UNIVERSITY OF CALIFORNIA, SAN DIEGO

Privacy vs. Authenticity

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Björn Markus Jakobsson

Committee in charge:

Russell Impagliazzo, Chairman
Mihir Bellare
Samuel Buss
Ramamohan Paturi
Nolan Wallach
Bennet Yee
Moti Yung

1997

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to begin by thanking my advisor, Russell Impagliazzo, for his support and endless patience: given my background in Engineering and my very limited exposure to mathematical proofs when I started working with Russell, this must not have been an easy task. During our weekly meetings at Samson's (Russell's eternal breakfast restaurant), Russell has, throughout my years in San Diego, helped me finding the problems with many, many revisions of papers (some of which never became papers as a result) and helped me to learn to think. Apart from being grateful to Russell for introducing me to cryptography, for teaching me a profession, for forcing me not to take shortcuts, for all the interesting discussions, and for analyzing all my ideas (which sometimes did not take all that long due to their silliness), I also have Russell to thank for being my big brother (which is not meant in the sense of surveillance) and buddy, and for introducing me to his friends and colleagues, many of whom are now also *my* friends and colleagues.

One of these very important introductions was to Moti Yung. Interestingly, Moti immediately made it to my list of jerks, which I have later confessed to him and apologized for: I presented a poster at the rump session of Crypto '92, and Moti was one of the few to come by and look. He commented that he thought of something similar a couple of years ago, but never got around to write it up. I was furious; this was my first paper (read: my big pride,) and one of the few people who cared to look claimed to have done it already, *but* did not find it interesting enough to write down. Later, I realized that I should instead feel happy that I apparently was only a couple of years behind in some (albeit insignificant) sense. Much of my efforts have later been guided by Moti, starting during my internship at IBM T.J. Watson in the summer of '95, and I am grateful for all the time and attention he has given my ideas. One of my favorite work environments has been the coffee room of the Computer Science Department at Columbia University, where Moti and I often discussed and debugged each other's ideas. Moti would listen for a while, then think, then give a quick reply, all mixed with short blitzes of snoring sleep and jokes. What was probably most important for me was that Moti taught me how to write.

Another person of importance to my work is David Chaum; during my internship at DigiCash in the summer of '94, David inspired me to think of payment related problems and solutions, and gave me an insight into the business side of cryptography. I also got to know Niels Ferguson this summer, and have had many interesting discussions with Niels since. I tried time after time, for almost a year, to come up with efficiency improvements on Niels' Crypto '93 payment scheme, each one of which got quickly executed by Niels. After this year of trying, I changed my strategy (and this was an important change): Instead of patching up problems one by one, I learned to start from square one. Later, by the time I started to develop the schemes constituting this thesis, I got to know Stefan Brands. There was nothing published in terms of payment schemes with revokable anonymity at this time, and we discussed what such a scheme would mean, and how it could possibly be achieved. My first construction (which is an unreadable UCSD Tech report - this was before Moti taught me to write) was strongly inspired by these discussions.

During the development of the articles that constitute this thesis, I have benefitted from discussions with several people, who have analyzed the protocols and given valuable suggestions. These people are (in alphabetical order) Jan Camenisch, Giovanni Di Crescenzo, Markus Michels, Tal Rabin, Matthias Schunter, Markus Stadler, and Rebecca Wright. I have also enjoyed and benefitted from discussions with Mihir Bellare, Joan Boyar,

Ivan Damgård, Matt Franklin, Dalia Malki, David M'Raïhi, Torben Pedersen, and Heather Woll.

I would like to thank Robert Hecht-Nielsen for inspiring and encouraging me to apply to the PhD program, and for believing in me; I want to thank my friends for helping me through those times that were not entirely joyous, and for making other times compensate for the tough times; also, I want to thank for the financial support I have received, from: Russell Impagliazzo (National Science Foundation, NSF YI Award CCR-92-570979, and Sloan Research Fellowship BR-3311,) E. Lundströms Stiftelse, Lars Hiertas Minnesfond, The Sweden-America Foundation, The California Grant, Thomsens legat, Michael Hansens stipendium, Tranchells stipendium, Helge Ax:son Johnsons Stiftelse, Anna Whitlocks Minnesfond, The Paulson Award, S-E Bankens Skånestipendium, and The Royal Swedish Academy of Sciences (von Beskows Stiftelse, G.S. Magnussons Stiftelse, and Claes Adelskjölds Stiftelse.)

Last, but not least, I want to thank my parents for teaching me to question everything. Poor them, what a terrible thing to teach your kid.

VITA

July 18, 1968	Born, Malmö, Sweden
1991	Civilingenjör i Datateknik, Lund Institute of Technology, Sweden
1997	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

M. Bellare, M. Jakobsson, M. Yung, "Round-Optimal ZK Arguments based on any One-Way Function," Advances in Cryptology – Proceedings of Eurocrypt '97, pp. 280–305.

A. Herzberg, M. Jakobsson, S. Jarecki, Hugo Krawczyk, M. Yung, "Proactive Public Key and Signature Systems," Proceedings of the 4th ACM Conference on Computer and Communications Security, 1997, pp. 100–110.

M. Jakobsson, M. Yung, "Distributed 'Magic Ink' Signatures," Advances in Cryptology – Proceedings of Eurocrypt '97, pp. 450–464.

M. Jakobsson, M. Yung, "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System," Advances in Cryptology - Proceedings of Financial Cryptography '97.

M. Jakobsson, M. Yung, "Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers," Advances in Cryptology – Proceedings of Crypto '96, pp. 186–200.

M. Jakobsson, Kazue Sako, Russell Impagliazzo, "Designated Verifier Proofs and Their Applications," Advances in Cryptology – Proceedings of Eurocrypt '96, pp. 143–154.

M. Jakobsson, M. Yung, "Revokable and Versatile Electronic Money," Proceedings of the 3rd ACM Conference on Computer & Communications Security, 1996, pp. 76–87.

M. Jakobsson, "Ripping Coins for a Fair Exchange," Advances in Cryptology – Proceedings of Eurocrypt '95, pp. 220–230.

M. Jakobsson, "Blackmailing using Undeniable Signatures," Advances in Cryptology – Proceedings of Eurocrypt '94, pp. 425–427.

M. Jakobsson, "Reducing costs in identification protocols," Rump session, Crypto '91.

M. Jakobsson, "Machine-Generated Music with Themes," International Conference on Artificial Neural Networks, 1992.

G. Jakobsson, M. Jakobsson, M. Persson, "NO till vardags," ISBN 91 88070 14 X

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Cryptography, with a focus on electronic commerce.
Associate Professor Russell G. Impagliazzo

ABSTRACT OF THE DISSERTATION

Privacy vs. Authenticity

by

Björn Markus Jakobsson
Doctor of Philosophy in Computer Science
University of California, San Diego, 1997
Associate Professor Russell G. Impagliazzo, Chair

In many cryptographic settings, there is a trade-off between *privacy* and *authenticity*. We analyze this trade-off in the context of electronic commerce: On one hand, we have schemes whose perfect and un-revokable privacy makes them susceptible to attacks such as blackmail and money-laundry. On the other hand, we have schemes where the authenticity of the funds (in the sense of ownership) is guaranteed by sacrificing user privacy in its entirety. In this work, we propose a model and protocols balancing the needs for privacy against those of authenticity.

In our proposed e-money system, all users enjoy full privacy, but both value of funds and user anonymity can be *revoked* or suspended unconditionally, by the cooperation of a quorum of banks and consumer rights organizations. Our method employs diffusion of a task into distributed modules; doing so, it enables a stronger and more realistic adversarial setting, and achieves increased security, privacy, availability and functionality without introducing any noticeable disadvantage. The result is a scheme that protects against privacy aided attacks, such as blackmail and money-laundry, as well as the "ultimate crime," where an active attacker gets the bank's secret key or forces the bank to give "unmarked bank notes". Our system, unlike all previous anonymous systems, can prevent all such crimes from successfully being perpetrated, and employs *revocation* to do so.

One important building block implements the desired balance between privacy and authenticity for digital signatures. We introduce *magic ink signatures*; such signatures require a quorum of servers to be produced, and a (possibly different) quorum to be unblinded. We present and use an efficient and robust scheme for magic ink DSS signatures.

The mechanisms introduced to balance the need for anonymity against the need to be able to revoke it, together with the notion of *challenge semantics* that we introduce, provide us with a very *versatile system*, a second important goal of our investigation. The proposed scheme is efficient and allows for numerous modes of payments.

Chapter I

Introduction

I.A The Need for Balanced E-Money Systems

It is provisioned that in the near future, electronic money systems will be part of the economy, exploiting advances and accessibility of networking technologies to homes, offices, and other locations. Electronic money, associated with cryptographic security against forgery and user anonymity with respect to payments, allows a practical and flexible solution to the problem of how to administer the transfer of funds. This flexible tool can enhance the prevalence and capability of fund transfers while simultaneously increasing its users' privacy.

Originally, perfect anonymity of users of e-money was advocated very strongly, granting privacy to users with respect to their purchase patterns. However, as recently noted, it may be a tool that can easily be abused by criminals lest extreme caution is taken when designing the system. The availability of such a strong, flexible access-control capability as anonymous money may become dangerous (serving as a double-edged sword): Perfect anonymity enables "perfect" crimes like blackmailing [76], money laundry and bank robberies [49]. The reason why we label these crimes *perfect* is that in a setting with perfect anonymity, the skilled perpetrator would be provably untraceable. Thus, we argue that a solution that obtains perfect privacy in the *cryptographic sense* is far from perfect in the *social or economical sense*. In fact, the danger of anonymous e-money may prevent its legality ([34, 83]). We therefore argue that the anonymity must be *revokable* in cases where it is authorized by court due to immense suspicion of a violation of the law, or due to legal information inquiries, regulatory acts, coin losses, etc.

However, it is unsatisfactory if a coin or its anonymity only can be revoked by its owner (where he gives the bank the blinding factors used when the coin was withdrawn, so that the bank can blacklist the coin). Whereas this *is* a possibility if the user loses his smart card with the coins on, and has a backup, it is *not* possible if the user has been the victim of a blackmail attack in which he has never learnt the blinding factors. This could happen if the attacker forced the victim to use blinding factors *set by the attacker* in the withdrawal protocol of the coin from the victim's account. Furthermore, it may be that the user whose coin needs to be traced is not willing to cooperate with the bank, e.g., if he obtained the coin through a bank robbery. Still yet, anonymity can be abused for money laundering and the sale of illegal goods. Here, it has been argued that a criminal would not dare to use e-money for illicit transactions, since it could be proven that he

received the corresponding payments. We dispute this opinion: Since the criminal can easily ascertain that he cannot be tied to the delivery (into a back room operating shop), and since he can claim that the payment was for other merchandise (using a front room operating shop) the ability to prove that a certain payment was made will not provide any evidence of the alleged offense. Therefore, the revokational ability must reside in an entity that will not cooperate in the illegal conduct we are trying to discourage. However, giving the revocation power to a *single entity* (such as the bank) enables abuse against individuals, thereby bringing us back to square one. Electronic money systems represent a model of highly sensitive systems where trust and protection are much needed, and for which it is vital to win consumer confidence and support. We believe that this support has to be gained by not forcing users to put a lot of trust in individual entities for the security of their funds and integrity, and by supplying the user with a practical product. We do so by distributing the power for removing the privacy so that entities with different goals (e.g., commercial entities, law enforcement, and consumer representative organizations) are required to cooperate in order to revoke anonymity. In order to increase availability and maintain security against a small subset of malicious revokational entities, we want to allow any valid quorum of revokation entities to be able to perform their task, even if they were not involved in preceding operations.

I.A.1 Outline

In this thesis, which covers material mainly from [49, 50, 51], an electronic money system with revokable anonymity is presented, first on a high level, outlining the main protocols and their properties; then, we suggest one possible implementation of an important building block, a magic ink signature scheme for DSS signatures.

We start by explaining what our electronic money system achieves, what attacks we consider, and, in general terms, the structure of our solution and our proposed building blocks. We then put our work in the context of related work (section I.B). The main body of the paper contains two (somewhat independent) parts: First, the electronic money scheme is presented on a high level, using magic ink signatures as a building block. Then, we introduce this signature type and present a DSS based implementation.

An Electronic Money Scheme: We begin by presenting the system model in section II.A. Here, the setting is explained, the different participants introduced, and system events outlined. Then, in section II.B, we define the different attacks we consider, and state the requirements our solution satisfies. We define concepts in section II.C, and introduce our system in section II.E. Here, we use magic ink signatures as a building block. In section II.F, we discuss how the scheme is made more versatile using challenge semantics. This allows us to add functionality such as divisibility, checks, credit purchases, micro payments and a fair exchange without any noticeable increase of complexity. In section II.G, we discuss the efficiency of the scheme, and in II.H, we introduce a method for database reduction to minimize the amount of data to be stored by the Bank. Finally, we outline security aspects of the system in section IV.B.

Magic Ink DSS Signatures: In sections III.A– III.D, we specify the requirements on the signature generation and tracing protocols, review DSS signatures, and describe the model. Then, in section III.E, we communicate the intuition of our magic ink scheme, using a simplified version, followed by a description of our assumptions in section III.F. We explain

what tools we will use in section III.G, and then introduce the Magic Ink DSS signature generation and tracing protocols in section III.H– III.I. Security claims and proofs are presented in section IV.A.

Novel methods: The work constituting this thesis introduces:

- A stronger attack model than other schemes with revokable anonymity, and methods to ensure security in this attack model.
- The notion of *dual verification signatures*, signatures with two different modes of verification, where one can be off-line and the other require interaction with an authority.
- The notion of *challenge semantics*, used to increase the functionality of the payment scheme.
- The notion of coins with no pre-set number of allowed spendings, but instead with a limited allowed *value* of spendings.
- The use of threshold cryptography to control when actions can be performed.
- A flexible tool for generation of blind signatures, and selective unblinding of these, in a distributed setting where participants do not trust each other.
- Two new ways of implementing robustness: (1) using a verification protocol for undeniable signatures, and (2) *destructive robustness*, in which protocol properties may be cancelled for failed transcripts in order to achieve greater simplicity and efficiency.

I.A.2 What we achieve

Our **first objective** is to implement payer anonymity, and to aid law-enforcement by allowing revocation of anonymity and value of funds, allowing traceability and blacklisting as part of an off-line e-money system (once anonymity is broken, further measures can be taken in accordance with the law). We want the revocational power to be distributed, allowing *any* valid quorum to revoke anonymity (or value) selectively. This should hold even if the same entities were not involved in previous transactions (i.e., withdrawals), and even if some of the participants of these previous transactions are corrupt. In other words, we want to implement robustness in the context of several interacting protocols between replaceable participants. The system achieves *unforgeability* of funds, *anonymity* for the honest user, and legal *traceability* of funds. Furthermore, the system achieves *revocability* of funds, *refundability* (in case of incorrect action by the Bank, the Judge can enforce a refund), and *framing-freeness* (i.e., that a user cannot falsely be accused of performing an attack).

Our **second objective** is to achieve high versatility and efficiency, meeting flexibility and convertibility requirements put forth in [61]. Our system is easily extendible to include practical functions like coin divisibility (the ability to spend any fraction of a coin and save the rest for later payments,) check payments, credit card payments, micropayments (e.g., [42, 52, 53, 71, 80]), and a fair exchange [46] (ensuring that neither the payer nor the payee cheats each other). Similarly, we can allow certain coins to be eligible for deposit only after certain triggering events have occurred (e.g., implementing surety bonds [54].) These expansions in terms of functionality are obtained by employing the

means used for revocation and by introducing the use of *challenge semantics*, i.e., an extension of functionality achieved by the use of special forms of challenges, used to signal behavior/function.

Our **third objective** is robustness against spurious faults (overspending robustness). In contrast, systems like [9, 10, 12, 13, 84] have a weakness in that if there is a fault in a user module, allowing a coin to be overspent, then the transcripts of the spent coin allow anybody who sees them to *further* overspend the coin, *without limitations* (and in a way that makes it appear as if it was the withdrawer who performed the overspendings.) Apart from making the liability of the user disproportionately large, it may also be very dangerous in the context of national (economic) security. Therefore, we will require our system to implement *overspending robustness*, i.e., if a coin is overspent, no information enabling further overspending is leaked. Thus, in our system the user will only be liable for the overspendings he or she actually performed.

I.A.3 Avoiding abuse

By striking an appropriate balance between the rights of the users and those of commercial and legal entities, we aim to limit the threat of the following types of attacks:

- **User abuse:**
  - **Forgery:** This is the generation of valid funds representations by an entity other than the Bank.
  - **Overspending:** This is when a user by repeated spending spends a properly withdrawn coin for a higher value than it has.
  - **Money laundry:** Money laundry is when one or several participants hides income of questionable nature by making it appear that the income was generated by another business venture. Whereas this problem is of a more social than technical nature, we need to be able to trace payments in order to secure evidence for prosecution in cases where a crime is believed to have taken place.
  - **Blackmail:** Blackmail is when an attacker forces a user to withdraw money for him, in a way that only the attacker has a representation of the funds. Today's system of physical representation of value does not *prevent* this type of crime, but stepping over to perfectly private electronic money would in fact *aid* such criminal abuse. Allowing selected funds to be traced restricts the criminal to today's methods.
  - **Bank robbery:** We consider two types of bank robbery attacks. In the first, an issuer (comprising banks and other entities holding secret keys used for generation or tracing of funds) may be coerced by an attacker to reveal its secret key (e.g., internal fraud). In the second, which is possible if withdrawals can be blinded, an attacker may try to coerce the issuer to get e-money by forcing the issuer to engage in a blinded, (perhaps non-standard) protocol for withdrawal. We call this forced blinding *blindfolding*, and the two attacks *bank robberies*. If the authenticity of a coin is solely based on the signature function of the issuer, then it is not practically possible for the signers to trace these types of illegally issued coins, leaving the offender totally untraceable. This, of course, can be a major

problem, particularly in light of international terrorism and adversarial foreign governments.

- **Bank abuse:**
  - **Malicious tracing:** The Bank, being a commercial entity, may wish to sell usage statistics to advertising agencies and other companies with specific interest in user behavior. We aim to limit this type of abuse by the introduction of the Ombudsman[1], who must be involved in each tracing. The Ombudsman cannot be tricked to perform a tracing he does not agree to (or is forced to by a judge.)
  - **Framing:** For legal reasons (to make evidence useful in court) it must be impossible for the Bank (and the Ombudsman) to frame a user, making it appear that he or she engaged in a particular transaction, whereas this in fact is not the case.
  - **Embezzlement:** This is an attack in which a user loses funds due to cancellation of the same.
- **Ombudsman abuse:**
  - **Malicious tracing:** It must not be possible for the Ombudsman to obtain any data about user payment behavior, even when cooperating with the Bank in order to allow the Bank to trace.
  - **Framing:** (as described in "Bank abuse" above.)
  - **Forgery:** (as described in "User abuse" above.)
  - **Criminal Cooperation:** The Ombudsman may be interested in cooperating with an attacker for a trace not to be possible, or to reveal incorrect results.
  - **Embezzlement:** (as described in "Bank abuse" above.)

