Single Term Off-Line Coins

Niels Ferguson

CWI

P.O. Box 4079, 1009 AB Amsterdam

The Netherlands

e-mail: `niels@cwi.nl`

Abstract

We present a new construction for off-line electronic coins that is both far more efficient and much simpler than previous systems. Instead of using many terms, each for a single bit of the challenge, our system uses a single term for a large number of possible challenges. The withdrawal protocol does not use a cut-and-choose methodology as with earlier systems, but uses a direct construction.

1991 Mathematics Subject Classification: 94A60
1991 CR Categories: D.4.6
Keywords and Phrases: Cryptography, Electronic Cash, One-show credentials.

1 Introduction

Electronic coins, or one-show credentials, were first introduced in [CFN90]. Since then there have been several improvements and new constructions [CdBvH$^+$90, vA90, OO92]. All of these, however, are very complicated and inefficient compared to our new construction.

The main requirements for an electronic cash system are the following:

- Security. Every party should be protected from a collusion of all other parties (multi-party security).
- Off-line. There should be no need for communication with a central authority during payment.

- Fake Privacy. The bank and all the shops together should not be able to derive any knowledge from their protocol transcripts about where a user spends her money.

- Privacy (unlinkable or untraceable). The bank should not be able to determine whether two payments were made by the same user, even if all shops cooperate. This obviously implies fake privacy. Under real-world circumstances, a user is identified by means outside the electronic cash system during some of the payment transactions. If payments can be "linked" as originating from the same user, then knowing the identity of the payer during a single payment anywhere allows the tracing of *all* other payments made by that user. This is why fake privacy provides no privacy at all under real-world circumstances.

The scheme in [CFN90] (and later improved in [CdBvH$^+$90]) meets all of these requirements, but is quite complex. The authors also give an extension to electronic "checks" that allow a variable amount of money to be payed with a single signature. In essence, these checks can be thought of as a bunch of coins sharing common overhead.

In [vA90] Hans van Antwerpen described a different scheme which is more efficient than [CFN90], but has the same basic properties.

Okamoto and Ohta introduced the idea of divisibility of electronic cash in [OO92]. This allows a piece of money to be divided into smaller pieces each of which can be spent separately. Their construction satisfies all of the above requirements except the privacy, and the fake privacy of the user is only computationally protected. (In general the entity that chooses the security parameters (bank) should be the one that is computationally protected, while the other parties should be unconditionally protected.)

The most difficult fraud to counter in electronic cash systems is the double-spending. A user can always spend the same coin in two different shops. This fraud cannot be detected at the time of spending as the payments are off-line. The solution that all electronic cash systems use is to detect the double-spending after the fact. At each payment the user is required to release some information in response to a challenge from the shop. One such release of information provides no clue to the user's identity, but two such releases are sufficient to identify the user uniquely.

In all earlier schemes, the identification of double-spenders requires multiple terms in each coin. Each term is used to answer one bit of challenge from the shop during payment. If both possible answers for any term is ever given, the user's identity is revealed. To achieve an acceptable probability of detection, a large number of terms is required. These schemes are therefore inefficient if a piece of money has a fixed value, so all of them provide some way to pay a variable amount of money with a single 'check'. The resulting refunds for partially spent checks pose further security and privacy problems [Hir93] as well as user-interfacing problems. Another major disadvantage of check systems is that they are very complex (the full but concise mathematical description of the protocols in [vA90] requires 40 pages), which makes them extremely difficult to understand, verify, implement or debug.

Recently, in independent work, Franklin and Yung showed a system with a single challenge term, but which still has some security problems [FY92]. In [FY93] they give an improved system which uses multiple challenge terms again and uses a trusted centre to provide 'blank' coins. Both systems seem to be significantly less efficient than our protocols.

Up to this point all electronic cash schemes have used a cut-and-choose protocol for constructing the money. These protocols are by their very nature inefficient. To get a low enough probability of cheating, the cut-and-choose must consist of many terms, half of which are thrown away while the other half are combined into the piece of cash.

