Electronic Cash with Blind Deposits: How to Have No Spare Change

Moses Liskov

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
mliskov@theory.lcs.mit.edu

Abstract. Electronic cash schemes in which the bank authenticates many coins at once suffer from the problem that coins that are authenticated together can be linked to one another. Unfortunately, unless a user spends coins in a closely prescribed manner, different batches of coins ("wallets") will be linked together in these schemes. This is illustrated by the problem of what a customer does with the "spare change" – an unusable small amount of money left in a wallet.
We propose a new protocol to be used in e-cash schemes: blind deposits. In a blind deposit, a customer returns a coin to the bank without revealing the coin. We present a secure and efficient e-cash scheme with this added feature based on that of Liskov-Micali [LM01].

1 Introduction

Electronic cash ("e-cash") [Cha82] is an electronic version of real-world cash designed to mimic certain key properties. E-cash is supposed to be *anonymous* and *unlinkable*, so that individual coins cannot be linked either to a user, or to other coins of the same anonymous user. E-cash is also supposed to be *off-line*, so that it can be spent without the user having to wait for the bank to approve each coin before the merchant accepts it.

Because of the off-line property, double-spending is a problem in e-cash – at best, double-spending can be prevented by tamper-resistant devices, and if that fails, be detected after the fact [Bra93]. Schemes therefore typically require that a customer embed his identity in the coins he generates. Thus, minting each individual coin requires a signature for authentication and a zero-knowledge proof to ensure the coins were properly generated, making coin generation a fairly expensive operation.

Unfortunately, we anticipate that in order to be useful, each individual coin must be of a small denomination. Thus, in order to make electronic cash efficient, it is desirable to amortize the cost of the zero-knowledge and signature computation by using them to produce many coins. Prior work in this area has produced many schemes [Bra93,Fer93,EO94,Oka95,CFT98] in which many low-denomination coins are minted together.[1]

[1] Here, we use the term "coin" to refer to any spendable unit of electronic currency, though this doesn't exactly match up with some of the terminology in those papers.

These schemes have a drawback, in that coins authenticated together can be linked together if the merchants collude. Formally, we can at best require that coins and wallets (batches of coins authenticated together) be anonymous – that they cannot be linked to the user's identity – and that coins from different wallets are unlinkable [LM01].

Because of the less than ideal anonymity properties of these schemes, circumstances (which we discuss in section 2) may arise in which a customer wishes to get rid of the e-cash in a wallet without spending it. Thus, we consider a new additional tool in an electronic cash scheme: blind deposits. In a blind deposit, a customer interacts directly with the bank and "spends" (deposits) a coin without revealing the coin or the wallet it came from.

STRUCTURE OF THE PAPER. In section 2 we discuss the problems a customer may run into in a less-than-anonymous e-cash scheme, and how blind deposits solve them. We discuss relevant prior work in section 3. In section 4 we discuss the notion of security of an e-cash scheme with blind deposits. We give a secure wallet-based e-cash scheme with blind deposits based on the scheme of Liskov and Micali [LM01] in section 4.1, and sketch a proof of its security.

2 Motivation

When we consider an e-cash scheme, we should imagine the life of a wallet in the hands of a customer as follows: The customer withdraws the wallet, and then the customer uses the wallet in a series of transactions. It is a bad design idea to try to control (or make assumptions about) the transactions a customer makes in order to ensure the security of the e-cash scheme; if we do this, customers will be unhappy, since either they will have to obey certain restrictions placed on them by the e-cash scheme, or they will not be guaranteed the security they want. Thus, any proposal which relies on restricting the transactions a customer chooses to engage in is unacceptable.

2.1 The spare change problem

Normally, the customer has some money in a wallet and spends it on whatever transactions he wants. But this is not so simple when the amount of money left in the wallet is small. Suppose the customer simply does the natural thing – finds something to buy and pays for it by using all remaining coins out of the partially depleted wallet, and pays for the rest with coins from a new wallet. This is insecure! Now the merchant links the two wallets together. If this is always a customer's solution, it is fairly unlikely that any two consecutive wallets will *not*

They use the terms "multi-spendable coin" or "divisible coin" where we would use the term "wallet."

be linkable, so a conspiracy of merchants could actually link quite a number of transactions together.[2] What other options does the customer have?

