_{j+1})$ and $\texttt{SN}_j = f_{\texttt{SN}}(n_j, sk_j)$, then $\mathcal{A}'$ outputs the signature and message from the coin where extraction failed, and breaks simulation-unforgeability.

2. If the extractor is successful when run on these two coins and produces $\texttt{SN}_j = f_{\texttt{SN}}(n_j, sk_j)$ and $\texttt{SN}'_j = f_{\texttt{SN}}(n'_j, sk'_j)$ where $\texttt{SN}_j = \texttt{SN}'_j$ but $sk_j \neq sk'_j$ or $n_j \neq n'_j$, then we could construct an algorithm that runs like $\mathcal{A}'$ and breaks collision resistance of $f_{\texttt{SN}}$.

3. If the extractor is successful when run on these two coins and extracts $sk_j = sk'_j$ and $I_j \neq I'_j$, then we can construct a similar algorithm which will choose one of the two coins at random and output $(pk_j, I_j), cert_j$ as a forgery. Note that by construction, we only ever issue one signature for each I_i (because I_i is the joining order), so this algorithm will break unforgeability of the signature scheme.

   Note that if none of the above failures happen, then we are guaranteed by DS-security that the `Deposit` algorithm will extract the same $I_j = I'_j$ as was produced by the extractor, and that the `Deposit` algorithm will produce an accepting proof.

4. Finally, if double spending was detected, and none of the above occur, but $pk_{i^*} \notin UL$ or $pk_{i^*} \in UL$ but the user hasn't registered i.e. $cert_i = \perp$, then as argued above SigExt will extract the same ID_j as is produced by `Deposit`, along with a valid certificate. However, if $pk_{i^*} \notin UL$ or $pk_{i^*} \in UL$ but the user hasn't registered i.e. $cert_i = \perp$, then that means that this ID_j was never signed. Thus we can define an algorithm that runs like $\mathcal{A}'$ and outputs the extracted certificate to break unforgeability.

□

Lemma 7.6.4 (Weak Exculpability)**.** *If* (MSGen, MSKeyGen, MSign, MSVerify, MSigEval) *is simulatable, as defined in Definition 2.3.5,* (ComSetup, Com) *is computationally hiding,* $f_{\mathtt{SN}}, f_{\mathtt{DS}}$ *are anonymous, and* $f_{\mathtt{DS}}$ *is exculpable with respect to* `KeyGen`*, then the construction in Section 7.5.2 describes a transferable e-cash scheme secure against weak exculpability.*

Proof. Assume that there exists an adversary $\mathcal{A}$ that breaks weak exculpability of the transferable e-cash scheme. Then, we can use $\mathcal{A}$ to construct an algorithm $\mathcal{A}'$ that breaks unforgeability of the signature scheme or exculpability of the double-spending function. In the exculpability game $\mathcal{A}$ plays the role of the bank. In order for $\mathcal{A}$ to win it will have to either (1) make an honest user spend two coins with the same n_i (without intentionally double-spending), or (2) forge the proof Π.

The first case only occurs with negligible probability, since whenever an honest user receives a coin with `Receive`, he chooses a fresh value n_i. So every coin he receives has a different n_i.

We now modify the game by replacing all malleable signatures by simulated signatures. (By simulatability (Definition 2.3.5) this is indistinguishable even when the adversary generates the signing key.) Next, we guess which user i the adversary will attempt to frame, and replace each pair $f_{\mathtt{SN}}(n_i, sk_i), f_{\mathtt{DS}}(I_i, n_i, sk_i, \mathtt{SN}_{i+1})$ with a pair constructed with a random sk_i (if we have guessed correctly and this honest user never double-spends, i.e. $uds_i = 0$, then this will be indistinguishable by the anonymity of $f_{\mathtt{SN}}$ and $f_{\mathtt{DS}}$). Finally, we observe that in this new game $\mathcal{A}$ is only given pk_i and we never use sk_i, thus if $\mathcal{A}$ successfully frames user i, then we can break exculpability of the double-spending function. □

Note that our proposed construction does not satisfy the (strong) exculpability property defined in Section 7.1.

Lemma 7.6.5 (OtR-FA)**.** *If* (MSGen, MSKeyGen, MSign, MSVerify, MSigEval) *is simulation context hiding,* $f_{\mathtt{DS}}$ *and* $f_{\mathtt{SN}}$ *are computationally hiding and we are assuming secure channels, then, the construction in Section 7.5.2 describes an Observe-then-Receive anonymous transferable e-cash scheme.*

Proof. Given the use of secure channels when the adversary observes a coin transfer/withdrawal does not get see the coin itself. When the adversary picks two coin indices, j_0, j_1, that belong to honest users and have not been deposited, the only information he has about them is their transfer "history" (i.e. who were the previous user owners). Now assume that the coin j_b is transferred to $\mathcal{A}$. Then, $\mathcal{A}$ receives $c_b = (\sigma', (C_{\overline{\mathtt{SN}_i}}, C_{\overline{\mathtt{DS}_{i-1}}}), R_{\mathtt{SN}_i}$. Given that $\mathcal{A}$ owns the the decryption secret keys, he can obtain $\mathtt{SN} = \mathtt{SN}_1 \parallel \cdots \parallel \mathtt{SN}_i$ and $\mathtt{DS} = \mathtt{DS}_1 \parallel \cdots \parallel \mathtt{DS}_{i-1}$, however given the hiding properties of $f_{\mathtt{SN}}$ and $f_{\mathtt{DS}}$ he does not learn anything about the encoded secret keys. Thus, he doesn't learn anything about who were the previous owners of the coin and cannot link it to j_0, j_1. Also, given that the malleable signature, σ, is simulation context hiding the adversary learns nothing about the transformation description, since the signature is indistinguishable from a fresh simulated signature

on the transformed message. The proof is similar when $\mathcal{A}$ receives the coin as part of a deposit. □

Lemma 7.6.6 (StO-FA)**.** *Assuming secure channels the construction in Section 7.5.2 describes a Spend-then-Observe anonymous transferable e-cash scheme.*

Proof. In the spend then observe definition the adversary picks two coin indices j_0, j_1 and a user index i such that the coins belong to uncorrupted users, and user i is also uncorrupted. Then, observes a spending of the coin j_b, for $b \in \{0, 1\}$, to the user i. Even if the adversary owned the coin with index j_b before, i.e. know the full serial number and double spending tags, he can still not link it to a coin he observes being transferred since during transfer he sees no information about the coin itself due to secure channels. □

Lemma 7.6.7 (StR-FA)**.** *If* $(\mathsf{MSGen}, \mathsf{MSKeyGen}, \mathsf{MSign}, \mathsf{MSVerify}, \mathsf{MSigEval})$ *is simulation context hiding and simulation unforgeable,* $f_{\mathtt{DS}}$ *and* $f_{\mathtt{SN}}$ *are computationally hiding and* $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{REnc}, \mathsf{Dec})$ *is a re-randomizable encryption scheme, then, the construction in Section 7.5.2 describes a Spend-then-Receive anonymous transferable e-cash scheme.*

Proof. In this game the adversary picks two coin indices j_0, j_1 belonging to uncorrupted users. Remember that now $\mathcal{A}$ does not know the bank's deposit secret key, thus cannot decrypt. If at some point the adversary owned the j_0 coin he has seen $C_{\overline{\mathtt{SN}_k}}$ and $C_{\overline{\mathtt{DS}_{k-1}}}$. Also, when he further transferred the coin to an honest user, he received $\mathtt{SN}_{k+1}$ and computed $\mathtt{DS}_k$ (similar for j_1). Say that now $\mathcal{A}$ receives one of the coins back and he sees $C_{\overline{\mathtt{SN}_i}}$ and $C_{\overline{\mathtt{DS}_{i-1}}}$. Given that the encryption scheme is statistically re-randomizable he cannot link these encryptions to the previous ones he had, thus he cannot link the two values. Also, since the encryption scheme is semantically secure, he cannot recognize the encryption of the values $\mathtt{SN}_{k+1}$ and $\mathtt{DS}_k$ that he knows. As part of the coins the adversary also sees malleable signatures σ_k and σ_i. Because of the simulation context-hiding property of the malleable signature scheme, these values are also unlinkable.

Finally, the adversary has access to the Deposit oracle. Given the simulation unforgeability and simulatability properties of malleable signatures, we can extract all the information from σ and

simulate deposit to $\mathcal{A}$. The adversary cannot distinguish between the real and the simulated game, by simulatability of the malleable signatures. □

Lemma 7.6.8 (StR*-FA)**.** *If* (MSGen, MSKeyGen, MSign, MSVerify, MSigEval) *is simulation context hiding,* $f_{\mathtt{DS}}$ *and* $f_{\mathtt{SN}}$ *are computationally hiding and* (EKeyGen, Enc, REnc, Dec) *is a re-randomizable encryption scheme, then, the construction in Section 7.5.2 describes an Spend-then-Receive* anonymous transferable e-cash scheme.*

Proof. In the strong version of Spend then Receive anonymity game the adversary knows the deposit secret key, thus, for a coin he receives he can tell whether he owned it before. However, he should not be able to tell who of the honest users owned the coin in between. The adversary's coin is transferred to a random user ${i^*}_b$, and then the adversary picks pairs of honest users and the coin is randomly transferred to one of them. When $\mathcal{A}$, transfers the coin ${c^*}_1$ to user ${i^*}_b$ he receives $\mathtt{SN}_{i+1} = f_{\mathtt{SN}}(n_{i+1}, {sk_{i^*}}_b)$ but because of the hiding property of $f_{\mathtt{SN}}$ he cannot tell anything about ${sk_{i^*}}_b$. Then, when $\mathcal{A}$ receives coin ${c^*}_k$, he can decrypt $C_{\overline{\mathtt{SN}_k}}$ and $C_{\overline{\mathtt{DS}_{k-1}}}$ but again because of the hiding properties of $f_{\mathtt{SN}}, f_{\mathtt{DS}}$ he cannot tell anything about the secret keys of the users who owned the coin. Finally, the malleable signature σ which is part of ${c^*}_k$ also does not reveal anything because of the simulation context-hiding property of malleable signatures. □

7.7 Instantiation

In order to instantiate our scheme we need to make concrete choices for a malleable signature scheme which supports the allowable transformations $\mathcal{T}_{CSpend}$ and $\mathcal{T}_{CWith}$, a signature scheme for the signing of certificates, a randomizable public key encryption scheme, a commitment scheme (ComSetup, Com) and a zero knowledge proof system $\langle P, V \rangle$.