Our new system is a coin scheme in which each piece of money has a fixed value. Each coin consists of 3 numbers plus 2 signatures and can be stored in about 250 bytes. During a payment (of possibly several coins) the user has to perform only about 30 modular multiplications plus two multiplications per coin being payed. The withdrawal protocol constructs the coins directly without resorting to cut-and-choose methods.

I would like to thank David Chaum for all his support, and for helping to clean up and improve the withdrawal protocol. Stefan Brands and Ronald Cramer provided many helpful comments.

2 Efficient payments

The initial motivation for the current scheme was the complexity of the electronic cash system described in [vA90]. To simplify the double-spending detection we want to use a polynomial secret sharing scheme [Sha79]. The user (Alice) gives a share of her identity to the shop at each payment in response to a random challenge. If Alice spends the same coin twice, she has to give two different shares which will allow the bank to recover her identity eventually. This solution gets rid of the large number of challenge terms in all previous systems.

The central idea (developed in cooperation with Hans van Antwerpen) is to represent a coin as 3 numbers, $C := f_c(c)$, $A := f_a(a)$, and $B := f_b(b)$

where the $f_{..}$ functions are suitable oneway functions. Alice also gets two RSA-signatures from the bank: $(C^k A)^{1/v}$ and $(C^U B)^{1/v}$, where v is a prime, U is Alice's identity and k is a random number.

When Alice wants to pay, she sends the numbers c, a, and b to the shop. The shop replies with a randomly chosen (non-zero) challenge x. Finally Alice sends $r := kx + U \pmod v$ and the signature $(C^r A^x B)^{1/v}$ which she can easily compute from the two signatures she got from the bank. The shop, in turn, can verify the consistency of these two responses. When spending several coins at the same time, the same challenge can be used and Alice can send the product of all the signatures to the shop. (Coins of different values use a different v but the same modulus.) This reduces the computations done by Alice to two multiplications per coin plus one exponentiation to the power x. For a 20-bit challenge the exponentiation requires about 30 multiplications.

The payment protocol can obviously be converted to a one-move protocol by choosing x as a hash value on the coin(s) and the shops identity. This requires a larger challenge (say 128 bits) but eliminates the interaction between the payer and payee.

At the end of the day, the shop sends c, a, b, the challenge and the response to the bank. The bank can verify the correctness of the coin and credit the shop with the corresponding amount. If Alice spends the same coin twice, she must reveal two different points on the line $x \mapsto Ux + k$ which immediately allows the bank to determine her identity U.

Given this idea for the payment protocol, the problem of withdrawing the coin from the bank remains. Alice must get the signatures on $C^k A$ and $C^U B$ while the bank learns nothing about c, a, b or k.

3 Randomized blind signatures

For our construction of an efficient withdrawal protocol we need a signature scheme with the following properties:

- Alice receives an RSA-signature on a number of a special form, which she cannot create herself.
- The bank is sure that the number it signs was randomly chosen.
- The bank receives no information regarding which signature Alice gets.

We call such a signature a *randomized blind signature*. The scheme that we use here is due to David Chaum [Cha92].

$$
\begin{array}{lcr}
\textsf{Alice} & & \textsf{Bank} \\
a_1 \in_{\mathcal{R}} \mathbb{Z}_n^* & & \\
\gamma \in_{\mathcal{R}} \mathbb{Z}_n^* & & \\
\sigma \in_{\mathcal{R}} \mathbb{Z}_v & & \\
 & \xrightarrow{\gamma^v a_1 g^\sigma} & \\
 & & a_2 \in_{\mathcal{R}} \mathbb{Z}_n^* \\
 & \xleftarrow{a_2} & \\
e \leftarrow f(a_1 \cdot a_2) - \sigma & & \\
 & \xrightarrow{e} & \\
 & & \overline{A} \leftarrow \gamma^v a_1 g^\sigma \cdot a_2 \cdot g^e \\
 & \xleftarrow{(\overline{A})^{1/v}} & \\
a \leftarrow a_1 a_2 & & \\
S \leftarrow (\overline{A})^{1/v} / \gamma & & \\
S^v \stackrel{?}{=} a g^{f(a)} & &
\end{array}
$$

