Publication date: Januari 2, 1995. This paper will appear in the Proceedings of the Second International Symposium of Latin American Theoretical Informatics (LATIN '95), Valparaíso, Chili, April 3–7, 1995.

Off-Line Electronic Cash Based on Secret-Key Certificates

Stefan Brands

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
E-mail: `brands@cwi.nl`

Abstract. An off-line electronic coin system is presented that offers multi-party security and unconditional privacy of payments. The system improves significantly on the efficiency of the previously most efficient such system known in the literature, due to application of a recently proposed technique called secret-key certificates.
By definition of secret-key certificates, pairs consisting of a public key and a matching certificate can be simulated with indistinguishable probability distribution. This allows a variety of polynomial-time reductions from a well-known signature scheme to the cash system. In particular, the withdrawal protocol can be proved to be restrictive blind with respect to one account holder, relying only on a standard intractability assumption; no such result has been proved before in the literature.
Another consequence of the application of the secret-key certificate technique is that the withdrawal protocol is not a blind signature issuing protocol. This falsifies the popular belief that efficient privacy-protecting off-line electronic cash systems must be based on withdrawal protocols that are blind signature issuing protocols.

1 Introduction

The information that is transferred in an electronic cash system has intrinsic value: it represents the digital equivalent of cash. One can distinguish between on-line and off-line systems, depending on whether the bank needs or doesn't need to verify payments in real-time. Off-line cash systems are obviously highly preferable over on-line systems in case of low-value payments. Adequate security mechanisms, based on public-key cryptographic techniques, can guarantee security against double-spending of electronic coins. Another important characteristic of cash systems is whether privacy of payments is offered or not. Payments in a privacy-protecting electronic cash system are untraceable and unlinkable, just as ordinary coins are.

A fairly large body of cryptographic literature is devoted to the design of privacy-protecting off-line electronic cash systems. Almost all of this literature [7, 10, 11, 13, 16, 17, 18, 19, 23, 25, 28, 29, 32, 33, 36, 37, 38, 39] is concerned with systems that only offer traceability as a security measure against double-spending, based on a paradigm of Chaum, Fiat and Naor [11]. This measure clearly does not offer a security level that is adequate for practical purposes.

If an electronic cash system is to be of practical value, then it must offer prior restraint of double-spending, at least as the first line of defense.

An important reason for the lack of prior restraint in the abovementioned references is that prior restraint can only be incorporated in an off-line system by using tamper-resistant user-modules, but naive adoption of tamper-resistant devices has the effect that account holders can no longer guarantee the privacy of their own payments. The only configuration that can ensure prior restraint, while at the same time maintaining the ability of account holders to ensure their own privacy, is one that has been proposed by Chaum [9] (see Chaum and Pedersen [12] for formal definitions and methodology, and Bos and Chaum [2] for the very first appearance of this approach in the cryptographic literature). In the configuration proposed by Chaum, an account holder interfaces a tamper-resistant device of the bank to his own computing device in such a way that all flow of information between the tamper-resistant device and the outside world must pass through his computer.

Developing off-line electronic cash protocols for this configuration, without giving up on the traceability of double-spenders (which can serve as a second line of defense in case tamper-resistance is compromised), is far from trivial: the configuration forces the use of secure three-party protocols, which are obviously a great deal harder to construct than secure two-party protocols. Maintaining traceability after the fact as as second line of defense nevertheless seems imperative, since practice indicates that any tamper-resistant device can be compromised if only sufficient resources are at hand.

In [3], I presented the first privacy-protecting off-line electronic cash system that is based on the configuration proposed by Chaum, and that offers both lines of defense against double-spending. It moreover offers an even greater level of privacy than that envisioned by Chaum, in that the computer of the account holder can prevent the development of so-called shared information, a privacy notion that has been introduced by Cramer and Pedersen [15]. (For an incomplete, and hence rather dubious, description of how this functionality might be achieved in the system of Ferguson [23], see Ferguson [24].) Apart from the greatly improved security due to the incorporation of prior restraint of double-spending, the system described in [3] compares very favorably to previously proposed privacy-protecting off-line electronic cash systems with respect to efficiency. This success is due to a new technique, called restrictive blinding, in combination with the so-called representation problem in groups of prime order.

As with any privacy-protecting off-line electronic cash system that features traceability of double-spenders, regardless of whether it serves as the first line of defense or only as the second, by far the most difficult aspect of the security analysis is to prove that the "identity" of the account holder indeed ends up being encoded in his electronic coins. In the system in [3] such a proof was not provided; instead, this property simply was assumed to hold, based on evidence provided by partial proofs. Assuming this non-standard assumption allowed all other statements to be proved on the basis of a standard intractability assumption that is related to the Schnorr identification scheme [35], which certainly was a big step forward in comparison to the provability of earlier "practical" proposals in the literature.

The particular restrictive blind signature scheme used in [3] was derived from an "ordinary" blind signature issuing protocol due to Chaum and Pedersen [12], and is a particular instance of a public-key certificate scheme. (See Chaum [8] for introduction of blind signatures.) As such, it is a blind signature issuing protocol, albeit a very special one. Subsequent attempts by me, and others, to design similar other restrictive blind signature schemes failed. The experienced difficulty in designing other restrictive blind signature schemes seems to be due to the tight relation between a public key and a matching certificate; such pairs by definition cannot be forged in secure public-key certificate schemes.

Recently, in part (i) of [4], I described a new technique, called secret-key certificates. This technique is much more suitable for the design of restrictive blind signature issuing protocols than the public-key certificate technique (see part (ii) of [4]). More specifically, the new technique allows any "Fiat-Shamir type" signature scheme (*e.g.*, [6, 21, 22, 27, 30, 35]) to be converted to a restrictive blind signature scheme, if only it can be converted to an ordinary blind signature issuing protocol by applying a blinding technique due to Okamoto and Ohta [31] (see also Okamoto [30]). The resulting signature schemes can rigorously be proved to be restrictive blind with respect to a single receiver, relying only on a standard intractability assumption. The new restrictive blind signature schemes are more efficient for the receiver than the scheme used in [3], both in on-line and off-line computational requirements. Somewhat surprisingly, the new restrictive blind signature schemes are not ordinary blind signature schemes; see part (ii) of [4] and the discussion in Sect. 7.

The off-line electronic cash system that is proposed in this paper is based on the secret-key certificate technique. It achieves exactly the same properties as does the system in [3], except for the ability to mathematically disprove false claims of double-spending. It thereby falsifies the popular belief that efficient privacy-protecting off-line electronic cash systems (and privacy-protecting credential mechanisms in general) must be based on public-key certificates, and hence on withdrawal protocols that are special instances of ordinary blind signature issuing protocols.

To facilitate comparison to [3] the description in this paper is explicitly in terms of a secret-key certificate scheme based on the Schnorr signature scheme. The description in part (ii) of [4] provides enough handles to easily make the conversion for any of the other secret-key certificate schemes that are described in that reference.

In view of the number of pages that is absorbed by the analysis of correctness, a motivation of the design criteria for the system has been refrained from. Instead, the reader is referred for background information to the references mentioned above (in particular [9, 11, 12, 15]), and to Sections 2 and 4 of [3] for the complete picture. For the same reason, a description of an intermediate version (*i.e.*, one that only offers traceability as a defense against double-spending), however helpful in understanding the complete system, has been omitted.

2 A Brief Overview of the New System

A high-level overview of the new system is presented below. This should provide a clear picture of the flow of electronic cash in the new system.

The bank issues an electronic coin to a user by issuing in a *withdrawal protocol* a certified key pair to him. A certified key pair is a triple consisting of a secret key, a public key and a certificate of the bank on the public key. The pair consisting only of the public key and the certificate will be called a certified public key; since we use secret-key certificates, such pairs can be generated by anyone, without cooperation of the bank (see part (i) of [4]). Each electronic coin is represented by a different certified public key.

The user has at his disposal a computer that is trusted by him (typically a personal computer or a hand-held computer), and a tamper-resistant device that has been issued to him by the bank (typically a smart card or a PCMCIA card). The configuration of the user's computer and his tamper-resistant device is such that all flow of information between the tamper-resistant device and the outside world must pass through the computer; see [9, 12]. By construction of the protocols, the computer of the user can perfectly blind the certified public key when performing the withdrawal protocol, but not a certain "blinding-invariant" part of the secret key that corresponds to the certified public key. The construction furthermore ensures that this blinding-invariant part of the secret key is known only to the tamper-resistant device.

To spend the electronic coin in a succeeding *payment protocol* at a service provider, the computer of the user computes with respect to the (certified) public key a digital signature on a message of the service provider. It then sends the certified public key and the signature to the service provider. Prior restraint of double-spending the electronic coin is due to the fact that not the entire secret key is known to the user's computer, and so computation of a digital signature requires the assistance of the tamper-resistant device. Of course, the tamper-resistant device has been programmed by the bank such as to assist only once in computing a signature with respect to a certified public key of the user.

Because the certified public key has been perfectly blinded by the user's computer in the withdrawal protocol, the revelation of the certified public key in the payment protocol does not leak any information that is correlated to the identity of the user. Moreover, the computer of the user moderates on the flight all flow of information between the tamper-resistant device and the service provider. The moderation ensures that *all* subliminal channels are prevented. Furthermore, the computations performed by the user's computer prevent the development of randomly generated numbers that are known to both the tamper-resistant device and the service provider. Although mutually known random numbers cannot serve as a subliminal channel, they would readily enable the payments of the user to be traced in case they are retrieved by the bank upon return of the tamper-resistant device; see [15]. In all, there are three potential ways to compromise the privacy of the user, each of which is prevented information-theoretically by virtue of the protocol design.

To deposit the electronic coin, at a later stage, the transcript of the payment protocol is sent by the service provider to the bank in a *deposit protocol.* If two different payment transcripts involve the same certified public key, then the same electronic coin has been double-spent by a party that managed to physically extract the secret information of a tamper-resistant device. By virtue of the protocol design, the two respective signatures enable the bank to compute the secret key that corresponds to the certified public key, and in particular the blinding-invariant part thereof; because the bank knows which user it issued the tamper-resistant device with this blinding-invariant part to, it can trace the double-spender.

3 The Blind Schnorr Signature Scheme

Because the security analysis of the cash system will be in terms of polynomial-time reductions from a well-known protocol to the cash system (which can be seen as one huge multi-party protocol), the key generation algorithms in the cash system must inherit certain characteristics of the key generation algorithm for the well-known protocol. We therefore first study this protocol.

Let us start by recapitulating the Schnorr identification scheme [35]. The arithmetical operations in the Schnorr identification scheme are performed in a group G_q of prime order q for which polynomial-time algorithms must be known to multiply, determine equality of elements, test membership, and to randomly select elements. Furthermore, no feasible algorithms for computing discrete logarithms in G_q should be known. Various types of such groups are well-known in the literature, and for this reason no specific such type will be fixed. For simplicity, and without loss of generality, we will assume that the random generation of a group G_q is completely specified by generating at random a prime q. By this convention, the Discrete Log assumption for G_q is the following: no probabilistic polynomial-time algorithm, on input a random triple (q, g, h), can output $\log_g h$ with overwhelming probability of success (and hence also not with non-negligible probability of success). We will henceforth assume without loss of generality that each of the elements in the input triple is generated independently at random according to a uniform probability distribution over the following sets: q is chosen from the set of primes of length k, g is chosen from $G_q \setminus \{1\}$, and h is chosen from G_q. (We exclude $g = 1$ merely for convenience, because we want to avoid having to mention in several proofs later on that there is a negligible probability that g is not a generator, or that the simulator in the proof generates a statistically indistinguishable probability distribution rather than an identical one.)