We can use the malleable signature construction given by Chase et al. They provide a generic construction of malleable signatures is based on cm-NIZKs [55]. There exist two constructions of cm-NIZKs, both due to Chase et al.: the first [55] is based in Groth-Sahai proofs [88], the second [56],

less efficient but simpler one is based on succinct non-interactive arguments of knowledge (SNARGs) and fully homomorphic encryption. Since efficiency is critical for payment systems we suggest using the Groth-Sahai instantiation. To do this we have to extend the $\mathcal{T}_{CSpend}$, $\mathcal{T}_{CWith}$ transformations to include the identity and show that our relation and transformations are *CM-friendly* which means that all of the objects (instances, witnesses and transformations) can be represented as elements of a bilinear group so that the system is compatible with Groth-Sahai proofs. Regarding the rest of the building blocks, a possible candidate for the signature scheme is the structure preserving signature due to Abe et al. [4] and the El Gamal encryption scheme [73]. Finally, we need to modify the above construction very slightly, to map elements of Z_p (like n_i, sk_i, I_i) into the pairing group for the transformation, but this is straightforward and does not affect the security. A full instantiation is left as an open problem since it is rather complicated especially when trying to improve efficiency of malleable signature.

7.8 Related work

Transferable e-cash was originally proposed in 1989 by Okamoto and Ohta [114, 115], who gave e-cash schemes that satisfy various properties such as divisibility and transferability but only provide weak levels of anonymity. While an adversary cannot link a withdrawal to a payment, it can link two payments by the same user; this property was called *weak anonymity* (WA). A notable work is due to Chaum and Pedersen [62], who proved that (1) transferred coins have to grow in size and (2) an unbounded adversary can always recognize coins he owned when seeing them spent later. Moreover, they extended the scheme due to van Antwerpen [137] to allow coin transfer.

The resulting scheme satisfies *strong anonymity* (SA), guaranteeing that an adversary cannot decide whether two payments were made by the same user. However, he can recognize coins he observed in previous transactions. Strong anonymity is also satisfied by the schemes constructed in [51, 27].

Anonymity for transferable e-cash has been a pretty subtle notion to define. In 2008 Canard and Gouget [50] gave the first formal treatment of anonymity properties for transferable e-cash. In addition to weak and strong anonymity, which do not yield the guarantees one would intuitively expect, they defined *full anonymity* (FA): an adversary, impersonating the bank, cannot link a coin previously (passively) observed to a coin he receives as a legitimate user (Observe-then-Receive). They also define *perfect anonymity* (PA): an adversary, impersonating the bank, cannot link a coin previously owned to a coin he receives. They showed that PA $\Rightarrow$ FA $\Rightarrow$ SA $\Rightarrow$ WA. Chaum and Pedersen [62] showed that perfect anonymity cannot be achieved against unbounded adversaries. In the same paper Canard and Gouget [50] prove that it cannot be achieved against bounded adversaries either. They therefore introduce two modifications of perfect anonymity which are incomparable to PA, namely PA1: an adversary, impersonating the bank, cannot link a coin previously owned to a coin he passively observes being transferred between two honest users (Spend-then-Observe); and PA2 (Spend-then-Receive): an adversary cannot link a coin previously owned to a coin he receives, assuming the bank is honest (If the adversary could impersonate the bank, the notion is not achievable due to the impossibility results mentioned above.) In the same paper they present a construction which satisfies all achievable anonymity properties, but is only of theoretical interest due to its inefficiency.

The first practical scheme that satisfies all FA, PA1 and PA2 is the scheme due to Fuchsbauer et al. [79], who base their construction on commuting signatures [78]. However it has two main drawbacks: (1) the users have to store the data of all transactions they were involved in to prove innocence in case of fraud; and even worse (2) when a double-spending is detected, all users up to the double spender lose their anonymity. Blazy et al. [29] addressed this problem and proposed a new scheme that overcomes the above drawbacks by assuming the existence of a trusted entity called the *judge*. This entity is responsible for the tracing of double spenders, but can also trace all the coins and users in the system at any time. This clearly contradicts one of the main goals of e-cash: as long as users do not double-spend, they remain anonymous.

7.9 Conclusions

I this Chapter we presented the first transferable e-cash scheme that satisfies all anonymity properties from the literature (FA, PA1, PA2) and more, is practical and does not assume any trusted party. We gave a formal treatment of the security and anonymity properties of transferable e-cash in a game-based fashion and defined a new anonymity requirement that was not captured before. Namely, we introduced a strengthening of Spend-then-Receive anonymity (a.k.a. PA2), which guarantees that an adversary, impersonating the bank—although able to link a coin that he previously owned to one he receives—should not be able to tell anything about the honest users who possessed the coin in between.

We presented a construction, where a coin withdrawn by the bank is signed using a malleable signature scheme. Whenever a user wishes to transfer a coin to another user he computes a mauled signature on a valid transformation of the coin. A valid transformation guarantees that the transferred coin is valid, it is indeed owned by the sender (i.e. the sender's secret key corresponds to the information encoded in the coin) and the new coin/signature created will encode the right information of the receiver i.e. the serial number $\mathtt{SN}_{i+1}$ picked by the receiver and the double spending tag $\mathtt{DS}$ formed by the owner. After k transfers, the coin's serial number is $\mathtt{SN} = \mathtt{SN}_1 \parallel \cdots \parallel \mathtt{SN}_k$, and there are $k-1$ double spending tags $\mathtt{DS}_1, \ldots, \mathtt{DS}_{k-1}$. The serial number and the double spending tags are encrypted under the bank's public key, allowing it to check for double spending on deposit. Moreover, the encryptions are re-randomized in every transfer, which ensures anonymity. We proved our scheme secure and anonymous and discussed an instantiation that can be proved secure under standard assumptions (Decision Linear and SXDH). Finally we proposed an efficient double-spending detection mechanism, which is independent of our scheme and could be used by other transferable e-cash constructions.

8 Anonymous Revocation

In the previous Chapters we presented new constructions for anonymous credentials as well as electronic payment schemes. In this Chapter we are going to discuss how is it possible to revoke a user who misbehaves. We are going to propose a generic revocation mechanism called "Anonymous Revocation Data Structure" (ARDS) that can be instantiated in several ways depending on the desired efficiency and security level. This is achieved by building generallity into the definitions already: a user's membership certificate consists of a signature on her joining order and secret key (the secret key is not revealed on the clear, the signature is the result of a two party protocol). Whenever a user is involved in an algorithm, such as Join, ProveMembership or Veirfy, a commitment to her secret key is given as input. This allows to use it in combination with group signatures or anonymous credentials schemes that require commitments to user secrets and certain types of proofs. We will present two possible instantiations of our generic construction; one based on range proofs and one based on dynamic accumulators.

8.1 Anonymous Revocation Data Structure

In this section we give the definition of a new building block called: *ARDS - anonymous revocation data structure*. ARDS is a data structure of maximum size N that allows the addition of new members, revocation of old ones and supports efficient membership queries. ARDS allows members/users to anonymously show that they belong to the current data structure, or in other words, haven't been revoked.

We assume, without loss of generality, that every time the revocation data structure is updated, this is a new revocation epoch t. Let $Commit(x; r)$ be a non-interactive commitment scheme and let par be its corresponding public parameters selected by a trusted party by running *CommitParams*. An anonymous revocation data structure scheme consists of seven algorithms which describe the interactions between Users $\mathcal{U}$, the ARDS manager $\mathcal{M}$ and the opening authority $\mathcal{OA}$:

1. *Setup*$(1^\lambda, N, par)$*:* on input a security parameter 1^λ, the upper bound on the size of the data structure N and the parameters of the commitment scheme par, this probabilistic algorithm outputs the public key of the scheme $\mathtt{PK}$ (which from now on will also include the parameters of the commitment scheme par), the ARDS manager's secret key $\mathtt{SK}_M$ and the opening authority's secret key $\mathtt{SK}_{\mathtt{OA}}$. It also initializes the revocation list for epoch $t = 0$ to be empty: $RL_t = \epsilon$ and finally sets an index $i = 0$ which will denote the users joining order.[1]

2. *UserKeyGen*$(1^\lambda, par)$*:* this is a probabilistic algorithm that on input the security parameter 1^λ and the parameters of the commitment scheme par, outputs a pair of secret and public key for a user (sk_U, pk_U).