I.A.4 Method

At the heart of our method are a number of techniques and notions: first is the distribution of the Bank and the Ombudsman. Secondly, we design the system to allow for the use of proactive sharing of the Bank/Ombudsman signature function, protecting the system against a strong infiltrating attacker. Then, we reorganize the memory, making one entity (the Bank) keep storage for another entity (the Ombudsman) in a secure repository. This reorganization and diffusion of function is a strengthening mechanism that protects against insider attacks (which is the most prevalent attack on financial institutes and systems,) and also makes the distribution of the (virtually) storage-free entity – the Ombudsman – inexpensive to perform.

To cope with bank robbery attacks (which is the strongest suggested attack on a monetary system) as well as the other attacks, without abandoning user anonymity, we use *dual verification signatures* [49], i.e., signatures such that their verifications under some circumstances need an "authenticator", e.g., the Bank or the Ombudsman. We will need two primitives:

[1] Ombudsman (pl. ombudsmän): word of Scandinavian origin for a government official appointed to represent individuals against abuses and capricious acts of public officials.

1. **A signature scheme that is not blindfoldable by the signature receiver.** In order to prevent a bank robbery, we need a signature scheme that cannot be blinded. A publicly verifiable signature scheme cannot be used, since it can always be blinded (in principle, if not efficiently, using Yao's secure computation protocol where the Bank secretly employs its signing key and the attacker gets the resulting value on his secret input [86]). We will employ a three-party protocol for payments where the Ombudsman gets involved in the signing.

2. **A mechanism for anonymity.** In order not to sacrifice user anonymity, we need a method to blind the above signatures *to* the Bank and Ombudsman *by* the Bank and Ombudsman. These entities will cooperate in producing the above signature in a way that prevents either of them from associating coins with identities without the cooperation of the other.

The above two primitives are implemented by a tagging of each coin, where the coin corresponds to a signature by the Bank and the Ombudsman. These tags are stored together with withdrawer information, such as identity, but cannot be correlated to a coin unless a quorum of Bank and Ombudsman servers cooperate in a tracing protocol. A coin is valid if its signature is valid (which is publicly verifiable, and is the normal-case condition) and, (in special cases, such as after a bank robbery) if the Bank/Ombudsman database contains a tag corresponding to the coin. Thus, the tagging and tracing functions further use cryptographic tools that are applicable under the suggested distributed control. In order to limit the trust assumption to a minimum, we need a *quorum* blind signature generation scheme, and a *quorum* unblinding scheme, where it is always possible for an arbitrary Bank/Ombudsman quorum to perform its task. To solve this problem, we introduce what we call *magic ink signatures* [50].

I.A.5 Tools for Privacy and Authenticity

Generally, when an authority signs "access tokens", "electronic coins", "credentials" or "passports", it makes sense to assume that whereas the users can typically enjoy the disassociation of the blindly signed token and the token itself (i.e., anonymity and privacy), there may be cases which require "unblinding" of a signature by the signing authority itself (to establish what is known as "audit trail" and to "revoke anonymity" in case of criminal activity).

The physical analog of "blind signatures" of Chaum is a document and a carbon paper put into an envelope, allowing the signer to transfer his signature onto the document by signing on the envelope, and without opening this. Only the receiver can present the signed document and the signer cannot "unblind" its signature and get hold of the document signed.

This leads us to consider a new type of signature with the following physical parallel: The signer places a piece of paper and a carbon paper in an envelope as before (but the document on the paper is not yet written). A second piece of paper is placed on top of the envelope. The receiver then writes the document on this second piece of paper using *magic ink*, i.e., ink that is only visible after being "developed". Due to the carbon paper, this results in the document being written in visible ink on the internal paper. Then, the signer signs the envelope on the second piece of paper, also using magic ink. Again, the

writing is being copied to the internal document. The receiver gets the envelope containing the signed message (the internal paper) and the signer retains the second piece of paper with the message and signature written in magic ink. Note that the signing is not blinded forever to the signer: Should the signer need to unblind the document, then he can develop the magic ink and get the document copy. We call this new type of signature a *magic ink signature*. We present an efficient method for distributively generating magic ink signatures, requiring a quorum of servers to produce a signature and a (possibly different) quorum to unblind a signature. What the distribution enables us is to implement the unblinding with separation in time – i.e., allowing the development of the magic ink at some point, but not earlier. This is impossible in the centralized case (what can be done late can be performed earlier if there is no limiting factor such as the quorum control in a distributed case).

Note that requiring various actions of a quorum of distributed agents regarding a specific signature value needs a careful flexible design. For example, we cannot require that in each action the same identical quorum of agents be present. This may otherwise, paradoxically, reduce the availability of the service as the distribution level grows (whereas one of the initial reasons in distributing the service was increased availability). For the same reason, and quite counter-intuitively, it may also force us to put *more* trust in individual servers with a higher degree of distribution, unless care is taken.

The scheme is robust, and the unblinding is guaranteed to work even if up to a threshold of signers refuse to cooperate, or actively cheats during either the signing or the unblinding protocol.

I.A.6 Tools for Robustness

We introduce two new methods of obtaining robustness:

- **Undeniable Signature Based Robustness** [50]
  We employ a verification protocol for undeniable signatures [15, 16] for the protocol participants to prove to each other that they performed the correct computation.

- **Destructive Robustness** [50]
  We introduce *destructive robustness* for increased efficiency. This is a method comprising two protocols: In a first protocol, some entity or entities can detect incorrect results, but cannot pinpoint who cheated. In a second protocol, the cheater is pinpointed, while protocol properties (such as privacy) may be destroyed for the related session.

I.B Related Work

Since the introduction of anonymous payment schemes by Chaum [17], and practical such schemes by Chaum, Fiat and Naor [18], a large variety of electronic cash schemes offering user privacy has been designed, e.g., [9, 22, 29, 32, 63, 64, 62]. Recently, though, it has been noticed that perfect privacy in the cryptographic sense is seldom perfect in an economical or social sense. The reason is that the privacy may be abused if it cannot be controlled; one example of this was early pointed out by von Solms and Naccache [76], who showed that schemes with user anonymity are susceptible to *blackmailing* attacks. Later, it was also noticed that *money laundry* would be facilitated by anonymity. The existence of

such attacks may endanger government support of suggested schemes and make it a risky commercial venture as well; thus, the recent trend has been towards schemes with revokable anonymity.

Whereas privacy in the e-money schemes with *perfect* privacy typically was implemented using some form of blind signatures (introduced by Chaum [17]), restricted versions employing a third party are used to obtain revokable privacy. In [12], Brickell, Gemmell and Kravitz developed a trustee-based electronic cash scheme that can be used to prevent some anonymity-aided attacks. In their model, there is a Bank and a trustee. The trustee can, given a spent coin calculate the identity of the withdrawer. This holds for all *correctly* withdrawn coins (thereby excluding coins obtained through a bank robbery.) Camenisch, Piveteau and Stadler [14, 78, 79] independently introduced "fair blind signatures," with a second tracing option added, from a given withdrawal transcript to a description of the corresponding coin. In such a signature scheme (similar to what was used in [12]), the signature receiver puts a pseudonym into the signature, allowing a judge later to unblind the signature by calculating a pseudonym from a signature or vice versa. This two-party relative of magic ink signature schemes is readily applicable to electronic money schemes, obtaining a scheme similar to that presented in [12]. Cut-and-choose based solutions enabling the trustee to be involved only in tracings were introduced in [12, 79], and further work in this area gave rise to efficient schemes (avoiding cut-and-choose based methods) where the trustee does not need to be engaged in the withdrawal protocol. This was done independently by Camenisch, Maurer and Stadler [13] and by Davida, Frankel, Tsiounis and Yung in [31, 23, 82], the latter using so called *indirect discourse proofs*, in which the withdrawer proves to the Bank that the trustee will be able to perform the trace if needed. In the work by M'Raïhi [58], a smart-card adapted scheme based on an approach reminiscent to the fair blind signatures, but with slightly stronger trust assumptions, was suggested. Another related solution has recently been proposed by Petersen and Poupard [67].

Functionally, these schemes have many similarities to the one we propose, although we strengthen the protection against attacks, increase the degree of generality and the level of distribution, and allow for customization of the resulting scheme, giving us a strong, flexible and efficient e-money scheme. Using challenge semantics, we lower the storage costs of users, and increase coin functionality compared to other proposed schemes. We also introduce a method for reduction of the bank database, allowing the amount of data stored to be drastically reduced. We note that both the challenge semantics and the database reduction methods are made possible by the general architectural framework we introduce. Also, our implementation allows mutually distrusting parties to perform the wanted calculation in a robust manner, whereas some of the previously proposed protocols instead assume that some entities and servers of the same entity trust each other.

We also strengthen the protection against weaknesses due to failures. In schemes like [9, 10, 12, 13, 84], the secret key needed to perform a payment is leaked in its entirety if the corresponding coin is overspent (which may occur due to hardware failures of the user device.) This allows participants other than the withdrawer of a coin to *further* overspend the coin; these are payments that the withdrawer of the coin will be responsible for. Our scheme does not have this weakness: it is *overspending robust*.

Whereas it is easy to see the relationship between our scheme and other schemes with revokable or perfect privacy, there are also schemes based on other types of architectures, such as the credit card architecture. Our e-money solution is (remotely) related to

the anonymous credit card of Low, Maxemchuk and Paul [55], in which they suggest an extension to the standard credit card system, allowing an intermediate degree of anonymity combined with the ability to trace purchases of credit card type (i.e., not between arbitrary participants, but only between users and registered shops). Other extensions to the existing credit card payment system are [3, 56, 61, 75]; here, however, anonymity is not an issue, but backwards compatibility and simplicity are emphasized.

None of the previously introduced signature-based schemes protect against the bank robbery attack introduced in our work. Whereas this stronger attack model and its realization in principle are perhaps mainly of theoretical interest, the means we employ to foil the attack and the added functionality arising from the use of these methods are of a practical nature. Protecting against bank robberies will require methods different from those proposed earlier, as this attack is much stronger than other attacks. We claim that for each normal withdrawal protocol in schemes based on publicly verifiable digital signatures, there exists a *blindfolded* withdrawal protocol, i.e., a protocol (that is easy to calculate) producing blind signatures, whereas the intended withdrawal protocol is not necessarily blind. This is a special case of a result of Yao's [86], stating that each publicly verifiable function can be calculated from private inputs in a way that does not betray the secret inputs, which in this case are the exact form of the coin and the secret used to sign. Yao's result thus implies that if only a publicly verifiable signature is used to authenticate a coin, then a blindfolded withdrawal can be made such that neither the attacker has to reveal his identity or the form of the withdrawn coin, nor has the Bank to reveal the secret allowing it to sign.

This work is based on three articles by Jakobsson and Yung. In [49], we introduced the model we employ, and gave a two-party protocol solution that protects against the user attacks considered, and partially protecting against the Bank and Ombudsman attacks. Our solution bears resemblance to the *weak blind signatures* of Franklin and Yung [33]. Two important differences are that first, we allow for an off-line verification in the common case, whereas they default on each (weak) signature verification being on-line with a checking center; second, we have mechanisms to enforce legal responsibilities from the Bank and the Ombudsman (which imply the need for real signatures rather than weak ones).

In [51] we distribute the above mentioned two-party solution to avoid the disadvantage of non-threshold systems with their centralized and highly sensitive functions. We believe that for an e-money system to be usable and economically viable, it is crucial to increase the overall trust. This issue is pursued in [51]. Here, we distributed the sensitive functions into a system where components may be held by different entities (this was typically done for a one-party function in a system, but there we did it for a function that is already a two-party construction.) We employed proactive tools to also enable resistance to strong attacks that are expected over open networks (the Internet.) At the same time, the storage requirements were reduced, and the system simplified. The sensitive actions of producing valid coins and revoking anonymity were done only under quorum agreement among the distributed entities, giving additional control via the distribution of authority amongst various bodies.

This result uses magic ink signatures, which we introduced in [50]. In recent years, various notions of distribution of cryptographic functions (signature and encryption) among independent agents were considered. The typical added functionality of such distributed functions include increased security of the secret key, increased availability of service, and

increased flexibility of access, the latter by requiring a quorum to access information (as in, e.g., [25, 36, 57]). All these notions are defined as distributed computing functions. In [50] we suggest that the distributed signature setting also provides for extended functionality by enabling a new notion of signature which is otherwise impossible (owing to the added control in this case). We also propose a method to produce (computationally) blinded DSS signatures [60] in an arbitrarily distributed manner, so that these can later be unblinded by a threshold of servers. This makes it possible to distribute the signature generating and signature unblinding servers, and to use proactive methods [43, 44] for maximum security and availability. We note that other implementations of magic ink signature schemes (and its relatives) may be used in place of the one for DSS signatures we present, although this may affect the high level description, adversarial model, and functional description slightly.

Chapter II

A Versatile and Efficient E-Money Scheme

II.A System Model

The model follows the one suggested in [49] and augmented in [51]:

II.A.1 Participants

The system can be modelled by seven types of (polynomial-time limited) participants/entities : Users (1) who withdraw money and perform payments. Users enjoy computational anonymity. Shops (2) get money from users and attackers, and deposit it in the Bank, while banks (3) manage user accounts, issue and redeem money. The banks are able to alert the shops to engage in a non-standard (on-line) payment procedure. Based on court orders banks may engage in blacklisting and tracing (as a crime prevention mechanism). The Ombudsman (4) participates in withdrawals and assists in reacting to court orders. The judge (5) will employ enforcement mechanisms, and issues court orders. Finally, there is a certification authority (6) and key directory (7) for public keys. More detailed, the participants are as follows:

1. Users

   Users withdraw funds from the Bank, and make payments to shops. They have certified public keys associated with themselves, allowing the users to identify themselves, and sign agreements. It is the goal of the honest users to transfer funds in return for merchandise or information. Users are concerned with being defrauded of their money, being refused service, having their spending habits monitored, and being falsely accused of a crime. It is the goal of the honest users to transfer funds in return for merchandise or information in a way that does not allow cheating protocol participants to defraud them. We will demand that the honest users enjoy (computational) anonymity.

2. Shops

   The shops receive payments from users and attackers. It is important for the shops to be able to verify that they have received information that corresponds to funds,

so that when the coins representing funds are deposited, the shops' accounts will be credited with the amount the coins represent. The shops may be interested in tracing purchases, i.e., to match a spent coin to a user identity.

3. Banks

   The Bank (who is distributed w.r.t. the ability to sign and trace) manages user accounts, issues money to users and receives spent coins from the shops. The Bank may be interested in tracing purchases, and may cooperate with attackers and shops in order to try to do so. Multiple, independent banks will issue and accept coins of distinctive types. The banks are able to issue *alerts*, forcing shops receiving coins issued by the alerting bank for specified time periods to engage in a protocol preventing an attacker to successfully spend money obtained by robbing the bank. Furthermore, each bank can after a court order blacklist coins or have payments traced.

4. Ombudsman

   The Ombudsman (who is distributed w.r.t. the ability to sign and trace) is the representative of the honest user. The Ombudsman will after a court order assist the Bank in tracing coins of a suspected attacker, but will not assist anybody in any other tracing. We require the Ombudsman to be available for the tracing of attackers by the Bank, and (in the case of on-line withdrawal protocols, such as the one we use here) to be available to the Bank during each withdrawal session. Besides availability and non-cooperation with the Bank regarding traceability, the Ombudsman will not have to be trusted in any other way, neither by the Bank nor the users. Specifically, the Ombudsman will not be able to frame users or successfully cheat any participant. Efficiently, there may be one Ombudsman per Bank, but we will treat the Ombudsman as one single (although distributed) entity in our discussions. The Ombudsman and the Bank have a private communication channel that allows them to communicate the status of transactions, e.g., alert another if a bank robbery is launched.

5. Judge

   The judge has the task of resolving conflicts between the above participants after analyzing the corresponding transcripts. The judge will issue court orders forcing the Ombudsman to cooperate with the Bank to trace coins or user funds specified by the judge. The judge will not have to be trusted in any way by any participant, apart from fulfilling these functions. The judge may be modelled by a multiplicity of independent judges with the same goals and behavior, but we will treat it as a single entity in our discussions.

6. Certification authority

   The certification authority certifies public keys of participants, including keys (of time-limited validity) used by the Bank and the Ombudsman to produce authentications on coins. The certification may have a hierarchical structure, but, again, we will think of it as a single participant.

7. Key directory

   The key directory stores all currently certified public keys. It is the purpose of the key directory to inform all participants what the currently used public keys are.

The Bank, the Ombudsman, the judge and the certification authority will each have a public key associated with themselves. These public keys (but for that of the certification authority) will be certified by the certification authority. Individual shops will have a (not necessarily certified) public key associated with them.

Remark: It is possible to envision a scenario in which no *outside* Ombudsman is needed, but where we allow a conglomerate of banks to control this entity.

II.A.2 Attackers

An attacker is any coalition of protocol participants that deviate from the specified protocols. Attackers may corrupt any set of the Bank and Ombudsman servers and may depart from the prescribed protocols in any way, including forcing other participants to engage in protocols different from those that are prescribed, in order to obtain spendable money or services for a greater amount than their accounts are billed. An attacker may force any entity to give out its secret keys. It is the goal of the attackers to obtain spendable money or services for a greater amount than their accounts are billed. We distinguish between two different attackers: (1) the *weak* attacker may only corrupt a non-quorum of Bank and Ombudsman servers in each time period (see [44, 43]), as well as any number of users and shops, and (2) the *strong* attacker, who can corrupt *any number* of Bank and Ombudsman servers.

II.A.3 Time

Time is divided into (possibly overlapping) time periods[1] of publicly known starts and lengths. A withdrawn coin is only accepted by the Bank for credit within its corresponding time frame, specified in the coin. Thus, each coin will have an explicit expiration date, after which it will not be accepted for deposit (and therefore not as payment either.) The expiration date of the coin will be part of the coin, and set by the Bank in a non-blindable fashion. It will be readable by all participants, but alterable by nobody.

II.A.4 Trust model

The following basic assumptions underlie the architecture:

1. The users trust that a quorum of the Bank and the Ombudsman servers will not conspire against them. (A signing quorum is a set of servers containing at least one Bank server; a tracing quorum at least one Ombudsman server – but typically more than that.)

