Untraceable Off-line Cash in Wallet with Observers

(Extended abstract)

Stefan Brands

CWI, PO Box 4079 Amsterdam, The Netherlands. E-mail: brands@cwi.nl

Abstract. Incorporating the property of untraceability of payments into off-line electronic cash systems has turned out to be no easy matter. Two key concepts have been proposed in order to attain the same level of security against double-spending as can be trivially attained in systems with full traceability of payments.
The first of these, one-show blind signatures, ensures traceability of double-spenders after the fact. The realizations of this concept that have been proposed unfortunately require either a great sacrifice in efficiency or seem to have questionable security, if not both.
The second concept, wallets with observers, guarantees prior restraint of double-spending, while still offering traceability of double-spenders after the fact in case tamper-resistance is compromised. No realization of this concept has yet been proposed in literature, which is a serious problem. It seems that the known cash systems cannot be extended to this important setting without significantly worsening the problems related to efficiency and security.
We introduce a new primitive that we call restrictive blind signatures. In conjunction with the so-called representation problem in groups of prime order this gives rise to highly efficient off-line cash systems that can be extended at virtually no extra cost to wallets with observers under the most stringent of privacy requirements. The workload for the observer is so small that it can be performed by a tamper-resistant smart card capable of performing the Schnorr identification scheme.
We also introduce new extensions in functionality (unconditional protection against framing, anonymous accounts, multi-spendable coins) and improve some known constructions (computional protection against framing, electronic checks).
The security of our cash system and all its extensions can be derived directly from the security of the well-known Schnorr identification and signature schemes, and the security of our new primitive.

1 Introduction

It is clear that the level of efficiency and security attainable in an off-line electronic cash system with fully traceable payments always outperforms that attainable in a system with the additional property of privacy of payments. This

D.R. Stinson (Ed.): Advances in Cryptology - CRYPTO '93, LNCS 773, pp. 302-318, 1994.

is caused by the urgent need to protect against account-holders who double-spend their electronic cash, since hardly anything is easier to copy than digital information.

In literature, various realizations have been proposed for untraceable off-line electronic cash. Untraceability is an important asset, but one should not ignore the fact that it is hard to realize at little cost. For this reason, we present in Section 2 an analysis of the cost it takes in terms of efficiency and security to incorporate untraceability of payments. We argue that known realizations of untraceable off-line cash systems offering only traceability of double-spenders after the fact already require a large sacrifice in either efficiency or (provability of) security, if not both. More seriously, no realizations of untraceable off-line cash systems have been proposed yet that can offer prior restraint of double-spending, whereas this property can be trivially attained in fully traceable off-line systems. Thirdly, various other useful extensions in functionality seem hard to achieve in the known systems.

To overcome these drawbacks, we propose in Section 4 the primitive of restrictive blind signatures, and use it in combination with the representation problem in groups of prime order (described in Section 3) to create untraceable off-line electronic cash systems that can offer not only traceability of double-spenders after the fact (Section 5), but more importantly also prior restraint of double-spending under the most stringent of privacy requirements (Section 6). These systems are almost as efficient as fully traceable off-line systems.

In addition, three new extensions can be realized: unconditional protection against framing, anonymous accounts, and multispendable coins. We refer the interested reader to [2, 3] for this. The new approach also supports a better construction for two known extensions, computational protection against framing and electronic checks. The extension to computational protection against framing is incorporated in Sections 5 and 6, and we refer to [2] for a description of the extension to checks.

All the statements we make in this extended abstract have been fully proven. We refer to [2, 3] for these proofs.

2 The cost of incorporating untraceability

2.1 Privacy-compromising systems

An off-line electronic cash system with full traceability of payments can be simply realized using only the basic cryptographic concept of a digital signature. Each coin is represented by a unique piece of digital information with a corresponding digital signature of the bank. If an account-holder ever double-spends then he will be identified by the bank *after* the corresponding payment transcripts have been deposited, if only the bank conscientiously keeps track for each coin to which account-holder it issued that coin. Since off-line cash systems are a medium for low-value payments only (high-value payments are made on-line), this traceability after the fact by itself will discourage many account-holders from double-spending.

If the bank in addition has the payment devices of its account-holders manufactured such that they are tamper-resistant, then a level of prior restraint of double-spending is attained that can only be withstood by an organization with the capabilities of a national laboratory. By maintaining the database concerning issued coins, the bank can still trace a double-spender after the fact in case he unexpectedly breaks the tamper-resistance of the payment device.

Such a system can be realized very efficiently. In each of the three protocols for withdrawal, payment, and deposit of a coin, only one signed number has to be transmitted (in practice other information will be sent along as well, such as signatures that serve as receipts), and a computational effort for each type of participant to verify the validity of the signature of the bank is required. The bank has to compute a digital signature for each coin it issues, and maintain a database with information about the coins issued to the account-holders. This database has to be searched on a regular basis to find out if double-spending has occurred.

The level of security of the system is also very satisfactory. In principle, the bank can use a digital signature proposed by [1], which is provably secure against adaptively chosen message attacks (assuming the existence of one-way permutations). However, since these signatures grow in size, and require quite some computational effort, they are inefficient for practical use in systems such as cash systems, where enormous amounts of signatures are routinely produced and verified. In practice one hence must inevitably sacrifice some provability of security and use e.g. signatures of the Fiat/Shamir type ([13]), such as Schorr signatures ([15]).