The *key generation algorithm* for the Schnorr identification scheme, on input a security parameter k, generates a public key (q, g, h) and a corresponding secret key $\log_g h$, to be used by a probabilistic polynomial-time prover $\mathcal{P}$. The prime q and the generator g are chosen as specified for the Discrete Log assumption. The secret key is chosen at random from $\mathbb{Z}_q$, and h is correspondingly computed as $h := g^x$. (We will not consider the key generation algorithm to generate in addition a polynomial-size certificate of primality of q.)

$\mathcal{P}$ can prove knowledge of its secret key to a probabilistic polynomial-time verifier $\mathcal{V}$ by means of the following challenge–response *identification protocol*:

Step 1. $\mathcal{P}$ generates at random a number $w \in \mathbb{Z}_q$, and sends $a := g^w$ to $\mathcal{V}$.
Step 2. $\mathcal{V}$ generates a challenge $c \in \mathbb{Z}_{2^n}$, and sends it to $\mathcal{P}$.
Step 3. $\mathcal{P}$ sends its response $r := cx + w \bmod q$ to $\mathcal{V}$.

$\mathcal{V}$ accepts if and only if $g^r h^{-c} = a$.

The number n is a security parameter that is super-logarithmical in k. Note that we have not assumed it to be generated as part of the public key, since it can be taken to be a predetermined constant fraction of q.

As shown by Schnorr, this protocol constitutes a proof of knowledge when $\mathcal{V}$ generates its challenge according to a uniform probability distribution, or one close to it. (See Feige, Fiat and Shamir [21] for a treatment of proofs of knowledge, and Bellare and Goldreich [1] for improvements related to the definition.) Although the protocol presumably is not zero-knowledge, it is generally believed to be witness hiding (as defined by Feige and Shamir [20]).

By applying a general technique, originating from Fiat and Shamir [22], for converting sound proofs of knowledge into signature issuing protocols, the protocol can be converted into a signature issuing protocol. To sign a message m (which may be a *vector* of numbers), the challenge c should hereto be taken equal to $\mathcal{H}(m, a)$. Here, $\mathcal{H}(\cdot)$ is a description of a collision-intractable hash-function, of size polynomial in k, that maps its inputs to $\mathbb{Z}_{2^n}$. (Actually, it is preferable that $\mathcal{H}(\cdot)$ is correlation-free, as defined by Okamoto [30], since the ability to determine a signature on a new message, by algebraically combining previously received signatures, is not excluded by collision-intractability.)

Note that we are implicitly modifying the key generation algorithm. The new key generation algorithm, on input k, generates a public key $(q, g, h, \mathcal{H}(\cdot))$ and a secret key $\log_g h$. The hash-function, $\mathcal{H}(\cdot)$, is generated at random from some suitable family of collision-intractable hash functions, and the other elements are generated in accordance with the key generation algorithm for the Schnorr identification algorithm.

The interaction can be removed in the new protocol because c can be determined by $\mathcal{P}$ itself. Furthermore, because n typically can be taken to be much smaller than the size of elements in G_q, the most compact representation of a signature on m is the pair (c, r), where c is equal to $\mathcal{H}(m, a)$. The pair (c, r) is called a Schnorr signature on m if and only if c is equal to $\mathcal{H}(m, g^r h^{-c})$.

By retaining the interaction in the signature issuing protocol, $\mathcal{V}$ can receive a signature on a message m that is unknown to $\mathcal{P}$. In particular, as shown by Ohta and Okamoto [31], retaining the interaction enables $\mathcal{V}$ to in addition blind the signature (c, r); hereto $\mathcal{P}$ must allow the challenge c in Step 2 to be in $\mathbb{Z}_q$. The resulting pairs, consisting of a message and a corresponding Schnorr signature, can be shown to be uncorrelated to views of $\mathcal{P}$ in executions of the signature issuing protocol. In other words, this protocol is an "ordinary" blind signature issuing protocol, as (informally) defined by Chaum (see [8] and later work). Note that we have once more modified implicitly the key generation algorithm, since $\mathcal{H}(\cdot)$ now maps its arguments to $\mathbb{Z}_q$.

Henceforth, we will refer to the final protocol as the *blind Schnorr signature issuing protocol*. Similarly, we will refer to the ensemble of key generation algorithm and protocol as the blind Schnorr signature *scheme*.

4 The System

Before providing the mathematical description of the system, some conventions are explained.

The distinction is made between two kinds of account holders: parties that only deposit electronic coins, and parties that also have the ability to withdraw electronic coins. Parties of the first kind are called *service providers*, and parties of the second kind are called *users*. All parties in the description are denoted by calligraphic letters: $\mathcal{B}$ for bank, $\mathcal{C}_i$ for the computer controlled by a user $\mathcal{U}_i$, $\mathcal{T}_i$ for the tamper-resistant device of $\mathcal{U}_i$, and $\mathcal{S}_j$ for a service provider, where $i, j \in \mathbb{N}$. Each of $\mathcal{B}$, $\mathcal{C}_i$, $\mathcal{T}_i$ and $\mathcal{S}_j$ should be thought of as a polynomial-time interactive Turing machine [21, 26] and $\mathcal{U}_i$ should be thought of as the pair ($\mathcal{C}_i$, $\mathcal{T}_i$). The running times and the total number of parties are polynomial in the security parameter k.

It is implicitly assumed in the descriptions of the protocols that a party halts the execution of a protocol in case it does not accept at a certain stage. A number that is said to be chosen at random from some set is always implicitly assumed to be generated according to a uniform probability distribution over the specified set, independently of any other event. Assignments will always be denoted by the symbol ":=."

We are now prepared to describe the system.

4.1 Initial Key Generation

On input a security parameter k, the key generation algorithm of $\mathcal{B}$ generates a secret key (x_{00}, x_{10}) and a public key $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$. Since we have assumed in the previous section independent uniform probability distributions for the key generation algorithm for the blind Schnorr signature scheme, the following distributions apply to the key generation algorithm for $\mathcal{B}$:

- q is a randomly chosen prime of length k.
- x_{00} and x_{10} are two randomly chosen elements in $\mathbb{Z}_q$.
- g_0 is a randomly chosen element in $G_q \setminus \{1\}$.
- h_0 denotes $g_0^{x_{00}}$, and g_1 denotes $g_0^{x_{10}}$.
- $\mathcal{H}(\cdot)$ is the description, of size polynomial in k, of a randomly chosen collision-intractable hash-function that maps its inputs to $\mathbb{Z}_{2^n}$.

(If we had not explicitly assumed an independent uniform probability distribution for the elements generated by the key generation algorithm for the blind Schnorr signature scheme, then its sub-algorithms would have had to be called, for which we would have had to describe, and motivate, an appropriate dissection into sub-algorithms. This would have obscured the description of the cash system, and even more so the polynomial-time reductions from the blind Schnorr signature scheme that will be shown later on. Note that this is not just being meticulous; it is imperative for the correctness of the simulations.)

In terms of the overview in Sect. 2, a certified public key is a pair (h_i, a_i), (c_0, r_0) such that

$$c_0 = \mathcal{H}((h_0 h_i, a_i), g_0^{r_0}(h_0 h_i)^{-c_0}).$$

A certified key pair, issued in the withdrawal protocol, is a triple

$$((x_{0i}, x_{1i}), (w_{0i}, w_{1i})),\ (h_i, a_i),\ (c_0, r_0)$$

such that $(h_i, a_i), (c_0, r_0)$ is a certified public key, and $h_i = g_0^{x_{0i}} g_1^{x_{1i}}$ and $a_i = g_0^{w_{0i}} g_1^{w_{1i}}$. We will from now leave out brackets, and simply write $\mathcal{H}(h_0 h_i, a_i, \cdot)$.

In this definition of a certified public key, the multiplication of h_i by h_0 in the hash-value is not needed; its only purpose is to make Assumption II in Sect. 5 look cleaner. (Alternatively, Assumption II could be adapted.)

$\mathcal{B}$ also sets up two tables, one representing an account database to store information related to its account holders, the other representing a deposit database to store deposited electronic coins.

4.2 Opening an Account

When service provider $\mathcal{S}_j$ opens an account, an entry is added by $\mathcal{B}$ to the account database. This entry consists of an appropriate description of the "identity" of $\mathcal{S}_j$, and a counter representing its cash balance.

When user $\mathcal{U}_i$ opens an account, the same procedure is followed. In addition, $\mathcal{B}$ provides $\mathcal{U}_i$ with a tamper-resistant device $\mathcal{T}_i$, and stores a randomly chosen *identification number* $x_{1i} \in \mathbb{Z}_q$ in $\mathcal{U}_i$'s entry in the account database. The identification number is the secret key of $\mathcal{T}_i$. The corresponding public key of $\mathcal{T}_i$ is $g_1^{x_{1i}}$, and will henceforth be denoted by h_i. (Again, had we not explicitly assumed an independent uniform probability distribution for elements generated by the key generation algorithm for the blind Schnorr signature scheme, then its sub-algorithms for generating a secret key x and corresponding $h := g^x$ would have had to be called in order to generate x_{1i} and h_i.)

Note that $\mathcal{S}_j$ does not need a tamper-resistant device. Furthermore, the entry of $\mathcal{S}_j$ may contain a pseudonym instead of a description of its identity.

4.3 The Withdrawal Protocol

To withdraw an electronic coin, $\mathcal{U}_i$ (after having proved ownership of the account) performs the following protocol with $\mathcal{B}$:

Step 1. $\mathcal{T}_i$ generates at random a number $w_i \in \mathbb{Z}_q$, and sends $a_i := g_1^{w_i}$ to $\mathcal{C}_i$. $\mathcal{T}_i$ then stores w_i in memory, for later use in the payment protocol.

Step 2. $\mathcal{B}$ generates at random a number $w_0 \in \mathbb{Z}_q$, and sends $a_0 := g_0^{w_0}$ to $\mathcal{C}_i$.

Step 3. $\mathcal{C}_i$ generates at random six numbers $x_{0i}, w_{0i}, w_{1i}, s_1, s_2, s_3 \in \mathbb{Z}_q$. It computes $h_i' := g_0^{x_{0i}} h_i$, $a_i' := g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} a_i$, $c_0' := \mathcal{H}(h_0 h_i', a_i', g_0^{s_1}(h_0 h_i)^{s_2} a_0)$, and sends $c_0 := c_0' + s_2 \bmod q$ to $\mathcal{B}$.

Step 4. $\mathcal{B}$ sends $r_0 := c_0(x_{00} + x_{10}x_{1i}) + w_0 \bmod q$ to $\mathcal{C}_i$, and debits the account of $\mathcal{U}_i$ by the value of the electronic coin.

$\mathcal{C}_i$ accepts if and only if

$$g_0^{r_0}(h_0 h_i)^{-c_0} = a_0.$$

If this verification holds, $\mathcal{C}_i$ computes $r_0' := r_0 + c_0' x_{0i} + s_1 \bmod q$, and stores a_i, s_3 and the triple (x_{0i}, w_{0i}, w_{1i}), (h_i', a_i'), (c_0', r_0') for later use in the payment protocol.

To withdraw l electronic coins at once, $\mathcal{U}_i$ is allowed to perform l executions of the withdrawal protocol in parallel. However, two *different* $\mathcal{U}_i$'s (who each have a different identification number x_{1i}) are not allowed to perform their executions of the withdrawal protocol fully in parallel; only after $\mathcal{B}$ has received a challenge c_0 of the first user will it send a number a_0 to the second user. In other words, withdrawals by different users can overlap only partially. In Sect. 6 a simple modification is described that is believed to enable executions of the withdrawal protocol to be run in parallel without any such restriction.