2. The Bank trusts a threshold of the Ombudsman servers to be available during withdrawals, for traces, and immediately after a bank robbery has taken place.

3. All the participants trust the judge to be honest and fair.

4. All the participants trust the certification authority only to perform certifications of accurate documents and correctly associated pairs of public keys and names of owners.

[1] These will be of a length sufficient to strike a good balance between storage requirements and degree of anonymity, two contrasting goals. The time intervals are for simplicity multiples of the refresh intervals in the proactive secret sharing.

5. The users trust the Bank not to incriminate them, or to steal their money by annulling their accounts, etc.[2]

II.A.5 Events

The following events are part of the system description:

- Open an account

  A user opens an account with the Bank by identifying himself (physically, or by an accepted alias,) supplying other personal information (such as address,) and giving the Bank a key[3] to be used for the user to prove his identity using a standard proof of identity.

- Withdraw money

  A user withdraws money by first engaging in an identification session with the Bank, proving his identity in order for the Bank to be able to associate the proper account with the withdrawal session. Then, the user, the Bank and the Ombudsman engage in a three-party protocol giving the user a coin that represents funds, and giving the Bank (and possibly also the Ombudsman) a record of the transaction. Different types of coins will exist, representing different values and expiration dates.

- Refresh money

  In effect, this combines a spending and deposit protocol with a withdrawal protocol, thus updating a coin with respect to the expiration date. Alternatively, a shop or a user can exchange coins of different expiration dates (using two simultaneous protocols for spending money with opposing flow of coins,) allowing the customer to obtain coins that can be deposited at a later date than his original coins would be spendable.

- Spend money (off-line version)

  A user spends money by engaging in a protocol with a shop, providing the shop with a transcript that could not have been constructed using any public algorithm available to the user or the shop, or any protocol other than the payment protocol. The shop is able to verify that the coin received is of a form that will be accepted for credit when given to the Bank.

- Whitelist

  After a bank robbery has taken place, the Bank and the Ombudsman cooperate to produce a list describing all properly withdrawn coins (the so called *white list*), to which deposited coins later can be compared. The white list is produced in a manner that does not allow any set of servers that does not include a quorum of Bank and

[2] This assumption can easily be avoided to the price of an inefficient system in which each transaction has to be signed by as well the Bank as the users, and signatures exchanged using methods for simultaneous exchange of secrets.

[3] Remark: if the user does not desire to withdraw money without physically appearing in the Bank and identifying himself, the user does not need to establish such a key. Furthermore, whereas public key and zero-knowledge methods may be involved for the proof of identity, other methods can be employed if the Bank is trusted not to steal.

Ombudsman servers to correlate any withdrawal view to the coin withdrawn in the corresponding session.

- Spend money (on-line version)

  This protocol is the same as above, with the exception that the shop needs to engage in an interaction with the Bank (or proxy) in order to verify that the coin will be accepted for credit when given to the Bank, i.e., that the spent coin is on the white list. (We will specify later how the shop knows whether to engage in the off-line or on-line version.)

- Deposit money (off-line version[4])

  The shop can deposit coins by sending them to the Bank[5], which will verify that they are of a correct form and have not been previously deposited, and will credit the wanted account with the corresponding funds. (The name of the account can be made explicit in the coin by the shop during the protocol during which he receives the coin from the user, thereby allowing the coin to be associated only with this account.) The Bank will give the shop a signed receipt specifying the transaction. The transcript is saved by the Bank until the end of the time interval associated with the coin.

- Deposit money (on-line version)

  This protocol is the same as above, with the exception that the Bank will also verify that the deposited coin is on the white list before accepting it for credit.

- Register new keys

  The Bank and Ombudsman will generate and by means of the key directory distribute authenticated public keys to be used for verification of Bank/Ombudsman signatures on coins. Since the exact form of a correctly formed authenticated coin will be a function of time (due to the expiration dates of coins,) the Bank will have to generate and have distributed such keys repeatedly. The Bank and Ombudsman will not publish the public keys of future time intervals too early in advance. The Bank and Ombudsman will sign the new public keys using a public key used solely for this purpose.

- Detect overspending

  If the Bank receives transaction transcripts whose total value transferred exceeds the value represented by the corresponding coin, these transcripts are evidence that the coin was used in an illegal fashion. The transcripts can be used in place of a court order, forcing the Ombudsman to interact with the Bank to trace the identity of the withdrawer of the coin.

[4] An on-line protocol can trivially be obtained by combining the protocol for spending money with that for depositing money. In such a case, the shop would deposit funds before accepting the payment. It may be required by the Bank that certain types of payments be on-line, or may be desirable by the merchant to minimize its risk. Since this type of on-line payment protocol is a special case of the off-line payment and deposit protocols, we will henceforth deal only with the more general case of off-line payments. Note that this form of on-line protocol would differ from the one associated with a bank alert.

[5] If there is more than one bank, the coin can be deposited in any bank still, but this bank will have to send it to the issuing bank for overspending verification, etc.

- Trace

  The Bank and the Ombudsman interact in a protocol allowing the Bank to match the traced coin or traced identity to the corresponding identity or coin, or to verify if a particular pair of a coin and withdrawal transcript correspond to each other, without revealing any additional information.

- Blacklist

  Based on a court order or an overspending, the Bank can publish a list of coins to be put on vs. taken off the list of coins not to be accepted. This prevents coins received in so called blackmail attacks, and similar attacks, from being used.

- Fund freezing

  The same as blacklisting, with the difference that it may not be permanent.

- Fund "thawing"

  The Bank which issued the freezing of funds can by sending out a second certified message remove coins from the list of frozen funds, allowing them to be used for future transactions.

- Alert (on/off)

  The Bank can alert all the shops to use the on-line protocols when receiving a payment. This is done with respect to coins of certain expiration dates (which can be identified by the shops during the payment protocol.) Similarly, the Bank can *remove* the alert for coins of certain expiration dates. The alert mechanism is used to prevent coins received during a so called bank robbery from being used[6]. (Notice that such a coin may be "withdrawn" using a blindfolded withdrawal protocol, thereby making a normal tracing or blacklisting impossible.)

- Arbitration

  If there is any disagreement or suspicion of criminal activity, the case can be taken to the judge. There are protocols specifying the interaction between the involved parties and the judge in such cases; these include requests for court orders and the issuing of these. We will elaborate on this below, under the mechanisms to ensure justice.

- Certification

  There will be protocols for obtaining and denouncing certifications on documents such as public keys, as well as publicly available algorithms for verifying the certifications made.

[6]For the bank robbery attack that corresponds to a *known* attack (i.e., in which the issuer is forced to give out signatures or secret keys), we assume that the propagation rates and read priorities of messages that are alerts are higher than those of coins being withdrawn and spent, i.e., that all shops will be made aware of an alert before a coin withdrawn at the same time as the alert is made can be successfully spent. This is a realistic assumption since the Bank is in command of both the issuing and the alert, and can delay the execution of the former just a little. For the bank robbery attack corresponding to an insider attack (which may go undetected for some amount of time,) only unspent funds can be blocked by the alert; however, the Bank will still be able to earmark what already spent funds were received in the attack, which may give important information in itself.

- Update public keys and verification algorithms

  After a change to the public data has been made, this will be made public by the key directory, and all participants can obtain and perform the updates.

II.A.6 Mechanisms to ensure justice

There are complaint procedures allowing participants to prove the possible wrongdoings of other participants to the judge. Specifically, there will be ways to prove that some Bank or Ombudsman servers do not follow the withdrawal protocol, or that the Bank refuses to accept and give credit for a properly withdrawn but not previously deposited coin. Also, there will be specific methods for obtaining court orders, making sure that a malicious bank cannot trace coins that were not meant to be traced by using "bait and switch". Furthermore, different ways to ensure a fair exchange of payments and merchandise (i.e., that neither the payer nor the payee can cheat another) can be implemented (see [46],) adding procedures for obtaining the fair exchange and for filing grievances.

II.B Attacks and Requirements

II.B.1 Attacks

An attacker is somebody who deviates from the prescribed protocols. With an honest bank, we mean a set of honest Bank servers, of sufficient size to make a decision, such as accepting a payment. We consider the following attacks, some of which are only presented in an informal way, but which are covered by formal requirements:

Attack: Forgery
When a set of users, shops, and Bank and Ombudsman servers, not including a quorum of the latter two, after engaging in withdrawal protocols withdrawing funds for a value of $\mathcal{V}$, are able to perform payments for a value exceeding $\mathcal{V}$, which are later accepted by an honest bank as valid.

Attack: Impersonation
The attacker is a coalition of users, shops, Bank and Ombudsman servers. In a successful attack, an honest user is charged more than his withdrawals total.

Attack: Overspending
The attacker is a coalition of users, shops and Bank and Ombudsman servers, not including a quorum of the latter. A successful attack is when the attackers after withdrawing an amount $\mathcal{V}$ perform payments for a value $\mathcal{V}_1 > \mathcal{V}$, such that: (a) these are accepted as valid by a set of honest shops, but (b) not all of these are accepted by an honest Bank as valid.

Attack: Illegal purchases
These are transactions that, whereas perfectly valid from a money-flow point of view, are not legal by the nature of the products or services paid for.

Attack: Money laundering
An illegal transfer of funds performed in order to misrepresent the distributions of incomes of (or hide the existence of) organizations.

Attack: Blackmail
An attacker $\mathcal{A}$ directs a user $\mathcal{U}$ to engage in a withdrawal protocol using a protocol P_1, possibly different from any protocol in use, but generating a Bank/Ombudsman view indistinguishable from a normal withdrawal from user $\mathcal{U}$'s account. $\mathcal{A}, \mathcal{U}$ and the Bank/Ombudsman then run the protocol P_2, such that $\mathcal{A}$'s view during P_1 is indistinguishable from his view during P_2. As a result, $\mathcal{A}$ is able to generate a withdrawal view, indistinguishable from the view of $\mathcal{U}$ during a normal withdrawal.

Attack: Bank robbery
We assume an attacker $\mathcal{A}$ threatens and directs the Bank and the Ombudsman to use a protocol P_1, possibly different from any protocol in use. The attacked entities agree to engage in a protocol P_2, possibly different from P_1, but with a view that is indistinguishable to the attacker from that of P_1. As a result, $\mathcal{A}$ is able to generate a withdrawal view, indistinguishable from the view during a normal withdrawal. We assume that the attacker cannot spend the obtained money within a time Δ after the threat is over, and that this time is longer than the propagation time for an alert[7]. After the threat is gone, the Bank may temporarily change its behavior (e.g., requiring deposits to be made, or clearance from a proxy received, before a payment is accepted.)

Attack: Illegal Untraceability
The victim of this attack is a quorum of Bank and Ombudsman servers, and the attacker is a coalition of users, shops, and Bank and Ombudsman servers, not including any member of the victim. The attack is successful if an honest bank accepts a deposit that if traced by the quorum would not identify any member of the set of attackers, whereas at least one of these participants would be identified if the attack was not performed.

Attack: Malicious tracing
This is an attack in which Bank and/or Ombudsman servers without legal permission trace coins, i.e., match a withdrawal session to an identifier of a payment transcript.

Attack: Framing
Let $\mathcal{U}$ be a set of honest users. Consider a set $\mathcal{A}$ of dishonest users, shops, Bank and Ombudsman servers, not including any member of $\mathcal{U}$. A framing attack is when $\mathcal{A}$ produces a set of transcripts, that, if a tracing is performed with these as input, the output would identify a member of $\mathcal{U}$ with non-negligible probability.

Attack: Embezzlement
Let $\mathcal{U}$ be a set of honest users, performing withdrawals for a value $\mathcal{V}$. An embezzlement is an attack in which $\mathcal{A}$, a set of users, shops, Bank and Ombudsman servers, not including any member of $\mathcal{U}$, makes only a value $\mathcal{V}_1 < \mathcal{V}_2$ be accepted as valid by an honest set of Bank servers, after members of $\mathcal{U}$ spend funds with a value $\mathcal{V}_2 \leq \mathcal{V}$ from the said withdrawals.

II.B.2 Requirements

Before stating the requirements, let us specify the relationship between withdrawals and deposits. In the following definitions, we assume the payment method has the following general form: After each withdrawal protocol, the Bank stores a record *tag*

[7]This attack models well an electronic bank robbery in which the attacker wants to spend the loot on physical merchandise, whose delivery takes time Δ.

(and the withdrawer's identity, if it is not a coerced withdrawal). The shop has a procedure $Correct(coin)$ which specifies whether a coin is valid. In the off-line mode (which is the normal case) this is publicly computable, and in the on-line mode (which the Bank may require after a successful bank robbery attack has been performed) it can only be computed by involvement of a quorum of Bank and Ombudsman servers. There is a relation $Corresponds(tag, coin)$ representing whether $coin$ is a spending of the coin withdrawn when tag was generated. This relation is computable only by a quorum of Bank and Ombudsman servers. Let $\mathcal{T}$ be the set of tags, and let $\mathcal{C}$ be the set of identifiers of valid spending transcripts.

In order to prevent or limit the impact of the previously specified attacks, we put forth the following requirements:

Requirement EM1: Unforgeability
Forgery is infeasible.

Requirement EM2: Impersonation safety
Impersonation is infeasible if the coalition of attackers does not involve a quorum of Bank and Ombudsman servers, or if the transaction can be legally challenged by taking the case to a judge.

Requirement EM3: Overspending detection
Let $\mathcal{A}$ be a set of attackers performing an overspending attack in which a value $\mathcal{V}$ is withdrawn and a value $\mathcal{V}_1 > \mathcal{V}$ is spent. If the scheme satisfies overspending detection, then a quorum of the Bank and Ombudsman servers will be able to establish (a) the sum of the overspending, i.e., $\mathcal{V}_1 - \mathcal{V}$, and (b) the identity of at least one member of $\mathcal{A}$.

Requirement EM4: Overspending robustness
Let $\mathcal{A}_1$ be a set of attackers performing an overspending attack in which a value $\mathcal{V}$ is withdrawn and a value $\mathcal{V}_1 > \mathcal{V}$ is spent. Let $\mathcal{A}_2$ be a set of users, shops, Bank and Ombudsman servers, disjoint from $\mathcal{A}_1$. A scheme is overspending robust if it is infeasible for any set $\mathcal{A}_2$ to make an honest quorum of Bank and Ombudsman servers running the overspending detection protocol output a sum of overspendings $\mathcal{V}_2 - \mathcal{V} > \mathcal{V}_1 - \mathcal{V}$ and only identities of participants in $\mathcal{A}_1$.

Requirement EM5: Traceability
Any Bank and Ombudsman quorum can, regardless of the actual withdrawal protocol used, and regardless of whether other Bank and Ombudsman servers refuse to cooperate, perform the following actions:

1. Given a $tag \in \mathcal{T}$, calculate $coin \in \mathcal{C}$, such that $Corresponds(tag, coin)$.
2. Given a $coin \in \mathcal{C}$, calculate $tag \in \mathcal{T}$, such that $Corresponds(tag, coin)$.
3. Given a $tag \in \mathcal{T}$ and a $coin \in \mathcal{C}$, establish whether $Corresponds(tag, coin)$.

All of these actions are performed in a way that does not give any set of participants a non-negligible advantage (apart from the advantage gained by knowing the *result* of the above calculation) in establishing whether $Corresponds(tag', coin')$ for any $tag' \in \mathcal{T}$, $coin' \in \mathcal{C}$, but $(coin', tag') \neq (coin, tag)$.

Requirement EM6: Revocability
Any traced coin can be permanently resp. temporarily made unspendable by blacklisting resp. freezing. In such an action, the Bank announces that it will not accept funds descriptions with identifier $coin \in \mathcal{C}$ for credit, for a $coin$ corresponding to the funds blacklisted or frozen.

Requirement EM7: Anonymity
The probability for any coalition of participants not containing a quorum of Bank and Ombudsman servers to determine whether $Corresponds(tag, coin)$ holds for any particular pair $(tag, coin) \in \mathcal{T} \times \mathcal{C}$ is not non-negligibly better than that of a guess, uniformly at random from all possible pairs, given all the available pairs of descriptions of $tag' \in \mathcal{T}$ and $coin' \in \mathcal{C}$ and all the already computed relations $Corresponds$ available.

Requirement EM8: Framing-freeness
Framing attacks are infeasible if (a) the withdrawer has a public key associated with him, and he signs withdrawals, or (b) there is no quorum of Ombudsman and dishonest Bank servers.

Requirement EM9: Refundability
A victim of an embezzlement attack can prove that the attack took place, resulting in the identification of at least one of the attackers.

II.C Definitions

Definition 1: Cryptosystem
A cryptosystem has the following components:

- A *security parameter* k and a *message space*, $\mathcal{M}_k = \{0,1\}^k$, to which the encryption algorithm may be applied.
- A p-time *key generation algorithm* $\mathcal{KG}$ producing a random pair (PK, SK) of keys on input 1^k.
- A p-time *encryption algorithm* E. This is a probabilistic algorithm which given a message m and a public key PK outputs an encryption $\overline{m}$ of m with respect to PK.
- A p-time *decryption algorithm* D. This is an algorithm which given E, $\overline{m}$, and (PK, SK), outputs m.

Security:
For public key systems, (PK, SK) is a pair of a public and secret keys, whereas for private key systems $PK = SK$. Intuitively, a cryptosystem is *secure* if there does not exist a p-time decrypter $\mathcal{D}$ that (in the public-key case gets PK and) for infinitely many k succeeds to decrypt an encrypted message $\overline{m}$ with some non-negligible probability in k, over all choices of $\mathcal{KG}$ and m, for $\overline{m} = E(PK, m)$. More formally, we adopt the polynomial security of **Probabilistic Encryption:** (see [40]), where there is no p-time algorithm $\mathcal{A}$ that has a non-negligible advantage (in k, over all choices of $\mathcal{KG}$ and m_0, m_1) in distinguishing between $E(PK, m_0)$ and $E(PK, m_1)$, given (m_0, m_1, PK).

Definition 2: Signature Scheme [26, 41] (by exposition of [4])
A digital signature has the following components:

- A *security parameter* k, a *message space* and a *key generation algorithm* as above.
- A *signing scheme*[8] $S = (S_S, S_R)$, where S_S and S_R are probabilistic p-time algorithms run by the signer vs. the receiver of a signature s on a message m. The signer knows the pair (PK, SK) of matching public and secret keys, and the receiver knows PK.
- A *verification algorithm* V. This is a p-time algorithm which given s, m, and PK (as well as some private input in special cases, which will be discussed below) outputs 1 if s is a valid signature for the message m with respect to the public key PK, and 0 otherwise.

