Revokable and Versatile Electronic Money

(Extended Abstract)

Markus Jakobsson* **Moti Yung†**

Abstract

We present an e-money system where both value of funds and user anonymity can be *revoked* or suspended unconditionally, but only by the cooperation of banks and consumer rights organizations. We introduce the "ultimate crime," where an active attacker gets the bank's key or forces the bank to give "unmarked bank notes". Our system, unlike all current anonymous systems, can prevent such a crime from successfully being perpetrated, and employs *revocation* to do so.

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 easily extends the basic needs of a practical payment scheme to allow for coin divisibility, checks, credit card purchases and surety bonds. Moreover, the system (unlike some previous ones) is robust against problems arising from spurious equipment.

1 Introduction

E-cash enables the exchange of digital coins with value assured by the bank's signature and with concealed user identity. This flexible tool can enhance the prevalence and capability of fund transfers while simultaneously increasing its users' privacy. Yet, the availability of such a strong, flexible access-control capability may become dangerous (serving as a double-edged sword): robberies, blackmailing, money-laundering and illegal purchases are possible misuse of money, and must not be eased by the choice of the paradigm for the digital transfer of funds. In fact, the danger of anonymous e-cash may prevent its legality ([17, 42]). Thus, provision is needed to allow for legal cancellation of the power encompassed in the anonymous e-cash mechanism. 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.

*Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093, Email: markus@cs.ucsd.edu. Research supported by NSF YI Award CCR-92-570979, Sloan Research Fellowship BR-3311 and The Royal Swedish Academy of Sciences. Part of the work was done while visiting IBM T.J. Watson.

†IBM T.J. Watson Research Center Yorktown Heights NY 10598 Email: moti@watson.ibm.com

What we achieve:
Our **first objective** is to aid law-enforcement by allowing revocation of user anonymity and value of funds, allowing traceability and blacklisting as part of an off-line e-cash system (once anonymity is broken, further measures can be taken in accordance with the law). The system achieves *unforgeability* of funds, *anonymity* for the honest user, *blindfolded-freeness* (that an attacker cannot force a withdrawal of a coin from the bank by coercion) 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 [28]. 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 and a fair exchange (ensuring that neither the payer nor the payee cheats each other). This is 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. We also design a mechanism by which certain coins are eligible for deposit only after certain triggering events have occurred (e.g. implementing surety bonds [25]).

Our **third objective** is robustness against spurious faults (overspending robustness). In contrast, a system like [2, 3] has 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.) This may be extremely dangerous. In our system the user will only be liable for the overspendings he or she actually performed.

2 Preliminaries

In order to allow funds to be traced, frozen and revoked, absolute anonymity has to be excluded. But we do not wish tracing to be possible at the whims of the bank, as transaction analysis can be used or sold for direct marketing, junk mail targeting and spying on people. Therefore, we will split the tracing function between the bank and a consumer rights organization, the *ombudsman*[1]. The ombudsman will register information about each withdrawal, and will release tracing information to the bank when presented with an appropriate court order, allowing the bank either to obtain the identity of the withdrawer of a coin or a list of what coins

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

were withdrawn by a certain person. This will allow the bank to blacklist certain coins, effectively freezing or revoking the corresponding funds, depending on whether the blacklisting is temporary or not.

To allow this splitting of the tracing capability, we need the withdrawal protocol not to be solely based on the bank's signature function. In fact, allowing the withdrawal of funds to be based solely on a publicly (directly) verifiable equation is dangerous in various respects. First, the bank may be coerced by an attacker to reveal its secret key (e.g., internal fraud). Second, if withdrawals are blinded, an attacker may try to coerce the bank to get e-cash by forcing the bank 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 bank, then it is not possible for the bank 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. 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 will introduce *dual verification signatures*, i.e., signatures such that their verifications under some circumstances need an "authenticator", the ombudsman. We will need two primitives:

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 [43]). We will employ a three-party protocol for payments where the ombudsman gets involved in the signing.

2. *A mechanism for anonymity revocation.* In order not to sacrifice user anonymity, we need a method to blind the above signatures by the ombudsman, who will not engage in unauthorized tracings, i.e., unblind signatures. The ombudsman will cooperate in producing the above signature in a way that prevents the bank from associating coins with identities without involving the ombudsman.

Related Work:
A large variety of electronic cash schemes has been designed in recent years [7, 2, 10, 13, 15, 32, 33, 31]. Most off-line schemes (introduced by Chaum, Fiat and Naor) use a restricted form of *blind signatures*, (introduced by Chaum [6]) to implement anonymity.

However, recently, von Solms and Naccache [38] show that schemes with user anonymity are susceptible to two attacks: *money laundering* and *blackmailing*. Brickell, Gemmell and Kravitz [5] have developed a trustee-based electronic cash scheme that can be used to prevent the above attacks. Also Stadler, Piveteau and Camenisch [40] gave blind signature schemes that can be used to prevent money-laundering. (Neither of these schemes protects against the the bank robber introduced here, who may actively use coercion against the bank).

Our solution that protects against bank robbery and other attacks bears resemblance to the *weak blind signatures* of Franklin and Yung [16]. 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 an on-line checking center, and 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). Our solution is also (remotely) related to the anonymous credit card of Low, Maxemchuk and Paul [26], 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 [27, 28, 21, 37]; here, however, anonymity is not an issue, but backwards compatibility and simplicity are emphasized.

3 Model

The system is modelled by eight types of participants. A participant is an interactive (possibly distributed) polynomial-time randomized computation. Participants are connected by a communication network that all can write to and read from. Moreover, the bank and the ombudsman will share a private channel so that they can communicate secretly the status of a transaction.