Figure 1: Randomized blind signature scheme

The protocol to get a randomized blind signature is shown in figure 1. All computations (except for the exponents) are done in an RSA system where the bank knows the factorization of the modulus n. The public exponent of the RSA system is v, a reasonably large prime (say 128 bits). Alice starts by choosing a random a_1, and two blinding factors σ and γ. She computes $\gamma^v a_1 g^\sigma$ where g is a (publicly known) element of large order in $\mathbb{Z}_n^*$ and sends the result to the bank. The bank chooses its own contribution a_2 and sends this back to Alice. Alice replies with $f(a_1 a_2) - \sigma$ where $f(\cdot)$ is a suitable oneway function mapping $\mathbb{Z}_n^*$ into $\mathbb{Z}_v$. The bank multiplies $\gamma^v a_1 g^\sigma$ by a_2 and $g^{f(a_1 a_2) - \sigma}$ to get $\gamma^v a_1 a_2 g^{f(a_1 a_2)}$. The bank computes the v'th root of this number and sends it to Alice. Alice divides out the blinding factor γ to get the pair $(a, (a g^{f(a)})^{1/v})$. The number a is called the base number of the signature.

Note: To make the blinding perfect, all computations involving exponents are done modulo v. For example, e is computed as $(f(a_1 a_2) - \sigma) \bmod v$. Alice can correct for the possible additional factor of g^v by multiplying the final signature by $g^{(f(a)-\sigma) \div v}$. In the rest of this paper we will assume implicitly that all computations involving exponents are done modulo v and that the necessary corrections are applied to the resulting signatures.

Assumption 1. It is computationally infeasible to forge a signature pair of the form $(a, (ag^{f(a)})^{1/v})$.

Reasoning. This assumption is a special case of the RSA signature assumption. Suppose Alice tries to forge a signature pair (a, A). If we define $t := f(a)$, then she must have solved the equation $t = f(A^v g^{-t})$. Two ways to solve this equation spring to mind. The first one is to choose A and then try different values of t until you get lucky (probability of success is $1/v$). The second one is to choose $A^v g^{-t}$, compute t, and then try to compute A. This requires the computation of a v'th root. Neither of these methods seems feasible.

Even if Alice has a large number of valid signature pairs, it is still difficult for her to find new ones. A result of Evertse and van Heyst [EvH92] implies (loosely stated) that the only new RSA signatures that can be computed from old ones are multiplicative combinations of the old signatures. If you multiply two signatures of the form $ag^{f(a)}$, you do not get another valid signature, unless $f(ab) = f(a) + f(b)$. We require of f that it is infeasible to find relations of this kind. □

Proposition 2. The bank gets no information regarding $(a, (ag^{f(a)})^{1/v})$ from the protocol in figure 1.

Proof. We define the view of the bank as all the communication to and from the bank, plus all the random choices that the bank made. Given the banks view of a run of the protocol, we will show that for any legal pair (a, A) there is exactly one set of choices that Alice could have made which would result in her receiving the signature pair (a, A) while the bank would get that view. This makes all possible signature pairs equally likely, given the knowledge of the bank.

Given a (from the pair) and a_2 (from the view), a_1 must have been chosen as a/a_2. Alice's choice of σ is given by subtracting the value in the third transmission $(f(a) - \sigma)$ from $f(a)$ (computed from a). Finally, Alice's choice of γ is uniquely determined by the first transmission.