4.4 The Payment Protocol

If $\mathcal{C}_i$ accepted the withdrawal, then the electronic coin can be spent by $\mathcal{U}_i$ at a service provider $\mathcal{S}_j$ by performing the following protocol:

Step 1. $\mathcal{C}_i$ computes $d := \mathcal{H}(h_i', \texttt{spec}, a_i')$. $\mathcal{C}_i$ then sends $d' := d + s_3 \bmod q$ to $\mathcal{T}_i$. The format of `spec` is discussed below.

Step 2. If w_i is stored in memory, then $\mathcal{T}_i$ sends $r_i := d' x_{1i} + w_i \bmod q$ to $\mathcal{C}_i$. $\mathcal{T}_i$ then erases w_i from memory.

Step 3. If $g_1^{r_i} h_i^{-d'} = a_i$, then $\mathcal{C}_i$ accepts the response of $\mathcal{T}_i$. $\mathcal{C}_i$ then computes $r_{0i} := d x_{0i} + w_{0i} \bmod q$ and $r_{1i} := r_i + w_{1i} \bmod q$, and sends the triple (h_i', a_i'), (c_0', r_0'), (r_{0i}, r_{1i}) to $\mathcal{S}_j$.

$\mathcal{S}_j$ computes $d := \mathcal{H}(h_i', \texttt{spec}, a_i')$, and accepts if and only if

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i' \quad \text{and} \quad \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'}(h_0 h_i')^{-c_0'}) = c_0'.$$

The number `spec` is a concatenation of several fields, in a format predetermined by $\mathcal{B}$. It comprises a first field that uniquely specifies the account of $\mathcal{S}_j$ with $\mathcal{B}$, and a second field that contains a distinct value for each payment involving $\mathcal{S}_j$. If $\mathcal{S}_j$ uses a value for the second field that it has used before, then $\mathcal{B}$ will reject the corresponding deposit. (Alternatively, $\mathcal{B}$ could require $\mathcal{S}_j$ to use a distinct value for the second field only in case it receives a certified public key that it has received before. It could even require $\mathcal{S}_j$ to reject the payment in that case. Since this approach requires $\mathcal{S}_j$ to compare each payment to earlier ones, and we have to fix a strategy in view of the statements that will be proved in Sect. 5, we do not adopt this strategy here.)

Note that it has implicitly been assumed in the description of the payment protocol that $\mathcal{C}_i$ can determine the appropriate values for `spec` by itself. Alternatively, $\mathcal{S}_j$ could provide $\mathcal{C}_i$ with `spec` or the appropriate value of one of the fields. Since it is of no concern to the payer whether d has been formed correctly, $\mathcal{S}_j$ could even provide $\mathcal{C}_i$ with d. None of these alternative approaches, or a combination of them, makes a difference with respect to the statements that are proved in Sect. 5, and the adopted approach is only for definiteness.

4.5 The Deposit Protocol

If $\mathcal{S}_j$ accepted in the payment protocol, then it can deposit the electronic coin in its account with $\mathcal{B}$ by performing the following protocol:

Step 1. $\mathcal{S}_j$ sends the payment transcript, consisting of (h'_i, a'_i), (c'_0, r'_0), (r_{0i}, r_{1i}) and $\mathtt{spec}$, to $\mathcal{B}$.

Step 2. $\mathcal{B}$ computes $d := \mathcal{H}(h'_i, \mathtt{spec}, a'_i)$, and accepts the payment transcript if and only if $\mathtt{spec}$ has not been used before, and

$$g_0^{r_{0i}} g_1^{r_{1i}} (h'_i)^{-d} = a'_i \quad \text{and} \quad \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0} (h_0 h'_i)^{-c'_0}) = c'_0 .$$

If $\mathcal{B}$ accepts the payment transcript, it credits the account that is specified by the value of the first field of $\mathtt{spec}$ by the value of the electronic coin.

$\mathcal{B}$ accepts if and only if it accepts the payment transcript, and (h'_i, a'_i), (c'_0, r'_0) is not already in the deposit database (as part of some other payment transcript). If $\mathcal{B}$ accepts, it stores (h'_i, a'_i), (c'_0, r'_0), $(\mathtt{spec}, r_{1i})$ in its deposit database.

Note that we have defined two different notions of acceptance for $\mathcal{B}$. Furthermore, a payment transcript encompasses a certified public key, which makes sense because we identified each electronic coin with a unique certified public key.

4.6 Tracing a Double-Spender

In case $\mathcal{B}$ accepted the payment transcript but did not accept, the deposited electronic coin must have been double-spent (implying that the tamper-resistance of at least one device has been compromised). $\mathcal{B}$ then proceeds as follows. Using the pair $(\mathtt{spec}^*, r^*_{1i})$ for the payment transcript that is already in the deposit database, it computes $(r_{1i} - r^*_{1i})/(d - d^*) \bmod q$, where $d^* := \mathcal{H}(h'_i, \mathtt{spec}^*, a'_i)$. It then searches its account database for an entry that contains this identification number; the identity description in the resulting entry reveals the user that is responsible for the fraud.

5 Correctness of the System

In this section we will assess the correctness of the presented cash system. The presentation is divided into three parts: completeness, privacy of payments, and security.

5.1 Completeness

Following Feige, Fiat and Shamir [21], we denote by a $\overline{\mathcal{Z}}$ a party $\mathcal{Z}$ that follows the protocols, and by $\widetilde{\mathcal{Z}}$ a party $\mathcal{Z}$ with unlimited computing power that may deviate from the protocols in an arbitrary way.

The following result states that withdrawn electronic coins can be spent, if only the (computer of the) user follows the protocols.

Proposition 1. *If $\overline{\mathcal{C}_i}$ accepts in the withdrawal protocol, and accepts the response of $\widetilde{\mathcal{T}_i}$ in the payment protocol, then $\overline{\mathcal{S}_j}$ accepts in the payment protocol.*

Proof. $\overline{\mathcal{S}_j}$ accepts if

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i' \quad \text{and} \quad \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} (h_0 h_i')^{-c_0'}) = c_0',$$

where $d := \mathcal{H}(h_i', \texttt{spec}, a_i')$. In Step 2 of the withdrawal protocol, $\overline{\mathcal{C}_i}$ computes

$$a_i' := g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} a_i \quad \text{and} \quad c_0' := \mathcal{H}(h_0 h_i', a_i', g_0^{s_1} (h_0 h_i)^{s_2} a_0).$$

Therefore it suffices to prove that

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} a_i \quad \text{and} \quad g_0^{r_0'} (h_0 h_i')^{-c_0'} = g_0^{s_1} (h_0 h_i)^{s_2} a_0$$

for the assignments made by $\overline{\mathcal{C}_i}$ in the protocols. The first equality follows from

$$\begin{aligned} g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} &= g_0^{r_{0i}} g_1^{r_{1i}} (g_0^{x_{0i}} h_i)^{-d} \\ &= g_0^{r_{0i} - d x_{0i}} g_1^{r_{1i}} h_i^{-d} \\ &= g_0^{w_{0i}} g_1^{r_{1i}} h_i^{-(d' - s_3)} \\ &= g_0^{w_{0i}} g_1^{r_i + w_{1i}} h_i^{s_3 - d'} \\ &= g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} g_1^{r_i} h_i^{-d'} \\ &\overset{(\star)}{=} g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} a_i, \end{aligned}$$

and the second from

$$\begin{aligned} g_0^{r_0'} (h_0 h_i')^{-c_0'} &= g_0^{r_0 + c_0' x_{0i} + s_1} (h_0 g_0^{x_{0i}} h_i)^{-c_0'} \\ &= g_0^{r_0 + s_1} (h_0 h_i)^{-c_0'} \\ &= g_0^{s_1} g_0^{r_0} (h_0 h_i)^{s_2 - c_0} \\ &= g_0^{s_1} (h_0 h_i)^{s_2} g_0^{r_0} (h_0 h_i)^{-c_0} \\ &\overset{(\star\star)}{=} g_0^{s_1} (h_0 h_i)^{s_2} a_0. \end{aligned}$$

The substitution in $(\star)$ is allowed because $\overline{\mathcal{C}_i}$ accepts the response of $\widetilde{\mathcal{T}_i}$ in the payment protocol only if $g_0^{r_i} h_i^{-d'} = a_i$, and that in $(\star\star)$ is allowed because $\overline{\mathcal{C}_i}$ accepts in the withdrawal protocol only if $g_0^{r_0} (h_0 h_i)^{-c_0} = a_0$. □

Note that this completeness result holds for all $\widetilde{\mathcal{B}}$ and $\widetilde{\mathcal{T}_i}$, not only for $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$.

Our next result shows that the service provider can deposit the electronic coins that it is being paid with.

Proposition 2. *If $\overline{\mathcal{S}_j}$ accepts in the payment protocol, and deposits the payment transcript in the deposit protocol, then $\overline{\mathcal{B}}$ accepts.*

Proof. This is immediately clear from the fact that $\overline{\mathcal{S}_j}$ does not use the same value for the second field in two different payments, since the verification relations that are applied by $\overline{\mathcal{B}}$ in the deposit protocol are the same as those applied by $\overline{\mathcal{S}_j}$ in the payment protocol.

5.2 Privacy

We will now investigate in what sense the privacy of honest payers is guaranteed.

Lemma 3. *For any $\overline{\mathcal{C}_i}$, for any possible view of $\widetilde{\mathcal{B}}$ in an execution of the withdrawal protocol in which $\overline{\mathcal{C}_i}$ accepts, for any possible view of $\widetilde{\mathcal{S}_j}$ in an execution of the payment protocol in which the computer controlled by the other party follows the protocol, and for any possible view of $\widetilde{\mathcal{T}_i}$ in (i) the execution of a withdrawal protocol in which $\overline{\mathcal{C}_i}$ accepts and (ii) a corresponding execution of the payment protocol in which $\overline{\mathcal{C}_i}$ accepts the response of $\widetilde{\mathcal{T}_i}$, there is exactly one set of random choices that $\overline{\mathcal{C}_i}$ could have made in the execution of the withdrawal protocol such that the views of $\widetilde{\mathcal{B}}$, $\widetilde{\mathcal{S}_j}$ and $\widetilde{\mathcal{T}_i}$ correspond to the withdrawal and payment of the same electronic coin.*

Proof. We first consider the relations that must be satisfied by definition. The response r_0 of $\widetilde{\mathcal{B}}$ in the withdrawal is such that $g_0^{r_0}(h_0h_i)^{-c_0} = a_0$, since $\overline{\mathcal{C}_i}$ accepts in the withdrawal protocol. By Proposition 1, we can assume that the relations $g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i'$ and $\mathcal{H}(h_0h_i', a_i', g_0^{r_0'}(h_0h_i')^{-c_0'}) = c_0'$ are satisfied in all views of $\widetilde{\mathcal{S}_j}$ in executions of the payment protocol with a party whose computer follows the payment protocol. Since $\overline{\mathcal{C}_i}$ accepts the response of $\widetilde{\mathcal{T}_i}$ in the payment protocol, it must be that $g_1^{r_i} h_i^{-d'} = a_i$.