- **Discard the coins.** The customer can simply never spend the leftover coins. This is unacceptable: the customer pays a monetary penalty for security.
- **Spend only money remaining in the wallet.** The customer makes a purchase which costs at most the amount of money remaining. First of all, if the amount isn't exact, there will still be leftover money, just less of it. But more importantly, this restricts what the customer can do with his money. This is unacceptable: the customer pays a penalty of his freedom for security.
- **Give the coins back to the bank.** The customer spends the remaining coins with the bank, and gets the money added to their account. Unfortunately, this is insecure: the bank sees what a merchant sees in a spending protocol, but also sees the customer's identity (so the money can be deposited in the right account), so the customer's anonymity is broken.
- **Use the remaining money with a trusted merchant.** If the customer trusts the merchant, it is acceptable to spend all the remaining change from one wallet and finish with money from a new wallet. However, this restricts the customer to dealing with a trusted merchant in the transaction, which is unacceptable. Furthermore, the customer may not be willing to trust any merchants.

So what is a customer to do with spare change? If the customer wishes to use the change at all, as a matter of practicality, the customer must trust either the bank or the merchants, in which case we don't have security. Blind deposits provide a safe way of returning a customer's spare change for credit, and so they solve this problem.

ATTAINING UNLINKABLE WALLETS WITH NO TRUSTED MERCHANTS. There are two ways in which blind deposits can be used to solve the spare change problem. Suppose that a customer has a (partially) depleted wallet and an unused wallet. Now, the customer can simply blind deposit the coins in the depleted wallet and obtain a new wallet, and pay for the transaction with coins from the unused wallet. Then, after the transaction the customer will still have one unused wallet.

There is no reason to do the blind deposit first, unless the customer doesn't have enough money in his account to withdraw a new wallet otherwise. Thus, a more natural and desirable solution would be for the customer to spend money out of the unused wallet, and then later cash in the depleted wallet and obtain a new wallet. This takes the transaction off-line.

However, the frequency of (and possibly, the number of) blind deposits can be reduced if a user is willing to have spare change sitting around for a while. If the customer has many wallets, and doesn't perform a blind deposit until several of them are reduced to unusable amounts, then the customer might perform fewer

[2] What's more, since the merchants control the price of their merchandise, they could make the prices unlikely to come out even in the end so that this happens even more often.

blind deposits if he engaged in some transactions which had low enough costs to be paid entirely from one depleted wallet. In the worst case, this will not happen, and the customer will have to perform the same number of blind deposits.[3]

ATTAINING UNLINKABLE WALLETS WITH SOME TRUSTED MERCHANTS. Actually, we can attain unlinkable wallets without blind deposits if we are willing to trust some merchants. The customer simply lets many wallets with spare change accumulate until the customer spends money with a trusted merchant, at which point the customer uses as many spare change wallets as possible to pay for the transaction.

This is a reasonable solution, but it has a minor problem: the customer is rewarded financially for spending money with the trusted merchants more often, since every dollar the customer has in spare change is a dollar he can't use. Perhaps the customer would normally transact with trusted merchants very infrequently, but would be compelled to do so more frequently in order to recover the spare change regularly. In order to alleviate this pressure, the customer could simply use blind deposits. Now, the customer is free to engage in whatever transactions he wishes to engage in, and his spare change won't become a problem.

ALLOWING SAFE EXITS. Suppose a customer has been involved in an e-cash scheme for a while and decides to leave. This would be fine if the customer had no e-cash left, but if he did, what could he do with it? Effectively, the customer would be forced to either trust the bank and give the cash back, breaking anonymity, or find a transaction to engage in. The former is clearly unacceptable, and the latter is also unacceptable since the customer's freedom is impinged. Blind deposits allow the customer a way to cash in his remaining coins without losing anonymity, and thus exit the system safely.