If Alice had indeed chosen a_1, σ and γ in this way, then she would have gotten (a, A) as a signature pair. So, from the banks point of view, all signature pairs are equally likely. □

3.1 'Abuses' by Alice

There are several ways in which Alice can deviate from this protocol which we will investigate briefly. For this we rewrite the protocol as shown in figure 2.

$$
\begin{array}{lcl}
\textsf{Alice} & & \textsf{Bank} \\
A \in \mathbb{Z}_n^* & & \\
B \in \mathbb{Z}_v & & \\
 & \xrightarrow{\quad Ag^B \quad} & \\
 & & a_2 \in \mathbb{Z}_n^* \\
 & \xleftarrow{\quad a_2 \quad} & \\
C \in \mathbb{Z}_v & & \\
 & \xrightarrow{\quad C \quad} & \\
 & \xleftarrow{(Aa_2g^{B+C})^{1/v}} & \\
D \in Z_n^* & & \\
E \in Z_v & & \\
R \leftarrow D(Aa_2g^{B+C})^{E/v} & &
\end{array}
$$

Figure 2: Possible behaviours of Alice

We assume that Alice is choosing the numbers A–E in some clever way. For practical reasons we have to restrict Alice's behaviour a bit. In choosing the numbers, she can use any construction, but we assume that the only computations that she does with the final signature are an exponentiation and a multiplication. (Other operations don't seem to make sense on an RSA signature.) Furthermore, Alice should end up with a v'th root on a number of the form $Kg^{f(K)}$ for some K that Alice can compute. Any other results are not of interest in the coin system.

First of all, observe that B only occurs as $C + B$ in the result. As C is chosen later then B, we can assume $B = 0$ without loss of generality.

To get a useful signature, Alice must have $R = (Kg^{f(K)})^{1/v}$, which is equivalent to $D^vA^Ea_2^Eg^{EC} = Kg^{f(K)}$. We can assumed that Alice doesn't use a factor g^x in A for some x. (Alice can get the same effect by adding x to C.) The exponents on g are modulo v, so D cannot contribute to them, and a_2 certainly doesn't contribute to the powers of g. As Alice doesn't know $g^{1/v}$, she cannot put any extra factors in herself. To get something useful she must therefore solve the following two simultaneous equations.

$$
\begin{aligned}
EC &= f(K) \pmod{v} \\
D^vA^Ea_2^E &= K
\end{aligned}
$$

There seems only one way to do this, and that is to fix K by choosing D, A, and E, and then choosing C to fit the first equation. (Any other way would involve inverting the oneway function, or computing a root.) But this means that D and E must have been fixed before sending C to the bank.

We conclude that Alice can 'abuse' this protocol by sending a slightly different reply in the third message. She can choose any D and E such that she gets a valid signature (on a number of the form $xg^{f(x)}$) after raising it to the E'th power and multiplying it by D. The factor D doesn't help Alice much in cheating. Alice can modify the base number after the bank has revealed a_2, but she can only multiply the base number by a D^v. Because Alice cannot compute roots, any v'th power is essentially random to her, thus she cannot control the way in which she changes the base number. Alice could at most use the D factor to shift the uniform distribution of the base number slightly, but in the large set of possible base numbers this is hardly significant. The same basically holds for the E power.

Unfortunately, we cannot prove that there is no other way for Alice to cheat. The attacks allowed in figure 2 are only the most obvious ones. There might for example be an attack in which Alice computes the cosine of the last reply to get something useful, but this seems somewhat unlikely to give any useful result. At present, the state of the art in cryptography does not allow us in general to prove the security of such a protocol.

Notice that when $E \neq 1$ it is essential for Alice to be able to compute a_2^E after receiving a_2. For our withdrawal protocol we also need a restricted version which does not allow Alice to choose $E \neq 1$. Instead of a_2, the bank sends $h^{a_2} \bmod p$ where p is a prime of the form $un + 1$ (n is the RSA modulus), and h is a publicly known element of order n in $\mathbb{F}_p$. The exponents of h are reduced modulo n (as h is of order n) so the numbers in the exponent behave in exactly the same way as in the RSA system. The final form of the signature will not be $(ag^{f(a)})^{1/v}$ but rather $(ag^{f(h^a)})^{1/v}$.

The modified protocol is shown in figure 3. It depends on the fact that Alice cannot compute $h^{(t^k)}$ from h^t. We will investigate this problem further in a separate paper.