The eight participants are: Users (1) who honestly withdraw and pay money. Users enjoy computational anonymity. Attackers (2), on the other hand, depart from the specified protocols and may *actively* force action on other participants. Shops (3) get money from users and attackers, and deposit it in the bank, while banks (4) 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 (5) participates in withdrawals and assist in reacting to court orders. The judge (6) will employ enforcement mechanisms, and issues court orders. Finally, there is a certification authority (7) and key directory (8) for public keys. The banks, the ombudsman, the judge and the certification authority will each have a public key associated with. These public keys will be certified by the certification authority. Individual shops will have a (not necessarily certified) public key associated with them.

System Events (overview):

1. Open an account: A user opens an account with a bank.

2. Withdraw money: A user withdraws money from a bank.

3. Spend money: A user performs a payment by sending a coin to a shop.

4. Spend money (on-line version): A user performs a payment by sending a coin to a shop; the shop contacts the ombudsman.

5. Deposit money: A shop deposits money received by a user at a bank.

6. Deposit money (on-line version): A shop deposits money received by a user at a bank. The bank contacts the ombudsman.

7. Refresh money: A user updates unspent coins with respect to the expiration date.

8. Register new keys: The banks certify and publish their public keys.
9. Detect overspending: A bank detects and identifies coins which were overspent above their denomination.
10. Trace: A bank and the ombudsman interact in a protocol to trace coins.
11. Blacklist: A bank sends out a certified list of coins not to be accepted, possibly together with a court order allowing this.
12. Fund freezing: As above, but with the difference that items can later be taken off the black list.
13. Fund liquefying: Taking a coin off the black list.
14. Alert (on/off): A bank alerts shops to verify the correctness of coins with the ombudsman before accepting them as payment.
15. Arbitration: If there is any disagreement or suspicion of criminal activity, the case can be taken to the judge, who will settle the dispute after examining the transcripts exhibited.
16. Certification: Certification of public keys and documents.
17. 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.

TIME:
Time is divided into (possibly overlapping) time periods of predetermined and publicly known starts and lengths. This division implies a reduction of the storage space required for the bank's money data-base. A withdrawn coin is accepted by the bank only within its marked period.

TRUST MODEL:
The following assumptions underlie the architecture:

1. The users trust the bank and the ombudsman not to conspire against them.
2. The honest users trust the ombudsman not to give the bank any information to compromise their anonymity.
3. The bank trusts the ombudsman to give it sufficient information to allow tracing in case of an overspending, or when the judge issues a court order.
4. The bank trusts the ombudsman to be available during withdrawals, traces, and alerts.
5. All the participants trust the judge to be honest and fair.
6. All the participants trust the certification authority to perform only certifications of accurate documents and correctly associated pairs of public keys and names of owners.
7. The users trust the bank not to try to steal their money[2]

[2] This requirement can 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.

MECHANISMS TO ENSURE JUSTICE:
There are complaint procedures allowing participants to prove the possible wrongdoings of other participants to the judge.

4 Attacks and Requirements

ATTACKS: we consider the following attacks (which are more active than what has been previously considered in our setting):

1. *Forgery:* When a set of participants, excluding the bank, 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 accepted by the bank as valid.
2. *Overspending:* When a user spends coins for a total value exceeding the allowed value.
3. *Impersonation:* An attack in which an attacker gains access to another user's funds without this user's cooperation.
4. *Money laundering:* An illegal transfer of funds performed in order to misrepresent the distributions of incomes of (or hide the existence of) organizations. (While the system is not actively preventing these attacks, it nevertheless traces them.)
5. *Illegal purchases:* Examples of illegal purchases are payments for drugs, etc. (Again, the system only traces these events upon suspicion).
6. *Blackmail:* When an attacker forces a user to perform a withdrawal from his own account, and transfer the funds to the attacker in a way that allows the attacker (only) either to deposit or spend the coins without being traceable.
7. *Bank robbery:* When a set of attackers force the bank to engage in (possibly blindfolding) protocols (e.g., for withdrawal of funds) in order to obtain coins that later can be successfully spent. This needs a more careful modelling: We assume the attacker threatens and directs the bank to give him coins. When the attack is over (after a short[3] withdrawal process) the threat that forced the bank to comply will have vanished and an alert may be issued to merchants. There is a delay Δ associated with each payment, and Δ is longer than the propagation time for the alert.[4]

4.1 Requirements

Our solution fulfills the following requirements:

Unforgeability: Forgery as defined above is infeasible.

Anonymity w.r.t. the Bank: It is infeasible for the bank to match a coin to the identity of a honest payer without the cooperation of the ombudsman.

[3] Since the attacker does not want to be traced physically over the network.

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

Anonymity w.r.t. the Ombudsman: It is infeasible for the ombudsman to match a coin to the identity of a honest payer without the cooperation of the bank.

Blindfolded-freeness: It is not possible to obtain a blinded coin without the bank's knowledge of the fact that this particular coin has been blinded.

Traceability: The bank and the ombudsman can cooperate and match any coin to its withdrawer[5], regardless of the withdrawal protocol.

Revocability: Any coin, including those obtained in bank robberies, can be permanently/temporarily made unspendable by resp. blacklisting/freezing.

Refundability: It is possible to prove to a judge that the ombudsman has not followed the withdrawal protocol, as long as the bank and the ombudsman do not conspire. It is possible to prove to a judge that the ombudsman or a bank does not accept a correctly withdrawn coin. If the judge rules for the user, she forces a refund to be given.

Framing-freeness: The bank[6] or the ombudsman cannot falsely incriminate a user.

Overspending Robustness: If an overspending is performed then: no coalition of shops, users (other than the coin owner), and the ombudsman, with access to all spending transcripts, can generate a new transcript for the coin with public key y, that will be accepted by the bank when deposited.

Efficiency: The scheme proposed is efficient in storage, communication and computation.