We correspondingly define the following sets:

$$
\begin{aligned}
\mathrm{Views}(\widetilde{\mathcal{B}}) &= \{(a_0, c_0, r_0) \mid a_0 \in G_q \text{ and } c_0, r_0 \in \mathbb{Z}_q \text{ such that} \\
&\qquad g_0^{r_0}(h_0h_i)^{-c_0} = a_0\}, \\
\mathrm{Views}(\widetilde{\mathcal{S}_j}) &= \{(h_i', a_i', c_0', r_0', r_{1i}, r_{0i}, d) \mid h_i', a_i' \in G_q,\ \ c_0', r_0', r_{1i}, r_{0i} \in \mathbb{Z}_q \\
&\qquad \text{and } d \in \mathbb{Z}_{2^n} \text{ such that } g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i' \text{ and} \\
&\qquad \mathcal{H}(h_0h_i', a_i', g_0^{r_0'}(h_0h_i')^{-c_0'}) = c_0'\}, \\
\mathrm{Views}(\widetilde{\mathcal{T}_i}) &= \{(a_i, d', r_i) \mid a_i \in G_q \text{ and } d', r_i \in \mathbb{Z}_q \text{ such that } g_1^{r_i} h_i^{-d'} = a_i\}, \\
\mathrm{Choices}(\mathcal{C}_i) &= \{(x_{0i}, w_{0i}, w_{1i}, s_1, s_2, s_3) \mid x_{0i}, w_{0i}, w_{1i}, s_1, s_2, s_3 \in \mathbb{Z}_q\}.
\end{aligned}
$$

We will show that for all $\widetilde{\mathcal{B}}$-view $\in \mathrm{Views}(\widetilde{\mathcal{B}})$, for all $\widetilde{\mathcal{S}_j}$-view $\in \mathrm{Views}(\widetilde{\mathcal{S}_j})$, and for all $\widetilde{\mathcal{T}_i}$-view $\in \mathrm{Views}(\widetilde{\mathcal{T}_i})$, there is exactly one tuple $(x_{0i}, w_{0i}, w_{1i}, s_1, s_2, s_3) \in$

Choices($\mathcal{C}_i$) such that $\widetilde{\mathcal{B}}$-view, $\widetilde{\mathcal{S}_j}$-view, and $\widetilde{\mathcal{T}_i}$-view correspond to the withdrawal and payment of the *same* electronic coin. We must take into account that $\widetilde{\mathcal{B}}$ can make smart choices for its public key $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$ and for h_i.

Suppose that $\widetilde{\mathcal{B}}$-view, $\widetilde{\mathcal{S}_j}$-view, and $\widetilde{\mathcal{T}_i}$-view correspond to the withdrawal and payment of the same electronic coin. We will successively determine uniquely the numbers $x_{0i}, w_{0i}, w_{1i}, s_1, s_2, s_3$ that must have been chosen by $\overline{\mathcal{C}_i}$. First, x_{0i} is uniquely determined from h_i, h_i' as $x_{0i} = \log_{g_0}(h_i'/h_i)$. Note that x_{0i} exists and is uniquely defined, since q is prime and hence $g_0 \neq 1$ is a generator of G_q. From r_{0i}, d, and x_{0i} we see that the choice $w_{0i} = r_{0i} - dx_{0i} \bmod q$ must have been made, and from r_{1i}, r_i it follows that $w_{1i} = r_{1i} - r_i \bmod q$ must have been chosen. The choice for x_{0i} together with r_0, r_0' and c_0' determines s_1 as $s_1 = r_0' - r_0 - c_0' x_{0i} \bmod q$, and s_2 is uniquely determined from c_0, c_0' as $s_2 = c_0 - c_0' \bmod q$. Finally, the numbers d, d' determine s_3 as $s_3 = d' - d \bmod q$. Note that each of $w_{0i}, w_{1i}, s_1, s_2, s_3$ exists and is uniquely defined, because $\mathbb{Z}_q$ is a field.

For these choices of the six variables all the assignments and verifications in the two protocol executions would be satisfied by definition, except maybe for the assignments $a_i' := g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} a_i$ and $c_0' := \mathcal{H}(h_0 h_i', a_i', g_0^{s_1}(h_0 h_i)^{s_2} a_0)$ that must have been made by $\overline{\mathcal{C}_i}$ in the withdrawal protocol. To prove that these assignments hold as well, we notice that from $\widetilde{\mathcal{S}_j}$-view $\in$ Views($\widetilde{\mathcal{S}_j}$) we have that

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i' \quad \text{and} \quad \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'}(h_0 h_i')^{-c_0'}) = c_0'.$$

Therefore, the proof is completed if

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = g_0^{w_{0i}} g_1^{w_{1i}} h_i^{s_3} a_i \quad \text{and} \quad g_0^{r_0'}(h_0 h_i')^{-c_0'} = g_0^{s_1}(h_0 h_i)^{s_2} a_0$$

for the choices for $x_{0i}, w_{0i}, w_{1i}, s_1, s_2$ and s_3 made above. This can be derived exactly as in the proof of Proposition 1, considering that in this case the substitution in ($\star$) is allowed because $\widetilde{\mathcal{T}_i}$-view $\in$ Views($\widetilde{\mathcal{T}_i}$), and ($\star\star$) because $\widetilde{\mathcal{B}}$-view $\in$ Views($\widetilde{\mathcal{B}}$). $\square$

Proposition 4. *If $\mathcal{C}_i$ follows the protocols, and does not double-spend, then no shared information can be developed between $\mathcal{B}$, $\mathcal{T}_i$, and all service providers $\mathcal{S}_j$ in the executions of the withdrawal and payment protocols that $\mathcal{C}_i$ takes part in.*

Proof. This is an immediate consequence of Lemma 3 and the fact that $\overline{\mathcal{C}_i}$ in the withdrawal protocol generates tuples $(x_{0i}, w_{0i}, w_{1i}, s_1, s_2, s_3)$ uniformly at random from Choices($\mathcal{C}_i$). $\square$

In other words, the privacy of the payments of honest users is protected with respect to the most stringent criterion conceivable. (See Cramer and Pedersen [15] for the definition of shared information.)

5.3 Security

We first will state the assumptions that underly the security of the cash system, and investigate their plausibility.

Although in principle it is easier to forge Schnorr signatures in the blind Schnorr signature scheme than in the non-interactive scheme, since m can be chosen depending on a and the challenge can be freely chosen, it is generally believed to be unforgeable. This motivates our first assumption.

Assumption I. *For any $l \geq 0$, no probabilistic polynomial-time verifier can determine with non-negligible probability of success $l+1$ distinct pairs, consisting of a message and a corresponding Schnorr signature, by performing l executions of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$.*

A result of Chen, Damgard and Pedersen [14] provides some additional evidence in support of this assumption (albeit that the attack that they rule out only relates to sequential executions of witness hiding proofs of knowledge).

We can distinguish in Assumption I between sequential and parallel executions of the blind Schnorr signature issuing protocol; it is not impossible, although highly unlikely, that the sequential version is secure whereas the parallel version is not. For the purpose of Propositions 10, 14 and 15 we will allow the attacker in Assumption I to run executions of the protocol in parallel.

To introduce our second assumption, we consider the following modification of the Schnorr identification scheme. Instead of using a fixed, randomly chosen h, $\mathcal{P}$ generates h at random in each execution of the protocol, and transfers it along with a in Step 1. We can consider h, or (h, a) for that matter, as a one-time public key of $\mathcal{P}$. The proof of soundness for the Schnorr identification scheme clearly applies also to the modified protocol. In particular, if $\mathcal{P}$ can compute correct responses with respect to two different challenges of $\mathcal{V}$, then $\mathcal{P}$ "knows" $\log_g h$. We again apply the general technique of [22], in order to convert this modified protocol into a signature issuing protocol; this time we must correspondingly take $c := \mathcal{H}(h, m, a)$. The applicability of the general technique suggests that it should be infeasible to find triples $h, m, (c, r)$ such that $c = \mathcal{H}(h, m, g^r h^{-c})$. We now state our second assumption:

Assumption II. *There exists a probabilistic polynomial-time Turing machine M (the knowledge extractor) such that for any probabilistic polynomial-time Turing machine A, any initial contents of A's work tape WT and random tape RT, and any sufficiently large k, if A on input (q, g) outputs with nonnegligible probability of success a triple $h, m, (c, r)$ such that*

$$c = \mathcal{H}(h, m, g^r h^{-c}),$$

then $M(A, WT, RT, (g, q)) = \log_g h$ with overwhelming probability.

Since h can be chosen by A, this assumption is at least as strong as the assumption that matching the verification relation for the Schnorr signature scheme is infeasible without knowledge of the secret key: there may be negligibly many known values of h for which forgery is easy, while the probability that such an h is chosen as the public key in the Schnorr signature scheme is negligible. However, the fact that we arrived at the modified scheme by applying the technique of [22] to a sound proof of knowledge that is very similar to the Schnorr identification scheme suggests that there must be a fundamental problem with this technique in case Assumption II turns out to be false. Moreover, the attacker in Assumption II only is allowed to attempt to "forge" triples *from scratch*. In light of this, Assumption II certainly seems very plausible. Furthermore, it seems that the security statements that are based on Assumption II still hold if the knowledge extractor can only extract a "non-trivial" part of $\log_g h$ (*e.g.*, half of the bits), although I have not been able to come up with corresponding proofs.

To show that the system is secure for the bank, three *key properties* have to be proved. Informally, these are:

1. No conspiracy can forge a payment transcript.
2. If the bank discovers that a coin has been double-spent, then its procedure for tracing a double-spender results in the identification number of a member of the conspiracy that has committed the fraud (second line of defense).
3. No conspiracy can double-spend a coin without first having to *physically* extract the secret key of at least one tamper-resistant device (first line of defense).

Once these three key properties have been shown to hold, other properties (*e.g.*, the infeasibility to deposit a wire-tapped payment transcript to another account, or to spend a wire-tapped coin of another party) can easily be proved. For this reason, we will focus on proving the key properties.

In all the statements that we will prove, the bank is assumed to be honest. This implies that all the tamper-resistant devices follow the protocols as well. Almost all our proofs involve the construction of a simulator that interacts with a particular prover, in order to be able to respond to all possible challenges of a conspiracy (with the probability distribution that applies when the honest bank is involved, regardless of the strategy that is being followed by the conspiracy); the result of the attack of the conspiracy can then be shown to contradict some appropriate initial assumption on the amount of information that can be retrieved from the prover (*e.g.*, Assumptions I and II). Of course, in the situation where a conspiracy can *physically* extract the secret key of a tamper-resistant device of one of its members, it in fact makes no sense to simulate it, but we will not make this distinction in the description of the simulators. We will furthermore describe only the vital parts of each simulator. For instance, to simulate the opening of an account, we will only describe the information that the bank makes available to the party that opens an account. Likewise, we will not explicitly describe the simulation of the payment and deposit protocols in cases in which it is obvious how to do this.

In most proofs we will construct a simulator A that moves to a third step only after l executions of the withdrawal protocol (and payment protocol, on occasion) have been simulated. In these cases, there may be a possibility that all the users in the system stop to perform the withdrawal protocol before l executions have been performed. To ensure that A always halts in polynomial time we can of course let A halt if some time has expired without any requests for executions of the withdrawal protocol. For clarity in exposition, we will not introduce a notion of *time*, and instead leave it to the reader to fill in this detail. The presence of such a mechanism will be assumed implicitly.

There are some subtle issues involved in defining the notion of a conspiracy. Without going into further detail on this matter, we will adopt the following definition. We divide the set of all users and service providers in the system into two subsets. One subset contains all the users and service providers that *always* follow the protocols, and the complementary subset is called the conspiracy. A conspiracy can be viewed as one probabilistic polynomial-time algorithm, composed of its members. Note that the conspiracy does not necessarily have to consist of parties that cooperate; every party that, at one time or another, deviates from an execution of the protocol, is a member of the conspiracy. We will furthermore assume at all times that *a conspiracy can physically extract the secret keys of the tamper-resistant devices of its members* (unless explicitly indicated otherwise).