3 Prior work

The blind deposit notion is new, so far as we know. However, the main problem (spare change) it is intended to solve can be solved in two other ways. First of all, in non-wallet based cash schemes, such as Chaum's original scheme [Cha82], coins were independently minted and so were not linkable. In such schemes, we can achieve safe deposits simply by spending a coin with the bank – since there are no other coins linkable with the deposited coin, the bank learns useless information. The spare change problem is inherent to *wallet-based* e-cash schemes only. The other way to solve the spare change problem is to use an unlinkable wallet-based e-cash scheme. The scheme of Nakanishi, Haruna, and Sugiyama [NHS99] is the only known example However, their scheme requires the existence

[3] This tradeoff benefits the bank at the expense of the customer. The bank wishes to minimize blind redeposits, but the customer doesn't care – he just wants to deposit his coins as soon as they become spare change. This impasse could possibly be resolved if the bank and customer came to some mutual understanding.

of a trustee.[4] We improve on this in that we give a solution to the spare change problem without relying on trustees, and furthermore provide the blind deposit mechanism, which has other uses.

The idea of blind deposits is not entirely new: Chaum [Cha89] and Brands [Bra] both proposed a similar idea for electronic checks.

4 Security of wallet-based e-cash with blind deposits

It should be clear now that blind deposits would be a useful tool to have in an e-cash scheme.

We expand on the definitional framework of Liskov and Micali [LM01]. Generally speaking, a wallet-based e-cash scheme with blind deposits involves three players: the bank $\mathcal{B}$, the merchant $\mathcal{M}$ and the customer $\mathcal{C}$.[5]

A wallet-based e-cash scheme with blind deposits consists of five protocols. In E (account establishment), the customer sets up whatever keys are to be used throughout. In W (wallet withdrawal), the bank issues the customer a batch of coins. In S (spending), the customer interacts with a merchant to spend a coin. In MD (merchant depositing), the merchant deposits a coin with the bank. Finally, in BD (blind depositing), the customer performs a blind deposit with the bank.

The basic properties we require are as follows:

Honest operation. Simply put, everything works as it should when all parties are honest. After running E, $\mathcal{C}$ and $\mathcal{B}$ can engage in the withdrawal protocol W, agreeing on a size for the wallet, 2^d. After W, the customer obtains a wallet. When i is an index of a coin, S provides a mechanism for spending the i^{th} coin of the wallet, where the merchant $\mathcal{M}$ involved will accept. The transcript of this spending protocol can be used by the merchant to deposit with the bank in protocol MD, and the bank will accept.

Finally, $\mathcal{C}$ may use BD to deposit a coin with the bank; if the coin was generated honestly, $\mathcal{B}$ will accept.

No forgery. A malicious customer cannot use more distinct coins than he withdrew from the bank, where "use" encompasses both spending and blind depositing. To be more specific, no malicious customer can interact with $\mathcal{B}$ in W where the size agreed to was 2^d and produce a valid wallet with size more than 2^d. Furthermore, no malicious customer can successfully spend or blind deposit a coin i successfully if $i < 1$ or $i > 2^d$.

Double-use. Here, there is a difference between the security we require and that required for other e-cash schemes. When blind deposits are allowed, we need to

[4] The trustee's purpose is to provide revocable anonymity, but they reveal the identity of double-spenders through the trustee, so the trustee cannot be removed

[5] In this paper we do not consider schemes with trustees.

make sure that not only is double-spending prevented, but that no customer can use a coin twice in *any* way.

There are three new kinds of attacks similar to double-spending:

- Blind depositing a coin which has previously been spent
- Blind depositing a coin which has been previously blind deposited
- Spending a coin which has previously been blind deposited

For ease of terminology, we will refer to the double-spending and related attacks as *double-use.*

Thus, we require that double-use of a coin reveals the customer's identity. Alternatively, we require that a customer cannot blind deposit a coin he has previously used, and that if a customer spends a coin he has previously used, the customer's identity will be revealed. Since blind depositing is effectively an online protocol it is reasonable to imagine that the bank could discover double-use before finishing, and thus reject.