5 Definitions

Next, we present basic definitions (known concepts are only described informally).

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 (polynomial-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 polynomial time 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 a polynomial time algorithm which given E, $\overline{m}$, and (PK, SK), outputs m.

[5] If, during a bank robbery, the "withdrawer" does not correctly identify himself (as can be expected), it will be impossible to match to the identity of the attacker to the withdrawn coins, only to the withdrawal session. This, however, is sufficient in order for the bank to blacklist these coins.

[6] Only applicable if the withdrawer has a public key associated with him, and it signs withdrawals.

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 a random encrypted message $\overline{m}$ with some non-negligible probability (in k). More formally, we adopt the polynomial security of **Probabilistic Encryption:** (see [19]), where there is no p-time algorithm $\mathcal{A}$ that can distinguish between pairs $(m, E(m))$ and (m, r) for random strings $r \in E(\cdot)$.

Definition 2: Signature Scheme [11, 20]
A digital signature has the following components:

- A *security parameter* k, a *message space* and a *key generation algorithm* as above.
- A *signing scheme* $S = (S_S, S_R)$, 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 outputs 1 if s is a valid signature for the message m with respect to the public key PK, and 0 otherwise.

Security:
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** [20], 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.

History-Free:
A signature scheme is *history-free*[7] if the signing scheme is not a function of previously signed messages.

Message Recovery Signature Scheme: [22]
A message recovery signature scheme is a signature scheme (S, V) such that given a valid signature s of some unknown message m, the corresponding message m can be obtained by application of the verification algorithm V.

Blind Signature: [6]
A *blind signature* scheme is a pair $S = (S_S, S_R)$ that allows the receiver to obtain a valid signature (m', s') that is statistically uncorrelated to the transcript seen by the signer during the protocol.

Transparently Blindable Signature:
A signature function $S = (S_S, S_R)$ is *transparently blindable* if there exists a blind signature scheme (S_S, S'_R) giving signatures 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:
The signature scheme $S = (S_S, S_R)$ is *blindfoldable* if there exists a blind signature protocol (S'_S, S'_R) giving signatures with the same distribution as $S = (S_S, S_R)$ does, based any on the signer's key.

[7] A signature scheme that is not history free may have problems if used in anonymous e-cash schemes, as shown in [34].

Dual Verification Signature Scheme:
A *dual verification signature scheme* is a seven-tuple $(k, \mathcal{M}_k, \mathcal{KG}, S, V_1, V_2, t)$, with the following property: $(k, \mathcal{M}_k, \mathcal{KG}, S, V_1)$ is a signature scheme with publicly verifiable signatures, and $(k, \mathcal{M}_k, \mathcal{KG}, S, V_2)$ is a signature scheme where a signature can only be verified by interaction with a authenticator. Here, (m, s) may be a correct message-signature pair w.r.t. one of the schemes, but not the other. We call t the triggering condition; this decides whether V_1 or V_2 shall be used for verification of a signature.

Definition 3: Challenge Semantics
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.

Assumption 1:
If (S, V) is a signature scheme that is secure against a random message attack, then (S', V'), *a signature with preprocessed message*, is secure against a chosen message attack, where $S' = S \circ g$, and $V'(m, s) = V(g(m), s)$, and g is a one-way function that is assumed to acts like a random oracle[8] (For example we can use exponentiation over a finite field). Or, when we need message recovery g is assumed to act like a random permutation oracle with reversible message extraction. (For example, the function can be the following: take the message and append its message digest [35] to add redundancy to legally signed messages, then take the first 7-byte block of this input and concatenate to it a CBC encryption (using the first block as a key) of the rest of it, then apply Feistel transformations and permutations. This finishes the preprocessing, now apply the signature function). See [1] for theoretical justification.

5.1 Building Blocks

There are five basic building blocks to be used in our protocols (The building blocks are instantiated at random and independently of each other.)

1. A public-key probabilistic encryption schemes (of the bank and the ombudsman), (E, D) such that E is public and D is secret. (We may use [36] [12], or key agreement [11].)

2. A symmetric encryption scheme (like DES [29]), (E_K, D_K), where K is the session key[9].

3. $(S_{Ombudsman}, V_{Ombudsman})$, the ombudsman signature scheme, which is existentially unforgeable (with preprocessed message as defined above) [10] and 1-1.

4. The bank signature scheme. This is a blindable, history-free, message recovery[11] signature scheme (S_{Bank}, V_{Bank}). (We can use [36, 12] with preprocessed message and assume that inverting the signature amounts to querying a random oracle).

5. The *coin signature* scheme, (S_{Coin}, V_{Coin}). This is an existentially unforgeable signature scheme (which can practically be obtained using a "random oracle-like function", just like for $(S_{Ombudsman}, V_{Ombudsman})$.) If x is the secret key of the signature scheme, then $y = f(x)$ is the corresponding public key, where no participant can invert f on a random instance in polynomial time. The function f behaves like a random oracle; in practice, this can be based on exponentiation in a finite field (of a preprocessed value). It is advantageous to choose a scheme based on discrete log, since this makes it easier to select good secret keys. We can use [12, 39, 30, 14] for (S_{Coin}, V_{Coin}).

We note that applying a one-way function to a message before calculating the signature (as is done by both the ombudsman and the user) is practically used to prevent against chosen message attacks or resp. to avoid transparent blinding. If rigorous provability is required, we can choose *provably* existentially unforgeable signature schemes, e.g. [20], for the ombudsman and coin signature schemes instead.

6 The Basic System

Next we present our solution. Some straightforward functions, such as certification, blacklisting, etc., will not be elaborated on.

We let $\mathcal{X}_{Coin}$ denote the set of possible secret keys x for (S_{Coin}, V_{Coin}), $\mathcal{K}_O$ the set of possible session keys K_O, and $\mathcal{K}_B$ the set of possible session keys K_B.