Although the system sketched thus far is highly satisfactory from both the efficiency and security points of view, it does not protect the interests of the account-holders. As we discussed, by the very nature of the system the bank has to maintain databases to keep track of the information issued in executions of the withdrawal protocol and the deposited payment transcripts. Since a payment transcript encompasses the withdrawn coin, per definition the entire payment history of all account-holders is stored in computer files by the bank. Hence, not only is such a system not privacy-protecting, it in fact is the extreme opposite. This can have considerable social and political impact (see e.g. [5, 6, 16]). Henceforth, we will refer to such a system as a privacy-compromising system.

2.2 Privacy-protecting cash systems

Two ingenious key concepts have been developed to enable the incorporation of full untraceability of payments while maintaining the level of security against double-spending of the privacy-compromising system.

Concept I. The first key concept is *one-show blind signatures*, introduced in [8]. One-show blind signatures enable traceability of a payment if and only if the account-holder double-spent the coin involved in that payment. That is, traceability after the fact can be accomplished only for double-spenders.

Realizing this concept has turned out to be no easy matter. For traceability, the identity of the account-holder must be encoded into the withdrawn information, whereas this information is not known to the bank by virtue of the blind signature property needed to achieve untraceability of payments. If we forget about the even less efficient theoretical constructions proposed for this (although they seem to guarantee the same level of security as can be achieved in privacy-compromising systems), then only cut-and-choose withdrawal protocols seem to remain. These still cause an enormous overhead in computational and communication complexity that we believe is unacceptable.

Our new primitive, *restrictive* blind signature schemes, in combination with the representation problem in groups of prime order, allows us to construct a three-move withdrawal protocol (i.e. no cut-and-choose) in which the computational effort required by the bank is almost equivalent to that required to compute Schnorr signatures. In the payment protocol, only two (!) modular multiplications are required of the account-holder in order to pay. The database that must be maintained by the bank is almost of the same size as that in the privacy-compromising system.

We refer to [3] for an overview of the cryptographic literature on untraceable off-line electronic cash systems. We confine ourselves here to the remark that in concurrent work ([11]), a system offering traceability after the fact is proposed that also does not use a cut-and-choose withdrawal protocol. Unfortunately, its security seems highly questionable. This is caused by the use of many unspecified one-way hash functions, nested within one another up to four levels deep, and a strange construction to create an element with an order equal to the order of the multiplicative group modulo a composite.

As we show, there is no need at all to resort to such "ad hoc" constructions. In fact, our approach allows for greater efficiency, security, and extendibility in functionality.

Concept II. The privacy-compromising system offered prior restraint of double-spending. Using the second key concept (see [7]), *wallets with observers*, this can also be achieved in privacy-protecting systems. In this setting, a tamper-resistant device that takes care of prior restraint of double-spending, called an *observer*, is embedded into the payment device of the account-holder in such a way that a payment can only be successfully executed if the observer cooperates. The ensemble of payment device and observer is called a wallet. In order to guarantee the untraceability of payments, the embedding must be such that any message the observer sends to the outside world passes through the payment device. This enables the payment device to recognize attempts of the observer to leak information (*outflow*) related to its identity, and vice versa (*inflow*).

If the observer stores all information it receives during the period it is embedded within the payment device, it might still be that the bank can trace payments to account-holders afterwards by comparing this information with the deposited payment transcripts (and possibly also its view in executions of the withdrawal protocol). Mutually known information which enables traceability is called *shared information*; it comprises both inflow and outflow. This concern, al-

though not specifically in the context of off-line cash systems, was raised in [10]. Although it might seem unrealistic to worry about the development of shared information, it is not hard to construct withdrawal and payment protocols with no inflow and outflow, whereas all payments can be traced virtually effortlessly by the bank once the observer is handed in. A trivial example of development of shared information without inflow or outflow is a payment protocol in which the observer and the payment device generate mutually at random a number, known to both, which the payment device sends to the shop.

As in the privacy-compromising system, the bank should not rely solely on tamper-resistance. If an account-holder unexpectedly breaks the tamper-resistance of the observer and double-spends, then he should still be traceable after the fact. This implies that the first concept acts as a safety net, and hence a realization of the second concept must be an extension of a realization of the first concept. For this reason, we refer to a system realizing the first concept as a basic cash system.

Contrary to the first concept, no realizations of an untraceable off-line cash system satisfying these conditions have been proposed yet. It seems that the known cash systems that provide realizations of the first concept cannot be extended to this important setting without worsening the problems related to efficiency and security. Our system can be extended to meet all the requirements of the second concept at virtually no extra cost in efficiency and security. Only a minor modification of the basic system is required. The workload for the observer is so small that it can be performed by a smart card capable of performing the Schnorr identification scheme.

Recently ([12]), Ferguson sketched how to extend his basic system ([11]) to wallets with observers; however, this seems to significantly worsen the problems related to security present in his basic system, as well as efficiency. As with the first concept, we show that there is no need for ad hoc constructions.

3 The representation problem in groups of prime order

All arithmetic in this article is performed in a group G_q of prime order q for which polynomial-time algorithms are known to multiply, invert, determine equality of elements, test membership, and randomly select elements. There is a vast variety of groups known to satisfy these requirements.

Definition 1. Let $k \geq 2$. A *generator-tuple* of length k is a k-tuple $(g_1, \ldots, g_k)$ with $g_i \in G_q \setminus \{1\}$ and $g_i \neq g_j$ if $i \neq j$. For any $h \in G_q$, a *representation* of h *with respect to* a generator-tuple $(g_1, \ldots, g_k)$ is a tuple $(a_1, \ldots, a_k)$, with $a_i \in \mathbb{Z}_q$ for all $1 \leq i \leq k$, such that $\prod_{i=1}^{k} g_i^{a_i} = h$.

Usually, it will be clear with respect to what generator-tuple a representation is taken, and we will not mention it. If $h = 1$, one representation immediately springs to mind, namely $(0, \ldots, 0)$. We call this the *trivial* representation.