We will also prove some results with respect to a conspiracy that has only one user as its members. More specifically, we allow the conspiracy to only make use of the (views in the) protocol executions of one user. In other words, such a conspiracy cannot "wire-tap" into the protocol executions of other users, or cooperate with other users. The results of these proofs provide good evidence that the particular attacks that they exclude are infeasible *even* when applied by a general conspiracy. The reason for this is that *different* users in the cash system are not allowed to run their executions of the (withdrawal) protocol in parallel, and so a general conspiracy hardly has any advantage over the conspiracy that has a limited view; executions of the withdrawal protocol with respect to one account can only benefit from executions with respect to other accounts *that have already been completed*, such as previously retrieved triples that meet a certain verification relation. In effect, there is no co-operation possible between different users while performing executions of the withdrawal protocol. (See Sect. 6 for a good example of the added power that comes from the ability of different users to cooperate in determining their challenges in the withdrawal protocol.) The conspiracy with a limited view will be denoted by $\widehat{\mathcal{U}_i}$, to emphasize that it can only make use of the protocol executions of $\mathcal{U}_i$.

In all statements, the involved probabilities are taken over the coin tosses of the conspiracy, over the public keys of $\overline{\mathcal{B}}$ and (all) $\overline{\mathcal{T}_i}$, and their coin tosses in the protocols. In case the security statements apply to *any* conspiracy, the probabilities are also taken over the coin tosses of the *honest* parties in the system, since their views might be of help to a conspiracy.

We start with two lemmas, in order to prove our first proposition.

Lemma 5. *If the blind Schnorr signature scheme is witness hiding, then no conspiracy can compute* $\log_{g_0} g_1$ *with non-negligible probability of success.*

Proof. Suppose that a conspiracy can misuse any l executions of the withdrawal protocol to extract with non-negligible probability of success $\log_{g_0} g_1$. We will construct a polynomial-time algorithm A for extracting the witness of the Prover $\overline{\mathcal{P}}$ in the blind Schnorr signature scheme.

Algorithm A, on input a random public key $(q, g, h, \mathcal{H}(\cdot))$ of $\overline{\mathcal{P}}$ (*i.e.*, as generated by the key generated algorithm for the blind Schnorr signature scheme), performs the following steps:

Step 1. (Simulate the initial key generation.) Set $g_0 := g$ and $g_1 := h$. Generate at random an element $x_{00} \in \mathbb{Z}_q$, and compute $h_0 := g_0^{x_{00}}$. The simulated public key of $\overline{\mathcal{B}}$ is $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$.

Step 2. For each party in the cash system, simulate the actions that $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ would perform. For a user $\mathcal{U}_i$, perform hereto the simulation as follows:

- (Opening an account.) Generate at random a secret key $x_{1i} \in \mathbb{Z}_q$ for $\mathcal{T}_i$, and the corresponding public key $h_i := g_1^{x_{1i}}$.
- (The withdrawal protocol.)
  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Generate at random a number $w_i \in \mathbb{Z}_q$, and send $a_i := g_1^{w_i}$ to $\mathcal{U}_i$.
  Step 2. [Simulate $\overline{\mathcal{B}}$.] Receive a from $\overline{\mathcal{P}}$, and pass $a_0 := a^{x_{1i}}$ on to $\mathcal{U}_i$.
  Step 3. [Simulate $\overline{\mathcal{B}}$.] Receive c_0 from $\mathcal{U}_i$, and pass $c := c_0$ on to $\overline{\mathcal{P}}$.
  Step 4. [Simulate $\overline{\mathcal{B}}$.] Receive r from $\overline{\mathcal{P}}$, and pass $r_0 := x_{1i}r + c_0 x_{00} \bmod q$ on to $\mathcal{U}_i$.
- (The payment protocol.) This can be handled as in the description of the cash system, since x_{1i} is known.

Continue this simulation until l executions of the withdrawal protocol have been performed.

Step 3. Check if the conspiracy has $\log_{g_0} g_1$ on its tapes. If not, then halt.

Step 4. Output $\log_{g_0} g_1$.

By definition of the key generation of $\overline{\mathcal{P}}$, and that of A, the public key in Step 1 is simulated with the same probability distribution as that by which $\overline{\mathcal{B}}$ generates its public key. Likewise, the key generation for $\overline{\mathcal{T}_i}$ is performed with the same probability distribution. The response that is computed by A in the simulated withdrawal protocol is the same as the response that $\overline{\mathcal{B}}$ would compute:

$$\begin{aligned}
g_0^{r_0} &= g_0^{x_{1i}r + c_0 x_{00}} \\
&= (g_0^r)^{x_{1i}} (g_0^{x_{00}})^{c_0} \\
&= (g^r)^{x_{1i}} h_0^{c_0} \\
&\overset{(\star)}{=} (h^c a)^{x_{1i}} h_0^{c_0} \\
&= g_1^{x_{1i} c_0} a_0 h_0^{c_0} \\
&= (h_0 g_1^{x_{1i}})^{c_0} a_0 \\
&= (h_0 h_i)^{c_0} a_0,
\end{aligned}$$

where the substitution in $(\star)$ is allowed because the response of $\overline{\mathcal{P}}$ in the blind Schnorr signature issuing protocol is always correct. From this it easily follows that the views provided by A are the same as those provided by $\overline{\mathcal{B}}$ in the real cash system (regardless of the probability distribution by which the conspiracy generates its challenges). This obviously holds also for the simulated executions of the payment protocol, and Step 1 of the simulated executions of the withdrawal protocol. Note that A provides for the possibility of parties to physically extract the secret key of $\mathcal{T}_i$, because it generates x_{1i} for all users by itself. Hence, Step 4 is reached by supposition with non-negligible probability.

To complete the proof, observe that an execution of each of Steps 2, 3 and 4 of the simulated withdrawal protocol constitutes exactly one execution of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$. For the output of A in Step 4 we have $\log_{g_0} g_1 = \log_g h$, which is the witness of $\overline{\mathcal{P}}$. Since A performs only polynomially many executions of the protocol with $\overline{\mathcal{P}}$, this contradicts the assumption that the blind Schnorr signature issuing protocol is witness hiding. □

The proof of the following lemma is trivial, and is therefore omitted.

Lemma 6. *If Assumption I is true, then the blind Schnorr signature scheme is witness hiding.*

Definition 7. A conspiracy is said to be able to *forge a certified key pair* if it can compute with non-negligible probability of success $l+1$ distinct certified key pairs by performing l executions of the withdrawal protocol with $\overline{\mathcal{B}}$, for some $l \geq 0$.

Proposition 8. *If Assumption I is true, then no conspiracy can forge a certified key pair.*

Proof. Suppose that a conspiracy can misuse any l executions of the withdrawal protocol to extract with non-negligible probability of success l^* *distinct* certified key pairs, with $l^* > l$. We will construct a polynomial-time algorithm A for breaking Assumption I.

Algorithm A, on input a random public key $(q, g, h, \mathcal{H}(\cdot))$ of $\overline{\mathcal{P}}$, performs the following steps:

Step 1. (Simulate the initial key generation.) Set $g_0 := g$ and $h_0 := h$. Generate at random an element $x_{10} \in \mathbb{Z}_q$, and compute $g_1 := g_0^{x_{10}}$. The simulated public key of $\overline{\mathcal{B}}$ is $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$.

Step 2. For each party in the cash system, simulate the actions that $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ would perform. For a user $\mathcal{U}_i$, perform hereto the simulation as follows:

- (Opening an account.) Generate at random a secret key $x_{1i} \in \mathbb{Z}_q$ for $\mathcal{T}_i$, and the corresponding public key $h_i := g_1^{x_{1i}}$.
- (The withdrawal protocol.)
  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Generate at random a number $w_i \in \mathbb{Z}_q$, and send $a_i := g_1^{w_i}$ to $\mathcal{U}_i$.
  Step 2. [Simulate $\overline{\mathcal{B}}$.] Receive a from $\overline{\mathcal{P}}$, and pass $a_0 := a$ on to $\mathcal{U}_i$.
  Step 3. [Simulate $\overline{\mathcal{B}}$.] Receive c_0 from $\mathcal{U}_i$, and pass $c = c_0$ on to $\overline{\mathcal{P}}$.

Step 4. [Simulate $\overline{\mathcal{B}}$.] Receive r from $\overline{\mathcal{P}}$, and pass $r_0 := r + c_0 x_{10} x_{1i} \bmod q$ on to $\mathcal{U}_i$.

– (The payment protocol.) This can be handled as in the description of the cash system, since x_{1i} is known.

Continue this simulation until l executions of the withdrawal protocol have been performed.

Step 3. Check if the conspiracy has l^* distinct certified key pairs on its tapes. If not, then halt.

Step 4. For each of the l^* distinct certified key pairs, $((x'_{0i}, x'_{1i}), (w'_{0i}, w'_{1i})), (h'_i, a'_i), (c'_0, r'_0)$, compute $m' := (h_0 h'_i, a'_i)$, $c' := c'_0$ and $r' := r'_0 - c'_0(x'_{0i} + x_{10} x'_{1i}) \bmod q$, and output $m', (c', r')$.

By definition of the key generation of $\overline{\mathcal{P}}$, and that of A in Step 1, the public key in Step 1 is simulated with the same probability distribution as that by which $\overline{\mathcal{B}}$ generates its public key. Likewise, the key generation for $\overline{\mathcal{T}_i}$ is performed with the same probability distribution. The response that is computed by A in the simulated withdrawal protocol is the same as the response that $\overline{\mathcal{B}}$ would compute:

$$\begin{aligned} g_0^{r_0} &= g_0^{r + c_0 x_{10} x_{1i}} \\ &= g^r (g_0^{x_{10}})^{c_0 x_{1i}} \\ &\overset{(\star)}{=} (h^c a)(g_1^{x_{1i}})^{c_0} \\ &= h_0^{c_0} a_0 h_i^{c_0} \\ &= (h_0 h_i)^{c_0} a_0, \end{aligned}$$

where the substitution in $(\star)$ is allowed because the response of $\overline{\mathcal{P}}$ in the blind Schnorr signature issuing protocol is always correct. From this it easily follows that the views provided by A are the same as those provided by $\overline{\mathcal{B}}$ in the real cash system (regardless of the probability distribution by which the conspiracy generates its challenges). This obviously holds also for the simulated executions of the payment protocol, and Step 1 of the simulated executions of the withdrawal protocol. Note that A provides for the possibility of parties to physically extract the secret key of $\mathcal{T}_i$, because it generates x_{1i} for all users by itself. Hence, Step 4 is reached by supposition with non-negligible probability.

We next show (i) that the output of A consists of l^* messages with corresponding Schnorr signatures, and (ii) that all these pairs are distinct with at least overwhelming probability. Property (i) follows from

$$\begin{aligned} c' &= c'_0 \\ &\overset{(\star)}{=} \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 h'_i)^{-c'_0}) \\ &= \mathcal{H}(m', g_0^{r' + c'_0(x'_{0i} + x_{10} x'_{1i})}(h_0 g_0^{x'_{0i}} g_0^{x_{10} x'_{1i}})^{-c'_0}) \\ &= \mathcal{H}(m', g_0^{r'} h_0^{-c'_0}) \\ &= \mathcal{H}(m', g^{r'} h^{-c'}), \end{aligned}$$

where the substitution in $(\star)$ is allowed by definition of a certified public key.