Let us now consider some properties of signature schemes, most of which are orthogonal to each other:

Security:
We say that an attacker succeeds to forge a signature if he manages to produce a signature on a message that was not previously already signed. A signature scheme is *secure* if there is no p-time forger $\mathcal{F}$ that, for infinitely many k, succeeds with a non-negligible probability to forge a signature s on a given message m so that $V(s, m, PK) = 1$. In particular, we can let the attacker use the signature device first and require security. We may require security against **Adaptive Chosen/Random Message Attacks** [41], namely that security holds w.r.t. an $\mathcal{F}$ performing a successful forgery after given access to a signature oracle a polynomial number of times on chosen (resp. random) messages. Schemes that are secure against adaptive chosen message attacks are also called *existentially unforgeable.*

History-Free:
A signature scheme is *history-free*[9] if the signing scheme is not a function of previously signed messages.

Blind Signature: [17]
A *blind signature* scheme is a pair $S = (S_S, S_R)$ that allows the receiver to obtain a valid signature that is computationally uncorrelated to the transcript seen by the signer during the protocol.

Transparently Blindable Signature: [49]
A signature function $S = (S_S, S_R)$ is *transparently blindable* if there exists a blind signature scheme (S_S, S'_R) producing an output with the same distribution as S does, and the signer's view is indistinguishable for the two protocols, i.e., the signer cannot tell whether he signs regularly or blindly.

Blindfoldable Signature: [49]
The signature scheme $S = (S_S, S_R)$ is *blindfoldable* if there exists a blind signature protocol (S'_S, S'_R) producing an output with the same distribution as $S = (S_S, S_R)$ does.

[8] Note that the signing scheme may be interactive. We make the distinction between interactive signing schemes and protocols in that we consider the protocol a particular implementation of the interactive signing scheme.

[9] A signature scheme that is not history free may have problems if used in anonymous e-money schemes, as shown in [68].

Dual Verification Signature Scheme: [49]
A *dual verification signature scheme* is a six-tuple $(\mathcal{M}_k, \mathcal{KG}, S, V_1, V_2, \tau)$, with the following property: $(\mathcal{M}_k, \mathcal{KG}, S, V_1)$ is a signature scheme with publicly verifiable signatures, and $(\mathcal{M}_k, \mathcal{KG}, S, V_2)$ is a signature scheme where a signature can only be verified by interaction with a authenticator. In other words, the function $Correct = (V_1, V_2, \tau)$ is publicly computable in the first case ($\tau = 1$,) but can only be done by the Bank and Ombudsman in the second case ($\tau = 2$.) Here, (m, s) may be a correct message-signature pair w.r.t. one of the schemes, but not the other. We call τ the triggering condition; this decides whether V_1 or V_2 shall be used for verification of a signature.

Definition 3: Challenge Semantics [49]
The challenge semantics of a coin describes the functionality of the coin by assigning different meanings to different bits of the challenge. It is not possible to alter the challenge semantics of a coin once it has been spent.

II.D Building Blocks

Coin Signature Scheme:
Each coin will be represented by a secret key / public key pair (x_{Coin}, y_{Coin}). The keys are associated with a signature scheme S that is *existentially unforgeable*, and that has a *public verification algorithm*, V. A lot of schemes may be used for this, e.g., RSA [72], ElGamal [27], and related schemes like [1, 77]. In order to heuristically produce existentially unforgeable signatures using message-recovery schemes like these, the standard method is to apply a collision resistant one-way function unrelated to the signature scheme to the message to be signed before the signature is calculated (see also [69]). More specifically, a random oracle is needed (see [5, 6].)

Bank/Ombudsman Signature Scheme:
The Bank and the Ombudsman will in a distributed fashion calculate a signature $s_{Coin} = S_{B/O}(y_{Coin})$ on the public key y_{Coin} associated with a properly withdrawn coin. The signature scheme is not *transparently blindable*, is *history-free*, and has a *dual verification scheme* where all the valid message-signatures pairs under V_2 also are valid under V_1. The non-public verification algorithm V_2 is implemented by a list of valid tags to which a potential signed message can be matched to for verification. For security reasons, we want the signature generation to be a distributed function, calculating a signature from shares of the message using shares of the secret. We employ methods for proactive secret sharing and function evaluation [43] to do this. We use the magic ink [50] generation of DSS [60] signatures, i.e., a distributed signature generation employing computational blinding, such that the blinding can (only) be removed by a quorum using an unblinding algorithm.

Probabilistic Encryption Scheme:
Let (E_{B_i}, D_{B_i}) denote the probabilistic encryption vs. decryption algorithms of the ith server of the Bank, using the Bank's public key for encryption and its secret key for decryption. Similarly, (E_{O_i}, D_{O_i}) are the public key probabilistic encryption and decryption algorithms of the ith server of the Ombudsman. Any public key encryption scheme can be employed for these. Furthermore, let (E_K, D_K) denote a symmetric key probabilistic encryption/decryption scheme with key K. The schemes will be augmented using standard methods [40] for making the encryption probabilistic.

A Mechanism for Tagging:
Each withdrawn coin will be tagged by the Bank and Ombudsman in order for it later to be traceable. The generation of the tag is a part of the signature generation scheme, which we will elaborate on later. The tag has the property that $Corresponds(tag, coin)$ is satisfied for the coin withdrawn.

II.E The Basic E-money System

II.E.1 Withdrawing a Coin

A coin is represented by a pair (x_{Coin}, s_{Coin}), where $s_{Coin} = S_{B/O}(y_{Coin})$ for the public key y_{Coin} corresponding to the secret key x_{Coin}. In order to perform the withdrawal of a coin, the following protocol is executed:

1. The withdrawer, whom we will call Alice, runs the key generation algorithm to get (x_{Coin}, y_{Coin}). She proves her identity to the Bank, potentially in a manner that does not allow an eavesdropper to get any information about her identity. This can be obtained by, e.g., probabilistic encryption of transcripts. If needed, Alice establishes shared session keys with each server of the Bank and the Ombudsman by randomly choosing such keys and encrypting these using the public keys of the intended receivers. (Which can be the same as the public key shares of the signature scheme.)

2. Using the magic ink signature generation scheme, Alice, the Bank and the Ombudsman servers compute an output so that Alice gets a Bank/Ombudsman signature $s_{Coin} = S_{B/O}(y_{Coin})$, and the Bank (and possibly the Ombudsman) servers get a tag *tag*, linked to the signed message, i.e., such that $Corresponds(tag, coin)$ is satisfied, where *coin* is a function of (y_{Coin}, s_{Coin}). Here, $Corresponds$ is always computable by a quorum of Bank and Ombudsman servers, but not computable by less than such a quorum. The Bank saves *tag* in a secure database, along with Alice's identity and an identifier or classification of the withdrawal session.

II.E.2 Spending a Coin

A coin (x_{Coin}, s_{Coin}) is spent in the following way:

1. The spender of the coin, Alice, sends (y_{Coin}, s_{Coin}) to the payee, Shop, who verifies the validity of the message-signature pair (y_{Coin}, s_{Coin}) using $Correct$.

2. Shop sends the challenge c to Alice. The challenge may to some part be of a predetermined form, using challenge semantics to implement functions like divisibility, etc., and to the other part random and set by Shop.

3. Alice sends the answer $a = S_{x_{Coin}}(c)$ to Shop, who verifies that $V_{y_{Coin}}(a, c) = 1$, and if the signature is valid, saves the transcript $(y_{Coin}, s_{Coin}, c, a)$.

We note that the above can be made non-interactive by the use of predetermined contracts (e.g., "by clicking here you agree to pay $1.50 for the document.")

II.E.3 Depositing a Coin

A spent coin is deposited by forwarding the transcript $(y_{Coin}, s_{Coin}, c, a)$ to the Bank. The Bank verifies that $Correct(y_{Coin}, s_{Coin})$, and that $V_{y_{Coin}}(a, c) = 1$. The Bank further verifies that the same transcript has not been deposited before, and then credits the depositor's account. Each time a spent coin is deposited, the Bank will after verifying the correctness of the related signatures perform the following accounting:

1. If that exact transcript $(y_{Coin}, s_{Coin}, c, a)$ has already been deposited, it will ignore the transcript, since it is only repeated. In this case, no credit is given.
2. Otherwise, it will add the value of the spending to the total value this coin has been spent for (0 if it has not previously been spent.) Then, it credits the depositor's account with the amount of the spending.
3. If the total value spent using any coin exceeds the value of a coin, then the corresponding coin has been overspent.

If an overspending has been performed, i.e., if there are signed messages using the public key of this coin, such that the sum of their implied values exceeds the allowed value, then these transcripts will constitute a tracing order, and a trace will be performed.

II.E.4 Tracing

When a coin needs to be traced, the Bank will either (case 1) have as input an identity *id* of a user, and will want to know what coin(s) (y_{Coin}, s_{Coin}) this person withdrew in a certain time interval or during a particular withdrawal session, or (case 2) have the description (y_{Coin}, s_{Coin}) of a coin and want to know the identity *id* of the withdrawer, or, finally (case 3) determine whether a certain coin (y_{Coin}, s_{Coin}) was withdrawn during a specific withdrawal session. In chapter III we show how traces can be performed.

II.E.5 Alert

If a successful bank robbery takes place, the Bank will immediately alert all shops that they now must use the on-line verification protocol for Bank/Ombudsman signatures, i.e., deposit each coin before it is accepted as a payment. In order not to have to involve the Ombudsman for each payment, and in order to be able to use proxies, the Bank and the Ombudsman can produce the whitelist from the list of tags affected (the tags with the same expiration date as the bank robbery coins), to which coin transcripts can be compared for validity check. The list will be produced in a way that does not compromise user privacy. This can be done using standard multi-party protocols that hide the secret input of the participants (see [86]) or using more efficient and specialized protocols (e.g., [48].) Before any coin is accepted, the Bank or a Bank proxy will verify that the coin is in the list of valid coins.

II.F Versatility of the Monetary System

The basic system can now be altered to encompass a wide variety of modular efficient extensions based on what we call *challenge semantics*. The idea is to extend functionality by letting some number of the bits used for the challenge represent the "meaning"

of the payment, e.g., the division of a coin, a check, a credit payment, a surety bond, etc. Some bits can represent the condition under which the coin can be cashed, and still other bits can be set to designate the identity of the payee, or the accounts in which the payment can be deposited, thereby making (among other things) hold-ups of shops futile.

Long contracts can be used by hashing them and using the resulting hash value as part of the challenge; part of the challenge can be encrypted to limit the readability to certain groups of parties or certain situations, or to communicate a distress situation (as in [23].)

Just like different maximum values are marked by different Bank signatures on the coins, different limitations of functionality can be imposed by the Bank in the same way. Thus, one type of signature can mean that the coin is limited for use as a check, etc.

II.F.1 An Extension to Divisible Coins

Change giving mechanisms are trivially part of any system as long as the shop has sufficient "change," and can spend the corresponding coins as a payment to the user, to negatively offset the sum paid. However, this method is inconvenient due to the fact that it requires the shop to have change, and the payer to store the same. Even worse, it is contradicting the anonymity requirements, as it would allow the change returned to be used to identify the customer of a certain transaction with the cooperation of the shop. Therefore, in order to avoid multiple coins to be required to match a particular sum (which is a second trivial solution to the problem,) we opt for the more attractive solution of coin divisibility.

Allowing coins to be divided into arbitrary fractions can trivially be obtained in our system (divisions are linkable, though). The Bank will accept deposited coins as before, and credit the accounts of the depositor by the amount indicated in the challenge, or the full value of the coin, whatever is lower. An overspending has occurred if one coin is used to transfer more funds than its related value allows. If $\mathcal{V}$ is the value of the coin, and v_i the value transferred in the ith spending of the coin, the coin is overspent when $\sum_{i=1}^{\kappa} v_i > \mathcal{V}$, where κ is the number of times the coin is spent. When a coin has been overspent, the Bank will, as before, show the corresponding transcripts to the Ombudsman, who after verifying that they indeed constitute an overspending will participate in a tracing of this coin.

II.F.2 Electronic Checks and Credit Cards

An electronic check is a transfer of funds with an amount that is not fixed at the time of the withdrawal. Similarly, a credit card purchase is a payment using a coin of no intrinsic value. After being spent, which in either of these cases will be done by "inscribing" the value spent, just as for the divisible coin, the coin is deposited. Then, since the value of a coin of this type in itself is zero, this corresponds to a (legal) overspending of funds, and the identity of the spender will be obtained using the normal procedure. Alternatively, a dedication of the usage of some bits in the challenge can be used to signal the difference, thus introducing further semantics of the challenge, and signaling to the Bank and the Ombudsman to treat the coin in a special way. Then, when the identity is obtained, the amount indicated on the check is subtracted from the balance of this person. It is possible to implement (bank) checks with maximum amounts, such that if these are exceeded, the

"overspending" is no longer legal and constitutes an actual overspending. Next, we briefly describe two alternative solutions, both granting anonymity:

1. The Bank keeps anonymous accounts (with positive funds for checking accounts and negative funds and limits for credit card accounts) that are replenished/paid off by anonymous user transfers (i.e., anonymous payments by users to the corresponding accounts). Here, each coin, which may be signed by the Bank/Ombudsman using another signature scheme than "cash" coins, corresponds to such an account.

2. We can, by further employing challenge semantics, use parts of the challenge to communicate the number to an anonymous account and a one-time password. Again, the account would be replenished (or the balance paid off) by means of normal e-money payments to the Bank, designated to the anonymous account.

In either of the two cases, user misbehavior can be detected, after which revokation methods would be applied. Particularly, if the balances are not paid off at the agreed time, then a normal trace is performed.

II.F.3 Obtaining a Fair Exchange

A fair exchange of payments for an item (goods or services) is a barter where no party obtains the item desired without handing over the item offered. We sketch how the concept, introduced in [46], can be applied to our payment system in a modular way, using a similar approach.

The idea is for the user to *rip* a coin, and in a first phase give the ripped coin to the shop, which sends the merchandise/information after verifying that the coin is correct. It is important that the shop cannot deposit the ripped coin for credit, but also that the user cannot use it for something else (without overspending it). Thus, the user will trust that the shop will send the merchandise/information, as he can verify the correctness of the funds (but not deposit them for credit). On the other hand, the shop knows that the user will send the "second half" after having received the merchandise/information, as the coin has already in a sense been spent, and cannot be used for anything else.

This can be achieved by the use of challenge semantics. In a first payment of a coin, the payer will use a challenge where one bit specifies that the coin is void. Thus, it will not be possible to deposit that coin for credit with the Bank. The "second half" is given using the very same challenge but toggling the void bit of the same coin. This is a payment that the Bank will give credit for. Should, however, the second half never be given, then the payee can use the first part to file a complaint, preventing the payer from being able to use the coin to its full value without overspending it.

II.F.4 Event Triggered Payments

Using challenge semantics, it is easy to create conditional payments, where the payment takes effect triggered by events. Examples of situations where this is useful are for insurance policies, surety bonds, lotteries, etc. (see [54]). This will be done by putting a (possibly hashed or encrypted) description of the triggering events in the challenge. Before giving credit for a deposited coin with a bit specifying conditional set, the Bank would have the corresponding triggering function evaluated to make sure that the coin can be cashed.

The events can be either of pure conditional character or just of a time dependent type. An example of the former type would be "if Alice cannot pay her rent, then this check can be used for that purpose," and an example of the latter is "this coin cannot be cashed until July 7th."

II.F.5 Micro-payments

Micro-payments can be implemented (as in [53, 71]) by using the last hash value in a chain of such values as a challenge; later, the spender can gradually pay by sending over hash preimages to the merchant. Since our scheme is relatively light-weight, it is well suited to implement anonymous micro-payments.

II.F.6 Negotiable Anonymity

Challenge semantics can also be used to sell one's privacy in any one particular payment. This can be done for marketing reasons, for example, and technically achieved simply by indicating in the challenge that no anonymity is desired. This can be combined with a conditional, giving the user losing his or her privacy a cash-back in those cases the anonymity is removed. Also, the challenge can specify who can obtain the information, whether partial information can be given (e.g., only age and profession, but never name, etc.) and conditions of the release.

II.G A Note On Efficiency

First, the computational and communicational costs for withdrawing and storing a coin do not depend on the number of times it can be spent. For other schemes offering κ-spendability, these costs are linear in κ. For example, using our proposed magic ink DSS scheme, we get the following: Apart from the counter and independently of the value of κ, where κ is the number of times the coin was spent, the user has to store only one DSS signature, i.e., 320 bits. This is significantly less than the amount stored in most other electronic cash schemes. It is *particularly* competitive for large values of κ. Likewise, there is no extra cost associated with divisibility of coins or checks, or with other extended functionality obtained by the use of challenge semantics.

Second, when a user spends a coin, the transcript $(y_{Coin}, s_{Coin}, c, a)$ will have to be transferred first between the user and the shop, then between the shop and the Bank, where it will be stored until the coin expires. For the maximum keylength of DSS signatures, y_{Coin} is 1024 bits long; s_{Coin} and a are each 320 bits; and c is the size of the challenge[10]. This, too, compares very well to other schemes.

II.H A Note on Database Reduction

Clearly, it does not make sense for the bank to store each received quadruple $(y_{Coin}, s_{Coin}, c, a)$ constituting a payment transcript: if one coin is used in two payments (which becomes natural given the simple coin divisibility), then both y_{Coin} and s_{Coin} will be identical for these two transcripts, since these two parameters are labels of the coin, and

[10]The challenge can be appropriately expanded before its usage, via a pseudo-random generator.

not of the payment itself. Whereas this simple observation allows a lot of storage space to be conserved, we can go much further.

The only reason why the Bank needs to store some parts of the transcript is to be able to convince the Ombudsman (and possibly a Judge) that a coin has been overspent, to initiate a tracing. This data can be cleverly reduced, since tracing is by quorum agreement in our architecture, and not based on solving some $\kappa+1$ equations of degree κ (which is how overspending tracing is implemented in other anonymous e-money systems.) If transcripts are inspected by the Ombudsman (or Judge), much of the data can later be erased, and a short Ombudsman signature be put in its place. Let us look at what data we have to store before and after such an inspection:

Data to be stored before inspection: For each coin, indexed by its signed public key (y_{Coin}, s_{Coin}), the following data is stored: $((y_{Coin}, s_{Coin}), (c_1, a_1), (c_2, a_2), \ldots (c_\kappa, a_\kappa))$. Here, κ is the number of spendings of the coin. The size of this record (using the maximum keylength of DSS signatures) is $1354 + 480\kappa$ bits.