Proposition 2. *For all $h \in G_q$ and all generator-tuples of length k there are exactly q^{k-1} representations of h.*

This simple result implies that the density of representations of h is negligible with respect to the set of size q^k containing all tuples $(a_1, \ldots, a_k)$. Therefore, any polynomial-time algorithm that applies an exhaustive search strategy in this set to find one has negligible probability $1/q$ of success. The following result shows that there is no essentially better strategy, assuming the Discrete Log assumption.

Proposition 3. *Assuming that it is infeasible to compute discrete logarithms in G_q, there cannot exist a number $h \in G_q$ and a polynomial-time algorithm that, on input a randomly chosen generator-tuple $(g_1, \ldots, g_k)$, outputs a (nontrivial if $h = 1$) representation of h with nonnegligible probability of success.*

Since the difference between two distinct representations of any number $h \in G_q$ is a nontrivial representation of 1, we get the following important result.

Corollary 4. *Assuming that it is infeasible to compute discrete logarithms in G_q, there cannot exist a polynomial-time algorithm that, on input a generator-tuple $(g_1, \ldots, g_k)$ chosen at random, outputs a number $h \in G_q$ and two different representations of h with nonnegligible probability of success.*

We next define the *representation problem in groups of prime order* (using a standard specification format).

Name: Representation problem in groups of prime order.
Instance: A group G_q, a generator-tuple $(g_1, \ldots, g_k)$, $h \in G_q$.
Problem: Find a representation of h with respect to $(g_1, \ldots, g_k)$.

Although our electronic cash system can be implemented with any group G_q that satisfies the listed conditions, and for which no feasible algorithms are known to compute discrete logarithms, we will for explicitness assume henceforth that G_q is the unique subgroup of order q of some multiplicative group $\mathbb{Z}_p^*$, for a prime p such that $q|(p-1)$.

4 Restrictive blinding in groups of prime order

In order to explain the notion of restrictive blinding, we give a high-level overview of the basic cash system, in which the primitive is put to use. In the following, (g_1, g_2) is a randomly chosen generator-tuple.

In setting up an account, the bank generates a unique number $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$ which is registered together with the identity of the account-holder with the newly created account. When the account-holder wishes to withdraw a coin from his account, the bank multiplies $I = g_1^{u_1}$ by g_2. Hence, the account-holder knows the representation $(u_1, 1)$ of the number $m = Ig_2$ with respect to (g_1, g_2). During the three-move withdrawal protocol, the account-holder will blind m to a number A, such that he ends up with a signature of the bank corresponding to A. A and the signature will be unconditionally untraceable to any specific execution of the withdrawal protocol. By construction of the payment protocol,

the account-holder at this stage *must* know a representation (x_1, x_2) of A with respect to (g_1, g_2) in order to be able to pay.

Here, the role of the *restrictive* blind signature protocol becomes clear.

Definition 5. Let $m \in G_q$ (in general, it can be a vector of elements) be such that the receiver at the start of a blind signature protocol knows a representation $(a_1, \ldots, a_k)$ of m with respect to a generator-tuple $(g_1, \ldots, g_k)$. Let $(b_1, \ldots, b_k)$ be the representation the receiver knows of the blinded number A of m after the protocol has finished. If there exist two functions I_1 and I_2 such that

$$I_1(a_1, \ldots, a_k) = I_2(b_1, \ldots, b_k),$$

regardless of m and the blinding transformations applied by the receiver, then the protocol is called a *restrictive* blind signature protocol. The functions I_1 and I_2 are called *blinding-invariant functions* of the protocol *with respect to* $(g_1, \ldots, g_k)$.

Intuitively, one can think of it as being a protocol in which the receiver can blind the "outside" of the message m (and signature), but not its internal structure.

For the application to untraceable off-line cash systems, in which the bank must be able to identify a payer if and only if he double-spends, we construct the payment protocol such that the account-holder not only has to reveal A and the signature, but also some additional information about the representation he knows of A. This additional information must be such that one such piece of information does not reveal any Shannon information about u_1 (the internal structure), whereas knowledge of two such pieces enables the bank to extract this number in polynomial time.

Clearly, if the account-holder in the payment protocol is able to also blind the internal structure of m, then he will not be identified after the fact when double-spending. Hence, it is absolutely essential that the receiver is restricted in the blinding manipulations he can perform, which explains the terminology *restrictive* blind signature scheme.

5 The basic cash system

In this section, we describe the most basic form of the cash system, involving only signed information (coins) of one value. We denote the bank by $\mathcal{B}$, a generic account-holder by $\mathcal{U}$, and a generic shop by $\mathcal{S}$. Although $\mathcal{U}$ will be a payment device (such as a smart card, palmtop or personal computer) in a practical implementation, we will often identify $\mathcal{U}$ with the account-holder.

The setup of the system. The setup of the system consists of $\mathcal{B}$ generating at random a generator-tuple (g, g_1, g_2), and a number $x \in_{\mathcal{R}} \mathbb{Z}_q^*$.

$\mathcal{B}$ also chooses two suitable collision-intractable (or even better, correlation-free one-way, as defined in [14]) hash functions $\mathcal{H}$, $\mathcal{H}_0$, with

$$\mathcal{H} : G_q \times G_q \times G_q \times G_q \times G_q \to \mathbb{Z}_q^*$$

and, for example,

$$\mathcal{H}_0 : G_q \times G_q \times \text{SHOP-ID} \times \text{DATE/TIME} \to \mathbb{Z}_q.$$

The function $\mathcal{H}$ is used for the construction and verification of signatures of $\mathcal{B}$, and the function $\mathcal{H}_0$ specifies in what way the challenges must be computed in the payment protocol. $\mathcal{B}$ publishes the description of G_q (which is p, q in the specific case of $G_q \subset \mathbb{Z}_p^*$), the generator-tuple (g, g_1, g_2), and the description of $\mathcal{H}$, $\mathcal{H}_0$ as its public key. The secret key of $\mathcal{B}$ is x.