No framing. Even if the bank and all merchants conspire, they cannot produce a pair of transcripts that show $\mathcal{C}$ spending a coin that $\mathcal{C}$ previously used.

Wallet anonymity. Even interacting in an adaptive manner, and even controlling the customer's honest actions, even the bank and merchants conspiring together cannot gain an advantage in linking a user's identity, established in E, with that user's wallets, compared to random guessing. This should remain true if the customer not only spends coins but also uses them in blind deposits. We must assume that in this experiment the adversary is unable to make any users redeposit different numbers of coins from one another, since this would surely give a way for the bank to distinguish between them. For example, the bank may observe one customer spending all but 8 of his coins. Then, if a customer redeposits 10 coins, the bank will know that this is not the same customer. Furthermore, as before, we require that all the users withdraw wallets of the same size.

To express this more formally, we set up a game in which the bank gets to, in a limited fashion, control the honest actions of m customers. Specifically, we give each customer a "real name" $\mathcal{C}_i : i \in [1, m]$, and a "pseudonym" $\mathcal{C}'_j = \mathcal{C}_{\pi(i)}$, where π is a permutation on m elements selected at random from S_m. The rules the customers will follow are:

1. No customer will ever spend or blind deposit any coin more than once.
2. All the customers will spend together or blind deposit together. That is, if a customer spends the first coin, all other customers will refuse to blind deposit any coins, and will refuse to spend a second coin, until all customers have spent their first coin.

If a customer breaks rule 1, the adversary could simply use the identity of the customer revealed through the double-use to win the game. If a customer breaks rule 2 even once, the adversary could win the game as follows: ask all the customers to spend all their coins but one, then ask a particular customer $\mathcal{C}$ to spend the last coin in his wallet. Finally, ask a customer $\mathcal{C}'$ to blind deposit a coin. If $\mathcal{C} = \mathcal{C}'$, the customer would have to refuse, and so the adversary would learn that $\mathcal{C} = \mathcal{C}'$, since any other customer would not refuse.

Given those rules, the adversary can act arbitrarily adaptively. So, we ask that for arbitrary $c, n > 0$ and $m > 1$, and for all adversaries A, there is a k_0 such that for all $k > k_0$, the probability that, using k as the security parameter everywhere,

1. Customers $\mathcal{C}_1, \ldots, \mathcal{C}_m$ run account establishment protocols with A, and do not reject.[6]
2. Customers $\mathcal{C}_1, \ldots, \mathcal{C}_m$ withdraw wallets $w_1, \ldots, w_m$ from A, and do not reject.
3. Permutation π is chosen randomly from S_n.
4. For $a = 1$ to n do:
    - 4a. A chooses "spend" or "blind deposit."
    - 4b. If A chose "spend," then for $b = 1$ to m do:
        - 4b*i*. A picks a customer j.
        - 4b*ii*. Customer $\mathcal{C}'_j$ spends a random unspent coin in his wallet with A and does not reject.
    - 4c. If A chose "blind deposit," then for $b = 1$ to m do:
        - 4c*i*. A picks a customer i.
        - 4c*ii*. Customer $\mathcal{C}_i$ blind deposits a random unspent coin in his wallet with A and does not reject.
5. A outputs a pair (i, j), such that $j = \pi(i)$.

is less than $1/m + 1/k^c$.

Wallet unlinkability. Even interacting in an adaptive manner, and even controlling the customer's honest actions, even the bank and merchants conspiring together cannot gain an advantage in linking separate wallets withdrawn by the same user, compared to random guessing. We assume the customer never spends coins from any wallet with coins from any other wallet here, though as we discussed in section 2, this may not be a realistic assumption, and the customer may *have* to use BD to avoid doing this.

4.1 The scheme

We now present a secure and efficient wallet e-cash scheme with blind deposits, based on the e-cash scheme of Liskov and Micali [LM01].

To embed the customer's identity we use a secure symmetric scheme (E', D'). (Our scheme actually construct (E', D') from an underlying secure symmetric encryption scheme (E, D). The precise description of how (E', D') are derived from (E, D) is given by Liskov and Micali [LM01].