3. *Join:* this is a protocol between the ARDS manager $\mathcal{M}$ and a user $\mathcal{U}$ who wishes to join the data structure, which we will denote as two interactive Turing machines J_M and J_{User}. By $[J_{\mathcal{U}}(\mathtt{PK}, sk_U, pk_U, r),\ J_{\mathcal{M}}(\mathtt{PK},\ \mathtt{SK}_M, i, C_U)]$ we refer to an execution of the protocol, where

[1] Note that we could have an extra algorithm called *OAKeyGen* preceding *Setup* that generates a key pair $(\mathtt{PK}_{\mathtt{OA}}, \mathtt{SK}_{\mathtt{OA}})$ for the opening authority and then $\mathtt{SK}_{\mathtt{OA}}$ is also given as input to *Setup*.

$C_U = Commit(sk_U; r)$ is a commitment to the user's secret key and r is some randomness[2]. The output of the *Join* protocol has two components:

- the user's output is a membership certificate *cert*,
- the public transcript *trans* which includes C_U and an index $i \leq N$ which denotes the user's joining order and uniquely identifies each user.

4. *Revoke(*$\mathtt{PK}, \mathtt{SK}_M, RL_{t-1}, R_t$*):* this algorithm is run by the ARDS manager to generate an updated revocation list RL_t for a new revocation epoch t. It takes as input the public key of the scheme $\mathtt{PK}$, the manager's secret key $\mathtt{SK}_M$, the revocation list of the previous epoch RL_{t-1} and the set of members to be revoked $R_t \subset \{1, \ldots, N\}$. The output is the revocation list $RL_t = \{\bigcup_{j=1}^{t} R_j, \text{aux}\}$, where aux is some additional cryptographic output. It is desirable that *Revoke* is sublinear in the size of RL_{t-1} (i.e. it works by editing RL_{t-1}).

5. *ProveMembership*$(\mathtt{PK}, RL_t, cert, sk_U, pk_U, C_U, r, i)$: is a probabilistic algorithm, run by a user, that takes as input the system public key $\mathtt{PK}$, a membership certificate *cert*, the user's secret and public keys (sk_U, pk_U) and a commitment to the user's secret key C_U with the corresponding randomness r (C_U is a fresh commitment, i.e. it doesn't have to be the same with the one that the user computed during *Join*). It also takes as input the user's joining order i and the revocation list RL_t for the specific revocation epoch t and outputs $\perp$ if the user's index i belongs to the set of revoked members i.e. $i \in \bigcup_{j=1}^{t} R_j$ or a proof of membership π_{memb} otherwise. It is desirable that *ProveMembership* is sublinear in the size of RL_t (i.e. it just performs a few lookups in RL_t).

6. *Verify(*$\mathtt{PK}, t, C_U, \pi_{memb}$*):* is a deterministic algorithm that gets as input the system public key $\mathtt{PK}$, the revocation epoch t, a commitment to the user's secret key C_U and a proof of membership π_{memb} and outputs 1 if the proof is valid and 0 otherwise (note that, interestingly, there is no need for RL_t at all).

[2]Using a commitment to the user's secret key in order to register makes our scheme anonymous. Alternatively, in a non-anonymous setting, we could have the user to register by simply showing his public key.

7. *Open(*$\mathtt{PK}, \mathtt{SK_{OA}}, t, \pi_{memb}$*):* takes as input a proof π_{memb}, the group's public key $\mathtt{PK}$, the opening authority's private key $\mathtt{SK_{OA}}$ and the revocation epoch t and if the membership proof is valid outputs the unique index i that identifies the user or outputs $\perp$ otherwise.

8.1.1 Security Model

An anonymous revocation data structure scheme will have to be correct, anonymous and secure. In this section, we formally define the above notions. When we write $cert \in Join(\mathtt{PK}, i, pk_U)$ we mean that *cert* was the output of an honest user with public key pk_U who ran the *Join* protocol with the honest manager and joined the structure with order i.

Correctness. An ARDS scheme is *correct* if the following conditions are both satisfied for any honestly generated $\mathtt{PK}, \mathtt{SK}_M, sk_U, pk_U$:

- For each revocation epoch t and any $cert \in Join(\mathtt{PK}, i, pk_U)$, if $i \notin \bigcup_{j=1}^{t} R_j$, it always holds that $Verify\ (\mathtt{PK}, t, C_U, \pi_{memb}) = 1$, where $\pi_{memb} = ProveMembership(\mathtt{PK}, RL_t, cert, sk_U, pk_U, C_U, r, i)$.
- For each revocation epoch t and any $cert \in Join(\mathtt{PK}, i, pk_U)$ such that $i \notin \bigcup_{j=1}^{t} R_j$, if $\pi_{memb} = ProveMembership\ (\mathtt{PK}, RL_t, cert, sk_U, pk_U, C_U, r, i)$, then

$$Open(\mathtt{PK}, \mathtt{SK_{OA}}, t, \pi_{memb}) = i.$$

Anonymity. Let's first discuss what anonymity means for ARDS. The idea is that a malicious group manager, when given a membership proof π_{memb} shouldn't be able to successfully find who the user that created the proof was. Given that ARDS allows the users to anonymously join the structure, anonymity will actually guarantee that the group manager cannot guess the joining order i of the user, when given π_{memb}.

In order to formally describe anonymity, we will use a simulation based definition. We create a

simulator S who is given some trapdoor information τ to the system parameters and then creates valid membership proofs without having any information about a user's membership certificate *cert* or having access to any pair of user public/secret key (sk_U, pk_U).

We will now define an adversary $\mathcal{A}$ who will play the role of the ARDS manager and will interact either with honest users who truthfully follow the protocols, this will be called the `real` game, or he will interact with the simulator and thus play the `ideal` game. We require that there is no adversary who can distinguish between the two games with non-negligible probability. Let's first define the two games:

$\texttt{REAL}(1^\lambda)$:

1. $par \leftarrow \mathit{CommitSetup}(1^\lambda)$;
2. $\texttt{PK} \leftarrow \mathit{Setup}(1^\lambda, N, par)$;
3. $\mathcal{A}^{J_{\mathcal{U}}(\texttt{PK},\cdot,\cdot,\cdot),\mathit{ProveMembership}(\texttt{PK},\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)}(par, \texttt{PK}, \texttt{SK}_M)$;
4. output b

$\texttt{IDEAL}(1^\lambda)$:

1. $(par, \tau) \leftarrow \mathit{SimCommitSetup}(1^\lambda)$;
2. $\texttt{PK} \leftarrow \mathit{Setup}(1^\lambda, N, par)$;
3. $\mathcal{A}^{J_{\mathcal{U}}(\texttt{PK},\cdot,\cdot,\cdot),\mathit{SimProve}(\texttt{PK},\tau,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)}(par, \texttt{PK}, \texttt{SK}_M)$;
4. output b

The oracle $J_{\mathcal{U}}$ represents the user side of the *Join* protocol where the $J_{\mathcal{M}}$ side is executed by the adversary. The oracles *SimCommitSetup* and *SimProve* are parts of the simulator S. The *SimCommitSetup* oracle takes as input a security parameter 1^k and outputs the parameters for the commitment scheme *par*, together with some trapdoor τ. The *SimProve* oracle takes as fixed input the public key of the scheme `PK` and the trapdoor information τ. The adversary then gives as input of his choice a revocation list RL_t for an epoch t, a membership certificate *cert*, a user's secret and public keys (sk_U, pk_U), a commitment to the user's secret key C_U with the corresponding

randomness r and the joining order of the user i. *SimProve* then will have to first run the actual *ProveMembership* algorithm and check whether the output verifies. If it does it will compute and output the new simulated proof using the trapdoor information τ. Otherwise it will just output $\perp$.

Definition 8.1.1. *An ARDS scheme is anonymous if there exists a polynomial time simulator* $S = (SimCommitSetup, SimProve)$ *such that for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *there exists a negligible function* ν *such that for all* λ *it holds that:*

$$\left| \begin{array}{l} Pr[b \leftarrow \mathtt{REAL}(1^\lambda)) : b = 1] \\ -Pr[b \leftarrow \mathtt{IDEAL}(1^\lambda)) : b = 1] \end{array} \right| = \nu(\lambda).$$

Note that in the definition given above we assume, without loss of generality, that the algorithm *ProveMembership* always outputs a proof π_{memb} that verifies, or otherwise outputs $\perp$.

Security Here we want to ensure that an unauthorized member cannot successfully prove membership and cannot make a membership proof correspond to a different sk_U that the one of the authorized user who owns the membership certificate and computes the proof.

Let $\mathcal{A}$ be an adversary who can corrupt the opening authority and obtain her secret key $\mathtt{SK}_{\mathtt{OA}}$. The adversary is also allowed to do the following:

- Generate honest users by having access to an oracle called *HonestJoin* which first triggers *UserKeyGen*() and then the interaction between $(J_{\mathcal{U}}, J_{\mathcal{M}})$. Let U^h be the set the of users generated by *HonestJoin*($\mathtt{PK}$). The adversary only learns the joining order i and the public key pk_U of the honest users.
- Introduce malicious users (i.e. users under $\mathcal{A}$'s control) by running the $J_M(\mathtt{PK}, \mathtt{SK}_M, \cdot, \cdot)$ side of the *Join* algorithm. Let U^m be the set of malicious users.
- Direct honest users to prove membership through an oracle *HonestProve*($\mathtt{PK}, \cdot$) which on input i causes the user with joining order i to output a proof of membership.
- Cause the ARDS manager $\mathcal{M}$ to run the *Revoke*($\mathtt{PK}, \mathtt{SK}_M$, $RL_{t-1}, \cdot$) protocol in order for users

in U^h or U^m of the adversary's choice to be revoked.