A Coin: A coin is a pair (x, s), where $x \in \mathcal{X}_{Coin}$ is a random number known only by the withdrawer. Then, $y = f(x)$, $\sigma = S_{Ombudsman}(y)$ and $s = S_{Bank}(\sigma)$. Different denominations can be represented by different bank signatures.

Withdrawing a Coin: Alice will withdraw a coin by engaging in the following 3-party protocol:

1. Alice picks

$$\begin{cases} K_B \in_u \mathcal{K}_B \\ K_O \in_u \mathcal{K}_O \\ x \in_u \mathcal{X}_{Coin} \end{cases}$$

and calculates[12]

$$\begin{cases} \overline{K}_B = E_{Bank}(K_B) \\ \overline{K}_O = E_{Ombudsman}(K_O) \\ y = f(x) \\ \overline{y} = E_{K_O}(y) \\ \overline{id} = E_{Bank}(id) \\ \overline{\overline{id}} = E_{K_O}(\overline{id}) \end{cases}$$

where id is Alice's identity. She sends $(\overline{K}_B, \overline{K}_O, \overline{y}, \overline{\overline{id}})$ to the Bank[13] and identifies herself to the bank. The identification session will be encrypted with their session key, K_B.

[8] The security of several schemes, including Schnorr and Feige-Fiat-Shamir signatures, is based on assumptions like this.

[9] To avoid excessive notation we will use the same symbol e.g. $E(\cdot)$ for public and symmetric encryptions

[10] This is a sufficient requirement, but it is only necessary that the signatures are not transparently blindable.

[11] Thus, there exists an algorithm $V_{Bank,Ombudsman}$ such that $V_{Bank,Ombudsman}(y, S_{Bank}(S_{Ombudsman}(y))) = 1$, but it is hard to find a pair (y, s) such that $V_{Bank,Ombudsman}(y, s) = 1$ even if $S_{Ombudsman}$ is known.

[12] For simplicity, we leave out the random string for the probabilistic encryption from the description.

[13] If framing-freeness w.r.t. the bank is required, Alice needs to sign the transcript to the Bank. If the bank is trusted not to frame users, then no user signature is needed.

2. The bank sends the ombudsman the quadruple $(\overline{K}_O, \overline{y}, \overline{\overline{id}}, n)$, where n is a unique withdrawal session number[14] set by the Bank.

3. The ombudsman calculates
$$\begin{cases} K_O = D_{Ombudsman}(\overline{K}_O) \\ y = D_{K_O}(\overline{y}) \\ \overline{id} = D_{K_O}(\overline{\overline{id}}) \\ \sigma = S_{Ombudsman}(y) \end{cases}$$
and sends $\overline{id}$ to the Bank, who verifies that $D_{Bank}(\overline{id}) = id$. Interacting with each other, and using a normal blinded signature protocol, the ombudsman and the bank produce $s = S_{Bank}(\sigma)$ in a way so that y and σ are blinded from the bank. The ombudsman verifies that $V_{Bank}(\sigma, s) = 1$, sends $\overline{s} = E_{K_O}(s)$ to the bank, and enters $(n, y, K_O, \overline{id})$ in its database. (Assuming the withdrawer is not performing a bank robbery.)

4. The bank sends $\overline{s}$ to Alice, and stores $(n, id, \overline{s}, \overline{\overline{id}})$ in its database. The bank subtracts the value of the coin from Alice's balance.

5. Alice calculates the value $s = D_{K_O}(\overline{s})$, and then verifies that $V_{Bank,Ombudsman}(y, s) = 1$, and then saves[15] (x, s).

Spending a Coin: Alice pays a shop by interacting in the following protocol:

1. Alice sends (y, s) to the shop.

2. Shop checks that $V_{Bank,Ombudsman}(y, s) = 1$, that the coin has not expired, and then chooses a random challenge c which it sends to Alice.

3. Alice responds to the challenge by calculating and sending the shop the value $a = S_{Coin}(x, c)$.

4. The shop verifies that $V_{Coin}(y, a, c) = 1$. If no bank robbery attempt has been reported for the corresponding bank and time period, then the system is off-line, and the shop accepts the payment. Otherwise, it proceeds:

5. The shop sends (y, s) to the ombudsman. The ombudsman verifies that the coin has not expired and that $V_{Bank,Ombudsman}(y, s) = 1$. Then, it verifies that y appears in the list of legal withdrawings[16] and sends a signed acknowledgement if so; otherwise it takes actions to have the attacker caught.

6. When the shop has received a signed acknowledgement from the ombudsman, it accepts the payment.

Depositing a Coin: The shop can deposit received payments at its leisure. It does so by sending (y, s, c, a) to the bank. The bank verifies that

1. $V_{Bank,Ombudsman}(y, s) = 1$,

2. $V_{Coin}(y, a, c) = 1$.

3. the coin has not expired,

4. the transcript was not deposited already.

It then credits the shop's account with the proper amount.

Preventing Overspendings: If the Bank receives k distinctive and correct quadruples
$$(y, s, c_1, a_1) \ldots (y, s, c_k, a_k),$$
such that their cumulative value exceeds the value of the coin with public key y, then the user has overspent the corresponding coin. The bank sends the quadruples to the ombudsman, who after verifying their correctness (as done by the bank for a deposit) answers with the withdrawal session n during which the coin with coin number y was withdrawn. To prevent the ombudsman from cheating in this step and falsely incriminate somebody, the ombudsman will also have to send the session key K_O and random bits used for the withdrawal. Thus, the bank can verify that the claimed coin indeed was withdrawn in this session by verifying that $s = D_{K_O}(\overline{s})$ was indeed sent to the withdrawer. (Note that,if desired, the ombudsman at withdrawal may add a tool that will help the bank finding (by itself) the user encrypted identity in case of overspending. This tool may be based on traditional methods [7, 15, 13, 2]; the procedure, however, extends the size of a coin).