4 Coin withdrawal protocol

For our system we need 3 numbers, C, A, and B. These will be of the form

$$\begin{aligned} C &= cg_c^{f(h_c^c)} \\ A &= ag_a^{f(a)} \\ B &= bg_b^{f(h_b^b)} \end{aligned}$$

$$
\begin{array}{lcr}
\textsf{Alice} & & \textsf{Bank} \\
a_1, \gamma \in_{\mathcal{R}} \mathbb{Z}_n^* & & \\
\sigma \in_{\mathcal{R}} \mathbb{Z}_v & & \\
 & \xrightarrow{\gamma^v a_1 g^\sigma} & \\
 & & a_2 \in_{\mathcal{R}} \mathbb{Z}_n^* \\
 & \xleftarrow{h^{a_2}} & \\
e \leftarrow f(h^{a_1 a_2}) - \sigma & & \\
 & \xrightarrow{e} & \\
 & & \overline{A} \leftarrow \gamma^v a_1 g^\sigma \cdot a_2 \cdot g^e \\
 & \xleftarrow{a_2, (\overline{A})^{1/v}} & \\
a \leftarrow a_1 a_2 & & \\
S \leftarrow (\overline{A})^{1/v}/\gamma & & \\
S^v \stackrel{?}{=} a g^{f(h^a)} & &
\end{array}
$$

Figure 3: Randomized blind signature scheme without exponential attack

where the numbers g_c, g_a and g_b are publicly known and of large order in the group Z_n^*. The numbers h_c and h_b are elements of order n from $\mathbb{F}_p$ where $p-1$ is a multiple of n. The use of three different g values ensures that the numbers are distinct and do not mingle when multiplied together in a signature.

The coin withdrawal protocol consists of three parallel runs of the randomized blind signature scheme. Two of the runs are the restricted version, and one is the unrestricted version. The E-attack is used here to allow Alice to randomise the k parameter of the secret sharing line herself while the other parameter is fixed by the bank.

There are 2 additions to this simple parallel-run view. One is that there is an extra one-way function that makes a depend on e_c and e_b. This prevents Alice form choosing e_c and e_b as a function of a. Were she able to do this, she could cancel some of the terms and get a signature on just C and B. Although this is not a threat as such against the payment scheme, it is undesirable that Alice has so much freedom.

The second modification is that the bank puts a random power on C in the first signature. Alice is going to end up with two signatures: $(C^k A)^{1/v}$ and $(C^U B)^{1/v}$. Here, U is the identity and k is a random number unknown to the bank. However, to prevent Alice from combining an old coin with the one currently being withdrawn the bank must ensure that k is indeed random. Therefore, the bank puts a random power on the C in the first signature, forcing Alice to choose k at random.

The coin withdrawal protocol is shown in figure 4. We assume that the Bank and Alice have agreed on an identity U to be encoded in the coin. The protocol consists of the following steps:

1. Alice starts by choosing the random numbers c_1, a_1, b_1, σ, τ, ϕ, α, β, and γ. The first three are Alice's contributions to the base numbers. The second triple are the exponential blinding factors, and the third triple are the multiplicative blinding factors. Alice computes $\gamma^v c_1 g_c^\sigma$, $\alpha^v a_1 g_a^\tau$ and $\beta^v b_1 g_b^\phi$, and sends these numbers to the bank.

2. The bank then chooses its three contributions to the base numbers c_2, a_2, b_2. It sends $h_c^{c_2}$, a_2, and $h_b^{b_2}$ to Alice. Sending a_2 directly allows Alice to raise one of the resulting signatures to a power she chooses.

3. Alice chooses a random k_1, and computes the exponents e_c and e_b as $f(h_c^{c_1 c_2}) - \sigma$ and $f(h_b^{b_1 b_2}) - \phi$. She computes a as $(a_1 a_2 f_2(e_c, e_b))^{k_1}$ where $f_2(\cdot)$ is a suitable oneway function. The exponent e_a is computed somewhat differently to get the right exponent after raising the signature to the k_1'th power. After computing e_a as $(1/k_1)f(a) - \tau$, Alice send all three exponents to the bank. Note: The subtractions and multiplication by $1/k_1$ are done modulo v. Any modulo reduction here has to be corrected in the final signature, by multiplying the signature by a suitable powers of g_c, g_a or g_b. These corrections are not shown.