His purpose is to come up with a proof π^*_{memb} and a commitment C^*_U that verifies with respect to RL_{t^*}, where t^* denotes the current revocation epoch. Let π^*_{memb} open to i when the *Open* algorithm is run. Then, $\mathcal{A}$ is successful if the opening of the produced proof, i, does not correspond to any unrevoked adversarially controlled user or if for the commitment C^*_U associated to π^*_{memb} and the commitment $C^{(i)}_U$ associated to user with joining order i, it holds that $sk^*_U \neq sk_U$, where $C^*_U = Commit(sk^*_U, r^*)$ and $C^{(i)}_U = Commit(sk_U, r^{(i)})$.

Definition 8.1.2. *An ARDS scheme is secure if, for any PPT adversary $\mathcal{A}$ involved in the experiment hereafter, we have $\boldsymbol{Adv}^{sec}(\mathcal{A}) := Pr[\boldsymbol{Expt}^{sec}(\lambda) = 1] \in negl(\lambda)$.*

Experiment $\boldsymbol{Expt}^{sec}(\lambda)$

$(par) \leftarrow CommitSetup(1^\lambda)$;

$\mathtt{PK} \leftarrow Setup(1^\lambda, N, par)$;

$(\pi^*_{memb}, C^*_U, t^*, (sk_U, pk_U), (sk^*_U, pk^*_U)) \leftarrow$

$\mathcal{A}(par, \mathtt{PK}, \mathtt{SK_{OA}})^{J_M, HonestJoin, HonestProve, Revoke}$

If $Verify(pk, \pi^*_{memb}, C^*_U, t^*) = 0$ *return 0;*

$i = Open(\mathtt{PK}, \mathtt{SK_{OA}}, t^*, \pi^*_{memb})$;

If $i = \perp$ *or* $i \notin U^m \setminus \cup_{j=1}^{t^*} R_j$ *or*

$sk^*_U \neq sk_U$, *where* $C^*_U = Commit(sk^*_U, r)$ *and*

$C^{(i)}_U = Commit(sk_U, r)$, *return 1;*

Return 0;

Let's now discuss how our security notion relates to the misidentification and framing properties of group signatures. It's pretty straightforward to see that misidentification is fully covered by our security defitinion. Framing is a little bit more complicated. The general idea of a framing attack is that a set of malicious colluding users can combine their keys to produce a valid proof of membership in such a way that the opening algorithm will attribute the signature to an honest user [22]. A flavor of framing attacks is already captured by the definition given above. The adversary cannot make a

proof of membership that he created open to an honest user i.e. if $i \in U^h$ the adversary wins in the $\mathbf{Expt}^{\text{sec}}(\lambda)$ experiment defined above.

For now we do not capture framing attacks if the ARDS manager is corrupted i.e. the adversary has $\mathtt{SK}_M$. In that case the adversary could assign a joining order of an honest user to a malicious one and thus later make a proof open to $i \in U^h$.

8.2 A generic construction of ARDS using Range Proofs

In this section we are going to give our first generic construction of an anonymous revocation data structure which is based on the use of range proofs. The idea is that in each revocation epoch, the group manager will partition the unrevoked users in integer intervals according to their joining order and then the users will use range proofs to anonymously prove that they belong to one of these intervals.

By $N \in \text{poly}(\lambda)$ we denote the maximum number of users that can join the data structure which we will realize as a list of size N. Let i be a counter of how many users have joined the structure so far. The first user who joins the structure will be placed in the first position and will receive a unique identification number ID that shows his joining order $ID = i = 1$, the second is placed in the second position and receives $ID = i = 2$ and so on. The user's membership certificate will be $\sigma_{cert} = Sign(\mathtt{SK}_M, (sk_U, ID))$ where sk_U is the user's secret key.

At the beginning of each revocation epoch t, the ARDS manager, $\mathcal{M}$, partitions the unrevoked users into m subsets/intervals $S_1, \ldots, S_m$ where $S_1 \cup \cdots \cup S_m$ is equal to the set of unrevoked users which we will call current membership list. If $R_t \subset \{1, \ldots, N\}$ is the set of users to be revoked in epoch t, the manager, M, computes the subsets as integer intervals that exclude the IDs of the revoked users. As an example, let $t = 1$ (i.e. there is no revoked user up to now) and two users are going to be revoked: $R_1 = \{ID = a, ID = b\}$ for $a < b$ and $a, b \in \{1, \ldots, N\}$. Then, M will compute the current membership list UL_t to be the intervals $[1, a-1], [a+1, b-1], [b+1, N]$ and will output

$UL_t = ([1, a-1], [a+1, b-1], [b+1, N])$ together with $\sigma_{UL_t} = Sign(\mathtt{SK}_M, [1, a-1], [a+1, b-1], [b+1, N], t)$ for authentication. $Sign()$ is a signature protocol on committed values and $\mathtt{SK}_M$ the secret key of the ARDS manager. The membership list consists of $r = |\cup_{j=1}^{t} R_j| + 1$ intervals at maximum and it takes $O(r)$ time to compute it.

Say now that an ARDS member, $\mathcal{U}$, wants to prove in an anonymous fashion that he belongs to the list of unrevoked users. First he looks into the current membership list UL_t to identify the interval S_i in which his *ID* lies in. Then, he computes a range proof that his identifier belongs in that interval. Range proofs allow the user to do that in a way that he keeps both his identifier and the interval secret. Finally, the user has to convince the verifier that the interval he belongs to is a valid one, i.e. is listed on the current membership list. To do so, $\mathcal{U}$ just proves knowledge of a signature on that interval [38].

8.2.1 Range Proofs Generic Framework

Let *Commit= (CommitSetup, Commit)* be a commitment scheme and *Π =(Π Setup, Π Prove, Π Commit)* be a NIZKPoK system. Let *par* be the corresponding parameters generated by running an augmented setup *AugSetup = (CommitSetup,Π Setup)*. Let *SigS=(SigKeyGen, Sign, Verify)* be a public key signature scheme that allows signing on committed values and allows efficient NIZK proofs of knowledge of signatures on committed values. Finally, assume a semantically secure encryption scheme *EncS= (EncGen, Enc, Dec)* and a key generation algorithm for a user *KeyGen*. Then, the generic framework is defined as follows:

Setup$(1^\lambda, N, par)$: the input is the security parameter 1^λ, the maximum number of users $N = 2^{\ell-1}$ for some integer ℓ and the parameters of the commitment scheme. During setup the following steps take place:

1. Run *SigKeyGen* twice to receive two signing key pairs for the group manager: $(\mathtt{SK}_M, \mathtt{PK}_M)$ and $(\mathtt{SK}'_M, \mathtt{PK}'_M)$. Those signing keys will be used for signing the user certificates and the ranges

respectively.

2. Set $t = 0$, $RL_0 = \epsilon$, $i = 0$.

3. Run *EncGen* to receive an encryption key pair for the opening authority ($\mathtt{SK}_{\mathtt{OA}}$, $\mathtt{PK}_{\mathtt{OA}}$).

4. Publish $\mathtt{PK}$ which includes ($\mathtt{PK}_M, \mathtt{PK}'_M, \mathtt{PK}_{\mathtt{OA}}$).

UserKeyGen($1^\lambda, par$): Output a pair of secret, public key for a user: (sk_U, pk_U) using *KeyGen*.

Join: is executed between a User, $\mathcal{U}_i$, and the ARDS manager, $\mathcal{M}$, and the following steps take place:

1. The User picks a random value r and generates a commitment $C_U = Commit(sk_U, r)$, C_U is sent to $\mathcal{M}$.

2. $\mathcal{M}$ increments i and then assigns $ID = i$ to $\mathcal{U}_i$. The user together with $\mathcal{M}$ together run a signature protocol on committed values to compute $\sigma_{cert} = Sign(\mathtt{SK}'_M, (sk_U, ID))$ to bind the user's position on the structure, i, to a commitment C_U for a user secret key sk_U.

3. The user outputs $cert_i = (ID, C_U, \sigma_{cert})$.

Revoke($\mathtt{PK}, \mathtt{SK}_M, RL_{t-1}, R_t$): for a revocation epoch t where R_t is the list of the users to be revoked.

1. The ARDS manager computes a cover of the unrevoked user set $\{1, \ldots, N\} \setminus \cup_{j=1}^{t} R_j$ as the union of intervals $S_1, \ldots, S_m$.

2. For $k = 1$ to m computes a signature $\sigma_{S_k} = Sign(\mathtt{SK}_M, (S_k, t))$ in order to authenticate.

3. Return $RL_t = (t, \cup_{j=1}^{t} R_j, \{S_k, \sigma_{S_k}\}_{k=1}^{m})$.

ProveMembership ($\mathtt{PK}, RL_t, cert_i, sk_U, C_U, r_u$): if $i \in \cup_{j=1}^{t} R_j$ then return $\perp$. Else, to prove membership do the following:

1. Using RL_t, find the interval S_k that contains the user's identifier ID. In order for the user to show that he belongs to this interval without leaking ID, $\mathcal{U}_i$ first commits to S_k, ID, and

σ_{S_k}: $C_{S_k}, C_{ID}, C_{\sigma_{S_k}}$ and generates a range proof π_{RP} that his index ID lies in the range $[a, b]$ defined by S_k.