Limiting the Privacy by Revoking the Anonymity: If somebody is suspected of a serious crime, the bank can obtain a court order allowing it to have the privacy removed for some specific withdrawal session n or some coin serial number y. The bank sends this court order to the ombudsman, along with either the specified n or y. The ombudsman searches its database and responds with the $(n, y, K_O, \overline{id})$ used for the withdrawal. The bank looks up $(n, id, \overline{s}, \overline{\overline{id}})$ in the database and verifies that
$$\begin{cases} s = D_{K_O}(\overline{s}) \\ V_{Bank,Ombudsman}(y, s) = 1 \\ \overline{id} = D_{Bank}(\overline{\overline{id}}) \\ id = D_{K_O}(\overline{id}). \end{cases}$$
The bank now knows the triple (id, y, s), and so, the coin has been traced.

User Complaints: If a withdrawer does not receive a correct coin in the withdrawal protocol, she will send the bank the triple (s, y, K_O), plus the random strings used for the probabilistic encryption of $\overline{K}_O$ and $\overline{y}$. The bank will calculate $(\overline{K}_O, E_{K_O}(y))$ and verify that these are identical to the values for $(\overline{K}_O, \overline{y})$ sent by the withdrawer in the withdrawal protocol. If the corresponding transcript was indeed sent and s is not the correct combined signature of the ombudsman and the bank, then the bank will know the withdrawer is right, and will credit her account again.

If a user complains that a shop would not accept a certain coin as a payment because the ombudsman did not find it in the list of legally withdrawn coins, then the user will show the bank the transcript where the ombudsman, in a signed message, tells the shop not to accept the coin (y, s, c, a). If the bank agrees that this is a correct coin one of the following must be true: Either the user is a bank robber or the ombudsman should have accepted the coin. The user will give the bank the (s, y, K_O) of the coin. The bank will look up

[14] This does not have to be random, and so, can be merely a counter, in which case it does not have to be sent over.

[15] K_O and the random strings used when encrypting $\overline{K}_O$ and $\overline{y}$ are saved, too, but not necessarily on the same media, as they are only to be used in case of a disagreement of the validity of a coin.

[16] This corresponds to the activation of the triggering condition of the dual verification signature scheme.

what transcript $(n, id, \overline{s}, \overline{\overline{id}})$ corresponds to the claimed coin. If this coin was withdrawn in a standard withdrawal and not through a bank robbery (those will certainly be marked!) then the ombudsman must be at fault. We discuss the complaint methods further in Appendix A.

7 Security of the System

We will now state the theorems and sketch the proofs, showing that the proposed system satisfies the specified requirements. Namely, the system achieves *unforgeability* (1,) *anonymity w.r.t. the bank* (2,) *anonymity w.r.t. the ombudsman* (3,) *blindfolded-freeness* (4,) *traceability* (5,) *revocability* (6,) *refundability* (7,) *framing-freeness* (8,) and *overspending robustness* (9.) Furthermore, as will be discussed in section 8.6, the system achieves *efficiency*.

Theorem 1: The system achieves *unforgeability*, i.e., a coalition of users, shops and the ombudsman cannot, after engaging in withdrawal protocols corresponding to an amount $\mathcal{V}$, perform valid payments for a value exceeding $\mathcal{V}$, without the bank being able to identify and prove this violation.

Lemma 1a: For each valid coin (x, s), the bank produced one signature.

Proof of Lemma 1a:
Assume the contrary. Then it is possible to construct a valid coin given $S_{Ombudsman}$ and the publicly available information.
First, note that a random permutation oracle is *almost* 1-1 (by counting arguments). Furthermore, $S_{Ombudsman}$ inversion is a 1-1 function. Recall that by the one-way properties of signature schemes (unless the secret key is known,) we assumed these act like random oracles.
We will now show that a valid coin (x, s) cannot be constructed, whether you start from a value x or from the signature s (or, from some value taken from the middle of the signing process):
Direction 1: $(x \rightarrow s)$ Since f acts like a random oracle, setting x and calculating $y = f(x)$ will give a random number. $S_{Ombudsman}(f(x))$, too, will be a random value, since $S_{Ombudsman}$ is 1-1. We cannot calculate S_{Bank} on $S_{Ombudsman}(f(x))$ since this amounts to finding a bank signature for a random value, which cannot be done but with a non-negligible probability.
Direction 2: $(s \rightarrow x)$ We start with a random bank signature s that the user may produce and (by the message recovery properties of the bank's signature scheme) calculate the corresponding message, σ, such that $s = S_{Bank}(\sigma)$. This is the ombudsman signature on a message $f(x)$. S^{-1}_{Bank} acts like a random oracle, and $S^{-1}_{Ombudsman}$ is 1-1, and thus, $f(x)$ will be randomly distributed. Finding the corresponding x is impossible but with negligible probability, since, by assumption, it is hard to invert f on random instances but with a non-negligible probability. □

Lemma 1b: It is only possible to produce a new correct challenge-response pair (c, a) for a coin with public key y if the corresponding secret key x is known, even if a polynomial number of correct pairs $\{(c_i, a_i)\}$ have been seen, where the c_i's are set by the attacker.

Proof of Lemma 1b:
Responses employ the coin's signature. So claim follows from its existential unforgeability. □

Lemma 1c: For each coin spent, the bank will find out its corresponding public key y when the coin is deposited. The bank will know the total value of all the spendings of each coin that has been deposited.