4. The bank now computes the blinded versions of C, A and B. $\overline{C}$ is computed as $\gamma^v c_1 g_c^\sigma \cdot c_2 \cdot g_c^{e_c}$ which is equal to $\gamma^v c g_c^{f(h_c^c)}$ for $c = c_1 c_2$. Note that the bank does not know the value of c. $\overline{A}$ and $\overline{B}$ are computed in a similar way, and the factor $f_2(e_c, e_b)$ is put into $\overline{A}$. The following relations hold between the blinded numbers and their unblinded values:

$$\begin{aligned} \overline{C} &= \gamma^v C \\ \overline{A} &= \alpha^v A^{1/k_1} \\ \overline{B} &= \beta^v B \end{aligned}$$

   The bank then chooses a random k_2, and sends c_2, b_2, k_2, $(\overline{C}^{k_2} \cdot \overline{A})^{1/v}$, and $(\overline{C}^U \cdot \overline{B})^{1/v}$ to Alice.

5. Using c_2 and b_2 Alice can compute c and b as $c_1 c_2$ and $b_1 b_2$ respectively. She now constructs the numbers C, A, and B as $c g_c^{f(h_c^c)}$, $a g_a^{f(a)}$, and $b g_b^{f(h_b^b)}$.

$$
\begin{array}{lcr}
\textsf{Alice} & & \textsf{Bank} \\
c_1, a_1, b_1 \in_{\mathcal{R}} \mathbb{Z}_n^* & & \\
\sigma, \tau, \phi \in_{\mathcal{R}} \mathbb{Z}_v & & \\
\gamma, \alpha, \beta \in_{\mathcal{R}} \mathbb{Z}_n^* & & \\
& \xrightarrow{\gamma^v c_1 g_c^{\sigma},\, \alpha^v a_1 g_a^{\tau},\, \beta^v b_1 g_b^{\phi}} & \\
& & c_2, a_2, b_2 \in_{\mathcal{R}} \mathbb{Z}_n^* \\
& \xleftarrow{h_c^{c_2},\, a_2,\, h_b^{b_2}} & \\
k_1 \in_{\mathcal{R}} \mathbb{Z}_v^* & & \\
e_c \leftarrow f(h_c^{c_1 c_2}) - \sigma & & \\
e_b \leftarrow f(h_b^{b_1 b_2}) - \phi & & \\
a \leftarrow (a_1 a_2 \cdot f_2(e_c, e_b))^{k_1} & & \\
e_a \leftarrow \frac{1}{k_1} f(a) - \tau & & \\
& \xrightarrow{e_c,\, e_a,\, e_b} & \\
& & \overline{C} \leftarrow \gamma^v c_1 g_c^{\sigma} \cdot c_2 \cdot g_c^{e_c} \\
& & \overline{A} \leftarrow \alpha^v a_1 g_a^{\tau} \cdot a_2 \cdot f_2(e_c, e_b) \cdot g_a^{e_a} \\
& & \overline{B} \leftarrow \beta^v b_1 g_b^{\phi} \cdot b_2 \cdot g_b^{e_b} \\
& & k_2 \in_{\mathcal{R}} \mathbb{Z}_v^* \\
& \xleftarrow{c_2,\, b_2,\, k_2,\, (\overline{C}^{k_2} \cdot \overline{A})^{1/v},\, (\overline{C}^{U} \cdot \overline{B})^{1/v}} & \\
c \leftarrow c_1 c_2 & & \\
b \leftarrow b_1 b_2 & & \\
k \leftarrow k_1 k_2 \bmod v & & \\
C \leftarrow c g_c^{f(h_c^c)} & & \\
A \leftarrow a g_a^{f(a)} & & \\
B \leftarrow b g_b^{f(h_b^b)} & & \\
S_a \leftarrow \left((\overline{C}^{k_2} \cdot \overline{A})^{1/v} / \gamma^{k_2} \alpha \right)^{k_1} & & \\
S_b \leftarrow (\overline{C}^{U} \cdot \overline{B})^{1/v} / \gamma^{U} \beta & & \\
S_a^v \stackrel{?}{=} C^k A & & \\
S_b^v \stackrel{?}{=} C^U B & &
\end{array}
$$