2. Then, $\mathcal{U}_i$ needs to prove that S_k is a certified interval for epoch t. So, the user provides a NIZK proof that σ_{S_k} verifies: π_{S_k}.

3. $\mathcal{U}_i$ also needs to prove that σ_{cert} is a signature on committed values (ID, sk_U) inside commitments C_U, C_{ID} by generating a NIZK proof π_{σ_i}.

4. Using $\mathtt{PK_{OA}}$ the user encrypts ID: $enc_{\mathtt{PK_{OA}}}(ID)$ together with a NIZK proof π_{ID} that the encrypted ID is the one that corresponds to the user's certificate.

5. By **com** we denote all the commitments generated and by $\boldsymbol{\pi}$ all the proofs. The proof of membership π_{memb} consists of: $(enc_{\mathtt{PK_{OA}}}(ID), \mathbf{com}, \boldsymbol{\pi})$.

Verify: In order to verify the membership proof π_{memb} one needs to verify all the proofs in $\boldsymbol{\pi}$.

Open: takes as input a membership proof π_{memb} and using the opening authority's secret key $\mathtt{SK_{OA}}$ and the revocation epoch t outputs the user's unique ID (or joining order) $ID = i$ by simply decrypting $enc_{\mathtt{PK_{OA}}}(ID)$ which is part of the membership proof. If the decryption is not successful output $\perp$.

8.2.2 Security Proof

In this section we will show that our proposed generic framework of ARDS with range proofs, satisfies the security definitions given in Section 8.1.1.

Theorem 8.2.1. *If the commitment scheme Commit is secure and the NIZK proof system Π is zero-knowledge, then the above construction is an anonymous ARDS scheme as defined in Definition 8.1.1.*

Proof. Remember that the simulator S was defined to consist of $S = (SimCommitSetup, SimProve)$. The *SimCommitSetup* oracle outputs the parameters for the commitment scheme *par* and the

NIZKPoK scheme parameters, together with some simulation trapdoor τ for the NIZK proof system. Then, *SimProve* outputs a simulated proof for a user with membership certificate *cert* and secret and public keys (sk_U, pk_U) of the adversary's choice. Let's take a closer look on how the simulator,S, works. Let S always commit to joining order $ID = 0$. In order to prepare a proof of membership π_{memb} § the simulator runs *KeyGen* to compute a secret/public key for a user: (sk_U, pk_U) and then for a revocation epoch t, commits to a random interval S_j and computes $C_{S_j}, C_{\sigma(S_j)}$ as well as a simulated range proof π_{RP} that his $ID = 0$ belongs to that range. Then, S also creates a NIZK proof that σ_{cert} is a signature on committed values $(ID = 0, sk_U)$ inside committments C_U, C_{ID}. Finally, the simulator produces an encryption of 0 under the opening authority's public key: $enc_{\mathtt{PK_{OA}}}$.

Now, let $\mathcal{A}$ be an adversary that can successfully distinguish between membership proofs which were honestly generated by *ProveMembership* and simulated proofs generated using *SimProve* with advantage ε. It is straightforward to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks security of the commitment scheme with the same advantage ($\mathcal{B}$ will simply output the same guess b as $\mathcal{A}$ does). □

Theorem 8.2.2. *If the commitment scheme Commit is computationally binding, the range proof is zero-knowledge and sound and the signature scheme SigS=(SigKeyGen, Sign, Verify) is existentially unforgeable then the above construction is a secure ARDS scheme as defined in Definition 8.1.1.*

Proof. Let $\mathcal{A}$ be an adversary as defined in Def. 8.1.1 who is given access to *HonestJoin, HonestProve* and *Join* oracles and outputs $(\pi^*_{memb}, C^*_U, t^*, (sk_U, pk_U), (sk^*_U, pk^*_U))$. Let i be the opening of π^*_{memb}. In order for $\mathcal{A}$ to win in the security experiment one of the following forgeries has to happen:

Type I: $i \notin U^m \setminus \cup_{j=1}^{t^*} R_j$. In order for this forgery to take place the adversary to either create a forged membership certificate and create a proof for it or to generate a false range proof that the index i of one of the users under his control has been revoked while it hasn't. The first case is impossible due to the assumed unforgeability of the signature scheme. The second case is also impossible since that would break the soundness of the underlying range proof.

Type II: $sk^*_U \neq sk_U$, where $C^*_U = Commit(sk^*_U, r)$ and $C^{(i)}_U = Commit(sk_U, r)$. This forgery

happens if the adversary manages to make a proof of membership for a different sk_u that the one, the membership certificate has been issued to. Luckily, due to the fact that the commitment scheme is binding this is not possible. □

8.3 A Generic Construction of ARDS Using Accumulators

Let's now see how it is possible to realize ARDS using accumulators. Here, the idea is that the joining order of the user will be used as the value to be accumulated. Whenevever a new user joins the structure he is assigned a unique ID (joining order) which also gets accumulated to the accumulator value. The user's membership certificate consists of a witness that his ID belongs to the current value of the accumulator $v = f(u, X \cup ID)$ given that u is the current value of the accumulator and X is the set of IDs of users who haven't been revoked and are members of the structure. In order for the user to prove membership in the structure he will have to provide a witness w such that $f(w, ID) = v$. Given that we want the proof of membership to happen in an anonymous fashion, the user instead of providing the actual witness will give a proof of knowledge of such a witness. Remember that this also serves as a unique identification for the user.

Let *Commit= (CommitSetup, Commit)*, *Π =(ΠSetup, ΠProve, ΠCommit)*, *par*, *SigS=(SigKeyGen, Sign, Verify)*, *EncS= (EncGen, Enc, Dec)* and *KeyGen* be as before. Let G be an efficient probabilistic algorithm that produces a secure, dynamic accumulator. Then, the generic framework of ARDS with accumulators is defined as follows:

Setup$(1^\lambda, N, par)$: the input is the security parameter 1^λ, the maximum number of users $N = 2^{\ell-1}$ for some integer ℓ and the parameters of the commitment scheme. During setup the following steps take place:

1. Run *SigKeyGen* twice to receive two signing key pairs for the group manager: $(\mathtt{SK}_M, \mathtt{PK}_M)$ and $(\mathtt{SK}'_M, \mathtt{PK}'_M)$. Those signing keys will be used for signing the accumulator value and the user certificates respectively.

2. Generate the accumulator $(f, aux_f, u) \leftarrow G(1^\lambda)$ and set $X = 0, v = u$.

3. Set $t = 0$ and $i = 0$.

4. Run *EncGen* to receive an encryption key pair for the opening authority ($\mathtt{SK_{OA}}$, $\mathtt{PK_{OA}}$).

5. Publish $\mathtt{PK}$ which includes $(\mathtt{PK}_M, \mathtt{PK}'_M, \mathtt{PK_{OA}}, (f, v))$.

UserKeyGen:$(1^\lambda, par)$: Output a pair of secret, public key for a user: (sk_U, pk_U) using *KeyGen*.

Join: is executed between a User $\mathcal{U}_i$ and the ARDS manager $\mathcal{M}$ and the following steps take place:

1. The User picks a random value r and generates a commitment $C_U = \mathit{Commit}(sk_U, r)$, C_U is sent to $\mathcal{M}$.

2. $\mathcal{M}$ increments i and then assigns $ID = i$ to $\mathcal{U}_i$. The manager computes the user's witness w such that $f(w, ID) = v$ where v is the current value of the accumulator. The user together with $\mathcal{M}$ run a signature protocol on committed values to compute $\sigma_{cert} = \mathit{Sign}(\mathtt{SK}'_M, (sk_U, ID, w))$ to bind the user's position on the structure, i, to a commitment C_U for user'a secret key sk_U and to the witness, w, that this value has been added to the accumulator.

3. The user outputs $cert_i = (i, C_U, \sigma_{cert})$.

Revoke$(\mathtt{PK}, \mathtt{SK}_M, RL_{t-1}, R_t)$: for a revocation epoch t where R_t is the list of the users to be revoked.

1. The ARDS manager removes from the accumulator all the values listed in R_t using the deletion algorithm. The new value of the accumulator will be $v' = f(v, X/\{R_t\})$ and the group manager also computes a signature on it: $\sigma_v = \mathit{Sign}(\mathtt{SK}_M, (v, t))$ in order to authenticate.

2. Return $RL_t = (t, \cup_{j=1}^{t} R_j, v, \sigma_v)$.

ProveMembership $(\mathtt{PK}, RL_t, cert_i, sk_U, C_U, r_u, v)$: if $i \in \cup_{j=1}^{t} R_j$ then return $\perp$. Else, to prove membership do the following:

1. Compute a witness w that verifies that the user's ID belongs to the accumulator value for epoch t and compute a proof of knowledge of this value π_w.

2. Then, $\mathcal{U}_i$ needs to prove that v is a certified accumulator value for epoch t. So, the user provides a NIZK proof that σ_v verifies: π_v.

3. $\mathcal{U}_i$ also needs to prove knowledge of that σ_{cert} is a signature on committed values (ID, sk_U) inside commitments C_U, C_{ID} by generating a NIZK proof π_{σ_i}.

4. Using $\mathtt{PK_{OA}}$ the user encrypts ID: $enc_{\mathtt{PK_{OA}}}(ID)$ together with a NIZK proof π_{ID} that the encrypted ID is the one that corresponds to the user's certificate.