The format of $\mathcal{H}_0$ assumes that each shop $\mathcal{S}$ has a unique identifying number $I_{\mathcal{S}}$ (this can be its account number at $\mathcal{B}$) known to at least $\mathcal{B}$ and $\mathcal{S}$; we denote above the set of all such numbers by SHOP-ID. The input from SHOP-ID ensures that two different shops with overwhelming probability will generate different challenges. The input from the set DATE/TIME is a number representing the date and time of transaction, which guarantees that the same shop will generate different challenges per payment. We stress that the format of $\mathcal{H}_0$ is just exemplary; other formats might do as well.

$\mathcal{B}$ also sets up two databases. One is called the account database and is used by the bank to store information about account-holders (such as their name and address), the other is called the deposit database and is used to store relevant information from deposited payment transcripts.

A signature $\text{sign}(A, B)$ of $\mathcal{B}$ on a pair $(A, B) \in G_q \times G_q$ consists of a tuple $(z, a, b, r) \in G_q \times G_q \times G_q \times \mathbb{Z}_q$ such that

$$g^r = h^{\mathcal{H}(A,B,z,a,b)} a \quad \text{and} \quad A^r = z^{\mathcal{H}(A,B,z,a,b)} b.$$

A *coin* is a triple $A, B, \text{sign}(A, B)$. If an account-holder knows a representation of both A and B with respect to (g_1, g_2), then we will simply say that he knows a representation of the coin.

Opening an account. When $\mathcal{U}$ opens an account at $\mathcal{B}$, $\mathcal{B}$ requests $\mathcal{U}$ to identify himself (by means of, say, a passport). $\mathcal{U}$ generates at random a number $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$, and computes $I = g_1^{u_1}$. If $g_1^{u_1} g_2 \neq 1$, then $\mathcal{U}$ transmits I to $\mathcal{B}$, and keeps u_1 secret. $\mathcal{B}$ stores the identifying information of $\mathcal{U}$ in the account database, together with I. We will refer to I as the account number of $\mathcal{U}$. The uniqueness of the account number is essential, since it enables $\mathcal{B}$ to uniquely identify $\mathcal{U}$ in case he double-spends.

$\mathcal{B}$ computes $z = (Ig_2)^x$, and transmits it to $\mathcal{U}$. Alternatively, $\mathcal{B}$ publishes g_1^x and g_2^x as part of his public key, so that $\mathcal{U}$ can compute z for himself.

The withdrawal protocol. When $\mathcal{U}$ wants to withdraw a coin, he first must prove ownership of his account. To this end, $\mathcal{U}$ can for example digitally sign a request for withdrawal, or identify himself by other means. Then the following withdrawal protocol is performed:

Step 1. $\mathcal{B}$ generates at random a number $w \in_{\mathcal{R}} \mathbb{Z}_q$, and sends $a = g^w$ and $b = (Ig_2)^w$ to $\mathcal{U}$.

Step 2. $\mathcal{U}$ generates at random three numbers $s \in_{\mathcal{R}} \mathbb{Z}_q^*$, $x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q$, and uses them to compute $A = (Ig_2)^s$, $B = g_1^{x_1} g_2^{x_2}$, and $z' = z^s$. $\mathcal{U}$ also generates at random two numbers $u, v \in_{\mathcal{R}} \mathbb{Z}_q$, and uses them to compute $a' = a^u g^v$ and $b' = b^{su} A^v$. He then computes the challenge $c' = \mathcal{H}(A, B, z', a', b')$, and sends the blinded challenge $c = c'/u \bmod q$ to $\mathcal{B}$.

Step 3. $\mathcal{B}$ sends the response $r = cx + w \bmod q$ to $\mathcal{U}$, and debits the account of $\mathcal{U}$.

$\mathcal{U}$ accepts if and only if $g^r = h^c a$ and $(Ig_2)^r = z^c b$. If this verification holds, $\mathcal{U}$ computes $r' = ru + v \bmod q$.

$\mathcal{U}$		$\mathcal{B}$
		$w \in_{\mathcal{R}} \mathbb{Z}_q$
		$a \leftarrow g^w$
$s \in_{\mathcal{R}} \mathbb{Z}_q^*$	$\xleftarrow{a, b}$	$b \leftarrow (Ig_2)^w$
$A \leftarrow (Ig_2)^s$		
$z' \leftarrow z^s$		
$x_1, x_2, u, v \in_{\mathcal{R}} \mathbb{Z}_q$		
$B \leftarrow g_1^{x_1} g_2^{x_2}$		
$a' \leftarrow a^u g^v$		
$b' \leftarrow b^{su} A^v$		
$c' \leftarrow \mathcal{H}(A, B, z', a', b')$		
$c \leftarrow c'/u \bmod q$	$\xrightarrow{c}$	
$g^r \stackrel{?}{=} h^c a$	$\xleftarrow{r}$	$r \leftarrow cx + w \bmod q$
$(Ig_2)^r \stackrel{?}{=} z^c b$		
$r' \leftarrow ru + v \bmod q$		

Proposition 6. *If $\mathcal{U}$ accepts in the payment protocol, then $A, B, (z', a', b', r')$ is a coin of which he knows a representation.*

Proposition 7. *Assume that it is infeasible to existentially forge Schnorr signatures, even when querying the prover in the Schnorr identification protocol polynomially many times. Then it is infeasible to existentially forge a coin, even when performing the withdrawal protocol polynomially many times and with respect to different account numbers.*

In other words, the number of coins in circulation can never exceed the number of executions of the withdrawal protocol. This obviously is an important fact, since one should not be able to create his own money. In fact, as Lemma 8 shows, the task is even much more difficult, since one in addition has to know a representation of a coin in order to be able to spend it.