Above, we used (y_{Coin}, s_{Coin}), the signed public key of the coin, as index. If we are using DSS signatures, we can instead use the (much shorter) signature invariant (which is a function of (y_{Coin}, s_{Coin})) as the index. Using DSS, this invariant, call it I_{Coin}, is only 160 bits long[11].

Inspection: Consider the following (potentially partially reduced) record:
Unreduced part: $((y_{Coin}, s_{Coin}), (c_l, a_l), (c_{l+1}, a_{l+1}), \ldots (c_\kappa, a_\kappa))$.
Reduced part: $(I_{Coin}, S, c_1, c_2, \ldots, c_{l-1})$.
Here, S is the Ombudsman signature (using an existentially unforgeable scheme with a secret key other than the one for signing coins in withdrawals) on $(I_{Coin}, c_1, \ldots, c_{l-1})$. During the inspection, the Ombudsman verifies that S is a valid signature of his on this message, and that I_{Coin} is the invariant corresponding to (y_{Coin}, s_{Coin}). The Ombudsman then verifies for each newly deposited transcript $(y_{Coin}, s_{Coin}, c_j, a_j)$, $l \leq j \leq \kappa$, that this is a valid payment transcript. Finally, the Ombudsman produces a new signature S on $(I_{Coin}, c_1, \ldots c_\kappa)$.

Data to be stored after inspection: For each coin, now indexed by its signed public key I_{Coin}, the following data has to be stored: $(I_{Coin}, S, c_1, c_2, \ldots c_k)$. The size of this record (using DSS for the signature S) is $480 + 160\kappa$ bits, which is approximately a third of the size of the record before the reduction, and a tenth of the 1824κ bits required if each one of the κ transcripts were saved in their entirety.

Implications of the reduction: Having reduced the database, it is no longer possible to prove to a third party that an overspending took place; however, if an overspending would occur, then the Bank will be able to convince the Ombudsman that a record (of a type shown in the inspection phase) corresponds to an overspending: The challenges $(c_1, \ldots, c_\kappa)$ indicate the value of the spendings, and S is evidence that at some earlier point, the Ombudsman agreed that the full transcripts corresponded to the saved challenges. The reduction does not affect any other tracing decision or tracing method (since these only use the coin invariants as input).

[11] For the DSS signature (r, s) on m, this invariant is $I_{Coin} = [mr^{-1}]_q$. We refer to the description of the signature scheme in next chapter, and in particular assumption 3, for a more complete treatment.

Chapter III

Magic Ink Signatures

III.A Outline

Magic ink signature generation is a method to enable the generation of blind signatures, which can later be unblinded by the signer (following the physical analogue given before). This is in sharp contrast to traditional blind signatures, which are information theoretically blinded to the signer. The typical application where the need for this functionality arises is for cases where privacy of individuals is assured until some criminal or otherwise unusual activity is detected. Upon detection, identification of the origin of a signature becomes important in identifying the source of the unwanted activity. This is applied to private access tokens, authorized anonymous accounts, and, here, electronic money.

Both the generation and the tracing are performed under quorum action. The generation protocol is distributed to increase the availability and security of the system; the tracing protocol is distributed in order to increase the availability of the system, and to introduce control. We present a method for magic ink generation (and tracing) of DSS signatures; this can be used in the e-money scheme presented earlier.

III.B Requirements

We wish to obtain a signature scheme where blind signatures can be distributively produced by a quorum of trustees, and these signatures *always* can be unblinded by a (possibly different) quorum of trustees. Let the signature key x be distributed using a (t_s, n) secret sharing scheme, and the tracing key x_t distributed using a (t_t, n) secret sharing scheme. We want a signature scheme with the following properties:

Requirement MI1: Correctness
Signatures can be correctly generated using a $(3t_s, n)$ threshold scheme, by any size-$(3t_s+1)$ quorum out of the n trustees, even in the case where a coalition of up to t_s dishonest participants attempt to disrupt the generation of valid transcripts.

Requirement MI2: Zero-Knowledge
Consider an attacker consisting of t_s trustees, the signature receiver, and any number of users and shops. Given one single access to a DSS* oracle[1] with secret key x, the view of

[1]The distinction between DSS and DSS* will be explained in section III.E.

this attacker, executing the signing and tracing protocols, can be simulated in p-time, and the simulated view is indistinguishable from the corresponding view of the real protocols.

Requirement MI3: Anonymity
The signatures are computationally blinded to any set of less than or equal to t_t trustees (i.e., the signature cannot be correlated to the blinded signature or the signing session by a set of less than $t_t + 1$ trustees.) More specifically, the probability for any coalition of participants not containing a quorum of Bank and Ombudsman servers to determine whether $Corresponds(tag, coin)$ holds for any particular pair $(tag, coin) \in \mathcal{T} \times \mathcal{C}$ is not non-negligibly better than that of a guess, uniformly at random from all possible pairs, given all the available pairs of descriptions of $tag' \in \mathcal{T}$ and $coin' \in \mathcal{C}$ and all the already computed relations $Corresponds$ available.

Requirement MI4: Quorum tracing
Valid signatures can always be unblinded, i.e., signatures matched to signing session or vice versa, by any size-$(t_t + 1)$ quorum out of the n servers. In other words, no coalition of less than $t_s + 1$ signature servers, interacting with honest servers a polynomial number of times in the signing and tracing protocols, can produce a valid signature that will not be traced to the tag of one of the signing sessions by any group containing at least $t_t + 1$ honest servers in the tracing protocol.

For the use of the magic ink signatures in the context of the previously presented e-money scheme, we have that the Ombudsman controls at most t_s signing servers (so that the Ombudsman cannot produce money without the cooperation of the Bank), and the Bank controls at most t_t of the tracing servers (so that the Bank cannot trace without the cooperation of the Ombudsman.) More generally, these servers can be distributed in themselves, the above only refers to the number of secret shares held by the different entities.

III.C The Digital Signature Standard (DSS)

We use the DSS (described herein) as the underlying signature algorithm [60].

Note: Since we use different moduli at different times, we use $[op]_z$ to denote the operation op modulo z, where this is not clear from the context.

Key Generation. A DSS key is composed of public information p, q, g, a public key y and a secret key x, where:
1. p is a prime number of length l, where l is a multiple of 64 and $512 \leq l \leq 1024$.
2. q is a 160-bit prime divisor of $p - 1$.
3. g is an element of order q in Z_p^*. The triple (p, q, g) is public.
4. $x \in_u Z_q$ is the secret key of the signer.
5. $y = [g^x]_p$ is the public verification key.

Signature Algorithm. Let $m \in Z_q$ be a hash of the message to be signed. The signer picks a random number $k \in_u Z_q$, calculates $[k^{-1}]_q$ (w.l.o.g. k and k^{-1} values compared to DSA description are interchanged), and sets

$$\begin{aligned} r &= [[g^{k^{-1}}]_p]_q \\ s &= [k(m + xr)]_q \end{aligned}$$

The pair (r, s) is a signature on m w.r.t the signer whose public key is $y = [g^x]_p$.

Verification Algorithm. A signature (r, s) of a message m can be publicly verified by checking that $r = [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q$.

III.D Communication and Threat Model

We assume the standard computation model of polynomial-time randomized Turing machines. Players are connected by an insecure broadcast medium, and an (also polynomial time limited) adversary can inject messages and eavesdrop, but can not delete messages of any other player. Furthermore, a weak signing (vs. tracing) adversary can corrupt up to t_s (vs. t_t) of the n players in the network, and by doing so, force the corrupted players from diverting from the specified protocol in any way. A strong adversary may temporarily corrupt all players. We refer to [36] for more details about the model.

III.E Single-Server (Pseudo) Magic Ink Signatures

In order to communicate the intuition of the scheme, we present a method for producing Magic Ink DSS Signatures using only one signature server. This clearly does not make sense practically, since this server, which knows the blinding factors, would be able to unblind the signature at will, and so, the signature would not have any noticeable blindness properties. However, when we later distribute the signature server, this problem is overcome.

As mentioned, x is the distributed secret key used to produce signatures, and $y = [g^x]_p$ its public correspondence. Similarly, we use a distributed secret key $x_t \in Z_q$ for tagging of signatures; $h = [g^{x_t}]_p$ is its public correspondence.

Let DSS* be the following modified version of DSS, in which the signature receiver gets to see r before he specifies m:

1. The signer S generates the temporary secret key k, and from this, the corresponding temporary public key $r = [[g^{k^{-1}}]_p]_q$, and sends r to the signature receiver R.
2. The signature receiver R sends the signer S the message m to be signed.
3. The signer S computes $s = [k(m + xr)]_q$, where x is the signer's secret key. s is sent to R.
4. The signature receiver R outputs the signed message (m, r, s).

Let MI-DSS* be the magic ink generation version of the above protocol. The non-distributed version of this is as follows:

1. The signer S generates his part of the temporary secret key, $\overline{k}$, and from this, the corresponding temporary public key $\overline{r} = [g^{\overline{k}^{-1}}]_p$, and sends $\overline{r}$ to the signature receiver R.

2. The signature receiver R generates two blinding factors $\alpha, \beta \in_u Z_q$, and computes the following:
$$\begin{cases} \mu = [m\alpha]_q \\ r = [[\overline{r}^{\beta}]_p]_q \\ \rho = [r\alpha]_q \end{cases}$$
R sends (μ, ρ) to the signer S.

3. The signer S computes $\sigma = [\overline{k}(\mu + x\rho)]_q$, and sends σ to R.

4. The signature receiver R calculates the unblinded signature $s = [\sigma\alpha^{-1}\beta^{-1}]_q$, and outputs the signed message (m, r, s).

If the signer and the signature receiver are both following the protocol, then (m, r, s) is a valid signed message using the secret key x. This is so since $r = [\overline{r}^{\beta}]_p$, $\mu = [m\alpha]_q$, $\rho = [r\alpha]_q$, and $s \equiv_q \sigma\alpha^{-1}\beta^{-1} \equiv_q \overline{k}\beta^{-1}(m\alpha + xr\alpha)\alpha^{-1} \equiv_q \overline{k}\beta^{-1}(m + x[g^{k^{-1}\beta}]_p)$. This is the same as saying $\overline{k}^{-1}\beta \equiv_q ms^{-1} + x[g^{k^{-1}\beta}]_p s^{-1}$, i.e., $r = [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q$. This is the format of a valid DSS signature.

We note that an attack on MI-DSS* implies an attack on DSS*, since an attacker can force a DSS* signer to behave like an MI-DSS* signer by the use of the following intermediary $\mathcal{I}$, that interfaces the receiver to simulate his view in MI-DSS*, and the signer to simulate his view in DSS*:

1. The signer S sends a value r to $\mathcal{I}$, who sets $\overline{r} = r$, and sends $\overline{r}$ to the receiver R.

2. R sends the pair (μ, ρ) to $\mathcal{I}$, who computes $m = [\mu r\rho^{-1}]_q$, and sends m to S.

3. S sends s to $\mathcal{I}$, who calculates $\sigma = [\rho r^{-1}s]_q$ and sends σ to R.

Given that the signer produced a valid DSS signature on m w.r.t. its secret key x and the temporary secret key corresponding to r, we have that $r = [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q$. Then, since $m = [\mu r\rho^{-1}]_q$ and $\sigma = [\rho r^{-1}s]_q$ (i.e., $s \equiv_q \sigma r\rho^{-1}$), we have that $r = [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q = [[g^{(\mu r\rho^{-1})(\sigma r\rho^{-1})^{-1}} y^{r(\sigma r\rho^{-1})^{-1}}]_p]_q = [[g^{\mu\sigma^{-1}} y^{\rho\sigma^{-1}}]_p]_q$. The distribution of (r, σ) given (μ, ρ) is identical to that in the real MI-DSS* protocol.

III.F Assumptions

Assumption 1:
Let p, q be primes such that $p - 1|q$, and let $m_1, \ldots m_\kappa$ be chosen uniformly at random from Z_q. The distribution $\{(m, m^x, m^{1/x})\}$ is indistinguishable from the distribution $\{(m, m^{r_1}, m^{r_2})\}$, for unknown values $x, r_1, r_2 \in_u Z_q$.

Assumption 2:
DSS* is existentially unforgeable, i.e., an adversary who gets access to a DSS* signature oracle a polynomial number of times on chosen messages will not be able to produce a valid message-signature (m, r, s) different from those produced by interaction with the oracle.

Assumption 3:
DSS* is existentially unforgeable with respect to the ratio $[mr^{-1}]_q$, i.e., an adversary who gets access to a DSS* signature oracle a polynomial number of times on chosen messages will not be able to produce a valid message-signature (m, r, s) with a ratio $[mr^{-1}]_q$ different from the ratios of the signatures produced by the oracle.

III.G Tools

Let us briefly describe what existing tools we will use in the protocol specification, before describing these in more detail:

- **Polynomial Interpolation Secret Sharing**[74]
  This is the well-known result in which a secret σ is shared by choosing at random a polynomial $f(x)$ of degree t, such that $f(0) = \sigma$, and distributing n points on this polynomial (where $(0, f(0))$ is not one of these points). Given any $t + 1$ out of the points, the entire polynomial f can be reconstructed, and specifically, $f(0)$ can be produced. However, given only t such points on the polynomial, each secret $\sigma = f(0)$ is equally likely. We call this a (t, n) sharing of σ.

- **Joint Random Secret Sharing**[28, 66]
  In a Joint Random Secret Sharing scheme the players *collectively* choose shares corresponding to a (t, n)-secret sharing of a random value. This is done by each participant selecting a degree-t polynomial, dealing n points on this (one to each participant); these polynomial values are added up to determine the shares of the resulting polynomial.

- **Joint Zero Secret Sharing**[7]
  This protocol generates a sharing of a "secret" whose value is zero. Such a protocol is similar to the above joint random secret sharing protocol but instead of local random secrets, each player deals a sharing of the value zero.

- **Computing Reciprocals**[2]
  Given a secret $k \in Z_q$, shared among players $P_1, ...P_n$, we want to generate the value $[g^{k^{-1}}]_p$ without revealing information on k or k^{-1}.

- **Multiplication of Two Secrets**[36]
  Given two secrets u and v, which are both shared among the players, compute the product uv, while maintaining both of the original values secret (aside from the obvious information which is revealed from the result).

We also use a new tool:

Destructive Robustness
We introduce a new method for making distributed protocols robust: Instead of verifying that each individual share of the calculation is correct, we first combine the shares and then verify that the combined result is correct. If it is not, then each share of the result is verified. A noticeable efficiency improvement can be obtained from doing so (mainly by making the common case efficient.) Moreover, this approach allows simpler and clearer protocol design. This is because we can allow the individual correctness verification to destruct important properties of the produced transcript, which, if the combined result is not correct, is a worthless transcript anyway. Therefore, we call this type of robustness *destructive robustness*. Destructive robustness involves two steps: (1) combination of shares of the result, and error detection, by verifying the correctness of the combined result. This check can be either internally (i.e., by the same entities that produced the shares) or externally. Then, if the combined result is not correct, the second step is invoked: (2) error tracing, in which it is determined what server(s) have deviated from the protocol. This kind of robustness is possible in protocols where partial incorrect results can be discarded and when we can withstand delays of malicious servers revealing themselves in a slow pace. We demonstrate an *external* method of destructive robustness for the magic ink generation of DSS signatures.

III.G.1 On Secret Sharing

In the following, we will use two types of secret sharing methods:

1. **Verifiable:** In such a scheme, introduced by [21], each secret share has a public correspondent, and the interpolation of all the secret shares corresponds to the interpolation of all the public correspondents. Examples of such schemes are those developed by Feldman [28] and Pedersen [66]; we will use the latter type.

2. **Non-verifiable:** In these schemes, there are no public correspondents of the secret shares. Such a scheme can trivially be obtained from any verifiable secret sharing scheme by stripping off the use of public versions.

It will be clear from the context what type we suggest for the different parts of the scheme; when we use a verifiable secret sharing scheme, we will implicitly name the public correspondents of the secret shares.

Distributed secret sharing: For both types of secret sharing, we can implement a dealer-free version (as in [66]) by the following approach: Assume we want to perform a (t, n) sharing of a number, and each participant S_i has a contribution α_i. S_i computes a (t, n) secret sharing of α_i, with shares $(\alpha_{i1}, \ldots, \alpha_{in})$, and sends α_{ij} to S_j. Then, S_i computes $a_i = \sum_{j=1}^{n} \alpha_{ji}$; now, $(a_1, \ldots, a_n)$ is a (t, n) secret sharing of $\sum_{i=1}^{n} \alpha_i$.

III.G.2 On Computing Reciprocals

Given a distributively held value $\overline{k}$, shared using a (t, n)-threshold scheme, we want to calculate $[g^{\overline{k}^{-1}}]_p$ without revealing any secret information. The following approach, developed in [2], is used: The servers share a secret a using a (t, n) secret sharing scheme,

and a value $b = 0$ using a $(2t, n)$ secret sharing scheme. Then, each server S_i calculates $v_i = [\overline{k}_i a_i + b_i]_q$ and $A_i = [g^{a_i}]_p$. A is calculated as the (t, n) interpolation of the shares of A, v as the $(2t, n)$ interpolation of the shares of v. The result $[A^{v^{-1}}]_p$, which will equal $[g^{\overline{k}^{-1}}]_p$, is output.

III.G.3 On Multiplication of Two Secrets

Two secrets can be multiplied without revealing any information about the shares (except the wanted product of the secrets) by the following approach, developed in [36]: let a and b be secrets in Z_q, both shared using (t, n) secret sharing schemes. Let c be a zero-sharing using a $(2t, n)$ secret sharing scheme. Then, the $(2t, n)$ interpolation of the shares $d_i = [a_i b_i + c_i]_q$ will equal $[ab]_q$. We note that the multiplication of two secrets easily extends to linear combinations and arbitrary threshold schemes without altering this method.

III.H Magic Ink Signature Generation

We assume that there are $n = 4t_s + 1$ signature servers, t_s of which are corrupted, and the remaining $3t_s + 1$ are honest. Let us now consider a distributed version of the protocols previously presented. Here, let Q be a quorum of $3t_s + 1$ servers in $S_1 \ldots S_n$, $x \in Z_q$ is the signature generation secret key (shared using a (t_s, n) secret sharing scheme) and $y = [g^x]_p$ the public correspondent; $x_t \in Z_q$ is their secret key for tagging (shared using a (t_t, n) secret sharing scheme); $h = [g^{x_t}]_p$ is the public correspondent. For our particular solution, we require that $t_t \leq t_s$, and guarantee that a transaction can be traced by any t_t servers, as long as the corresponding signature generation was performed by at least t_s honest servers.