Proof of Lemma 1c:
When a coin is spent and deposited, its public key y 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 on it. The corresponding secret key, x, 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.
When a spent coin is deposited, the bank will after verifying the validity of the related signatures check the following:

1. If that exact transcript (y, s, c, a) has already been deposited, it will ignore the transcript, since it is only repeated (the first deposit is respected due to unforgeability).
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 been spent,) where each coin is indexed by its public key y.
3. If the total value spent using any coin exceeds the value of a coin, the corresponding coin, indexed by its public key y, has been overspent (by its owner, due to unforgeability. (Otherwise, the shop is credited the amount).

Thus, for each coin, indexed by y, the bank will know the total value of the spendings, since by Lemma 1b, it is not possible for the value of a coin to be altered after the spending, by a party other than the spender. □

Proof of Theorem 1:
By Lemma 1a, 1b and 1c we have that for each valid payment transcript, the bank has had to put a signature on the corresponding public key of the coin, that each challenge signed in payment needs the knowledge of the private key x, and due to knowledge of the total spent, the bank detects overspendings of a coin y. Thus, each deposit is either fresh sent by the owner and previously signed by the bank–where it is accepted, or it is a replay of an old valid deposit (where it is ignored), or overspent by the user (again gets noticed and ignored). □

Theorem 2: The system achieves *anonymity w.r.t. the bank*, i.e., it is infeasible for the Bank to trace legal coins without the cooperation of the ombudsman.

Proof of Theorem 2:
The ombudsman must be present at a withdrawal (since the bank cannot forge the ombudsman's signature on a random message, nor invert f on a random message. Assume it could, i.e., assume that it could produce the bank signature S_{Bank} on a message that is $S_{Ombudsman}(y)$ on a given message y. Then, by the message recovery properties of S_{Bank}, it can also produce pairs $(y, S_{Ombudsman}(y))$, which contradicts the fact that $S_{Ombudsman}$ is assumed to be secure against existential forgery. Second, a bank cannot present overspending since (without knowing x) the signature by y is existentially unforgeable. Therefore, the bank cannot trick the ombudsman to give out tracing information due to overspending, prematurely. Third, given the protocol for obtaining court orders as in Appendix A, it is not possible for the bank to

trick the ombudsman to give it tracing information by getting a court order to release, for example, all coins withdrawn by a person Alice, and give the ombudsman a list of withdrawal sessions including withdrawals not made by Alice. This holds since $E_{K_O}(\overline{id})$ is specified in the court order for each coin to trace, making "bait and switch" impossible. (Otherwise, the bank could ask the judge to issue a court order allowing the bank to trace coins that the judge did not agree to have traced.) Therefore, assume that the bank – without the cooperation of the ombudsman – can correlate the payment transcripts seen with a coin described by (y, s). First, assume it can recognize s, but we know that the Bank's signature on σ is blinded from it, i.e., cannot in itself be correlated to any specific y. In order to be able to correlate the transcript seen with the coin withdrawn, the bank has therefore to be able to distinguish the transcript between Alice and the ombudsman, $(\overline{K}_O, E_{K_O}(y), E_{K_O}(s))$, from some other such withdrawal transcript, $(\overline{K}', E_{K'}(y'), E_{K'}(s'))$. This, however, enables the Bank to correlate K_O to $\overline{K}_O$ or K' with $\overline{K}'$, which is a contradiction since $(E_{Ombudsman}, D_{Ombudsman})$ is a probabilistic encryption scheme. □

Theorem 3: The system achieves *anonymity w.r.t. the ombudsman.*

Proof of Theorem 3:
Although the ombudsman knows the exact form of the withdrawn coins, it never gets to see the identity of the user withdrawing them. The proof of identity between the user and the bank will be probabilistically encrypted using the session key of the user and the bank, and the identity is probabilistically encrypted by the bank's scheme. So the identity remains hidden. □

Theorem 4: The system achieves *blindfolded-freeness.*

First, recall that a *transparently blindable* signature is a signature where the signer cannot tell whether he is signing a signature or a blinded signature, since the distribution of the transcripts he sees is indistinguishable, and he uses the same protocol for both. We will start by proving that existentially unforgeable signature schemes are not transparently blindable, or otherwise they would yield more than one correct message-signature pair per interaction. Thus, in order to get a blinded coin, the attacker must make the bank and the ombudsman aware of the attack. We will show that when this happens, the bank and the ombudsman can make the withdrawn coin invalid without the attacker being able to detect this, or in other words, that the attacker cannot obtain a valid coin in a blindfolded fashion.

Lemma 4: If (S, V) is an existentially unforgeable signature scheme vs. a chosen message attack, then it is not transparently blindable.

Proof of Lemma 4:
Assume the contrary. Then, (m', s') will be the blinded signature pair and (m, s) the unblinded pair. If the scheme is blindfoldable, the signer is not aware of the attack, and $V(m', s') = 1$ for $s' = S(m')$. If (m, s) is a correct signature, then $V(m, s) = 1$, as well. However, this means that there must exist at least one correct signature additional to the one the signer has produced, which contradicts the definition of existentially unforgeable vs. a chosen message attack. □

Proof of Theorem 4:
The ombudsman's signature is assumed to be existentially unforgeable. Therefore, according to Lemma 4, $(S_{Ombudsman}, V_{Ombudsman})$ is not transparently blindable. Therefore, if an attacker lets the bank and the ombudsman use the prescribed protocol, the coin cannot be blinded, and it must force these to use a blindfolded protocol. This will make the bank and the ombudsman aware of the attack.
Thus, an attacker cannot obtain a blinded signature on a message without the signers being aware of the attack. (An example of an attack would be if the attacker calculates and blinds $g(y)$, and forces the ombudsman to first sign this, and then re-blind it for the bank's signature and get a bank signature on the value.) If he would try to obtain a signature in a blindfolded fashion, then the bank and in particular the ombudsman (who privately can warn each other if they are under attack) would be aware of the attack, and would not enter the corresponding data to the list of "good" withdrawals. The attacker would not be able to tell this from looking at the signature, since it is only the private actions of the ombudsman that distinguishes a signature that will be accepted as valid from one that will not. The bank then alerts all the shops that they must engage in the online verification protocol for coins with expiration dates agreeing with those on the coins obtained in the bank robbery. When the alert has been sent out, it will be impossible to spend a coin received in the attack, since it will not be accepted by the ombudsman, not being in the list of good withdrawals. □