Assumption 1. *The withdrawal protocol is a restrictive blind signature protocol (with $m = Ig_2$) with blinding invariant functions I_1 and I_2 with respect to (g_1, g_2) defined by $I_1(a_1, a_2) = I_2(a_1, a_2) = a_1/a_2 \bmod q$.*

Although this assumption is stronger than the Diffie-Hellman assumption, there are convincing arguments based on partial proofs that suggest that breaking it requires breaking either the Schnorr scheme or the Diffie-Hellman assumption. For an extensive discussion, we refer to [3].

The payment protocol. When $\mathcal{U}$ wants to spend his coin at $\mathcal{S}$, the following protocol is performed:

Step 1. $\mathcal{U}$ sends $A, B, \mathrm{sign}(A, B)$ to $\mathcal{S}$.
Step 2. If $A \neq 1$, then $\mathcal{S}$ computes challenge $d = \mathcal{H}_0(A, B, I_{\mathcal{S}}, \mathrm{date/time})$, where date/time is the number representing date and time of the transaction. $\mathcal{S}$ sends d to $\mathcal{U}$.
Step 3. $\mathcal{U}$ computes the responses $r_1 = d(u_1 s) + x_1 \bmod q$ and $r_2 = ds + x_2 \bmod q$, and sends them to $\mathcal{S}$.

$\mathcal{S}$ accepts if and only if $\mathrm{sign}(A, B)$ is a signature on (A, B), and $g_1^{r_1} g_2^{r_2} = A^d B$.

$\mathcal{U}$		$\mathcal{S}$
	$\xrightarrow{A, B, \mathrm{sign}(A, B)}$	$A \overset{?}{\neq} 1$
		$d \leftarrow \mathcal{H}_0(A, B, I_{\mathcal{S}}, \mathrm{date/time})$
	$\xleftarrow{d}$	
$r_1 \leftarrow d(u_1 s) + x_1 \bmod q$		
$r_2 \leftarrow ds + x_2 \bmod q$	$\xrightarrow{(r_1, r_2)}$	
		Verify $\mathrm{sign}(A, B)$
		$g_1^{r_1} g_2^{r_2} \overset{?}{=} A^d B$

If $\mathcal{U}$ has access to a clock and the capability of looking up the identifying information $I_{\mathcal{S}}$ of $\mathcal{S}$ (which seems more plausible when the payment device is, say, a personal computer dialing in via a modem, then in the case where it is a smart card), this protocol can be collapsed to one single move since $\mathcal{U}$ can then compute d himself. In any case, it is of no importance to $\mathcal{U}$ whether d is correctly determined. This is only of concern to $\mathcal{S}$, since the bank will not accept the payment transcript in the deposit protocol if d is not of the correct form.

Lemma 8. *If $\mathcal{U}$ in the payment protocol can give correct responses with respect to two different challenges, then he knows a representation of both A and B with respect to (g_1, g_2).*

Since $\mathcal{H}_0$ is a randomizing hash function, this result implies that the probability that $\mathcal{S}$ accepts in the payment protocol, whereas $\mathcal{U}$ does not know a representation of both A and B with respect to (g_1, g_2), is negligible. Together with the completeness of the payment protocol this implies the following.

Corollary 9. *$\mathcal{U}$ can spend a coin if and only he knows a representation of it.*

The deposit protocol. After some delay in time (since the system is off-line), $\mathcal{S}$ sends the to $\mathcal{B}$ the payment transcript, consisting of $A, B, \text{sign}(A, B), (r_1, r_2)$ and date/time of transaction.

If $A = 1$, then $\mathcal{B}$ does not accept the payment transcript. Otherwise, $\mathcal{B}$ computes d using the identifying number of the shop $I_{\mathcal{S}}$ sending the payment transcript, and the supplied date/time of transaction. $\mathcal{B}$ then verifies that $g_1^{r_1} g_2^{r_2} = A^d B$ and that $\text{sign}(A, B)$ is a signature on (A, B). If not both verifications hold, then $\mathcal{B}$ does not accept the payment transcript. Otherwise, $\mathcal{B}$ searches its deposit database to find out whether A has been stored before. There are two possibilities:

- A has not been stored before. In that case, $\mathcal{B}$ stores $(A,$ date/time$, r_1, r_2)$ in its deposit database as being deposited by $\mathcal{S}$, and credits the account of $\mathcal{S}$. Note that not the entire payment transcript need be deposited.
- A is already in the deposit database. In that case, a fraud must have occurred. If the already stored transcript was deposited by $\mathcal{S}$, and date/time are identical to that of the new payment transcript, then $\mathcal{S}$ is trying to deposit the same transcript twice. Otherwise (the challenges are different), the coin has been double-spent. Since $\mathcal{B}$ now has at its disposal a pair (d, r_1, r_2) from the new transcript and a pair (d', r_1', r_2') from the deposited information (where $\mathcal{B}$ computes d' from the date/time of transaction of the stored information and the identifying number $I_{\mathcal{S}}$ of the shop who deposited the transcript), it can compute

  $$g_1^{(r_1 - r_1')/(r_2 - r_2')}.$$

  $\mathcal{B}$ then searches its account database for this account number; the corresponding account-holder is the double-spender. The number $(r_1 - r_1')/(r_2 - r_2') \bmod q$ serves as a proof of double-spending; it is equal to $\log_{g_1} I$, with I the account number of the double-spender.