Initialization:
All the signature generating servers generate keys for authentication of data (onwards we assume all communication to be authenticated using these keys). The servers distributively generate a random secret x using a (t_s, n) secret sharing scheme and a random secret x_t using a (t_t, n) secret sharing scheme; each server S_i publishes his share of the public key $y_i = [g^{x_i}]_p$ and $h_i = [g^{x_{ti}}]_p$, from which $y = [g^x]_p$ and $h = [g^{x_t}]_p$ are interpolated. Each server S_i then proves knowledge of his secret share x_i to the other servers; if some server fails, then he is replaced and the protocol restarts.

Signature Generation:

1. The set of servers $S_i | i \in Q$ distributively generate two random secrets, $\overline{k}, a \in_u Z_q$, both secret shared using a (t_s, n) secret sharing scheme. We let $\overline{k}_i$ and a_i denote the shares held by S_i. $S_i | i \in Q$ also distributively generate a $(2t_s, n)$ zero-sharing, where server S_i has the share b_i, and a $(3t_s, n)$ zero-sharing, where S_i has share c_i.

2. Using the method for computing reciprocals from [2], the servers compute $\overline{r} = [g^{\overline{k}^{-1}}]_p$:

   (a) Server S_i computes $v_i = [\overline{k}_i a_i + b_i]_q$, and $A_i = [g^{a_i}]_p$. He publishes (v_i, A_i).

   (b) The participants compute A by (t_s, n)-interpolation of the A_i's, v as the $(2t_s, n)$-interpolation of the v_i's and calculate $\overline{r} = [A^{v^{-1}}]_p$. Then, $\overline{r}$ is sent to R.

3. The signature receiver R has a message $m \in Z_q$ that he wants signed. He generates two blinding factors, $\alpha, \beta \in_u Z_q$. He computes the following:

$$\begin{cases} r = [[\overline{r}^{\beta}]_p]_q \\ \mu = [m\alpha]_q \\ \rho = [r\alpha]_q. \end{cases}$$

   R then computes a (t_s, n) secret sharing $(\mu_1, \ldots \mu_n)$ of μ, with public information $(M_1, \ldots, M_n) = (g^{\mu_1} \ldots g^{\mu_n})$, and a (t_s, n) secret sharing $(\rho_1, \ldots \rho_n)$ of ρ, with public information $(R_1, \ldots, R_n) = (h^{\rho_1} \ldots h^{\rho_n})$. R sends (μ_i, ρ_i) to S_i.

4. (a) If the $(M_i, R_i) \neq ([g^{\mu_i}]_p, [h^{\rho_i}]_p)$, then S_i publishes (μ_i, ρ_i) and halts; all servers verify whether the transcript sent to S_i is the encryption of (μ_i, ρ_i); if it is, and in fact $(M_i, R_i) \neq ([g^{\mu_i}]_p, [h^{\rho_i}]_p)$, then they halt.

   (b) The servers (t_s, n)-interpolate and store $tag = (g^{\mu}, h^{\rho})$.

   (c) The blinded signature σ is calculated: S_i generates $\sigma_i = [\overline{k}_i(\mu_i + x_i\rho_i) + c_i]_q$. The servers interpolate $\sigma = [\overline{k}(\mu + x\rho)]_q$, and send σ to R.

5. The signature receiver R unblinds the signature: $s = [\sigma\alpha^{-1}\beta^{-1}]_q$. He verifies that the triple (m, r, s) is a valid DSS signature on m; if not, then he complains to the signature servers.

6. If a complaint is received by the signature servers, then server S_i publishes μ_i, ρ_i, $\overline{K}_i = [g^{\overline{k}_i}]_p$, $B_i = [g^{b_i}]_p$, $C_i = [g^{c_i}]_p$; and proves to the other servers that

$$\begin{cases} log_{\overline{K}_i}((g^{\sigma_i} {C_i}^{-1} {\overline{K}_i}^{-\mu_i})^{{\rho_i}^{-1}}) = log_g y_i \\ log_{\overline{K}_i}(g^{v_i} {B_i}^{-1}) = log_g A_i. \end{cases}$$

   Each server verifies that the above proofs are valid, and that $M_i = [g^{\mu_i}]_p$, $R_i = [g^{\rho_i}]_p$. Then, the servers verify the correctness of $\overline{K}_i$, B_i, and C_i using the below described method. Any server who fails or refuses is replaced, and the signature generation restarts.

Method to verify the correctness of public shares without revealing the secret shares: Given a (t, n) secret sharing scheme and the public shares $(\Delta_1, \ldots, \Delta_n)$, out of which at most t are incorrect, we want to establish which ones are incorrect. Consider the following approach:

1. To establish Δ (if this is not already explicitly known, e.g., for zero-sharing), we pick N possible subsets of size $t + 1$ at random, compute the interpolation of the Δ_i's of these sets, and assign $\tilde{\Delta}$ to the most frequent result.

2. Let $\tilde{\Delta}$ be the assumed result of the interpolation, and randomly pick a set S of size $t + 1$ such that $\tilde{\Delta}$ is the interpolation of the Δ_i's of this set. For each other value Δ_j, calculate the interpolation of Δ_j and the values Δ_i, $i \in S$; if this does not equal $\tilde{\Delta}$, then Δ_j is assumed to be incorrect. If more than t public shares are assumed incorrect, then restart.

We have that $Corresponds(coin, tag)$ iff ${tag_a}^{rm^{-1}} \equiv_p tag_b$, for $tag = (tag_a, tag_b) = (g^\mu, h^\rho)$ and $coin = (m, r, s)$. The validity of a coin is described by $Correct(coin)$, which is satisfied iff $r = [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q$ and $Corresponds(coin, tag)$ for a tag tag in the Bank database.

Remark about robustness:
We see that the above method assures that cheating servers are caught, and that no transcript properties are lost when no complaint is filed. Also note that if R files a unjustified complaint, then *this* will be established, since it will be found that no server cheated. Finally, note that no secret information of honest servers will be leaked to R if R receives an invalid signature transcript. R has no motivation to complain on a good signature; this results in early "unblinding". Each time a threshold is used and opened, the misbehaving processors are eliminated and the process starts afresh (to avoid leaking information) based on new random choices. This may result in a delay of at most t_s times. Note that the method is applicable due to the probabilistic nature of the computation and the care in opening erroneous results.

III.I Tracing

We present tracing methods based on the verification of an undeniable signature (introduced by Chaum and van Antwerpen [15]). We use a (t_t, n) threshold scheme for the verification of the undeniable signature, such that the scheme is sound, correct and zero-knowledge (e.g., [16].)

There are three types of traces that can be performed: (1) given $coin$, find tag such that $Corresponds(coin, tag)$, (2) given a withdrawal session, and the corresponding tag, compute a description of the corresponding coin, and (3) given a coin and a tag, verify whether they correspond to each other.

We recall that the stored tag is of the form $(tag_a, tag_b) = (g^\mu, h^\rho)$, for $h = g^{x_t}$, where x_t is distributed over Bank and Ombudsman servers. The three different types of tracing are performed as follows:

1. $coin \rightarrow tag$.
   Given a description (m, r, s), for each potential withdrawal session, we calculate $(trace_a, trace_b) = ([{tag_a}^{rm^{-1}}]_p, tag_b)$. Using a protocol for verification of undeniable signatures, we verify whether $log_g h = log_{trace_a} trace_b$, which holds iff the coin corresponds to the tag.

2. $tag \rightarrow coin$.
   A description $(coin_a, coin_b) = ({tag_a}^{x_t}, tag_b)$ is calculated by a size-(t_t+1) quorum of x_t holders. This description is output. A certain coin, described by (m, r, s), corresponds to the tag iff ${coin_a}^{rm^{-1}} \equiv_p coin_b$.

3. $(coin, tag) \rightarrow$ yes/no.
   Given a description (m, r, s), and a tag, (tag_a, tag_b), we calculate $(trace_a, trace_b) = ([{tag_a}^{rm^{-1}}]_p, tag_b)$. Using a protocol for verification of undeniable signatures, we verify whether $log_g h = log_{trace_a} trace_b$, which iff the coin corresponds to the tag.

Chapter IV

Analysis

IV.A Analysis of the Magic Ink Scheme

We now prove that the demonstrated scheme satisfies the specification of Magic Ink Signature schemes. More specifically, we prove that our magic ink generation of the DSS signature satisfies requirement MI1-MI4: *Correctness* (Theorem MI1,) *Zero-Knowledge* (Theorem MI2,) *Anonymity* (Theorem MI3,) and *Quorum traceability* (Theorem MI4.)

Theorem MI1:
The magic ink signature scheme achieves correctness, i.e., signatures can be correctly generated using a $(3t_s, n)$ threshold scheme, by any size-$(3t_s+1)$ quorum out of the n signature servers.

Proof of Theorem MI1:
The signature protocol produces the correct result if all participants are honest, since then we have that

$$\begin{cases} \sigma = [\overline{k}(\mu + x\rho)]_q \\ \mu = [m\alpha]_q \\ r = [[\overline{r}^{\beta}]_p]_q \\ \rho = [r\alpha]_q \\ s = [\sigma\alpha^{-1}\beta^{-1}]_q \end{cases}$$

Therefore, $s \equiv_q \overline{k}\beta^{-1}(\mu + x\rho)\alpha^{-1} \equiv_q \overline{k}\beta^{-1}(m\alpha + xr\alpha)\alpha^{-1} \equiv_q \overline{k}\beta^{-1}(m + xr) \equiv_q \overline{k}\beta^{-1}(m + x[g^{k^{-1}\beta}]_p)$. Thus, the signature is valid.
The robustness of the signatures depends on our destructive robustness method for random signatures (on top of the non-robust threshold DSS), and the soundness of the undeniable signature verification protocol used therein: if the receiver obtains an incorrect signature and complains, then, by the soundness of the undeniable signature verification protocol used, any cheating signature server will be caught. (On the other hand, if the receiver complains after having received a *valid* signature, then this fact will be established by the signature servers.)
□

Let $\mathcal{D}$ denote the dishonest signature servers, and $\mathcal{H}$ the honest signature servers. By assumption, we have that $|\mathcal{D}| = k$, $|\mathcal{H}| = 3k + 1$. Let the attacker $\mathcal{A}$ consist of the signature receiver R and $\mathcal{D}$.

Theorem MI2:
There exists a polynomial-time machine $\mathcal{S}$, that given the public information (p, q, g, h, y), the secret keys and strategies of $\mathcal{A}$, and exactly one oracle call to an oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ generating a DSS* signature for the public signature key y, generates a probability distribution that is indistinguishable from the view of $\mathcal{A}$ during an execution of the magic ink signature generating and tracing protocols. Moreover, the simulated tag that $\mathcal{S}$ produces corresponds to the signature obtained from the oracle.

Proof of Theorem MI2:
The simulator $\mathcal{S}$ of the signature generation performs the following steps:

0. The simulator extracts the secret key x_j of each server $j \in \mathcal{D}$ that has at least an ϵ-chance of passing the zero-knowledge proof of knowledge during the identification phase. Those that do not pass are not allowed to participate in the remaining protocol.

1. The simulator generates the messages from the honest servers to the dishonest servers: it selects $\overline{k}_{ij}$, a_{ij}, b_{ij} and c_{ij} uniformly at random from Z_q, where these are the secret share components from server S_i, $i \in \mathcal{H}$ to server S_j, $j \in \mathcal{D}$. It sends $(\overline{k}_{ij}, a_{ij}, b_{ij}, c_{ij})$ to S_j (supposedly from S_i). Let $(\overline{k}_{ji}, a_{ji}, b_{ji}, c_{ji})$ be the message sent from S_j to S_i. Knowing all the $3t_s + 1$ share components of $\overline{k}$, a, b and c, sent from S_j the honest servers, the simulator calculates the share components that should be sent between the servers in $\mathcal{D}$. From all the components that were received by, or should be received by, S_j, the simulator calculates $\overline{k}_j$, a_j, b_j, and c_j, for $j \in \mathcal{D}$.

2. The simulator performs the following:

   (a) The simulator computes the anticipated value of v_j, $j \in \mathcal{D}$: $\hat{v}_j = [\overline{k}_j a_j + b_j]_q$. Similarly, the anticipated values $\hat{A}_j = [g^{a_j}]_p$ are computed. The simulator selects $t_s + 1$ values v_i, $i \in \mathcal{H}$ uniformly at random from Z_q – this defines $\hat{v}$, the $(2t_s, n)$-interpolation of the shares. The remaining values v_i, $i \in \mathcal{H}$ are set so that $\hat{v}$ remains the correct interpolation. The simulator makes a call to $\mathcal{O}_1$ to obtain $\hat{r}$, and sets $\hat{A} = [\hat{r}^{\hat{v}}]_p$. Then, the values A_i, $i \in \mathcal{H}$ are set so that $\hat{A}$ is the (t_s, n) interpolation of $\{\hat{A}_j\}_{j \in \mathcal{D}} \cup \{A_i\}_{i \in \mathcal{H}}$. The simulator publishes (v_i, A_i) for $i \in \mathcal{H}$, and receives (v_j, A_j) from $j \in \mathcal{D}$.

   (b) Just as for the real protocol, A and v are computed by interpolation, and $\overline{r} = [A^{v^{-1}}]_p$ is calculated and sent to R.

3. The simulator simulates the receiver R until it outputs (on the n private channels) $(\mu_1, \rho_1) \ldots, (\mu_n, \rho_n)$ and publishes $(M_1, \ldots, M_n)$, $(R_1, \ldots R_n)$, where M_i supposedly equals $[g^{\mu_i}]_p$, and R_i supposedly equals $[h^{\rho_i}]_p$. Given the $3t_s + 1$ pairs (μ_i, ρ_i), $i \in \mathcal{H}$, the simulator can calculate (μ_j, ρ_j), $j \in \mathcal{D}$ by interpolation.

4. (a) If for any $i \in \mathcal{H}$ the received shares do not correspond to the public versions, then S_i publishes the shares, and all servers in $\mathcal{H}$ halt.

   (b) The tag tag is computed as the interpolations (g^{μ}, h^{ρ}).

   (c) Set $\mu' = [\mu \overline{r} \rho^{-1}]_q$. The simulator makes the second part of the oracle call to receive σ' as the response from $\mathcal{O}_2(\mu')$; $\hat{\sigma} = [\rho \overline{r}^{-1} \sigma']_q$ is computed.
   Recall now that the simulator knows x_j, $\overline{k}_j$, and c_j, μ_j, ρ_j, $j \in \mathcal{D}$. It now

generates the anticipated share of σ from S_j: $\hat{\sigma}_j = [\overline{k}_j(\mu_j + x_j\rho_j) + c_j]_q$. It sets the shares σ_i, $i \in \mathcal{H}$ so that σ is the $(3t_s, n)$ interpolation of $\{\hat{\sigma}_j\}_{j\in\mathcal{D}} \cup \{\sigma_i\}_{i\in\mathcal{H}}$ (with the first $2t_s$ shares selected uniformly at random from Z_q.) The simulator outputs σ_i for server S_i, $i \in \mathcal{H}$, and receives σ_j from S_j, $j \in \mathcal{D}$. The servers interpolate σ from these shares, using $(3t_s, n)$ interpolation, and send σ to R.

5. The simulator waits.

6. If the receiver files a complaint then the following is performed: The simulator selects values $\overline{K}_i$, B_i, resp. C_i so that $\overline{K}$, 0, resp. 0 are the (t_s, n), $(2t_s, n)$, resp. $(3t_s, n)$ interpolations of the antipated values for $\overline{K}_j$, B_j, resp. C_j, for $j \in \mathcal{D}$, and these new values. Then, the simulator simulates S_i, $i \in \mathcal{H}$ to output $(\mu_i, \rho_i, \overline{K}_i, B_i, C_i)$, and simulates the zero-knowledge proofs of the protocol. The servers S_j, $j \in D$ are simulated to output the corresponding values and to prove the correct exponentiations. If any server fails or refuses, then the simulation halts.

□

Corollary MI2 :
No set of fewer than $t_s + 1$ servers can construct a valid Bank and Ombudsman signature, given that the secret key shares are distributed using a (t_s, n) secret sharing scheme.

Theorem MI3:
Assume that a set $\mathcal{D}$ of less than or equal to t_t signature servers participate in the signature generation protocol in which a signed message (m, r, s) and a tag tag is generated. If $\mathcal{D}$, without the help of the signature receiver, can, if later given (m, r, s), distinguish tag from a pair (g^{r_1}, g^{r_2}) for random numbers (r_1, r_2) with a non-negligible probability, then this would contradict assumption 1.

Proof of Theorem MI3:
Assume we have a potential undeniable signature (m', s') w.r.t., (g, h), and want to learn whether it is valid or not. Assume $t_t \leq t_s$ servers can do so; since the secret key x_t is distributed using a (t_t, n) secret sharing scheme, the servers do not have any information about the secret key.
We construct a signature generation simulator $\mathcal{S}$ generating a signed message (m, r, s) and a tag tag. $\mathcal{S}$ has the property that if (m', s') is a valid undeniable signature w.r.t. (g, h), then the view of the cheating servers is identical to that during a real signature generation, and tag is the tag on (m, r, s); if (m', s') is not a valid undeniable signature w.r.t. (g, h), then tag is a random number. Therefore, iff the dishonest servers, $\mathcal{D}$, can correlate this signature to its tag, then (m', s') is a valid undeniable signature w.r.t. (g, h).
In our simulation, we control the signature receiver R, and simulate the behavior of the $n - t_t$ honest signature servers $\mathcal{H}$. Pick the secret x uniformly at random from Z_q, by distributed secret sharing – note that this is not a problem since x and x_t are unrelated. Let (m, r, s) be an arbitrary, valid DSS signature using the secret key x. Select a tag $tag = (tag_a, tag_b)$ such that $([{tag_a}^r]_p, [{tag_b}^m]_p) = (m', s')$. The tag (tag_a, tag_b) is said to correspond to a signature (m, r, s) iff ${tag_b}^m \equiv_p {tag_a}^{rx_t}$, which for the above choice of the tag holds iff $s' \equiv_p {m'}^{x_t}$. In other words, the tag corresponds to the coin iff s' is an undeniable signature on m' w.r.t. (g, h); if (m', s') is a random pair w.r.t. (g, h), then tag is randomly distributed w.r.t. (m, r, s). The latter holds since $(tag_a, tag_b) = ({m'}^{r^{-1}}, {s'}^{m^{-1}})$: if (m', s') are randomly

distributed and (m, r, s) are fixed, then (tag_a, tag_b) is also randomly distributed. We now simulate the transcripts received by the servers in $\mathcal{D}$ as follows. This is similar to the the signature generation protocol (Theorem MI2), but here we control R. If (m', s') is valid, tag will correspond to the signature. If not, it will be random and independent of the signature. For denotational simplicity, we consider the worst case, $t_t = t_s$.