To prove property (ii), consider any two certified key pairs,

$$((x_{0i}, x_{1i}), (w_{0i}, w_{1i})), (h_i, a_i), (c_0, r_0)$$

and

$$((x^*_{0i}, x^*_{1i}), (w^*_{0i}, w^*_{1i})), (h^*_i, a^*_i), (c^*_0, r^*_0).$$

Suppose that the two corresponding pairs, as computed by A in Step 4, are identical. Denoting $(h_0 h_i, a_i)$ by m, and $(h_0 h^*_i, a^*_i)$ by m^*, the two pairs are m, $(c_0, r_0 - c_0(x_{0i} + x_{10}x_{1i}) \bmod q)$ and m^*, $(c^*_0, r^*_0 - c^*_0(x^*_{0i} + x_{10}x^*_{1i}) \bmod q)$. We will prove that if these two pairs are identical, then the two certified key pairs are identical. Applying the definition of a certified public key to the two certified key pairs, we have

$$c_0 = \mathcal{H}(h_0 h_i, a_i, g_0^{r_0}(h_0 h_i)^{-c_0})$$

and

$$c^*_0 = \mathcal{H}(h_0 h^*_i, a^*_i, g_0^{r^*_0}(h_0 h^*_i)^{-c^*_0}).$$

From $m = m^*$ it follows that $h_i = h^*_i$ and $a_i = a^*_i$. Since we also have $c_0 = c^*_0 \bmod q$ by equality of the two corresponding pairs, it follows that

$$\mathcal{H}(h_0 h_i, a_i, g_0^{r_0}(h_0 h_i)^{-c_0}) = \mathcal{H}(h_0 h_i, a_i, g_0^{r^*_0}(h_0 h_i)^{-c_0}).$$

Because $\mathcal{H}(\cdot)$ is collision-intractable, $g_0^{r_0}(h_0 h_i)^{-c_0} = g_0^{r^*_0}(h_0 h_i)^{-c_0}$ with overwhelming probability, and hence $r_0 = r^*_0 \bmod q$ with overwhelming probability. This leaves us with the possibility that $((x_{0i}, x_{1i}), (w_{0i}, w_{1i}))$ differs from $((x^*_{0i}, x^*_{1i}), (w^*_{0i}, w^*_{1i}))$. Suppose that $x_{0i} \neq x^*_{0i} \bmod q$ (the other possibilities are taken care of in exactly the same way). Then from $g_0^{x_{0i}} g_1^{x_{1i}} = h_i = h^*_i = g_0^{x^*_{0i}} g_1^{x^*_{1i}}$ it follows that

$$g_0 = g_1^{(x^*_{1i} - x_{1i})/(x_{0i} - x^*_{0i})},$$

and so $\log_{g_0} g_1 = (x^*_{1i} - x_{1i})/(x_{0i} - x^*_{0i}) \bmod q$. Since the certified key pairs in Step 4 are known by the conspiracy, while its view in the simulation is exactly the same as in the real cash system, this means that the conspiracy has been able to determine $\log_{g_0} g_1$. According to Lemma 6, the blind Schnorr signature scheme is witness hiding if Assumption I is true, and so by Lemma 5 we have a contradiction with Assumption I. So the certified key pairs are equal, and hence property (ii) holds.

To complete the proof, observe that an execution of each of Steps 2, 3 and 4 of the simulated withdrawal protocol constitutes exactly one execution of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$. Because of this one-to-one correspondence, A performs in total l executions of the blind Schnorr signature issuing protocol. This contradicts Assumption I. □

In combination with Proposition 1, which shows that an honest user receives a certified key pair when it performs an execution of the withdrawal protocol, this result tells us that there is a one-to-one correspondence between executions of the withdrawal protocol and certified key pairs.

Definition 9. A conspiracy is said to be able to *forge a payment transcript* if it can compute with non-negligible probability of success $l+1$ payment transcripts, the certified public keys of which are all distinct, by performing l executions of the withdrawal protocol with $\overline{\mathcal{B}}$, for some $l \geq 0$.

Note that a payment transcript is not considered to be forged if it encompasses a certified public key that is part of another payment transcript; as described in Sect. 2 each electronic coin is uniquely associated with a unique certified public key. The ability to output $l+1$ different payment transcripts that encompass the *same* certified public key, after having executed l executions of the withdrawal protocol, is captured by the notion of *double-spending*; the infeasibility to double-spend without physically compromising a tamper-resistant device (breaking the first line of defense) is covered by our assessment of the third key property, and the infeasibility to double-spend an electronic coin (after having physically extracted the secret key of a tamper-resistant device) without being traceable (breaking the second line of defense) is covered by our assessment of the second key property.

The result of Proposition 8 brings us a long way towards a proof of the first key property. The design of a clean-cut reduction to prove this key property unfortunately turns out to require a rather cumbersome formulation of a third plausible intractability assumption, which moreover is not needed in any of our other results. So we will instead only sketch here why the first key property should hold, on the basis of the result of the previous proposition. Consider a conspiracy that has forged a payment transcript (h'_i, a'_i), (c'_0, r'_0), (r_{0i}, r_{1i}) and $\mathtt{spec}$, and let $d = \mathcal{H}(h'_i, \mathtt{spec}, a'_i)$. Since $\overline{\mathcal{B}}$ accepts the payment transcript, the relations $g_0^{r_{0i}} g_1^{r_{1i}} (h'_i)^{-d} = a'_i$ and $\mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 h'_i)^{-c'_0}) = c'_0$ hold. From this, it seems that we may conclude that the conspiracy must know a pair (e_0, e_1) such that $h'_i = g_0^{e_0} g_1^{e_1}$. It follows that $a'_i = g_0^{r_{0i} - e_0 d} g_1^{r_{1i} - e_1 d}$, and so the triple $((e_0, e_1), (r_{0i} - e_0 d, r_{1i} - e_1 d))$, (h'_i, a'_i), (c'_0, r'_0) is a certified key pair. Since the payment transcript has been forged, the certified public key of this certified key pair differs from those extracted in executions of the withdrawal protocol. In other words, the forgery of a payment transcript seems to imply the forgery of a certified key pair, which contradicts Proposition 8.

This settles our assessment of the first key property. We now turn to the second key property.

The following proposition states that the identification number x_{1i} is invariant under all blinding operations that $\widehat{\mathcal{U}}_i$ can perform in the withdrawal protocol. The fact that we can prove this result under a rather weak assumption is perhaps the best demonstration of the power of the secret-key certificate technique. It is not a complete proof of the second key property, but certainly goes a long way.

Proposition 10. *If the Discrete Log assumption and Assumption II are true, then $\widehat{\mathcal{U}}_i$ cannot withdraw with non-negligible probability of success a certified key pair $((x_{0i}, x^*_{1i}), (w_{0i}, w_{1i}))$, (h_i, a_i), (c_0, r_0) for which x^*_{1i} differs from his identification number x_{1i}.*

Proof. Suppose that $\widehat{\mathcal{U}_i}$ can misuse l executions of the withdrawal protocol to extract with non-negligible probability of success a certified key pair such that $x^*_{1i} \neq x_{1i} \bmod q$. We will construct a polynomial-time algorithm A for computing discrete logarithms in G_q.

Algorithm A, on input a random triple (q, g, h), performs the following steps:

Step 1. (Simulate the initial key generation.) Set $g_0 := g$ and $g_1 := h$. Generate at random an element $x_{00} \in \mathbb{Z}_q$ and an element $x_{1i} \in \mathbb{Z}_q$, and compute $h_0 := g_0^{x_{00}} g_1^{-x_{1i}}$. Generate $\mathcal{H}(\cdot)$ in the same way as described in the key generation for the Schnorr signature scheme. The simulated public key of $\overline{\mathcal{B}}$ is $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$.

Step 2. Simulate for $\widehat{\mathcal{U}_i}$ the actions that $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ would perform, as follows:

- (Opening an account.) Use x_{1i} as the secret key for $\mathcal{T}_i$, and compute the corresponding public key as $h_i := g_1^{x_{1i}}$.
- (The withdrawal protocol.)
  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Generate at random a number $w_i \in \mathbb{Z}_q$, and send $a_i := g_1^{w_i}$ to $\widehat{\mathcal{U}_i}$.
  Step 2. [Simulate $\overline{\mathcal{B}}$.] Generate at random an element w_0 of $\mathbb{Z}_q$. Compute $a_0 := g_0^{w_0}$, and send a_0 to $\widehat{\mathcal{U}_i}$.
  Step 3. [Simulate $\overline{\mathcal{B}}$.] Receive c_0 from $\widehat{\mathcal{U}_i}$.
  Step 4. [Simulate $\overline{\mathcal{B}}$.] Compute $r_0 := c_0 x_{00} + w_0 \bmod q$, and send r_0 to $\widehat{\mathcal{U}_i}$.
- (The payment protocol.) This can be handled as in the description of the cash system, since x_{1i} is known.

Continue this simulation until l executions of the withdrawal protocol with $\widehat{\mathcal{U}_i}$ have been performed.

Step 3. Check if $\widehat{\mathcal{U}_i}$ has on its tapes a certified key pair $((x_{0i}, x^*_{1i}), (w_{0i}, w_{1i}))$, (h'_i, a'_i), (c'_0, r'_0) such that $x^*_{1i} \neq x_{1i} \bmod q$. If not, then halt.

Step 4. Extract $\log_{g_0}(h_0 h'_i)$ by running the knowledge extractor M on all tapes of A and $\widehat{\mathcal{U}_i}$. Compute $(\log_{g_0}(h_0 h'_i) - x_{00} - x_{0i})/(x^*_{1i} - x_{1i}) \bmod q$, and output the outcome.

Note that we have made use of Assumption II in Step 4 of this simulation.

By definition of the key generation of A in Step 1, the public key in Step 1 is simulated with the same probability distribution as that by which $\overline{\mathcal{B}}$ generates its public key. Likewise, the key generation for $\overline{\mathcal{T}_i}$ is performed with the same probability distribution. The response that is computed by A in the simulated withdrawal protocol is the same as the response that $\overline{\mathcal{B}}$ would compute:

$$
\begin{aligned}
g_0^{r_0} &= g_0^{c_0 x_{00} + w_0} \\
&= (g_0^{x_{00}})^{c_0} g_0^{w_0} \\
&= (h_0 g_1^{x_{1i}})^{c_0} a_0 \\
&= (h_0 h_i)^{c_0} a_0.
\end{aligned}
$$

From this it easily follows that the view of $\widehat{\mathcal{U}_i}$ that is provided by A is the same as that provided by $\overline{\mathcal{B}}$ in the real cash system (regardless of the probability distribution by which $\widehat{\mathcal{U}}_i$ generates its challenges), despite of the tricky way in which A generates h_0. This obviously holds also for the simulated executions of the payment protocol, and Step 1 of the simulated executions of the withdrawal protocol. Note that A provides for the possibility of $\widehat{\mathcal{U}}_i$ to physically extract the secret key of $\mathcal{T}_i$, because it generates x_{1i} by itself. Hence, Step 4 is reached by supposition with non-negligible probability.