Since not even $\mathcal{B}$ needs to know a non-trivial representation of 1 with respect to (g, g_1, g_2) in order to perform the withdrawal protocol, there obviously cannot exist an adaptively chosen message attack that enables account-holders to know more than one representation of a coin (assuming that there are polynomially many account-holders and shops). Therefore, we get the following:

Proposition 10. *If Assumption 1 holds, then the computation that $\mathcal{B}$ performs in the deposit protocol in case of double-spending, results in the account number of the double-spender.*

We next prove, informally speaking, that the privacy of payments of account-holders who follow the protocols and do not double-spend is protected unconditionally.

Proposition 11. *For any $\mathcal{U}$, for any possible view of $\mathcal{B}$ in an execution of the withdrawal protocol in which $\mathcal{U}$ accepts, and for any possible view of $\mathcal{S}$ in an execution of the payment protocol in which the payer followed the protocol, there*

is exactly one set of random choices that $\mathcal{U}$ could have made in the execution of the withdrawal protocol such that the views of $\mathcal{B}$ and $\mathcal{S}$ correspond to the withdrawal and spending of the same coin.

An immediate consequence of this proposition is the following.

Corollary 12. *Assuming the Discrete Log assumption, if $\mathcal{U}$ follows the protocols and does not double-spend, $\mathcal{B}$ cannot compute a proof of double-spending.*

That is, $\mathcal{U}$ is computationally protected against a framing.

In [14], Okamoto described a signature protocol that is structurally equivalent to our payment protocol. Existential forgery of these signatures is a harder task than existential forgery of Schnorr signatures.

Proposition 13. *Existential forgery of payment transcripts is a harder task than existential forgery of Okamoto signatures.*

The following two results imply that no additional encryption of messages that are transmitted is needed anywhere in our system.

Proposition 14. *Wire tapping an execution of the withdrawal protocol does not result in a coin.*

Proposition 15. *Wire tapping an execution of the payment protocol with $\mathcal{S}$ does not result in a payment transcript that can be deposited to another account than that of $\mathcal{S}$.*

6 Prior restraint of double-spending

We describe how to extend our basic cash system to the setting of wallets with observers in such a way that not even shared information can be developed. Even if the tamper-resistance is broken (and the account-holder can simulate the role of the observer), we still have the same level of security as in the original system, in fact the protocols reduce completely to those of the basic system. In particular, if one breaks the tamper-resistance and, as a result, can double-spend, one will still be identified after the fact.

The setup of the system. This is the same as in the basic cash system.

Opening an account. When $\mathcal{U}$ opens an account at $\mathcal{B}$, $\mathcal{B}$ requests $\mathcal{U}$ to identify himself (by means of, say, a passport). $\mathcal{U}$ generates at random a number $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$, and computes $g_1^{u_1}$. $\mathcal{U}$ transmits $g_1^{u_1}$ to $\mathcal{B}$, and keeps u_1 secret. $\mathcal{B}$ stores the identifying information of $\mathcal{U}$ in the account database, together with $g_1^{u_1}$.

$\mathcal{B}$ then provides $\mathcal{U}$ with an observer $\mathcal{O}$, with stored in its (ROM) memory a randomly chosen number $o_1 \in \mathbb{Z}_q^*$ which is unknown to $\mathcal{U}$. We will denote $g_1^{o_1}$ by $A_{\mathcal{O}}$. $\mathcal{B}$ computes $I = A_{\mathcal{O}}(g_1^{u_1})$ and $z = (Ig_2)^x$, and transmits $A_{\mathcal{O}}$ and z to $\mathcal{U}$. $\mathcal{U}$ stores $u_1, A_{\mathcal{O}}, z$.

We will refer to I as the account number. This number will perform the role that I performed in the basic cash system. Note that, contrary to the basic cash system, $\mathcal{U}$ by himself does not know $\log_{g_1} I$.

The withdrawal protocol. When $\mathcal{U}$ wants to withdraw a coin from his account, he first must prove ownership of his account, as in the basic cash system. Then the following withdrawal protocol is performed:

Step 1. $\mathcal{O}$ generates at random a number $o_2 \in \mathbb{Z}_q$, and computes $B_{\mathcal{O}} = g_1^{o_2}$. He then sends $B_{\mathcal{O}}$ to $\mathcal{U}$. Although this step is part of the protocol, $\mathcal{O}$ can send $B_{\mathcal{O}}$ to $\mathcal{U}$ at any time before Step 3.

Step 2. $\mathcal{B}$ generates at random a number $w \in_{\mathcal{R}} \mathbb{Z}_q$, and sends $a = g^w$ and $b = (Ig_2)^w$ to $\mathcal{U}$.

Step 3. $\mathcal{U}$ generates at random four numbers $s \in_{\mathcal{R}} \mathbb{Z}_q^*$, $x_1, x_2, e \in_{\mathcal{R}} \mathbb{Z}_q$, and uses them to compute $A = (Ig_2)^s$, $B = g_1^{x_1} g_2^{x_2} A_{\mathcal{O}}^{es} B_{\mathcal{O}}$, and $z' = z^s$. $\mathcal{U}$ also generates at random two numbers $u, v \in_{\mathcal{R}} \mathbb{Z}_q$, and uses them to compute $a' = a^u g^v$ and $b' = b^{su} A^v$. He then computes the challenge $c' = \mathcal{H}(A, B, z', a', b')$, and sends the blinded challenge $c = c'/u \bmod q$ to $\mathcal{B}$.