Our scheme uses Schnorr's signature scheme [Sch89] in the spending protocol, with public parameters p, q, g, and H. Recall that a signature under public key g^x of message t consists of a pair $(g^r, r + ex \bmod p)$ where $e = H(t, g^r)$.

Our scheme also uses a second, unspecified, signature scheme. To avoid confusion between the two schemes, the Schnorr keys and signatures are explicitly spelled out, while keys and signatures under the second scheme are denoted abstractly. These keys are presumed to be permanent. For simplicity, we assume that in this scheme the signature of a message M always includes M (in the clear).

This signature scheme is chosen so as to allow blind signatures [Cha82]. To highlight that any such a secure scheme can be used, we use the following abstract notation. There are two efficient randomized algorithms, F and G. The bank's blind signature of M is obtained by (1) computing $F(M) = (M', \rho)$; (2) asking the bank to sign M', and (3) computing $\mathrm{SIG}_{SK_B}(M)$ by running $G(\mathrm{SIG}_{SK_B}(M'), \rho)$.

The withdrawal protocol W**.** For simplicity, we assume that each wallet size is 2^d for some d.

Here is how a customer withdraws a wallet of size 2^d.

1. The customer generates a random public/private key pair (g^x, x) in the Schnorr scheme.
2. The customer uses his permanent secret key to sign the sentence "$\mathcal{C}$ has double spent"; that is, he computes $s = \mathrm{SIG}_{SK_\mathcal{C}}$("$\mathcal{C}$ has double spent"). Then, he uses the symmetric encryption scheme to encrypt s using as encryption key the just generated Schnorr secret signing key; that is, he computes $z = E'_x(s)$.
3. The customer then generates 2^d ephemeral key pairs $(g^{r_1}, r_1) \ldots (g^{r_{2^d}}, r_{2^d})$.
4. The customer generates 2^d "salts" in $\mathbb{Z}_p^*$, $s_1, \ldots, s_{2^d}$.
5. The customer creates a Merkle hash tree [Mer80], T, of depth $d+1$ which stores the public parts of the ephemeral keys $(g^{r_1}, \ldots, g^{r_{2^d}})$ and the salts in its leaves (where g^{r_i} is stored in leaf $2i$). The customer stores s_i in leaf $2i+1$. Denote by R the root value of T.
6. The customer generates a random value ρ, computes $M' = F(M, \rho)$, and sends M' to the bank, where $M = (R, d, z, g^x)$.
7. The customer proves interactively with the bank that M' is properly formed. Specifically, the customer proves that $M' = F(M, \rho)$ where M is a quadruple of values (a_1, a_2, a_3, a_4) such that (1) $a_2 = d$ and (2) a_3 is the encryption under E', using the discrete log of a_4 as a secret key, of $\mathcal{C}$'s permanent signature of the value "$\mathcal{C}$ has double spent". These proofs are accomplished via general zero-knowledge proof methods, such as those in [GMW91].
8. The bank provides a signature $\mathrm{SIG}_{SK_B}(M')$ and sends it to the customer.
9. The customer unblinds the signature, that is, he computes $w = G(\mathrm{SIG}_{SK_B}(M'), \rho)$. The public part of the wallet consists of (w, R, d, z, g^x), while the secret part of the wallet consists of the secret parts of the ephemeral keys $(r_1, \ldots, r_{2^d})$ along with the secret key x and the salts $s_1, \ldots, s_{2^d}$.

The spending protocol consists of two steps - payment ($S_\mathcal{C}$), computed by the customer, and payment verification ($S_\mathcal{M}$), computed by the merchant.