To prove that the output of A is equal to $\log_g h$, note that

$$\begin{aligned} g_0^{\log_{g_0}(h_0 h_i') - x_{00} - x_{0i}} &= g_0^{\log_{g_0}(h_0 h_i')} g_0^{-(x_{00}+x_{0i})} \\ &= (h_0 h_i') g_0^{-(x_{00}+x_{0i})} \\ &= (g_0^{x_{00}} g_1^{-x_{1i}} g_0^{x_{0i}} g_1^{x_{1i}^*}) g_0^{-(x_{00}+x_{0i})} \\ &= g_1^{x_{1i}^* - x_{1i}}. \end{aligned}$$

Since $x_{1i}^* \neq x_{1i} \bmod q$, and $g = g_0$ and $h = g_1$, it follows that

$$g^{(\log_{g_0}(h_0 h_i') - x_{00} - x_{0i})/(x_{1i}^* - x_{1i})} = h.$$

This contradicts the Discrete Log assumption. Hence, if Assumption II and the Discrete Log assumption are true then it cannot be the case that $x_{1i}^* \neq x_{1i} \bmod q$ with non-negligible probability of success. □

Note that this result holds independent of whether the executions of the withdrawal protocol by $\widehat{\mathcal{U}}_i$ are performed sequentially or in parallel.

In general no conspiracy must be able to compute a certified key pair for which x_{1i}^* is different from the identification numbers of each of its members. Proposition 10, and the fact that the withdrawals of different users can only be run sequentially, seem to justify the following conjecture.

Conjecture 11. *No conspiracy can withdraw with non-negligible probability of success a certified key pair* $((x_{0i}, x_{1i}^*), (w_{0i}, w_{1i}))$, (h_i, a_i), (c_0, r_0) *for which* x_{1i}^* *differs from the identification numbers of each of its members.*

We are now prepared to prove the second property. We start once more with a statement related to conspiracies with a limited view, to show that we do not need to rely on Conjecture 11 in this case.

Proposition 12. *If the Discrete Log Assumption and Assumption II are true, then the following holds. If* $\widehat{\mathcal{U}}_i$ *double-spends an electronic coin, and the corresponding payment transcripts are accepted by* $\overline{\mathcal{B}}$ *in the deposit protocol, then the procedure for tracing a double-spender results with overwhelming probability in the identification number of* $\widehat{\mathcal{U}}_i$.

Proof. Denote the two corresponding payment transcripts by

$$(h_i', a_i'), (c_0', r_0'), (r_{0i}, r_{1i}), \texttt{spec}$$

and

$$(h'_i, a'_i), (c'_0, r'_0), (r^*_{0i}, r^*_{1i}), \texttt{spec}^*.$$

Since $\mathcal{B}$ has accepted the payment transcripts, $\texttt{spec}$ is not equal to $\texttt{spec}^*$. Denoting $\mathcal{H}(h'_i, \texttt{spec}, a'_i)$ by d, and $\mathcal{H}(h'_i, \texttt{spec}^*, a'_i)$ by d^*, we hence have $d \neq d^* \bmod q$ with overwhelming probability; otherwise $\mathcal{H}(\cdot)$ is not collision-intractable.

It furthermore follows from the acceptance by $\overline{\mathcal{B}}$ of the payment transcripts that

$$h'_i = g_0^{(r_{0i}-r^*_{0i})/(d-d^*)} g_1^{(r_{1i}-r^*_{1i})/(d-d^*)}$$

and

$$a'_i = g_0^{(d^* r_{0i}-d r^*_{0i})/(d^*-d)} g_1^{(d^* r_{1i}-d r^*_{1i})/(d^*-d)}.$$

Hence the triple $(((r_{0i}-r^*_{0i})/(d-d^*) \bmod q, (r_{1i}-r^*_{1i})/(d-d^*) \bmod q), ((d^* r_{0i} - d r^*_{0i})/(d^*-d) \bmod q, (d^* r_{1i} - d r^*_{1i})/(d^*-d) \bmod q)), (h'_i, a'_i), (c'_0, r'_0)$ is a certified key pair. By Proposition 10, $(r_{1i} - r^*_{1i})/(d - d^*)$ must with overwhelming probability be equal to the identification number of $\widehat{\mathcal{U}}_i$. □

In exactly the same way, we can prove the following result.

Proposition 13. *If the Discrete Log assumption and Conjecture 11 are true, then the following holds. If a conspiracy double-spends an electronic coin, and the corresponding payment transcripts are accepted by $\overline{\mathcal{B}}$ in the deposit protocol, then the procedure for tracing a double-spender results with overwhelming probability in the identification number of one of its members.*

This concludes our assessment of the second key property. We now turn to the third key property.

Proposition 14. *Assume that the blind Schnorr signature scheme is witness hiding, and that Assumption II is true. If it is infeasible to physically extract the secret key of $\overline{\mathcal{T}_i}$, then $\widehat{\mathcal{U}}_i$ cannot withdraw with non-negligible probability of success one electronic coin that is accepted twice by $\overline{\mathcal{B}}$ in the deposit protocol.*

Proof. Suppose that $\widehat{\mathcal{U}}_i$ can misuse l executions of the withdrawal and payment protocol to extract with non-negligible probability of success one electronic coin that can be deposited twice. We will construct a polynomial-time algorithm for extracting the witness of $\overline{\mathcal{P}}$ in the blind Schnorr signature scheme.

Algorithm A, on input a random public key $(q, g, h, \mathcal{H}(\cdot))$ of $\overline{\mathcal{P}}$, performs the following steps:

Step 1. (Simulate the initial key generation.) Set $g_1 := g$. Generate at random an element $x_{00} \in \mathbb{Z}_q$ and compute $h_0 := g_0^{x_{00}} h^{-1}$, and generate g_0 at random from $G_q \setminus \{1\}$. The simulated public key of $\overline{\mathcal{B}}$ is $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$.

Step 2. Simulate for $\widehat{\mathcal{U}}_i$ the actions that $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ would perform, as follows:

- (Opening an account.) Let $h_i := h$ be the public key of $\overline{\mathcal{T}_i}$.
- (The withdrawal protocol.)

  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Receive a from $\overline{\mathcal{P}}$, and pass $a_i := a$ on to $\widehat{\mathcal{U}}_i$.

Step 2. [Simulate $\overline{\mathcal{B}}$.] Generate at random an element $w_0 \in \mathbb{Z}_q$, compute $a_0 := g_0^{w_0}$, and send a_0 to $\widehat{\mathcal{U}}_i$.

Step 3. [Simulate $\overline{\mathcal{B}}$.] Receive c_0 from $\widehat{\mathcal{U}}_i$.

Step 4. [Simulate $\overline{\mathcal{B}}$.] Compute $r_0 := c_0 x_{00} + w_0 \bmod q$, and send r_0 to $\widehat{\mathcal{U}}_i$.

- (The payment protocol.)

  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Receive d' from $\widehat{\mathcal{U}}_i$, and pass $c := d'$ on to $\overline{\mathcal{P}}$.

  Step 2. [Simulate $\overline{\mathcal{T}_i}$.] Receive r from $\overline{\mathcal{P}}$, and pass $r_i := r$ on to $\widehat{\mathcal{U}}_i$.
- (The deposit protocol.) Receive a payment transcript (h_i', a_i'), (c_0', r_0'), (r_{0i}, r_{1i}), $\mathtt{spec}$ from $\widehat{\mathcal{U}}_i$.

Continue this simulation until l executions of the withdrawal and payment protocol have been performed by $\widehat{\mathcal{U}}_i$.

Step 3. Check if one of the pairs (h_i', a_i'), (c_0', r_0'), received as part of a triple in simulated deposit protocol, has been deposited twice. If not, then halt.

Step 4. Denoting the additional information that has been deposited by $\widehat{\mathcal{U}}_i$ along with the pair (h_i', a_i'), (c_0', r_0') by $(\mathtt{spec}, r_{1i})$ and $(\mathtt{spec}^*, r_{1i}^*)$ respectively, and $\mathcal{H}(h_i', \mathtt{spec}, a_i')$ by d and $\mathcal{H}(h_i', \mathtt{spec}^*, a_i')$ by d^*, compute $(r_{1i} - r_{1i}^*)/(d - d^*) \bmod q$ and output the result.

By definition of the key generation of $\overline{\mathcal{P}}$, and that of A in Step 1, the public key in Step 1 is simulated with the same probability distribution as that by which $\overline{\mathcal{B}}$ generates its public key. Likewise, the key generation for $\overline{\mathcal{T}_i}$ is performed with the same probability distribution. The response that is computed by A in the simulated withdrawal protocol is the same as the response that $\overline{\mathcal{B}}$ would compute:

$$\begin{aligned} g_0^{r_0} &= g_0^{c_0 x_{00} + w_0} \\ &= (g_0^{x_{00}})^{c_0} g_0^{w_0} \\ &= (h_0 h)^{c_0} a_0 \\ &= (h_0 h_i)^{c_0} a_0. \end{aligned}$$

Likewise, the response that is computed by A in the simulated payment protocol is the same as the response that $\overline{\mathcal{T}_i}$ would compute:

$$\begin{aligned} g_1^{r_i} &= g^r \\ &\overset{(\star)}{=} h^c a \\ &= h_i^{d'} a_i, \end{aligned}$$

where the substitution in $(\star)$ is allowed because the response of $\overline{\mathcal{P}}$ in the blind Schnorr signature issuing protocol is always correct. From this it easily follows that the view of $\widehat{\mathcal{U}}_i$ that is provided by A in the withdrawal and payment protocol is the same as that provided by $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ in the real cash system (regardless of the probability distribution by which $\widehat{\mathcal{U}}_i$ generates its challenges), despite of the tricky way in which A generates the public key of $\mathcal{T}_i$. Note that we assumed that the secret key of $\overline{\mathcal{T}_i}$ cannot be physically extracted by $\widehat{\mathcal{U}}_i$, and so it is not a problem that A does not know it (in fact, this is essence of the simulation). Hence, Step 4 is reached by supposition with non-negligible probability.

According to Proposition 12, the output of A is equal to x_{1i} with overwhelming probability. To complete the proof, note that an execution of Step 1 of the simulated withdrawal protocol, and an execution of each of Steps 1 and 2 of the corresponding payment protocol, constitutes exactly one execution of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$, and that A performs polynomially many of these executions. But $x_{1i} = \log_{g_1} h_i = \log_g h$, and so A can apparently compute $\log_g h$ with non-negligible probability of success. This contradicts the assumption that the blind Schnorr signature scheme is witness hiding. □

Proposition 15. *Assume that the blind Schnorr signature scheme is witness hiding, and that Conjecture 11 is true. If it is infeasible to physically extract a secret key from any $\overline{\mathcal{T}_i}$, then no conspiracy can withdraw with non-negligible probability of success one electronic coin that is accepted twice by $\overline{\mathcal{B}}$ in the deposit protocol.*

Proof. Suppose that a conspiracy can misuse any l executions of the withdrawal and payment protocol to extract with non-negligible probability of success one electronic coin that can be deposited twice. We will construct a polynomial-time algorithm for extracting the witness of $\overline{\mathcal{P}}$ in the blind Schnorr signature scheme.

Algorithm A, on input a random public key $(q, g, h, \mathcal{H}(\cdot))$ of $\overline{\mathcal{P}}$, performs the following steps:

Step 1. (Simulate the initial key generation.) Set $g_1 := g$. Generate at random an element $x_{00} \in \mathbb{Z}_q$ and an element $x_{10} \in \mathbb{Z}_q^*$, and compute $g_0 := g_1^{x_{10}^{-1}}$ and $h_0 := g_0^{x_{00}}$. The simulated public key of $\overline{\mathcal{B}}$ is $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$.