0. A secret key x is generated as for the real protocol. The servers in $\mathcal{D}$ are simulated, and the messages to these constructed, so that h is the (t_t, n) - interpolation of the corresponding public shares (as done in the proof of Theorem MI2.) If any server fails, then they are excluded. (We note that x and x_t are independent.)

1. The simulator generates the messages from the honest servers to the dishonest servers: it selects $\overline{k}_{ij}$, a_{ij}, b_{ij} and c_{ij} uniformly at random from Z_q, where these are the secret share components from server S_i, $i \in \mathcal{H}$ to server S_j, $j \in \mathcal{D}$. It sends $(\overline{k}_{ij}, a_{ij}, b_{ij}, c_{ij})$ to S_j (supposedly from S_i). Let $(\overline{k}_{ji}, a_{ji}, b_{ji}, c_{ji})$ be the message sent from S_j to S_i. Knowing all the $3t_s + 1$ share components of $\overline{k}$, a, b and c, sent from S_j the honest servers, the simulator calculates the share components that should be sent between the servers in $\mathcal{D}$. From all the components that were received by, or should be received by, S_j, the simulator calculates $\overline{k}_j$, a_j, b_j, and c_j, for $j \in \mathcal{D}$.

2. The simulator performs the following:

   (a) The simulator computes the anticipated value of v_j, $j \in \mathcal{D}$: $\hat{v}_j = [\overline{k}_j a_j + b_j]_q$. Similarly, the anticipated values $\hat{A}_j = [g^{a_j}]_p$ are computed. The simulator selects $t_s + 1$ values v_i, $i \in \mathcal{H}$ uniformly at random from Z_q – this defines $\hat{v}$, the $(2t_s, n)$-interpolation of the shares. The remaining values v_i, $i \in \mathcal{H}$ are set so that $\hat{v}$ remains the correct interpolation. The simulator makes a call to $\mathcal{O}_1$ to obtain $\hat{r}$, and sets $\hat{A} = [\hat{r}^{\hat{v}}]_p$. Then, the values A_i, $i \in \mathcal{H}$ are set so that $\hat{A}$ is the (t_s, n) interpolation of $\{\hat{A}_j\}_{j \in \mathcal{D}} \cup \{A_i\}_{i \in \mathcal{H}}$. The simulator publishes (v_i, A_i) for $i \in \mathcal{H}$, and receives (v_j, A_j) from $j \in \mathcal{D}$.

   (b) Just as for the real protocol, A and v are computed by interpolation, and $\overline{r} = [A^{v^{-1}}]_p$ is calculated and sent to R.

3. The simulator generates random shares $\rho_j, \mu_j \in_u Z_q$, $(M_j, R_j) = (g^{\mu_j}, h^{\rho_j})$, for $j \in \mathcal{D}$. It then selects $\rho_i, \mu_i \in_u Z_q$, and calculates (M_i, R_i), $i \in \mathcal{H}$, such that the (t_s, n)-interpolation of $\{M_j\}_{j \in \mathcal{D}} \cup \{M_i\}_{i \in \mathcal{H}}$ equals tag_a, and the (t_s, n)-interpolation of $\{R_j\}_{j \in \mathcal{D}} \cup \{R_i\}_{i \in \mathcal{H}}$ equals tag_b.

4. The simulator simulates the dishonest servers; the servers in $\mathcal{H}$ do not complain that their shares (μ_i, ρ_i) do not correspond to (M_i, R_i). If any server in $\mathcal{D}$ would complain, then their shares are inspected, and the honest servers halt.
   The tag is computed as in the real protocol. By assumption 3, we have that tag will correspond to the signed message.
   The servers generate some response and send it to R (and R is sent the signature s on m for r, which does not need to involve the dishonest servers, whose output is ignored.)

If the dishonest servers output that the given signed message (m, r, s) corresponds to the tag tag, then output "valid signature", otherwise output "invalid signature". □

Theorem MI4:
The magic ink signature scheme achieves quorum traceability, i.e., valid signatures can always be traced by any size-$(t_t + 1)$ quorum out of the n servers. Here, traced means the following: Given a *coin*, the *tag* such that $Corresponds(coin, tag)$ can be computed; given *tag*, the *coin* such that $Corresponds(coin, tag)$ can be computed; and given a *coin* and a *tag*, it can be established whether $Corresponds(coin, tag)$. In other words, no coalition of less than $t_s + 1$ signature servers, interacting with honest servers a polynomial number of times in the signing and tracing protocols, can produce a valid signature that will not be traced to the tag of one of the signing sessions by any group containing at least $t_t + 1$ honest servers in the tracing protocol. Furthermore, we want signatures generated by an attacker who compromises the secret key of the signers (or forces the signers to sign using a protocol different from that specified by these) always to be possible to identify by any size-$t_t + 1$ quorum of signers.

Lemma MI4a:
Any valid signature (m, r, s) produced by a coalition of less than or equal to t_s dishonest servers interacting in a polynomial number of signing and tracing protocols, will correspond to one of the tags for the signing protocols they participated in.

Proof of Lemma MI4a:
By assumption 2, DSS* is existentially unforgeable, and by Corollary MI2, t_s servers are not able to generate valid signatures without interaction with other servers. Consider the simulation $\mathcal{S}$ in the proof of Theorem MI2. Let y be the public key for DSS*. We produce the correct view for the dishonest servers $j \in \mathcal{D}$ of the corresponding keys: We simulate server $j \in \mathcal{D}$ until it announces y_j, and we set y_i, $i \in \mathcal{H}$ so that y is the (t_s, n) interpolation of y. In more detail, we simulate the transcript received by $j \in \mathcal{D}$ in the distributed secret sharing of y. This is done the same way as in the proof of Theorem MI2. When an oracle call later is made in the simulation, assume that our supposed oracle makes the corresponding call to a DSS* signer with key y. By Theorem MI2, and assumption 3, the tags generated by the simulator will correspond to the signatures obtained. However, the transcripts received by the signature receiver and the dishonest servers in this (variation of the) simulation will be indistinguishable from transcripts received by them in the real protocol. If a new (and valid) signature is not in the same equivalence class as a signature seen by the DSS* signer (i.e., that the signature does not correspond to the same tag as any signature seen by DSS*), then this must also be the case for the simulation (since the simulated transcripts are identically distributed to the transcripts of the real protocol). This would contradict assumption 3. □

Lemma MI4b:
Assume that $Corresponds(coin, tag)$. If *coin* is the input to the tracing protocol (with tracing direction coin to tag), and at least $t_t + 1$ honest servers participate in this protocol, then the output of the protocol will be *tag*.

Proof of Lemma MI4b:
According to the protocol description, we have that $(trace_a, trace_b) = ({tag_a}^{rm^{-1}}, tag_b)$. Since x_t is shared using a (t_t, n) threshold scheme, $t_t + 1$ servers will be able to distributively and robustly verify whether $log_g h$ (which by definition equals x_t) equals $log_{trace_a} trace_b$, by the use of a sound and correct zero-knowledge protocol for verifying undeniable signatures. Iff this holds, the following holds: ${trace_a}^{x_t} \equiv_p trace_b$.By the format of (tag_a, tag_b), this is

the same as saying $\mu r m^{-1} \equiv_q \rho$, i.e., $Corresponds(coin, tag)$ for $coin = (m, r, s)$. Therefore, it will only be satisfied for a tag corresponding to the coin of the input, and this tag will be output. □

Lemma MI4c:
Assume that $Corresponds(coin, tag)$. If tag is the input to the tracing protocol (with tracing direction tag to coin), and at least $t_t + 1$ honest servers participate in this protocol, then the output of the protocol will be $coin$.

Proof of Lemma MI4c:
According to the protocol description, we have that $(coin_a, coin_b) = ({tag_a}^{x_t}, tag_b)$. Since x_t is shared using a (t_t, n) threshold scheme, $t_t + 1$ servers will be able to distributively and robustly compute this pair, by proving that they perform the correct exponentiation. Iff ${coin_a}^{rm^{-1}} \equiv_p coin_b$ holds, the following holds: $\mu r m^{-1} \equiv_q \rho$, i.e., $Corresponds(coin, tag)$ for a matched coin $coin = (m, r, s)$. Therefore, the coin corresponding to the tag that constitutes the public input to the protocol (and only such a coin) will yield equivalence, and therefore, the protocol output $(coin_a, coin_b)$ is a description of this coin. □

Lemma MI4d:
If $(coin, tag)$ is the input to the tracing protocol (for comparison tracing), and at least $t_t + 1$ honest servers participate in this protocol, then the output of the protocol will be yes iff $Corresponds(coin, tag)$.

The proof of Lemma MI4d follows from the proof of Lemma MI4b, since in the comparison tracing, we only limit the number of tags compared in the coin-to-tag tracing.

Lemma MI4e:
If and only if there is no tag tag for a valid coin $coin$ so that $Corresponds(coin, tag)$, then $coin$ must have been obtained through a bank robbery.

Proof of Lemma MI4e:
By Lemma MI4a, we have that for each time the standard protocol for generating magic ink signatures is used, a tag that corresponds to the coin being authenticated will be produced. Therefore, if there is no tag that corresponds to a valid coin, this coin must have been produced using a protocol other than the standard protocol (whether coerced or by an insider attack); this is what we call a bank robbery. □

Proof of Theorem MI4:
By Lemma MI4a, we have that for each coin produced using the standard magic ink signature protocol, the corresponding tag will be generated. By Lemma MI4b-MI4d, a quorum of $t_t + 1$ honest servers will be able to trace such a coin to such a tag; trace such a tag to such a coin, or verify whether such a coin corresponds to such a tag. If there is no tag that corresponds to a given coin, then, by Lemma MI4e, we have that this coin must have been obtained in a bank robbery attack. □

IV.B Analysis of the E-Money System

We now prove that the suggested e-money scheme, using the previously introduced primitive for distributed magic ink signatures, satisfies its specification. More specifically, we prove that our e-money scheme satisfies requirement EM1-EM9: *Unforgeability* (Theorem EM1,) *Impersonation safety* (Theorem EM2,) *Overspending detection* (Theorem EM3,) *Overspending robustness* (Theorem EM4,) *Traceability* (Theorem EM5,) *Revocability* (Theorem EM6,) *Anonymity* (Theorem EM7,) *Framing-freeness* (Theorem EM8,) and *Refundability* (Theorem EM9.)

Theorem EM1:
The system achieves *unforgeability*, i.e., a set of users, shops, and Bank and Ombudsman servers, not including a quorum of the latter, are not able to perform payments for a value exceeding $\mathcal{V}$, which are later accepted by an honest Bank as valid, after engaging in withdrawal protocols withdrawing funds for a value of $\mathcal{V}$.

Lemma EM1a:
For each spendable coin *coin*, the Bank cooperated to produce one signature, and has a tag *tag* such that $Corresponds(coin, tag)$ holds.

According to the requirements (see section III.B), the Ombudsman controls at most t_s signing servers, so according to Corollary MI2 we have that at least one Bank server must be involved in the transaction. The correctness of Lemma EM1a now follows from Lemma MI4a (the tag corresponding to the coin produced will be generated for each withdrawal session.)

Lemma EM1b:
We let an attacker not containing the withdrawer of a given properly withdrawn coin ask for a polynomial number of signatures using the secret key of the coin to be generated (in an adaptive chosen message manner for messages c_i.) It is not possible for this attacker afterwards to produce a valid signature a using the same key on a message c not previously signed.

Proof of Lemma EM1b:
Assume the contrary. This means that, given the public key y_{Coin} of the coin, it is possible to produce a message-signature pair (c, a) not earlier seen after seeing a polynomial number of such correct pairs. This is impossible since the coin signature scheme, (S, V), is assumed to be existentially unforgeable. □

Lemma EM1c:
For each deposited coin, the Bank will find out its corresponding public key y_{Coin} and the value of the spending, and will know the total value of all the spendings of each coin that has been deposited.

Proof of Lemma EM1c:
When a coin is spent and deposited, its public key y_{Coin} must be sent, by the specification of the protocol. The correctness of the public key will be authenticated by the combined Ombudsman and Bank signature s_{Coin} on it. The corresponding secret key, x_{Coin}, must for each spending be used to produce a signature a on the challenge c of the spending. Only coins of this format can be accepted by the Bank for credit when deposited. According to the book-keeping procedure given in section II.E.3, we have that for each deposited coin,

indexed either by (y_{Coin}, s_{Coin}) or an appropriately chosen function thereof, the Bank will know the total value of the spendings. □

Lemma EM1d:
If a coin *coin* gets overspent, then the Bank and the Ombudsman will be able to compute the identity of the party that identified itself during the withdrawal of *coin*.

Proof of Lemma EM1d:
For each coin *coin*, the Bank and the Ombudsman will be able to compute the tag *tag*, such that $Corresponds(coin, tag)$ by using the tracing protocol (see Theorem MI4). If the coin was withdrawn using the normal withdrawal protocol (as opposed to a bank robbery), then the Bank will have a record (id, tag) in its database of withdrawn coins, such that *id* is the identity of the participant who proved his identity to the Bank during the withdrawal protocol. □

Proof of Theorem EM1:
By Lemma EM1a, at least one honest Bank server must be involved in the generation of the signature on the coin public key y_{Coin}, that is a part of the deposited coin. By Lemma EM1b, the payer needs to generate a new signature w.r.t the public key y_{Coin} for each new payment and deposit transcript. According to Lemma EM1c, the Bank can tell payment transcripts of different coins from each other by (y_{Coin}, s_{Coin}), or an appropriate function thereof, and will know the total sum spent by its owner for each coin. Therefore, it is not possible to overspend coins without being detected. According to Lemma EM1d, it will be possible for the Bank and the Ombudsman to establish the identity of the participant who identified himself during the withdrawal of the overspent coin (unless the coin was obtained through a bank robbery, in which case this will be established.) Therefore, the scheme achieves unforgeability. □

Theorem EM2:
The system achieves impersonation safety, i.e., if there is no attacker consisting of a quorum of Bank and Ombudsman servers, or if transactions can be legally challenged by taking the case to a judge, then no coalition of users, shops, Bank and Ombudsman servers can succeed in charging an honest user more than what his withdrawals total.

Proof of Theorem EM2:
Since the identification scheme used is sound, it will not be possible for any set of participants not including the withdrawer $u \in \mathcal{U}$ to produce a valid withdrawal request from u. Since all communication is either authenticated or on dedicated channels, it will not be possible for any set of Bank and Ombudsman servers to substitute messages sent between u and the signing servers, without this being detected. Therefore, unless the attacker controls a quorum of Bank and Ombudsman servers, it will not be possible for him to perform a transaction after which he has a pair (x_{Coin}, s_{Coin}) constituting a spendable coin *coin*, and for which (u, tag) is recorded by the Bank, such that $Corresponds(coin, tag)$. The only time when any function of x_{Coin} (other than the value y_{Coin}) is revealed by u is during the payment protocol, when u signs a challenge c using the signature protocol associated with (x_{Coin}, y_{Coin}). This scheme is by assumption existentially unforgeable, and so, does not allow an attacker to produce a valid signature not generated by u after having received a polynomial number of signatures on chosen messages. Therefore, for each valid payment transcript $(y_{Coin}, s_{Coin}, c, a)$, corresponding to a tag linked to u, the signature a on c must have been produced by u, or a participant cooperating with u. □

Note: If the user sends a signed acknowledgement of having received the withdrawn coin after having obtained a valid pair (x_{Coin}, s_{Coin}), but not before that, and only u can produce this signature, then if the Bank does not have this signature on a withdrawal, then the Bank does not have a complete withdrawal view (making any claims of improper use invalid,) which would be recognized by a third party, such as a judge.

Theorem EM3:
The system achieves overspending detection, i.e., if a set $\mathcal{A}$ of attackers performs an overspending attack in which a value $\mathcal{V}$ is withdrawn and a value $\mathcal{V}_1 > \mathcal{V}$ is spent, then a quorum of the Bank and Ombudsman servers will be able to establish (a) the sum of the overspending, i.e., $\mathcal{V}_1 - \mathcal{V}$, and (b) the identity of at least one member of $\mathcal{A}$.

Proof of Theorem EM3:
The Bank will not accept a deposit unless it is of the proper format, $(y_{Coin}, s_{Coin}, c, a)$, where $s_{Coin} = s_{B/O}(y_{Coin})$, and $a = s_{x_{Coin}}(c)$. Here, the former is the Bank/Ombudsman signature on y_{Coin}, and the latter the signature, using the secret key corresponding to y_{Coin}, on c. Since the Bank/Ombudsman signature scheme by assumption is existentially unforgeable, we have that if the Bank receives such a deposit for a public key y_{Coin}, the same public key must have been previously signed by a Bank/Ombudsman quorum. The value of the spending, which will be associated with (y_{Coin}, s_{Coin}), or an appropriate function thereof, will be obvious to the Bank given the valid deposit transcript, since the challenge specifies this amount and the coin signature scheme is assumed to be existentially unforgeable, too. Therefore, if a coin is overspent, the Bank will know this, as soon as a value exceeding the legal value has been deposited. Furthermore, the Bank will be able to lower-bound the amount of the overspending at this time; the precise value of the overspending will be known at the end of the expiration time (for deposits) of the coin. By Theorem MI4, the Bank will be able to establish the tag corresponding to $(y_{Coin}, s_{Coin}, c, a)$ by cooperation with the Ombudsman. Each such tag is associated with the identity of the party who identified himself during the corresponding withdrawal protocol. □

Theorem EM4:
The system achieves overspending robustness, i.e., if a set $\mathcal{A}_1$ of attackers perform an overspending attack in which a value $\mathcal{V}$ is withdrawn and a value $\mathcal{V}_1 > \mathcal{V}$ is spent, then a set $\mathcal{A}_2$ of users, shops, Bank and Ombudsman servers, disjoint from $\mathcal{A}_1$, cannot make an honest quorum of Bank and Ombudsman servers running the overspending detection protocol output a sum of overspendings $\mathcal{V}_2 - \mathcal{V} > \mathcal{V}_1 - \mathcal{V}$ and only identities of participants in $\mathcal{A}_1$.

Proof of Theorem EM4:
For each time the user spends a coin, he will have to give a signature on a challenge using the secret key of the coin. As long as only a polynomial number of signatures are given, we have that, according to the definition of an existentially unforgeable signature scheme, that it will not enable an adversary to sign a new message. □

Theorem EM5:
The system achieves traceability, i.e., any Bank and Ombudsman quorum can, regardless of the actual withdrawal protocol used, and regardless of whether an "illegal untraceability" attack is executed, perform the following actions:

1. Given a $tag \in \mathcal{T}$, calculate $coin \in \mathcal{C}$, such that $Corresponds(tag, coin)$.
2. Given a $coin \in \mathcal{C}$, calculate $tag \in \mathcal{T}$, such that $Corresponds(tag, coin)$.
3. Given a $tag \in \mathcal{T}$ and a $coin \in \mathcal{C}$, establish whether $Corresponds(tag, coin)$.