The payment algorithm $S_\mathcal{C}$**.** On input $((w, R, d, z, g^x), (r_1, \ldots, r_{2^d}), x, i, t)$, the customer does the following:

1. The customer generates the ephemeral public key g^{r_i} from r_i and the public parameters. Then he computes $(g^{r_i} \bmod p, r_i + ex \bmod q)$, the Schnorr signature on the challenge t using secret key x and ephemeral key pair (g^{r_i}, r_i).
2. The customer generates the authentication path, *path*, in Merkle tree T for leaf $2i$.
3. The customer outputs $((g^{r_i} \bmod p, r_i + ex \bmod q, path), (w, R, d, z, g^x), i, t)$

The payment verification algorithm $S_{\mathcal{M}}$. On input $((a, b, path),$ $(w, R, d, z, g^x),\ i, t)$, the merchant does the following:

1. Check that (a, b) is a valid Schnorr signature relative to public key g^x on message t. If not, halt and output 0.
2. Check that w is a signature on (R, d, z, g^x) relative to the bank's public key $PK_{\mathcal{B}}$. If not, halt and output 0.
3. Check that *path* is of size $d+1$ and correctly authenticates that a is stored in the $2i^{th}$ leaf of a Merkle tree with root value R. If not, output 0.
4. (Else) output 1.

The merchant depositing algorithm MD.

The merchant deposit algorithm simply consists of the merchant passing the transcript $T = ((a, b, path), (w, R, d, z, g^x), i, t)$ to the bank, who verifies that $S_{\mathcal{M}}$ accepts T.

The blind deposit algorithm BD. If the customer wishes to blind deposit coin i with the bank, the customer and the bank execute this protocol:

1. The bank generates a transaction (challenge) t as if it were a merchant with public key $PK_{\mathcal{B}}$, and sends t to the customer.
2. The customer sends $a = g^{r_i}$ and $b = r_i + ex$ to B.
3. The bank checks that a does not match any previously spent or redeposited coin.
4. The customer proves to the bank in zero knowledge that (1) $\mathcal{C}$ knows a string w which specifies a wallet with public key K and hash tree root R and which has a signature verifiable by the bank's public key $PK_{\mathcal{B}}$, that (2) (a, b) form a valid signature on t with respect to public key K, and that (3) $\mathcal{C}$ knows a hash path that verifies that a occurs in the hash tree with root R.
5. The bank adds (a, b, T, t) to its database, where T is the transcript of the proof from step 4.

The identity revealing algorithm for double-spending. The algorithm REV, on input

$$y_1 = ((a_1, b_1, path_1), (w_1, R_1, d_1, z_1, g^{x_1}), i_1, t_1)$$

and

$$y_2 = ((a_2, b_2, path_2), (w_2, R_2, d_2, z_2, g^{x_2}), i_2, t_2)$$

runs as follows:

1. Check that (1) $S_{\mathcal{M}}(y_1) = 1$, (2) $S_{\mathcal{M}}(y_2) = 1$, (3) $w_1 = w_2$, (4) $i_1 = i_2$, and (5) $t_1 \neq t_2$. If any of these checks fail, halt and output ε.
2. Let $e_1 = H(t_1, a_1)$ and $e_2 = H(t_2, a_2)$, and compute $(e_1 - e_2)^{-1} \bmod q$.
3. Compute $x = (b_1 - b_2)(e_1 - e_2)^{-1} \bmod q$.
4. Compute $s = D'_x(z_1)$. Output s, which should be customer $\mathcal{C}$'s signature of "$\mathcal{C}$ has double spent."

The identity revealing algorithm for double-use. The algorithm REV', on input

$$y_1 = ((a_1, b_1, path_1), (w_1, R_1, d_1, z_1, g^{x_1}), i_1, t_1)$$

and $y_2 = ((a_2, b_2, T, t_2)$, runs as follows:

1. Check that (1) $S_{\mathcal{M}}(y_1) = 1$ and that (2) TR is a valid proof transcript for (a_2, b_2). If either of these checks fail, halt and output ε.
2. Let $e_1 = H(t_1, a_1)$ and $e_2 = H(t_2, a_2)$, and compute $(e_1 - e_2)^{-1} \bmod q$.
3. Compute $x = (b_1 - b_2)(e_1 - e_2)^{-1} \bmod q$.
4. Compute $(\mathcal{C}, s) = D'_x(z_1)$. If s is not a valid signature on "$\mathcal{C}$ has double-spent," output ε. Otherwise, output $\mathcal{C}, s$.