Theorem 5: The system achieves *traceability*, i.e., for all withdrawn coins, the bank will be able to match a coin to its related withdrawal session. For honestly withdrawn coins the transcripts can be matched to the identity of the withdrawer, whereas coins withdrawn during a bank robbery may only be matched to the attack.

Proof of Theorem 5:
Let (x, s) be an honestly withdrawn coin. When withdrawn, the bank will record the identity of the withdrawer and the (individual) number of the withdrawal session. Likewise, the ombudsman will record the session number and $y = f(x)$. By pooling the information, they will be able to construct lists, either of all coins honestly withdrawn by a specified user or of the users matching certain specified coins.
If (x, s) is a coin obtained in an attack using a non-standard withdrawal protocol, then the bank will know this fact and the (individual) number of the withdrawal session. As before, the ombudsman will know the session number and $y = f(x)$ (the latter follows from the fact that $S_{Ombudsman}$ is not blindfoldable. By pooling the information, they will be able to construct lists of all coins obtained through blindfolded withdrawal protocols. □

Theorem 6: The system achieves *revocability.*

Proof of Theorem 6:
Since, by Theorem 5, *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) not to be accepted or to have any status (whether temporarily or permanently) and distribute this (in an authenticated form) to all the shops. □

Theorem 7: The system achieves *refundability.*

Proof of Theorem 7:
Using the three complaint procedures of Appendix A, the user and one of the bank and the ombudsman can become convinced of the other's (the ombudsman's or the bank's) failure to behave as prescribed. If there are disagreements between the bank and the ombudsman, the judge can be asked to monitor the traffic between them so as to collect evidence what actually transpired. □

Theorem 8: The system achieves *framing-freeness.*

Proof of Theorem 8:
First, it is not possible for either the Bank or the Ombudsman to spend the coin, as they do not know the secret key x of the coin signature scheme of the coin, which is existentially unforgeable.
It will not be possible for the ombudsman to frame a user, i.e., claim that she performed a certain payment (possibly an overspending,) since the ombudsman will have to supply the bank with the session key K_O used in the claimed withdrawal, and, knowing K_O, the bank can verify that the traced coin indeed was withdrawn during this withdrawal session, by verifying that the signature sent was $s = D_{K_O}(\overline{s})$ and that the identity $id = D_{Bank}(D_{K_O}(\overline{\overline{id}}))$.
In order for the bank to frame a user, the bank needs to have the corresponding payment transcripts and the user's signature on the related withdrawal transcript (now we assumed a user signature due to the bank's potential misbehavior). Therefore, unless the bank can forge user signatures, it cannot frame a user. (Note that we can obtain framing-freeness of user w.r.t. the bank only if the withdrawer signs all withdrawal transcripts using his certified and registered public key, as the bank can otherwise fake withdrawals and overspend the corresponding coins, etc.) □

Theorem 9: The system achieves *overspending robustness.*

Proof of Theorem 9:
For each time the user spends a coin, she will have to give a signature on a challenge using the secret key of the coin. Due to existential unforgeability this will not enable an adversary to sign a new message. □

8 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 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. Just like different denominations are marked by different bank signatures on the coins, different types of functionality can be marked by the bank in this way. Thus, one signature can mean that the coin is limited for use as a check, etc.

8.1 An Extension to Divisible Coins

Change giving mechanism is 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. 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, 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 depositer 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_j the value transferred in the ith spending of the coin, the coin is overspent when $\sum_{i=1}^{k} v_i > \mathcal{V}$, where k 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 give the bank the information needed to perform the tracing.

8.2 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 the check 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 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). For each check or credit card purchase (the latter which would always be on-line to assure that the limit is not exceeded), the bank would give the coins to the ombudsman. The ombudsman, who would still keep the lists of the withdrawals, but who would also keep lists of withdrawal session numbers matched to anonymous account numbers (along with user proofs that they access the right anonymous account[17],) would answer the

[17] That is, that the user knows the secret key corresponding to the public key of the account. This public key would not be correlated to the user's identity or the possible public of any other account he keeps. It can, for example, be a zero-knowledge proof of knowledge of a square root or discrete log of the anonymous account identifier.

bank with the corresponding anonymous account number, and the bank would update the balance of the corresponding account. (The ombudsman knows the mapping between the identity and the anonymous account in case the bills are not paid.)

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-cash payments to the bank, designated to the anonymous account.

In either of the two cases, user misbehavior can be detected applying revocation.

8.3 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 [23], can be applied to our payment system in a modular way, but using a totally different 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 for toggling the void bit. 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.

8.4 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 [25]). This will be done by putting a (possibly hashed) 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 December 5th."

8.5 Security of the Extended System

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 (special way to challenge which will have to be agreed upon by all participants) the security of the extended system is unchanged.