All of these actions are performed in a way that does not give any set of participants a non-negligible advantage (apart from the advantage gained by knowing the *result* of the above calculation) in establishing whether $Corresponds(tag', coin')$ for any $tag' \in \mathcal{T}$, $coin' \in \mathcal{C}$, but $(coin', tag') \neq (coin, tag)$.

This follows automatically from Theorem MI3 (anonymity), Theorem MI4 (quorum traceability,) and the fact that for each withdrawn coin, the Bank stores the tag. If there for a given $coin$ is no tag such that $Corresponds(coin, tag)$, then $coin$ must have been obtained in a bank robbery attack. This fact will be established by the use of the tracing protocol.

Theorem EM6:
The system achieves revocability, i.e., any traced coin can be permanently resp. temporarily made unspendable by blacklisting resp. freezing.

Proof of Theorem EM6:
Since, by Theorem EM5, *any* coin can be traced at any time after its withdrawal, it will be possible for the Bank to construct a list of coins (or rather, their corresponding public keys y_{Coin}, or coin invariants I_{Coin}) not to be accepted (whether temporarily or permanently) and distribute this (in an authenticated form) to all the shops. Similarly, it will be possible for the Bank to remove coins from this list (corresponding to thawing of funds) by broadcasting a signed list of coins on the black list to be taken off the latter (or just an updated version of the black list, not containing descriptions of the thawed coins.) □

Theorem EM7:
The system achieves anonymity, i.e., the probability for any coalition of participants not containing a quorum of Bank and Ombudsman servers to determine for any particular pair $(tag, coin) \in \mathcal{T} \times \mathcal{C}$ whether $Corresponds(tag, coin)$, is not non-negligibly better than that of a guess, uniformly at random, given all the available pairs of descriptions of $tag' \in \mathcal{T}$ and $coin' \in \mathcal{C}$ and all the already computed relations R available.

This follows from Theorem MI3, anonymity for the magic ink signature scheme, and the fact that no other information linking withdrawals to identities or sessions is calculated in the payment scheme.

Theorem EM8:
The system achieves framing-freeness, i.e., it is not possible for a set $\mathcal{A}$ of dishonest users, shops, Bank and Ombudsman servers, not including any member of a set $\mathcal{U}$ to produce a set of transcripts, that, if a tracing is performed with these as input, the output would identify a member of $\mathcal{U}$ with non-negligible probability if (a) the withdrawer has a public key associated with him, and he signs withdrawals, or (b) there is no quorum of Ombudsman and dishonest Bank servers.

Whereas the requirement 'impersonation safety' is not implying or implied by the requirement 'framing-freeness', our proof that the system satisfies impersonation safety also proves that it satisfies framing-freeness. We therefore refer to the proof of Theorem EM2.

Theorem EM9:
The system achieves refundability, i.e., if $\mathcal{A}$, a set of users, shops, Bank and Ombudsman servers, not including any member of a set $\mathcal{U}$, makes only a value $\mathcal{V}_1 < \mathcal{V}_2$ be accepted as valid by an honest set of Bank servers, after members of $\mathcal{U}$ spend funds with a value $\mathcal{V}_2 \leq \mathcal{V}$ from the said withdrawals, then members of $\mathcal{U}$ can prove that the attack took place, resulting in the identification of at least one of the members of $\mathcal{A}$.

Proof of Theorem EM9:
If a user $u \in \mathcal{U}$ does not obtain a valid signature s on the public key y_{Coin} during a withdrawal, then he can prove this by presenting the session keys to the Bank and Ombudsman servers, who will then (using the method for destructive robustness) be able to identify the cheater. We refer to the proof of correctness (Theorem MI1.)
If a user $u \in \mathcal{U}$ has a valid pair (x_{Coin}, s_{Coin}), and is blocked from spending this coin to its full value, then he can complain to a Judge. Since the coin signature scheme is assumed to be existentially unforgeable, and only u can know (x_{Coin}, s_{Coin}), we see that the Bank will not be able to produce valid payment transcripts of the coin to a value exceeding that spent by the user (see the proof of Theorem EM2.) Therefore, the Bank will be identified as a cheater if it blocks a user from spending a coin to its full amount.
After a bank robbery, the Bank and the Ombudsman will in a robust fashion construct a list of descriptions of all properly withdrawn coins (but only these). By the robustness of this protocol, the Bank can not be misled by a cheating Ombudsman not to accept coins that are valid, or accept them to a lesser value. Thus, if a set of participants blocks some valid payments from being accepted, the identity of at least one such cheater will be known.
□

Remark: It is clear that the shop cannot change the semantics of a coin, or it would be able to change the challenge in the basic system, thus obtaining two valid coins (or more) to deposit, instead of one. Since the extensions are only based on the semantics (which is a convention how to set the challenge, agreed upon by all participants) the security of the extended system is unchanged.

Bibliography

[1] G. Agnew, R.C. Mullin and S. Vanstone, "Improved digital signature scheme based on discrete exponentiation," Electronics Letters, v. 26 1990, pp. 1024–1025.

[2] J. Bar-Ilan, and D. Beaver, "Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds," Proceedings of the ACM Symposium on Principles of Distributed Computation, 1989, pp. 201–209.

[3] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik and M. Waidner, "iKP - A Family of Secure Electronic Payment Protocols', Proceedings of the First USENIX Workshop on Electronic Commerce, New York, July 1995, pp. 89–106.

[4] M. Bellare and S. Micali, "How To Sign Given Any Trap-Door Permutation," Journal of the ACM, Vol. 39, No. 1, Jan 1992, pp. 214–233.

[5] M. Bellare and P. Rogaway, "Random Oracles are Practical: a paradigm for designing efficient protocols", The First ACM Conference on Computer and Communications Security, Nov. 1993, pp. 62–73.

[6] M. Bellare and P. Rogaway, "The Exact Security of Digital Signatures – How to Sign with RSA and Rabin," Advances of Cryptology, Eurocrypt '96, pp. 399–416.

[7] M. Ben-Or, S. Goldwasser and A. Wigderson, "Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations," In Proceedings of the twentieth annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 1–10.

[8] E. Berlekamp and L. Welch, "Error correction of algebraic block codes," US Patent 4,633,470.

[9] S. Brands, "Untraceable Off-line Cash in Wallets with Observers," Advances in Cryptology - Proceedings of Crypto '93, pp. 302–318.

[10] S. Brands, "An Efficient Off-line Electronic Cash Systems Based on the Representation Problem," C.W.I. Technical Report CS-T9323, The Netherlands.

[11] S. Brands, personal communication, Nov/Dec 1994.

[12] E. Brickell, P. Gemmell and D. Kravitz, "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change," Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1995, pp. 457–466.

[13] J. Camenisch, U. Maurer and M. Stadler, "Digital Payment Systems with Passive Anonymity-Revoking Trustees," Computer Security - ESORICS 96, volume 1146, pp. 33–43.

[14] J. Camenisch, J-M. Piveteau and M. Stadler, "An Efficient Fair Payment System," Proceedings of the 3rd ACM Conference on Computer and Communications Security, 1996, pp. 88–94.

[15] D. Chaum and H. van Antwerpen, "Undeniable Signatures," Advances in Cryptology - Proceedings of Crypto '89, pp. 212–216.

[16] D. Chaum, "Zero-knowledge undeniable signatures," Advances in Cryptology - Proceedings of Eurocrypt '90, pp. 458–464.

[17] D. Chaum, "Blind Signatures for Untraceable Payments," Advances in Cryptology - Proceedings of Crypto '82, 1983, pp. 199–203.

[18] D. Chaum, A. Fiat and M. Naor, "Untraceable Electronic Cash," Advances in Cryptology - Proceedings of Crypto '88, pp. 319–327.

[19] D. Chaum, "Achieving Electronic Privacy," Scientific American, August 1992, pp. 96–101.

[20] D. Chaum and T. Pedersen, "Wallet databases with observers," Advances in Cryptology - Proceedings of Crypto '92, pp. 89–105.

[21] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," Proceedings of the 26th Annual Symposium on Foundations of Computer Science, 1985, pp. 383-395.

[22] CitiBank and S. S. Rosen, "Electronic-Monetary System," International Publication Number WO 93/10503; May 27 1993.

[23] G.I. Davida, Y. Frankel, Y. Tsiounis, and M. Yung, "Anonymity Control in E-Cash Systems," Financial Cryptography 97.

[24] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to Share a Function Securely," In Proceedings of the twenty-sixth annual ACM Symposium on Theory of Computing (STOC), 1994, pp. 522–533.

[25] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," Advances in Cryptology - Proceedings of Crypto '89, pp. 307–315.

[26] W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Trans. Info. Theory IT-22, Nov. 1976, pp. 644–654.

[27] T. ElGamal "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," Advances in Cryptology - Proceedings of Crypto '84, pp. 10–18.

[28] P. Feldman, "A Practical Scheme for Non-Interactive Verifiable Secret Sharing" The twenty-eighth annual Symposium on Foundations of Computer Science (FOCS) 1987, pp. 427–437.

[29] N. Ferguson, "Extensions of Single-term Coins," Advances in Cryptology - Proceedings of Crypto '93, pp. 292–301.

[30] A. Fiat and A. Shamir, "How to Prove Yourself: a practical solutions to identification and signature", Advances in Cryptology - Proceedings of Crypto '86, pp. 186–194.

[31] Y. Frankel, Y. Tsiounis, and M. Yung, "Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-Cash," Advances in Cryptology - Proceedings of Asiacrypt 96, pp. 286–300.

[32] M. Franklin and M. Yung, "Towards Provably Secure Efficient Electronic Cash," Columbia Univ. Dept of C.S. TR CUCS-018-92, April 24, 1992. (Also in ICALP-93, July '93, Lund, Sweden, LNCS Springer Verlag).

[33] M. Franklin and M. Yung, "Blind Weak Signatures and its Applications: Putting Non-Cryptographic Secure Computation to Work," Advances in Cryptology - Proceedings of Eurocrypt '94, pp. 67–76.

[34] A. M. Froomkin, "Anonymity and its Enmities" Journal of Online Law, art. 4, 1995.

[35] E. Fujisaki, T. Okamoto, "Practical Escrow Cash System", LNCS 1189, Proceedings of 1996 Cambridge Workshop on Security Protocols, Springer, pp. 33 – 48.

[36] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, "Robust Threshold DSS Signatures", Advances in Cryptology - Proceedings of Eurocrypt '96, pp. 354–371.

[37] D. K. Gifford, L. C. Stewart, A. C. Payne, and G. W. Treese, "Payment Switches for Open Networks," Proceedings of Compcon '95, pp. 26–31.

[38] S. Glassman, M. Manasse, M. Abadi, P. Gauthier and P. Sobalvarro, "The millicent protocol for inexpensive electronic commerce," In World Wide Web Journal, Fourth International World Wide Web Conference Proceedings, O'Reilly, December 1995, pp. 603–618.

[39] S. Goldwasser, S. Micali and R. Rivest, "A 'Paradoxical' Solution to the Signature Problem," 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 441–448.

[40] S. Goldwasser and S. Micali, "Probabilistic Encryption" *JCSS*, 28(2), 1984, pp. 270–299.

[41] S. Goldwasser, S. Micali and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM Journal of Computing 17(2), April 1988, pp. 281–308.

[42] R. Hauser, M. Steiner and M. Waidner, "Micropayments based on iKP," 14th Worldwide Congress on Computer and Communications Security Protection, 1996, pp. 67–84.

[43] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk and M. Yung, "Proactive Public Key and Signature Systems," Proceedings of the 4th ACM Conference on Computer and Communications Security, 1997, pp. 100–110.

[44] A. Herzberg, S. Jarecki, H. Krawczyk and M. Yung, "Proactive Secret Sharing, or How to Cope with Perpetual Leakage," Advances in Cryptology - Proceedings of Crypto '95, pp. 339–352.

[45] ISO/IEC 9796 "Information Technology: security techniques– Digital signature scheme giving message recovery.

[46] M. Jakobsson, "Ripping Coins for a Fair Exchange," Advances in Cryptology - Proceedings of Eurocrypt '95, pp. 220–230.

[47] M. Jakobsson, "Escrow Cash,", Tech Report CS94-401, University of California, San Diego, December '94.

[48] M. Jakobsson, "A Practical Election Scheme Based on Homomorphic Mixes," Manuscript.

[49] M. Jakobsson and M. Yung, "Revokable and Versatile Electronic Money," 3rd ACM Conference on Computer and Communications Security, 1996, pp. 76–87.

[50] M. Jakobsson and M. Yung, "Distributed 'Magic Ink' Signatures," Advances in Cryptology - Proceedings of Eurocrypt '97, pp. 450–464.

[51] M. Jakobsson and M. Yung, "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System," Advances in Cryptology - Proceedings of Financial Cryptography '97.

[52] S. Jarecki and A. Odlyzko, "An efficient micropayment system based on probabilistic polling," Advances in Cryptology - Proceedings of Financial Cryptography '97.

[53] C. Jutla and M. Yung, "Paytree: 'amortized signature' for flexible micropayments," 2nd USENIX Workshop on Electronic Commerce, November 1996.

[54] C. Lai, G. Medvinsky, and B. C. Neuman, "Endorsements, licensing, and insurance for distributed system services," Proceedings of the Second ACM Conference on Computer and Communications Security, Nov. 1994, pp. 170–175.

[55] S. H. Low, N. F. Maxemchuk and S. Paul, "Anonymous Credit Cards," Proceedings of the Second ACM Conference on Computer and Communications Security, Nov. 1994, pp. 108–117.

[56] G. Medvinsky and B. C. Neuman, "Netcash: A design for practical electronic currency on the internet," Proceedings of the First ACM Conference on Computer and Communications Security, Nov. 1993, pp. 102–106.

[57] S. Micali, "Fair Cryptosystems," Advances in Cryptology - Proceedings of Crypto '92, pp. 113–138.

[58] D. M'Raïhi, "Cost-Effective Payment Schemes with Privacy Regulation," Advances in Cryptology - Proceedings of Asiacrypt '96.

[59] NBS FIPS PUB 46, "Data Encryption Standard," National Bureau of Standards, U.S. Department of Commerce, Jan 1977.

[60] NIST FIPS PUB XX, "Digital Signature Standard," National Institute of Standards and Technology, U.S. Department of Commerce, Draft, 1 Feb. 1993.

[61] B. C. Neuman and G. Medvinsky, "Requirements for Network Payment: The NetCheque^{TM} Perspective," Compcon '95, pp. 32–36.

[62] T. Okamoto, "An Efficient Divisible Electronic Cash Scheme," Advances in Cryptology - Proceedings of Crypto '95, pp. 438–451.

[63] T. Okamoto and K. Ohta, "Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash," Advances in Cryptology - Proceedings of Crypto '89, pp. 481–496.

[64] T. Okamoto and K. Ohta, "Universal Electronic Cash," Advances in Cryptology - Proceedings of Crypto '91, pp. 324–337.

[65] R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks," Proc. of the 10th ACM Symposium on the Principles in Distributed Computing, 1991, pp. 51–61.

[66] T.P. Pedersen, "Distributed Provers with Applications to Undeniable Signatures," Advances in Cryptology - Proceedings of Eurocrypt '91, pp. 221–242.

[67] H. Petersen, G. Poupard, "Efficient scalable fair cash with off-line extortion prevention," Technical Report LIENS-97-7, Ecole Normale Superieure, (1997), 33 pages.

[68] B. Pfitzmann and M. Waidner, "How to Break and Repair a 'Provably Secure' Payment System" Advances in Cryptology - Proceedings of Crypto 91, pp. 338–350.

[69] D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes," Advances in Cryptology - Proceedings of Eurocrypt '96, pp. 387–398.

[70] R. Rivest, "The MD5 Message Digest Algorithm," RFC 1321, Apr. 1992.

[71] R. Rivest and A. Shamir, "PayWord and MicroMint: two simple micropayment schemes," Cryptobytes, vol. 2, num. 1, 1996, pp. 7–11.

[72] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v. 21, n. 2, Feb 1978, pp. 120–126.

[73] Secure Electronic Transaction (SET) Specification.
`http://www.visa.com/set`, `http://www.mastercard.com/set`

[74] A. Shamir, "How to Share a Secret," Communications of the ACM, Vol. 22, 1979, pp. 612–613.

[75] M. Sirbu and J. D. Tygar, "NetBill: An Internet Commerce System Optimized for Network Delivered Services," Compcon '95, pp. 20–25.

[76] S. von Solms and D. Naccache, "On Blind Signatures and Perfect Crimes," Computers and Security, 11 (1992) pp. 581–583.

[77] C. P. Schnorr, "Efficient Signature Generation for Smart Cards," Advances in Cryptology - Proceedings of Crypto '89, pp. 239–252.

[78] M. Stadler, "Cryptographic Protocols for Revokable Privacy," PhD Thesis, ETH No. 11651, Swiss Federal Institute of Technology, Zürich, 1996.

[79] M. Stadler, J-M. Piveteau and J. Camenisch, "Fair Blind Signatures," Advances in Cryptology - Proceedings of Eurocrypt '95, pp. 209–219.

[80] J. Stern and S. Vaudenay, "SVP: a Flexible Micropayment Scheme," Advances in Cryptology - Proceedings of Financial Cryptography '97.

[81] J. M. Tenenbaum, C. Medich, A. M. Schiffman, and W. T. Wong, "CommerceNet: Spontaneous Electronic Commerce on the Internet," Compcon '95, pp. 38–43.

[82] Y. Tsiounis, "Efficient Electronic Cash: New Notions and Techniques," PhD Thesis, College of Computer Science, Northeastern University, 1997. `http://www.ccs.neu.edu/home/yiannis`

[83] B. Witter, "The Dark Side of Digital Cash," Legal Times, January 30, 1995.

[84] Y. Yacobi, "Efficient Electronic Money," Advances of Cryptology - Proceedings of Asiacrypt '94, pp. 153–164.

[85] Y. Yacobi, "On the continuum between on-line and off-line e-cash systems - I," Advances in Cryptology - Proceedings of Financial Cryptography '97.

[86] A. Yao, "How to Generate and Exchange Secrets," In Proceedings of the twenty-seventh annual Symposium on Foundations of Computer Science (FOCS) 1986, pp. 162–167.