5. By **com** we denote all the commitments generated and by $\boldsymbol{\pi}$ all the proofs. The proof of membership π_{memb} consists of: $(enc_{\mathtt{PK_{OA}}}(ID), \mathbf{com}, \boldsymbol{\pi})$.

Verify: In order to verify the membership proof π_{memb} one needs to verify all the proofs in $\boldsymbol{\pi}$.

Open: takes as input a membership proof π_{memb} and using the opening authority's secret key $\mathtt{SK_{OA}}$ and the revocation epoch t outputs the user's unique ID (or joining order) $ID = i$ by simply decrypting $enc_{\mathtt{PK_{OA}}}(ID)$ which is part of the membership proof. If the decryption is not successful output $\perp$.

8.3.1 Security Proof

We will now show that our proposed generic framework of ARDS with accumulators, satisfies the security definitions given in Section 8.1.1.

Theorem 8.3.1. *If the commitment scheme Commit is secure and the NIZK proof system Π is zero-knowledge, then the above construction is an anonymous ARDS scheme as defined in Definition 8.1.1.*

Proof. The proof is the same as for ARDS with range proofs. □

Theorem 8.3.2. *If the commitment scheme Commit is computationally binding, the accumulator is secure and the signature scheme SigS=(SigKeyGen, Sign, Verify) is existentially unforgeable then the above construction is a secure ARDS scheme as defined in Definition 8.1.1.*

Proof. The proof is again essentially the same with the range proofs case. □

8.4 Instantiations

The main purpose of our work is to give a generic framework for revocation that can be used as a building block in various scenarios. For this reason we chose not to give a single concrete instantiation; instead we discuss how to instantiate our generic constructions under various assumptions. In order to use ARDS for a specific scheme, one should select the right set of building blocks depenting on the system requirements.

8.4.1 ARDS With Range Proofs Instantiation

In order to instantiate ARDS with range proofs we need to make concrete choices for a commitment scheme *Commit*, a NIZKPoK system Π, a signature scheme *SigS* (it should allow signing on committed values and efficient NIZK proofs of knowledge of signatures on committed values) and for the encryption scheme *EncS*.

We first discuss an instantiation in the RO model under the Strong RSA assumption. A possible selection for a commitment scheme is the one due to Fujisaki and Okamoto [80] and for a NIZKPoK that allows range proofs the scheme due to Lipmaa [104]. As a signature scheme one could choose the Camenisch Lysyanskaya(CL) signature [44] and finally, an option for an encryption scheme is the Camenisch and Shoup verifiable encryption [48].

A bilinear pairing based instantiation in the standard model is also possible. Such an instantiation could use the Groth-Sahai(GS) proof techniques [88], the Abe et al. structure preserving signatures [5] and the Camenisch and Shoup verifiable encryption [48].

Schemes	Params Size	Memb. Cert. Size	Revocation List Size	Prove memb. Cost	Verify memb. Cost	Revocation Cost	Notes
NFHF1[107]	$O(N)$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$	RO model -GS
BS[32]	$O(1)$	$O(1)$	$O(r)$	$O(1)$	$O(r)$	$O(1)$	RO model -GS
LPY[103]	$O(\log N)$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$	Standard model -GS
This work (RP)	$O(1)$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$	RO & Standard model - GEN
This work (Accum)	$O(1)$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$	RO model - GEN

8.4.2 ARDS With Accumulators Instantiation

For the instantiation of the generic construction with accumulators we need concrete choices for a commitments cheme *Commit*, a NIZKPoK system Π, a signature scheme *SigS* and an encryption scheme *EncS*. Let G be an efficient probabilistic algorithm that produces a secure, dynamic accumulator.

A possible instantiation in the RO model could be given under the Fujisaki and Okamoto [80] commitment scheme, Schnorr proofs of knowledge [131] of discrete logarithms, Camenisch Lysyanskaya(CL) signatures [44], Camenisch and Shoup verifiable encryption [48] and the Camenisch Lysyanskaya dynamic accumulator.

A standard model instantiation would require pairing based accumulators as for example the one due to Camenisch et al. [42] or the one due to Nguyen [109]

8.5 Efficiency Comparisons

In this section we compare our two generic revocation mechanisms with some of the techniques used in related work. The comparisons are given in terms of the size of the public parameters, the membership certificate and the revocation list and the computational costs in order to prove membership, verify membership and perform the revocation. In the column "notes" we note in which model each revocation method is proven secure and whether it was defined a part of a group signature "GS", part of an anonymous credential scheme "AC" or is a generic mechanism "GEN". By n we denote the maximum number of user members, by R_{ac} the maximum number of accumulations and by r the number of revocations.

8.6 Related Work

Revocation has been extensively studied in the literature but almost exclusively in combination with other mechanisms. The only exception is the work due to Henry and Goldberg [94] where they formalize an *anonymous blacklisting system* (which is essentially an anonymous revocation system). They first survey the literature on anonymous blacklisting systems and categorize the existing solutions into three broad categories based on architecture, security, performance and functionality. Then, they describe an anonymous blacklist as a system that consists of five algorithms: registration, token extraction, authentication, revocation and blacklist audit protocol and also define the privacy and security properties. However, they do not provide a framework of how to construct such a scheme which is the goal of this work. Moreover, given their definitions it is not straightforward how use their defined scheme as a "plug and play" mechanism with anonymous credentials or group signatures.

The rest of the proposals of revocation mechanisms where tied to credential or signature schemes. Specifically:

Accumulators. A revocation mechanism based on cryptographic accumulators requires that the user update her witness every time that a revocation has occured; therefore, the total amount of work that each user needs to do is $\Omega(r)$, where r is the number of revoked users (if the user is very active, this can be amortized over many transactions in many time periods and thus may, overall, be very cheap); while the verifier's work is much lower: he just needs to keep track of the current value A. The credential issuer's work is constant per member addition or revocation, which is even more attractive than the trivial Merkle-tree-based non-anonymous solution outlined above.

The fact that each user needs to do $\Omega(r)$ work is unattractive. It would be more convenient, especially on constrained devices, if users did not need to do this much work. Indeed, in the non-anonymous case, this is not needed.

NNL Method. In two recent papers, Libert, Peters and Yung [103] adapt the Naor-Naor-Lotspiech [108]

(NNL) broadcast encryption scheme to the problem at hand. On a high level, the NNL algorithm allows, given a tree with $2^{\ell-1}$ leaves, of which r are revoked, to partition the set of unrevoked leaves into m disjoint sets; each such set can be represented using nodes (a, b) of the tree, such that b is a descendant of a and a leaf i is in the partition that corresponds to nodes (a, b) if i is a descendant of a but not a descendant of b. In the bilinear map setting, in the standard model, Libert, Peters and Yung construct revocable group signature schemes in which the group manager organizes the unrevoked users according to the NNL partition, and valid user, as part of his group signature, demonstrates that she is in a valid partition, i.e. that her membership certificate i corresponds to a valid NNL subset. In contrast to the accumulator-based solution, the amount of work each user has to do is independent of the total number of users revoked; on the other hand, this comes at a price: the group manager will have to perform $O(r)$ work in order to publish the updated revocation tree in each time period, where r is the number of users revoked so far. Another drawback of their solution, is that it is not generic: it is tailored for the bilinear-map setting. Also, it is a group signature, and it is not obvious how to extend the approach to anonymous credentials.

8.7 Conclusions

In this Chapter we presented a generic framework for revocation that can be used as a building block in various settings. We present an *anonymous revocation data structure (ARDS)*. We gave formal definitions of security and provide two generic constructions, one using revocation lists and range proofs and one using dynamic accumulators. Finally we discussed how our generic mechanism can be instantiated under various assumptions.

9 Conclusions

In this thesis we proposed new privacy preserving mechanisms for user authentication (anonymous credentials) and for electronic payments. We investigated the security of one of the most well-known and efficient anonymous credential schemes, U-Prove and we showed that his credential scheme cannot be proven secure under currently known techniques [15]. Specifically, we showed that the Brands blind signature (in the heart of his credential scheme) cannot be proven unforgeable in the random oracle model under any intractability assumption. Our impossibility result generalizes to a broader class of blind signatures, in particular, the blind Schnorr signature [131] and the blind GQ signature [90]. This result holds no matter how strong an assumption we make.

We then proposed a new anonymous credential scheme [14] called *Anonymous Credentials Light* that has comparable efficiency to the one due to Brands and it is provably secure (in the random oracle model). In particular, it is unlinkable under the decisional Diffie-Hellman assumption, and unforgeable under the Discrete-Logarithm assumption for sequential composition (the extension to concurrent self-composition is an open problem). For the construction, we defined a new cryptographic building block, called *blind signatures with attributes*, and discussed how it can be used in combination with a commitment scheme to directly get an anonymous credential system.

Our *Anonymous Credentials Light* can be extended to an efficient e-cash scheme [97] that moreover has the nice property of encoding of users attributes in the coins (i.e. user age, address etc.) We showed a real world application of our electronic cash scheme and specifically payment systems for subways and trains [11, 97]. We worked on a smartphone and the results are promising: a payment transaction takes about 400ms on a BlackBerry Bold 9900 (depending on the number of attributes being revealed), which we believe is quite practical.

The next proposed scheme was a transferable e-cash construction [10]. Our work had both a definitional and a constructive contribution. We gave formal definitions of transferable e-cash capturing properties that previous definitions failed to address. Then, we constructed a fully anonymous transferable cash that does not depend on any "judge" (a trusted authority responsible for double spending that can trace all coins and users in the system). The construction is based on malleable signatures for the transfer of the coins. Finally, we gave an independent mechanism for efficient double spending detection in transferable e-cash.