Step 4. $\mathcal{B}$ sends the response $r = cx + w \bmod q$ to $\mathcal{U}$, and debits the account of $\mathcal{U}$.

$\mathcal{U}$ accepts if and only if $g^r = h^c a$ and $(Ig_2)^r = z^c b$. If this verification holds, $\mathcal{U}$ computes $r' = ru + v \bmod q$.

$\mathcal{O}$		$\mathcal{U}$		$\mathcal{B}$
				$w \in_{\mathcal{R}} \mathbb{Z}_q$
$o_2 \in_{\mathcal{R}} \mathbb{Z}_q$				$a \leftarrow g^w$
$B_{\mathcal{O}} \leftarrow g_1^{o_2}$	$\xrightarrow{B_{\mathcal{O}}}$	$s \in_{\mathcal{R}} \mathbb{Z}_q^*$	$\xleftarrow{a,b}$	$b \leftarrow (Ig_2)^w$
		$A \leftarrow (Ig_2)^s$		
		$z' \leftarrow z^s$		
		$x_1, x_2, e \in_{\mathcal{R}} \mathbb{Z}_q$		
		$B \leftarrow g_1^{x_1} g_2^{x_2} A_{\mathcal{O}}^{se} B_{\mathcal{O}}$		
		$u, v \in_{\mathcal{R}} \mathbb{Z}_q$		
		$a' \leftarrow a^u g^v$		
		$b' \leftarrow b^{su} A^v$		
		$c' \leftarrow \mathcal{H}(A, B, z', a', b')$		
		$c \leftarrow c'/u \bmod q$	$\xrightarrow{c}$	
		$g^r \stackrel{?}{=} h^c a$	$\xleftarrow{r}$	$r \leftarrow cx + w \bmod q$
		$(Ig_2)^r \stackrel{?}{=} z^c b$		
		$r' \leftarrow ru + v \bmod q$		

If we concentrate on $\mathcal{O}$ and $\mathcal{U}$ as one party, then this is exactly the basic withdrawal protocol. Hence, Propositions 6 (with "$\mathcal{O}$ and $\mathcal{U}$ together know" substituted for "he knows") and 7 hold.

The payment protocol. When $\mathcal{U}$ wants to pay with the withdrawn information at $\mathcal{S}$, the following protocol is performed:

Step 1. $\mathcal{U}$ sends $A, B, \mathrm{sign}(A,B)$ to $\mathcal{S}$.

Step 2. If $A \neq 1$, $\mathcal{S}$ computes challenge $d = \mathcal{H}_0(A, B, I_{\mathcal{S}}, \mathrm{date/time})$, and sends it to $\mathcal{U}$.

Step 3. $\mathcal{U}$ computes $d' = s(d+e) \bmod q$, and sends this to $\mathcal{O}$.

Step 4. If o_2 is still in memory, then $\mathcal{O}$ computes the response $r_1' = d'o_1 + o_2 \bmod q$ and send it to $\mathcal{U}$. (If o_2 has already been erased, then $\mathcal{O}$ e.g. locks up.) Then $\mathcal{O}$ erases o_2 from its memory.

Step 5. $\mathcal{U}$ verifies that $g_1^{r_1'} = A_{\mathcal{O}}^{d'} B_{\mathcal{O}}$. If this verification holds, he computes $r_1 = r_1' + d(u_1 s) + x_1 \bmod q$ and $r_2 = ds + x_2 \bmod q$. He then sends (r_1, r_2) to $\mathcal{S}$.

$\mathcal{S}$ accepts if and only if $\mathrm{sign}(A,B)$ is a signature on (A,B), and $g_1^{r_1} g_2^{r_2} = A^d B$.

$\mathcal{O}$	$\mathcal{U}$	$\mathcal{S}$
		$A, B, \mathrm{sign}(A,B) \longrightarrow$ $A \overset{?}{\neq} 1$
		$d \leftarrow \mathcal{H}_0(A, B, I_{\mathcal{S}}, \mathrm{date/time})$
		$\longleftarrow d$
	$\longleftarrow d'$ $d' \leftarrow s(d+e) \bmod q$	
o_2 still in memory? $r_1' \leftarrow d'o_1 + o_2 \bmod q$		
	$\longrightarrow r_1'$ $g_1^{r_1'} \overset{?}{=} A_{\mathcal{O}}^{d'} B_{\mathcal{O}}$	
	$r_1 \leftarrow r_1' + d(u_1 s) + x_1 \bmod q$	
	$r_2 \leftarrow ds + x_2 \bmod q$	
		$\longrightarrow (r_1, r_2)$
		Verify $\mathrm{sign}(A,B)$
		$g_1^{r_1} g_2^{r_2} \overset{?}{=} A^d B$

As in the basic system, if $\mathcal{U}$ has a clock and the capability of looking up the identifying information $I_{\mathcal{S}}$ of $\mathcal{S}$, the protocol can be collapsed to one move from $\mathcal{U}$ to $\mathcal{S}$.

Since $\mathcal{U}$ by himself does not know a representation of I, it is easy to prove that he cannot know a representation of the coin by himself if the basic withdrawal protocol is a restrictive blind signature protocol. From Lemma 8 we hence get:

Proposition 16. *Assuming the tamper-resistance of $\mathcal{O}$ cannot be broken, $\mathcal{U}$ cannot spend a coin without cooperation of $\mathcal{O}$.*

Due to the important fact that $\mathcal{O}$ in the ensemble of withdrawal and payment protocols in effect performs exactly the Schnorr identification protocol, proving knowledge of $\log_{g_1} A_{\mathcal{O}}$, this result should hold even after polynomially many executions of the protocols.