Apart from the addition of BD, the only alteration to the Liskov-Micali scheme is this: when spending a coin in the prior scheme, the customer reveals the hash path that authenticates that coin as part of the tree with root R. This means, for example, that if we spend coin g^{r_1} we reveal g^{r_2} as part of the hash path. We offer a simple change in the tree structure which will solve this problem.

Previously, the hash tree was constructed with the leaves $g^{r_1}, \ldots, g^{r_{2^d}}$. Now, we ask the user to come up with 2^d random strings, the "salts" $s_1, \ldots, s_{2^d}$. Now, in place of the leaf g^{r_i} we put the internal node $H(s_i, g^{r_i})$, which has leaves g^{r_i} and s_i. Now the hash path includes the salt as the first sibling, but it never shows any other of the ephemeral keys.

4.2 Security sketch

To prove that this scheme is secure, we must prove a number of things. First of all, to satisfy basic properties of e-cash, we must prove that it is unforgeable, wallet anonymous, protects against double-use, and prevents framing, and we must also prove that blind redeposits are in fact blind. The proof that the scheme is unforgeable against spending is in the prior work [LM01]. To prove that this scheme is unforgeable against blind redeposits, we need only note that to successfully blindly deposit a coin, the adversary must either fake the zero-knowledge proof, or must be able to forge coins that can be used through spending, reducing to the previous result.

Wallet anonymity of the scheme was proved in the prior work, but with blind deposits allowed, this definition takes on new complexity. The proof, however, reduces to the previous one when we note that the blind deposit protocol can be simulated.

The mechanism to prevent double-spending is also previously described. As for the other double-use attacks, step 3 of the blind deposit protocol prevents any user from redepositing a previously used coin. Thus, the only kind of double-use attack not accounted for is the kind in which a coin is redeposited and then spent.

If this happens, and the user did not fake the zero knowledge proof in the blind deposit protocol, then the bank will know g^{r_i} and two different signatures made with respect to g^{r_i} and g^x, which can be used to solve for x. In this case, the bank already knows which customer deposited that coin, so it simply checks that x and the wallet w give back the right customer's identity.

To prevent framing, we use the same method that Liskov and Micali did – require that proof of double-spending constitute the inputs to the identity revealing protocol involved. This is now nontrivial to show, since the malicious bank has the option to try to frame the user through the identity revealing algorithm for double-use, REV'. If an adversary were capable of framing an honest customer through REV', then we could either forge the signature or break the encryption scheme, which should be impossible.

4.3 Efficiency

The efficiency of this scheme is a bit less than we would like. First of all, it requires general zero-knowledge proofs both in wallet withdrawal and in blind deposits. Secondly, there is no clear way to improve the efficiency of the blind deposit scheme when depositing many coins. Thus, blind deposits are about as expensive as wallet withdrawals. On the other hand, the security of Liskov-Micali, and of our adaptation of it, is provable under the discrete logarithm assumption in the random oracle model. However, it would clearly be desirable to adapt a more efficient scheme.

4.4 Adapting other schemes

Adding a blind deposit protocol to other existing e-cash schemes proves to be difficult, though possible. To exemplify this, we show how to adapt the scheme of Chan, Frankel, and Tsiounis [CFT98]. In this scheme, the customer has a Williams integer N and provides an encryption of his identity under RSA [RSA78], using N as the modulus. Each wallet consists of a pair (A, B) and a bank signature by that pair on the bank. In order to spend different coins from the same wallet, a customer produces a value Γ which depends on N, on (A, B), and which depends on the coin being spent. (In this scheme, as in that of Okamoto [Oka95], different denominations of coins can be spent, corresponding in an abstract way to internal nodes of a binary tree of a fixed maximum depth.) Because of the way Γ is generated, if a customer spends too many coins from the same wallet, the bank, using all the Γ values learned, could factor N and thus reveal the customer's identity.

In order to provide blind deposits, the customer would provide Γ, and the coin being spent, and instead of providing (A, B) and N, proves in zero knowledge that the customer knows (A, B) and N which check out correctly and to which the customer has a signature. This is not really practical, however, because if Γ is to be used to reveal the identity of a user, the bank wouldn't know which other values to use with that Γ and so would have to basically try all values it knows. This is still technically polynomial time but it is excessive computation.