Finally, we proposed a generic revocation mechanism [16] that can be used as a building block for various schemes. Apart from the definitions we gave two possible instantiations of our scheme: one using range proofs and one using dynamic accumulators.

Bibliography

[1] Near Field Communication Forum. http://www.nfc-forum.org/, 2008.

[2] An open letter from us researchers in cryptography and information security. http://masssurveillance.info/, January 2014.

[3] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151. Springer, 2001.

[4] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In *Proceedings of the 18th International Conference on The Theory and Application of Cryptology and Information Security, ASIACRYPT'12*, volume 7658 of *Lecture Notes in Computer Science*, pages 4–24. Springer-Verlag, 2012.

[5] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *Proceedings of the 30th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'10*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, 2010.

[6] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In *International Cryptology Conference on Advances in Cryptology, CRYPTO'00*, volume 1880 of *Lecture Notes in Computer Science*, pages 271–286, London, UK, 2000. Springer-Verlag.

[7] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In *Theory of Cryptography, TCC'12*, volume 7194 of *Lecture Notes in Computer Science*, 2012.

[8] Zack Anderson, RJ Ryan, and Alessandro Chiesa. The Anatomy of a Subway Hack: Breaking Crypto RFID's and Magstripes of Ticketing Systems. In *DEFCON'08*, 2008.

[9] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on authenticated data: New privacy definitions and constructions. In *International Conference on The Theory and Application of Cryptology and Information Security, ASIACRYPT'12*, volume 7658 of *Lecture Notes in Computer Science*, pages 367–385. Springer-Verlag, 2012.

[10] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. Fully anonymous transferable e-cash without a judge. In *Manuscript*, 2014.

[11] Foteini Baldimtsi, Gesine Hinterwalder, Andy Rupp, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. Pay as you go. In *Workshop on hot topics in privacy enhancing technologies, HotPETSs, Vigo, Spain*, 2012.

[12] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous Credentials Light. Cryptology ePrint Archive, Report 2012/298, 2012.

[13] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. Cryptology ePrint Archive, Report 2012/197, 2012.

[14] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous Credentials Light. In *ACM Conference on Computer and Communications Security, CCS'13*, pages 1087–1098. ACM-CCS, 2013.

[15] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'13*, volume 8270 of *Lecture Notes in Computer Science*, pages 82–99. Springer Berlin Heidelberg, 2013.

[16] Foteini Baldimtsi and Anna Lysyanskaya. An anonymous revocation data structure. In *Manuscript*, 2014.

[17] Niko Baric and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *International conference on Theory and application of cryptographic techniques, EUROCRYPT'97*, Lecture Notes in Computer Science, pages 480–494, Berlin, Heidelberg, 1997. Springer-Verlag.

[18] Lejla Batina, Jaap-Henk Hoepman, Bart Jacobs, Wojciech Mostowski, and Pim Vullers. Developing Efficient Blinded Attribute Certificates on Smart Cards via Pairings. In *CARDIS*, pages 209–222, 2010.

[19] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'09*, Lecture Notes in Computer Science, pages 108–125, Berlin, Heidelberg, 2009. Springer-Verlag.

[20] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In *Pairing-Based Cryptography, Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 114–131. Springer Berlin Heidelberg, 2009.

[21] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'92*,

volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer Berlin Heidelberg, 1993.

[22] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16:185–215, 2003.

[23] Mihir Bellare and Adriana Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Proceedings of the 22th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'02*, pages 162–177, 2002.

[24] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM conference on Computer and communications security, CCS'93*, CCS '93, pages 62–73. ACM, 1993.

[25] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *International Conference on the Theory and Application of Cryptographic Techniques, EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer, 1993.

[26] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard java card. In *ACM conference on Computer and communications security*, CCS '09, pages 600–610. ACM, 2009.

[27] Marina Blanton. Improved conditional e-payments. In *Proceedings of the 6th International Conference on Applied Cryptography and Network Security, ACNS'08*, pages 188–206, Berlin, Heidelberg, 2008. Springer-Verlag.

[28] Erik-Oliver Blass, Anil Kurmus, Refik Molva, and Thorsten Strufe. PSP: private and secure payment with RFID. In *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, WPES '09, pages 51–60, New York, NY, USA, 2009. ACM.

[29] Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In *4th International Conference on Cryptology, AFRICACRYPT'11*, volume 6737 of *Lecture Notes in Computer Science*, pages 206–223. Springer, 2011.

[30] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *International Conference on Practice and Theory in Public Key Cryptography, PKC'03*, Lecture Notes in Computer Science, pages 31–46, London, UK, 2003. Springer-Verlag.

[31] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Proceedings of the 24th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, 2004.

[32] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *ACM Conference in Computer and Communications Security*, CCS'04, pages 168–177, New York, NY, USA, 2004. ACM.

[33] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer Berlin Heidelberg, 1998.

[34] Joppe W. Bos, Marcelo E. Kaihara, Thorsten Kleinjung, Arjen K. Lenstra, and Peter L. Montgomery. On the security of 1024-bit rsa and 160-bit elliptic curve cryptography. Cryptology ePrint Archive, Report 2009/389, 2009.

[35] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'00*, Lecture Notes in Computer Science, pages 431–444, 2000.

[36] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'83*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1993.

[37] Stefan Brands. Rethinking public key infrastructures and digital certificates: Building in privacy. MIT Press, Cambridge-London, August 2000.

[38] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'08*, Lecture Notes in Computer Science, pages 234–252, Berlin, Heidelberg, 2008. Springer-Verlag.

[39] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In *ACM conference on Computer and communications security*, CCS'08, pages 345 – 356. ACM, 2008.

[40] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security*, CCS'06, pages 201–210, New York, NY, USA, 2006. ACM.

[41] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005.

[42] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *International Conference on*

Practice and Theory in Public Key Cryptography, PKC'09, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500. Springer, 2009.

[43] Jan Camenisch, Maciej Koprowski, and Bodgan Warinschi. Efficient blind signatures without random oracles. In *Proceedings of the 4th international conference on Security in Communication Networks*, SCN'04, pages 134–148, Berlin, Heidelberg, 2005. Springer-Verlag.

[44] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of the 3rd international conference on Security in communication networks*, SCN'02, pages 268–289, Berlin, Heidelberg. Springer-Verlag.

[45] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International Conference on the Theory and Application of Cryptographic Techniques, EUROCRYPT'01*, Lecture Notes in Computer Science, pages 93–118, London, UK, 2001. Springer-Verlag.

[46] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 56 – 72, London, UK, 2004. Springer-Verlag.

[47] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'99, pages 107–122. Springer-Verlag, 1999.

[48] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proceedings of the 23rd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

[49] Jan Camenisch, Fischer-Habner Simone, and Kai (eds.) Rannenberg. Privacy and identity management for life. In *ISBN 978-3-642-20316-9*. Springer, 2012.

[50] Sébastien Canard and Aline Gouget. Anonymity in transferable e-cash. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 5037 of *Lecture Notes in Computer Science*, pages 207–223, 2008.

[51] Sébastien Canard, Aline Gouget, and Jacques Traoré. Improvement of efficiency in (unconditional) anonymous transferable e-cash. In Gene Tsudik, editor, *Financial Cryptography and Data Security, FC 2008*, volume 5143 of *Lecture Notes in Computer Science*, pages 202–214. Springer-Verlag, Berlin, Heidelberg, 2008.

[52] Certicom Research. *Standads for Efficient Cryptography (SEC) 2: Recommended Elliptic Curve Domain Parameters*, version 1.0 edition, 2000.

[53] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A non-interactive range proof with constant communication. In *Financial Cryptography and Data Security, FC'12*, volume 7397 of *Lecture Notes in Computer Science*, pages 179–199. Springer, 2012.

[54] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come : Easy go divisible cash. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer Berlin / Heidelberg, 1998.

[55] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'12*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300. Springer Berlin Heidelberg, 2012.

[56] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Succinct malleable nizks and an application to compact shuffles. In *Theory of Cryptography, TCC'13*, Lecture Notes in Computer Science, pages 100–119, Berlin, Heidelberg, 2013. Springer-Verlag.

[57] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. In *to appear in Computer Security Foundations Symposium, CSF'14*, 2014.

[58] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'88*, Lecture Notes Computer Science, pages 319–327, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[59] David Chaum. Blind signatures for untraceable payments. In *Proceedings of the 2nd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'82*, Lecture Notes Computer Science, pages 199–203. Springer-Verlag, 1982.

[60] David Chaum. Blind signature system. In *Proceedings of the 3rd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'83*, page 153. Springer, 1983.

[61] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer, 1988.

[62] David Chaum and Torben P. Pedersen. Transferred cash grows in size. In *International conference on Theory and application of cryptographic techniques, EUROCRYPT'92*, Lecture Notes in Computer Science, pages 390–407. Springer-Verlag, 1992.

[63] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'92*, Lecture Notes in Computer Science, pages 89–105, London, UK, 1992. Springer-Verlag.

[64] Efrén Clemente-Cuervo, Francisco Rodríguez-Henríquez, Daniel Ortiz Arroyo, and Levent Ertaul. A PDA Implementation of an Off-line e-Cash Protocol. In Selim Aissi and Hamid R. Arabnia, editors, *Security and Management*, pages 452–458. CSREA Press, 2007.