8.6 Efficiency Analysis

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 k-spendability, these costs are linear in k. However, the costs in our scheme will, of course, depend on the signature functions used. For example, let (S_{Bank}, U_{Bank}) and $(S_{Ombudsman}, V_{Ombudsman})$ be RSA signatures, and x the secret key of a discrete log based signature scheme with the same size modulo. The user has only to store x and s (the combined ombudsman and bank signature,) along with a counter of how many times the coin has been spent. Therefore, apart from the counter and independently of the value of k, where k is the number of times the coin may be spent, the user has to store only two signatures, i.e., around 2048 bits for 1024-bit modulos. This is significantly less than the amount stored in most other electronic cash schemes (for example [2, 13].) It is *particularly* competitive for large values of k. 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, s, 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. Since y will be of the approximate size of x, as will a, and the challenge probably *much* smaller[18], the amount of information to be transmitted and stored per spending is still in the order of 3 RSA signatures. This, too, compares very well to other schemes. Furthermore, if a multi-spendable coin is spent several times at one occasion, the y and s of that coin will only have to be stored once, and the scheme can be altered to let a reflect several spendings by using challenges of a form where some bits reflect the number of spendings. Therefore, the amount stored by the bank per k-spendable coin will be in the order of $2 + k$ RSA signatures.

Acknowledgments

We would like to thank Rebecca Wright for her remarks on an earlier version. The first author wishes to thank Stefan Brands for very helpful discussions and important suggestions; Russell Impagliazzo, Giovanni Di Crescenzo and Matthias Schunter for important feedback, and David Chaum and Niels Ferguson for valuable discussions before the conception of the article.

References

[1] 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.

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

[18] The challenge will be appropriately expanded before it usage via a pseudo-random generator.

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

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

[5] E. Brickell, P. Gemmell, 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

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

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

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

[9] D. Chaum and T. Pedersen, "Wallet databases with observers," Crypto '92, pp. 89-105

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

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

[12] T. ElGamal "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," Crypto '84, pp. 10-18

[13] N. Ferguson, "Extensions of Single-term Coins," Crypto '93, pp. 292-301

[14] A. Fiat and A. Shamir, "How to Prove Yourself: a practical solutions to identification and signature", Crypto 86.

[15] 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).

[16] M. Franklin and M. Yung, "The Blinding of Weak Signatures," Eurocrypt '94.

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

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

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

[20] 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

[21] IBM Research, "iKP - A Family of Secure Electronic Payment Protocols", USENIX '95.

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

[23] M. Jakobsson, "Ripping Coins for a Fair Exchange," Eurocrypt '95, pp. 220-230

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

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

[26] S. H. Low, N. F. Maxemchuk and S. Paul, "Anonymous Credit Cards," The Second ACM Conference on Computer and Communications Security, Nov. 1994, pp. 108-117

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

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

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

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

[31] T Okamoto, "An Efficient Divisible Electronic Cash Scheme," Crypto '95, pp. 438-451

[32] T. Okamoto and K. Ohta, "Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash," Advances in Cryptology - Proceedings of Crypto '89, 1990, pp. 481-496

[33] T. Okamoto and K. Ohta, "Universal Electronic Cash," Advances in Cryptology - Proceedings of Crypto '91, 1992, pp. 324-337

[34] B. Pfitzmann and M. Waidner, "How to Break and Repair a 'Provably Secure' Payment System" Crypto 91.

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

[36] R. Rivest, A. Shamir, 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

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

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

[39] C. P. Schnorr, "Efficient Signature Generation for Smart Cards," Crypto '89, pp. 239-252

[40] M. Stadler, J-M. Piveteau, J. Camenisch, "Fair Blind Signatures," Advances in Cryptology - Proceedings of Eurocrypt '95, 1995, pp. 209-219

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

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

[43] A. Yao, "How to Generate and Exchange Secrets," Proceedings of the 27th FOCS, 1986, pp. 162-167

Appendix A: Mechanisms to Ensure Justice

We elaborate on the mechanisms sketched earlier:

Complaint methods:

- *The ombudsman does not follow the protocol.*

  If the signature given on the value y is incorrect, then the user can, given y and its encryption, the session key and its encryption, as well as the signature and its encryption convince the bank that the signature was indeed given to the user by the ombudsman, and that it is incorrect. The bank will testify before the judge.

- *The ombudsman refuses a properly withdrawn coin.*

  The ombudsman has to sign the message saying that a certain coin is rejected. This will be given to the shop, which will give it to the user. The user can take this and the coin to the bank, and by identifying the session the coin was withdrawn in, they can verify that the coin was indeed withdrawn in a way that was legal.

- *The bank refuses to accept and give credit for a properly withdrawn but not previously deposited coin. (This has to be by mistake, since we have assumed that the bank does not steal funds.) If mechanisms are added that prevents the bank from stealing funds (so we do not have to assume it is trusted), this complaint procedure will be of more importance.*

  The user together with the ombudsman would be able to identify the session during which the coin was withdrawn, and to verify that the it was of correct form. If the bank cannot produce withdrawal transcripts corresponding to a proof that the coin had already been spend to its full extent, then the judge will rule for the user, after having verified the correctness of the coin.

Obtaining Court Orders: Avoiding the preceding legal matters (such as convincing the judge that a trace should be performed)– the following will be performed. The judge can issue court orders for two different types of tracing:

- Trace a particular coin

  The judge will issue a court order requesting the withdrawal serial number (along with the rest of the related record) corresponding to the coin identifier y to be given to the bank.

- Trace coins of a particular person

  Before obtaining a court order, the bank will have to present the judge with a list of withdrawal sessions for which it wants tracing court orders to be issued. The bank will have to show the related identity, as well as the probabilistically encrypted identity $\overline{id}$ for each withdrawal, along with a proof that $\overline{id}$ indeed corresponds to the claimed identity. Then, the bank will issue a court order requesting the ombudsman to give out the coin identifiers (along with the rest of the related record) corresponding to the given pairs of withdrawal session numbers and $\overline{id}$'s. The ombudsman will verify that the session numbers given correspond to the $\overline{id}$'s given (and signed) by the judge before giving any information to the bank.