The advantage of adapting a scheme like Liskov-Micali is that each coin consists of a commitment to be used in a zero-knowledge based one-time signature scheme. Now, values can be sorted in a database by the commitment; two signatures will not produce a key unless they were both performed relative to the same signing key and the same commitment.

5 Conclusion

Blind deposits prove to be a useful function in an e-cash scheme. Unfortunately, their expense makes their use somewhat limited. Effectively, each blind deposit of a single coin requires as much computation as withdrawing an entire wallet of coins. Nonetheless, the scheme presented here is useful.

First of all, with blind deposits, we can attain unlinkable wallets without trusted parties. To do so requires one expensive operation for each coin that must be returned to the bank. However, with a non-wallet based scheme, in which each coin is authenticated individually, an expensive operation would be required for *every* coin, so even though blind deposits are expensive, this represents an improvement.

More importantly, it *allows* customers to use the wallet-based e-cash scheme securely without restricting their choice of transactions, and without trusting any parties. A customer could realize the full promised security of a wallet-based e-cash scheme by trusting some merchants, trusting the bank, or searching around for transactions which give value for small change. However, with blind deposits available, customers need not worry about using the scheme securely – the scheme *is* secure, and they just use it as they please.

We close with a few open problems. First of all, it would be an excellent improvement to find a more efficient scheme for realizing wallet-based e-cash with blind deposits. The most desirable efficiency improvement would be to find a way to blind deposit multiple coins at once without an $O(n)$ increase in computation. Another worthwhile improvement would be to realize a scheme without inefficient general zero-knowledge techniques which is more practical than the adaptation of Chan, Frankel, and Tsiounis given here.

Acknowledgements

The author would like to thank, in no particular order, Ron Rivest, April Rasala, Anna Lysyanskaya, Nathan Liskov, and David Liben-Nowell for valuable discussions, and would also further like to thank Nathan Liskov and the anonymous referees from the Financial Cryptography '02 program committee for reading over a draft of this paper and making valuable comments.

References

[Bra] S. Brands. An efficient off-line electronic cash system based on the representation problem.

[Bra93] S. Brands. Untraceable off-line cash in wallet with observers. In Stinson [Sti93].

[CFT98] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come – easy go divisible cash. In Kaisa Nyberg, editor, *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*. Springer-Verlag, May 31–June 4 1998.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of Crypto 82*. Plenum Press, New York and London, 1983, 23–25 August 1982.

[Cha89] David Chaum. Online cash checks. In Quisquater and Vandewalle [QV89].

[EO94] T. Eng and T. Okamoto. Single-term divisible coins. In Alfredo De Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995, 9–12 May 1994.

[Fer93] Niels Ferguson. Extensions of single-term coins. In Stinson [Sti93], pages 292–301.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[LM01] Moses Liskov and Silvio Micali. Amortized e-cash. In *Proceedings of Financial Cryptography '01*. Springer-Verlag, February 2001.

[NHS99] T. Nakanishi, N. Haruna, and Y. Sugiyama. Unlinkable electronic coupon protocol with anonymity control. In *Proc. of Second International Information Security Workshop ISW'99*, pages 37–46, 1999.

[Oka95] Tatsuaki Okamoto. An efficient divisible electronic cash scheme. In Don Coppersmith, editor, *Advances in Cryptology—CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*. Springer-Verlag, 27–31 August 1995.

[OO91] Kazuo Ohta and Tatsuaki Okamoto. Universal electronic cash. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992, 11–15 August 1991.

[QV89] J.-J. Quisquater and J. Vandewalle, editors. *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990, 10–13 April 1989.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Sch89] C. P. Schnorr. Efficient identification and signatures for smart cards. In Quisquater and Vandewalle [QV89], pages 688–689.

[Sti93] Douglas R. Stinson, editor. *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*. Springer-Verlag, 22–26 August 1993.

[Tsi97] Yiannis S. Tsiounis. Efficient electronic cash: New notions and techniques. In *PhD Thesis, Northeastern University*, 1997.