[65] Jean-Sébastien Coron. On the exact security of full domain hash. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'00*, Lecture Notes in Computer Science, pages 229–235. Springer-Verlag, 2000.

[66] C.P.Schnorr and M.Jakobsson. Security of discrete log cryptosystems in the random oracle + generic model. In *The Mathematics of Public-Key Cryptography, The Fields Institute*, 1999.

[67] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'94*, Lecture Notes in Computer Science, pages 174–187, London, UK, 1994. Springer-Verlag.

[68] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'94*, pages 174–187. Springer-Verlag, 1994.

[69] Howard A. Schmidt (National cybersecurity coordinator). National strategy for trusted identities in cyberspace. In *Cyberwar Resources Guide, Item 163, http://www.projectcyw-d.org/resources/items/show/163*, 2010.

[70] Ivan Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*. Springer-Verlag, 1999.

[71] Ivan Damgård. On σ- protocols. In *Course Notes, http://www.daimi.au.dk/~ivan/Sigma.ps*, 2002.

[72] 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, *INTRUST*, volume 7222 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2011.

[73] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of the 5th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'85*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[74] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'86*, pages 186–194. Springer-Verlag, 1986.

[75] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'13*, volume 7881 of *Lecture Notes in Computer Science*, pages 444–460. Springer Berlin Heidelberg.

[76] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'10*, Lecture Notes in Computer Science, pages 303–320. Springer Berlin Heidelberg, 2010.

[77] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 197–215. Springer Berlin Heidelberg, 2010.

[78] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'11*, volume 6632 of *Lecture Notes in Computer Science*, pages 224–245. Springer Berlin Heidelberg, 2011.

[79] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In *Proceedings of the 8th International Conference on Cryptology and Network Security, CANS'09*, pages 226–247, Berlin, Heidelberg, 2009. Springer-Verlag.

[80] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'97*, Lecture Notes in Computer Science, pages 16–30, London, UK, 1997. Springer-Verlag.

[81] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 34–41, New York, NY, USA, 2010. ACM.

[82] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *ESORICS*, pages 97–114, 2008.

[83] O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools*. Cambridge University Press, 2001.

[84] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *ACM symposium on Theory of computing, STOC'85*, pages 291–304, New York, NY, USA, 1985. ACM.

[85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[86] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. ACM Symposium on Theory of Computing, STOC'82. ACM, 1982.

[87] Philippe Golle and Kurt Partridge. On the anonymity of home/work location pairs. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A.J.Bernheim Brush, and Yoshito Tobe, editors, *Pervasive Computing*, volume 5538 of *Lecture Notes in Computer Science*, pages 390–397. Springer, 2009.

[88] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'08*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer Berlin Heidelberg, 2008.

[89] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. Anonymous credential schemes with encrypted attributes. In *Cryptology and Network Security*, pages 314–333. Springer, 2010.

[90] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'88*, Lecture Notes in Computer Science, pages 123–128. Springerg, 1988.

[91] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer Berlin Heidelberg.

[92] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[93] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'11*, volume 7073 of *Lecture Notes in Computer Science*, pages 70–88. Springer, 2011.

[94] Ryan Henry and Ian Goldberg. Formalizing anonymous blacklisting systems. In *IEEE Symposium on Security and Privacy*, pages 81–95, 2011.

[95] Thomas S. Heydt-Benjamin, Hee-Jin Chae, Benessa Defend, and Kevin Fu. Privacy for Public Transportation. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2006.

[96] Gesine Hinterwalder, Christof Paar, and Wayne P. Burleson. Privacy preserving payments on ultra-low power devices with application in intelligent transportation systems. In *Workshop on RFID Security – RFIDSec'12*, Nijmegen, Netherlands, June 2012.

[97] Gesine Hinterwälder, Christian T. Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. Efficient e-cash in practice: NFC-based payments for public transportation systems. In *Privacy Enhancing Technologies - PETS'13*, pages 40–59, 2013.

[98] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. "i know what you did last summer": Query logs and user privacy. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 909–914, New York, NY, USA, 2007. ACM.

[99] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In *Proceedings of the 17th annual international cryptology conference on advances in cryptology, CRYPTO'97*, pages 150–164. Springer-Verlag, 1997.

[100] Marc Langheinrich. Privacy by design principles of privacy-aware ubiquitous systems. In *Ubicomp 2001: Ubiquitous Computing*, volume 2201, pages 273–291. Springer Berlin, Heidelberg, 2001.

[101] Abua Abu Latanya Sweeney and Julia Winn. Identifying participants in the personal genome project by name. Harvard University. Data Privacy Lab. White Paper 1021-1. http://dataprivacylab.org/projects/pgp/1021-1.pdf.

[102] Ruogu Kang Lee Rainie, Sara Kiesler and Mary Madden. Anonymity, privacy, and security online. Pew Research Center's Internet at Carnegie Mellon University, 2013.

[103] Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In *Proceedings of the 32nd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2012.

[104] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'03*, Lecture Notes in Computer Science, pages 398–415. Springer-Verlag, 2003.

[105] Anna Lysyanskaya. Signature schemes and applications to cryptographic protocol design. In *PhD Thesis*. Massachusetts Institute of Technology, 2002. AAI0804606.

[106] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[107] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *International Conference on Practice and Theory in Public Key Cryptography, PKC'09*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480. Springer Berlin Heidelberg, 2009.

[108] Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'01*, Lecture Notes in Computer Science, pages 41–62, London, UK, UK, 2001. Springer.

[109] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology, CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer Berlin Heidelberg, 2005.

[110] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'03*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, 2002.

[111] NIST. The case for elliptic curve cryptography, 2009. `http://www.nsa.gov/business/programs/elliptic_curve.shtml`.

[112] Miyako Ohkubo and Masayuki Abe. Security of three-move blind signature schemes reconsidered. In *Symposium on Cryptography and Information Security, SCIS'03, Japan*, 2003.

[113] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In *Theory of Cryptography, TCC'06*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99. Springer, 2006.

[114] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'89*, pages 481–496. Springer-Verlag New York, Inc., 1989.

[115] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'91*, pages 324–337, London, UK, 1991. Springer-Verlag.

[116] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'99*, Lecture Notes in Computer Science, pages 223–238. Springer-Verlag, 1999.

[117] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'05*, volume 3788 of *Lecture Notes in Computer Science*, pages 1–20, 2005.

[118] Christian Paquin. U-prove cryptographic specification v1.1. In *Microsoft Technical Report, http://connect.microsoft.com/site1188*, February 2011.

[119] European Parliament and Council of the European Union. Directive 2009/136/ec. In *Official Journal of the European Union*, 2009.

[120] Rafael Pass. Limits of provable security from standard assumptions. In *ACM Symposium on Theory of Computing, STOC'11*, pages 109–118, 2011.

[121] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'91. Springer-Verlag, 1991.

[122] Martin Pirker and Daniel Slamanig. A Framework for Privacy-Preserving Mobile Payment on Security Enhanced ARM TrustZone Platforms. In Geyong Min, Yulei Wu, Lei (Chris) Liu, Xiaolong Jin, Stephen A. Jarvis, and Ahmed Yassin Al-Dubai, editors, *TrustCom*, pages 1155–1160. IEEE Computer Society, 2012.

[123] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'96*, volume 1163, pages 252–265. Lecture Notes in Computer Science, 1996.

[124] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[125] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. In *Journal Of Cryptology*, volume 13, pages 361–396, 2000.

[126] Wolfgang Rankl and Wolfgang Effing. *Smart Cards in Transportation Systems*, pages 869–891. John Wiley & Sons, Ltd, 2010.

[127] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, Pairing '09, pages 231–247, Berlin, Heidelberg, 2009. Springer-Verlag.

[128] Patrick F. Riley. The tolls of privacy: An underestimated roadblock for electronic toll collection usage. *Computer Law & Security Review*, 24(6):521 – 528, 2008.

[129] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. User Privacy in Transport Systems Based on RFID E-Tickets. In Claudio Bettini, Sushil Jajodia, Pierangela Samarati, and Xiaoyang Sean Wang, editors, *PiLBA*, volume 397 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[130] Bill Saderson. E-z pass could take toll on right to privacy. In *New York Post*, 1996.

[131] Claus P. Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'89*, Lecture Notes in Computer Science, pages 239–252. Springer-Verlag New York, Inc., 1989.

[132] Berry Schoenmakers. Some efficient zero knowledge proof techniques. Slides presented at: International Workshop of Cryptographic Protocols, March 2001.

[133] Dominique Schröder, Dominique Unruh, Sanjam Garg, Vanishree Rao, and Amit Sahai. Round optimal blind signatures. In *Proceedings of the 31st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'11*, volume 6841 of *Lecture Notes in Computer Science*, pages 630–648. Springer, 2011.

[134] Silvio Micali Shafi Goldwasser and Ronald R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17:281–308, February 1988.

[135] Latanya Sweeney. Simple demographics often identify people uniquely. Carnegie Mellon, Data Privacy Working Paper 3. http://dataprivacylab.org/projects/identifiability/.

[136] IBM Security Team. Specification of the identity mixer cryptographic library, version 2.3.0. In *IBM Research Report*, 2010.

[137] H. van Antwerpen and Technische Universiteit Eindhoven. *Off-line Electronic Cash*. Eindhoven University of Technology, 1990.

[138] Alan Westin. Privacy and freedom, 1967.

[139] Wikipedia. Prism (surveillance program). [Online; accessed 29-March-2014].