Figure 4: Coin withdrawal protocol

Alice computes the first signature S_a as $((\overline{C}^{k_2} \cdot \overline{A})^{1/v} / \gamma^{k_2} \alpha)^{k_1}$, and the second signature S_b as $(\overline{C}^U \cdot \overline{B})^{1/v} / \gamma^U \beta$. The k_1'th power is needed because the base number a was chosen as $(a_1 a_2)^{k_1}$ instead of $a_1 a_2$. All the necessary adjustments in the exponent of g_a were already made by Alice. The total effect of this k_1'th power is to get a v'th root on the number $C^k A$ where $k = k_1 k_2$.

Finally Alice checks that the signatures she received are correct by verifying that $S_a^v = C^k A$ and $S_b^v = C^U B$.

Alice ends up with the following set of numbers: c, a, b, k, S_a, and S_b which are the 3 base numbers, the random parameter for the secret sharing line and the 2 signatures. These 6 numbers plus the identity U are used as input to the payment protocol.

We still need a few additions to this protocol to protect Alice against framing by the Bank. To this end we let U be the concatenation of Alice's identity and a unique coin number. This makes the U's of all the coins distinct. Secondly, in the third transmission Alice includes a digital signature on U and all the data in the first three transmissions. If the bank now claims that Alice spent a coin twice, it must show a transcript of the withdrawal protocol for that coin. This transcript must include the correct data in the last transmission. (This is verifiable by a third party.) The bank also shows a, b and c from the doubly spent coin.

If Alice didn't spend the coin with that identity twice, then the bank can have *no* knowledge regarding a, b, or c. So if the bank tries to frame Alice, the triple (a, b, c) will (with high probability) be different from the actual values used by Alice. If Alice can provide a different triple (a, b, c) plus the corresponding blinding factors that match the transcript, then the bank must be framing Alice. Note that Alice cannot generate a new triple which matches a given transcript.

5 Discussion

We have presented an electronic coin scheme that is an order of magnitude simpler than the previous most efficient scheme. The efficiency of our scheme is difficult to compare; it uses more memory but far less computations than earlier schemes. The payment transaction is very efficient, requiring only 50 modular multiplications to pay any amount.

Work is currently under way to implement this scheme on workstations to provide e-mail money. An extension to n-spendable coins (which can be spent n times but not $n+1$ times) and the incorporation of observers have been developed and will be presented in a future paper.

References

[CdBvH+90] David Chaum, Bert den Boer, Eugène van Heyst, Stig Mjølsnes, and Adri Steenbeek. Efficient off-line electronic checks. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT '89*, Lecture Notes in Computer Science, pages 294–301. Springer-Verlag, 1990.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, Lecture Notes in Computer Science, pages 319–327. Springer-Verlag, 1990.

[Cha92] David Chaum. Randomized blind signature. Personal communications, April 1992.

[EvH92] Jan-Hendrik Evertse and Eugène van Heyst. Which new RSA-signatures can be computed from certain given RSA-signatures? *J. Cryptology*, 5(1):41–52, 1992.

[FY92] Matthew Franklin and Moti Yung. Towards provably secure efficient electronic cash. Technical Report TR CUCS-018-92, Columbia University, April 1992. Personal communications.

[FY93] Matthew Franklin and Moti Yung. Secure and efficient off-line digital money. Personal communications, January 1993.

[Hir93] Rafael Hirschfeld. Making electronic refunds safer. In *Advances in Cryptology—CRYPTO '92*, 1993. To appear.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, Lecture Notes in Computer Science, pages 324–337. Springer-Verlag, 1992.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[vA90] Hans van Antwerpen. Off-line electronic cash. Master's thesis, Eindhoven University of Technology, department of Mathematics and Computer Science, 1990.