We next investigate the privacy of the account-holders in this system.

Proposition 17. *If $\mathcal{U}$ follows the protocols, and does not double-spend, then no shared information can be developed between $\mathcal{O}$, $\mathcal{B}$, and all shops $\mathcal{S}$ in executions of the withdrawal and payment protocols he takes part in.*

Informally speaking, the privacy of payments of an account-holder who follows the protocols and does not double-spent is unconditionally protected, even if his observer's contents can be examined afterwards by the bank. If we encode denominations in g_2, then the property of no shared information even relates to the value of the coin.

The deposit protocol. This is exactly the same as in the basic system. In case the coin was double-spent, the number $(r_1 - r_1')/(r_2 - r_2') - o_1 \bmod q$ serves as a proof of double-spending.

Proposition 18. *If the tamper-resistance of $\mathcal{O}$ is broken (enabling $\mathcal{U}$ to simulate its role), then still the same level of security as in the basic cash system is guaranteed. In particular, if $\mathcal{U}$ double-spends, he will be identified after the fact.*

This follows immediately from the fact that the protocols in that case reduce to those of the basic cash system (view $\mathcal{U}$ and $\mathcal{O}$ as one entity).

7 Concluding remarks

In practice, the random number generator of $\mathcal{U}$ can be a quite simple pseudo-random bit generator. In that case, it might be preferable to reconstruct A, B, x_1,, x_2, s at payment time from the intermediary state of the generator.

The random number generators of $\mathcal{O}$ and $\mathcal{B}$ on the other hand must be cryptographically strong, since $\mathcal{U}$ can heavily analyze their outputs; preferably, $\mathcal{B}$'s pseudo-random numbers should be combined with numbers obtained from physical randomness (e.g. noise generators).

Certain security aspects can be straightforwardly strengthened by using the idea of [4]. However, this modification does not seem to increase the plausibility of Assumption 1, whereas it requires more computations of the payment device.

The only thing left open in mathematically proving the security of our system and its extensions to as great an extent as the current state of knowledge in cryptography seems to allow, is proving that the particular blind signature protocol we used is a restrictive one, without assuming non-standard assumptions. We do not know how to do this, although there are convincing partial proofs (see [3]) that suggest that breaking it requires breaking the Diffie-Hellman key assumption.

Nevertheless, this is not a serious problem; recently ([3]), we have come up with various other (even more efficient) restrictive blind signature schemes in groups of prime order, of which we can rigorously prove for fixed m that the security is equivalent to that of the Schnorr signature scheme. As should be obvious, any restrictive blind signature scheme can be substituted for the particular one used in this abstract, requiring only some minor modifications to the

protocols. In [3], we also describe similar constructions based on the representation problem in RSA-groups and restrictive blind signature schemes related to the Guillou/Quisquater signature scheme.

8 Acknowledgements

This system, in particular the extension to wallets with observers, is being studied by the European ESPRIT project CAFE. Various members of the project provided feedback on earlier versions of my technical report [2]. I am grateful to Ronald Cramer, Torben Pedersen and Berry Schoenmakers for their comments. Torben also provided part of the partial proofs that suggest that breaking Assumption 1 requires breaking the Schnorr signature scheme or the Diffie-Hellman assumption.

I especially want to thank David Chaum. This work was greatly inspired by his innovative work on untraceable electronic cash.

References

1. Bellare, Micali, "How To Sign Given Any Trapdoor Function," Proceedings of Crypto '88, Springer-Verlag, pages 200–215.
2. Brands, S., "An Efficient Off-line Electronic Cash System Based On The Representation Problem," CWI Technical Report CS-R9323, April 11, 1993.
3. Brands, S., "Untraceable Off-Line Cash Based On The Representation Problem," manuscript. To be published as a CWI Technical Report in Januari/Februari 1994.
4. Brickell, E. and McCurley, K., "An Interactive Identification Scheme Based On Discrete Logarithms And Factoring," Journal of Cryptology, Vol. 5 no. 1 (1992), pages 29–39.
5. Chaum, D., "Achieving Electronic Privacy," Scientific American, August 1992, pages 96–101.
6. Chaum, D., "Security Without Identification: Transaction Systems To Make Big Brother Obsolete," Communications of the ACM, Vol. 28 no. 10, October 1985, pages 1020–1044.
7. Chaum, D., "Card-computer moderated systems," (unpublished), 1989.
8. Chaum, D., Fiat, A. and Naor, M., "Untraceable Electronic Cash," Proceedings of Crypto '88, Springer-Verlag, pages 319–327.
9. Chaum, D. and Pedersen, T., "Wallet Databases With Observers," Preproceedings of Crypto '92.
10. Cramer, R. and Pedersen, T., "Improved Privacy In Wallets With Observers', Preproceedings of EuroCrypt '93.
11. Ferguson, N., "Single Term Off-Line Coins", Preproceedings of EuroCrypt '93.
12. Ferguson, N., "Extensions Of Single-Term Off-Line coins," these proceedings.
13. Fiat, A. and Shamir, A., "How To Prove Yourself: Practical Solutions To Identification And Signature Problems," Proceedings of Crypto '86, Springer-Verlag, pages 186–194.
14. Okamoto, T., "Provably Secure And Practical Identification Schemes And Corresponding Signature Schemes," Preproceedings of Crypto '92.

15. Schnorr, C.P., "Efficient Signature Generation By Smart Cards," Journal of Cryptology, Vol. 4 no. 3 (1991), pages 161–174.
16. "No Hiding Place / Big Brother Is Clocking You," The Economist, August 7th–13th 1993.