Step 2. For each party in the cash system, simulate the actions that $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ would perform. For a user $\mathcal{U}_i$, perform hereto the simulation as follows:

- (Opening an account.) Generate at random a number $s_i \in \mathbb{Z}_q^*$, and the corresponding public key $h_i := h^{s_i}$.
- (The withdrawal protocol.)
  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Receive a^* from $\overline{\mathcal{P}}$, and pass $a_i := (a^*)^{s_i}$ on to $\widehat{\mathcal{U}}_i$.
  Step 2. [Simulate $\overline{\mathcal{B}}$.] Receive a from $\overline{\mathcal{P}}$, and pass $a_0 := a$ on to $\widehat{\mathcal{U}}_i$.
  Step 3. [Simulate $\overline{\mathcal{B}}$.] Receive c_0 from $\widehat{\mathcal{U}}_i$, and pass $c := c_0 s_i \bmod q$ on to $\overline{\mathcal{P}}$.
  Step 4. [Simulate $\overline{\mathcal{B}}$.] Receive r from $\overline{\mathcal{P}}$, and pass $r_0 := c_0 x_{00} + x_{10} r \bmod q$ on to $\widehat{\mathcal{U}}_i$.
- (The payment protocol.)
  Step 1. [Simulate $\overline{\mathcal{T}_i}$.] Receive d' from $\widehat{\mathcal{U}}_i$, and pass $c^* := d'$ on to $\overline{\mathcal{P}}$.
  Step 2. [Simulate $\overline{\mathcal{T}_i}$.] Receive r^* from $\overline{\mathcal{P}}$, and pass $r_i := r^* s_i \bmod q$ on to $\widehat{\mathcal{U}}_i$.
- (The deposit protocol.) Receive a payment transcript (h_i', a_i'), (c_0', r_0'), (r_{0i}, r_{1i}), `spec` from $\widehat{\mathcal{U}}_i$.

Continue this simulation until l executions of the withdrawal and payment protocol have been performed by $\widehat{\mathcal{U}}_i$.

Step 3. Check if one of the pairs (h'_i, a'_i), (c'_0, r'_0), received as part of a triple in simulated deposit protocol, has been deposited twice. If not, then halt.

Step 4. Denoting the additional information that has been deposited along with the pair (h'_i, a'_i), (c'_0, r'_0) by $(\mathtt{spec}, r_{1i})$ and $(\mathtt{spec}^*, r^*_{1i})$ respectively, and $\mathcal{H}(h'_i, \mathtt{spec}, a'_i)$ by d and $\mathcal{H}(h'_i, \mathtt{spec}^*, a'_i)$ by d^*, compute $g_1^{(r_{1i}-r^*_{1i})/(d-d^*)}$. Search for a number h_i that was generated as $h_i := h^{s_i}$ in the simulated procedure for opening an account in Step 2. If such a number is not found, then halt.

Step 5. For the value of s_i, corresponding to the number h_i that has been found in the search in Step 4, compute $(r_{1i} - r^*_{1i})/s_i(d - d^*) \bmod q$ and output the result.

The response that is computed by A in the simulated withdrawal protocol is the same as the response that $\overline{\mathcal{B}}$ would compute:

$$\begin{aligned} g_0^{r_0} &= g_0^{c_0 x_{00} + x_{10} r} \\ &= (g_0^{x_{00}})^{c_0} (g_0^{x_{10}})^r \\ &= h_0^{c_0} g_1^r \\ &= h_0^{c_0} g^r \\ &\overset{(\star)}{=} h_0^{c_0} (h^c a) \\ &= h_0^{c_0} h^{c_0 s_i} a_0 \\ &= (h_0 h^{s_i})^{c_0} a_0 \\ &= (h_0 h_i)^{c_0} a_0. \end{aligned}$$

Likewise, the response that is computed by A in the simulated payment protocol is the same as the response that $\overline{\mathcal{T}_i}$ would compute:

$$\begin{aligned} g_1^{r_i} &= g_1^{r^* s_i} \\ &= (g^{r^*})^{s_i} \\ &\overset{(\star\star)}{=} (h^{c^*} a^*)^{s_i} \\ &= (h^{s_i})^{c^*} (a^*)^{s_i} \\ &= h_i^{d'} a_i. \end{aligned}$$

The substitutions in $(\star)$ and $(\star\star)$ are allowed because the response of $\overline{\mathcal{P}}$ in the blind Schnorr signature issuing protocol is always correct.

Since x_{10} is randomly generated from $\mathbb{Z}_q^*$, instead of from $\mathbb{Z}_q$, the computation of g_0 in Step 1 is always defined. This subtle difference in key generation implies that the views provided by A are not the same as those provided by $\overline{\mathcal{B}}$ and $\overline{\mathcal{T}_i}$ in the real cash system. However, it is easy to see that they are statistically indistinguishable, despite of the tricky way in which A generates the public keys of the $\mathcal{T}_i$'s. Note that we assumed that the secret keys of tamper-resistant devices cannot be physically extracted, and so it is not a problem that A does not know these secret keys (again, this is actually the essence of the simulation). Hence, Step 4 is reached by supposition with non-negligible probability.

According to Conjecture 11, $(r_{1i}-r^*_{1i})/(d-d^*) \bmod q$ in Step 4 must be equal to the identification number x_{1i} of one of the members of the conspiracy, with overwhelming probability. Since this identification number is equal to $\log_{g_1} h_i$ for the number h_i associated with this member, the search of A in Step 4 is successful with overwhelming probability.

So Step 5 is also reached with non-negligible probability. The output of A is equal to $\log_g h$, as can be seen as follows:

$$\begin{aligned} g^{(r_{1i}-r^*_{1i})/s_i(d-d^*)} &= (g_1^{(r_{1i}-r^*_{1i})/(d-d^*)})^{s_i^{-1}} \\ &= h_i^{s_i^{-1}} \\ &= h. \end{aligned}$$

To complete the proof, notice that an execution of each of Steps 2, 3 and 4 of the simulated withdrawal protocol constitutes exactly one execution of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$. Likewise, an execution of Step 1 of the simulated withdrawal protocol, and an execution of each of Steps 1 and 2 of the corresponding payment protocol, constitutes exactly one execution of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$. Hence A performs polynomially many executions of the blind Schnorr signature issuing protocol with $\overline{\mathcal{P}}$, and so the construction of A contradicts the assumption that the blind Schnorr signature scheme is witness hiding. □

Observe that A may need to perform executions of the blind signature issuing protocol with $\overline{\mathcal{P}}$ in parallel, *even* if no two executions of the withdrawal protocol were allowed to be run in parallel by $\overline{\mathcal{B}}$.

This concludes our assessment of the three key properties. It is important to observe that the presented results have been stated with respect to the worst conditions conceivable. Various statements can be shown to hold under weaker assumptions if the considered conspiracies are provided with less power. For example, all the results that have been proven in our assessment of the first two key properties are true under weaker assumptions, if the considered conspiracies cannot physically extract secret keys from tamper-resistant devices. We can also make use of the fact that conspiracies cannot learn anything from 'wire-tapping" executions of honest users, since these executions are perfectly simulatable, as we will prove in Proposition 17; only when they can wire-tap in addition the interaction between the tamper-resistant devices and these users in the payment protocol can something be learned (in an indirect way – which is highly unlikely to be useful).

Furthermore, there are additional results that one may wish to prove. For instance, we have not shown that it is infeasible to spend a coin *once* (more precisely, compute a payment transcript) without assistance of the tamper-resistant device. The reason for this is that there is no need to prove such a result: no damage is done if a user can withdraw a coin that can be spent *once* in cooperation with a cooperating service provider, without this requiring assistance of the tamper-resistant device. (Damage comes only from the ability to *forge* a coin or payment transcript, or the ability to spent a withdrawn coin a *second* time without assistance of a tamper-resistant device – and we have assessed these attacks). Other additional results that one may wish to prove are: the probability that two certified key pairs encompassing the same certified public key are withdrawn (by coincidence) is negligible ; it is infeasible to deposit a wire-tapped payment transcript to another account; and, it is infeasible to spend a wire-tapped coin of another party.

Since such modifications and additional results can easily be proven by applying the various proof techniques that have been demonstrated in this section, they are left as an exercise to the reader.

6 Immunization Against Attacks on Parallel Withdrawals

It can be shown that Conjecture 11 is false if $\mathcal{B}$ allows executions of the withdrawal protocol to be performed *in parallel* with respect to two users, that each have a *different* identification number. The two users can then join forces to extract a certified key pair in which neither of their identification numbers is encoded.

Let x_{1i} be the identification number of $\mathcal{U}$, and x_{1i}^* that of $\mathcal{U}^*$; the corresponding public keys of $\mathcal{T}$ and $\mathcal{T}^*$ are h_i and h_i^*. In its simplest form, the attack on the two parallel executions of the withdrawal protocol is the following (the steps marked with a period refer to the execution by $\mathcal{U}$, and those marked with a star refer to the execution by $\mathcal{U}^*$):

Step 1. (As in the withdrawal protocol.)
Step 1* (As in the withdrawal protocol.)
Step 2. $\mathcal{B}$ generates at random a number $w_0 \in \mathbb{Z}_q$, and sends $a_0 := g_0^{w_0}$ to $\mathcal{U}$.
Step 2* $\mathcal{B}$ generates at random a number $w_0^* \in \mathbb{Z}_q$, and sends $a_0^* := g_0^{w_0^*}$ to $\mathcal{U}^*$.

[**Preparation.**] $\mathcal{U}$ and $\mathcal{U}^*$ compute $h_i' := g_1^{x_{1i}'}$ for an arbitrary identification number x_{1i}' of their choice. They generate two random numbers w_{0i} and w_{1i} in $\mathbb{Z}_q$, and compute $a_i' := g_0^{w_{0i}} g_1^{w_{1i}}$, $c_0' := \mathcal{H}(h_0 h_i', a_i', a_0 a_0^*)$.

Step 3. $\mathcal{U}$ sends $c_0 := (c_0' x_{1i}^* - c_0' x_{1i}')/(x_{1i}^* - x_{1i}) \bmod q$ to $\mathcal{B}$.
Step 3* $\mathcal{U}^*$ sends $c_0^* := (c_0' x_{1i}' - c_0' x_{1i})/(x_{1i}^* - x_{1i}) \bmod q$ to $\mathcal{B}$.
Step 4. $\mathcal{B}$ sends $r_0 := c_0(x_{00} + x_{10} x_{1i}) + w_0 \bmod q$ to $\mathcal{U}$, and debits the account of $\mathcal{U}$ by the value of the electronic coin.
Step 4* $\mathcal{B}$ sends $r_0^* := c_0^*(x_{00} + x_{10} x_{1i}^*) + w_0^* \bmod q$ to $\mathcal{U}^*$, and debits the account of $\mathcal{U}^*$ by the value of the electronic coin.

$\mathcal{U}$ and $\mathcal{U}^*$ accept if and only if

$$g_0^{r_0}(h_0 h_i)^{-c_0} = a_0 \quad \text{and} \quad g_0^{r_0^*}(h_0 h_i^*)^{-c_0^*} = a_0^*.$$

If the verification holds, then $\mathcal{U}$ and $\mathcal{U}^*$ compute $r_0' := r_0 + r_0^* \bmod q$.

Proposition 16. *If $\mathcal{U}$ and $\mathcal{U}^*$ accept, then*

$$(0, x_{1i}'), (w_{0i}, w_{1i})), \ (h_i', a_i'), \ (c_0', r_0')$$

is a certified public key.

Proof. It is clear that $h_i' = g_0^0 g_1^{x'_{1i}}$ and $a_i' = g_0^{w_{0i}} g_1^{w_{1i}}$. It remains to prove that (h_i', a_i'), (c_0', r_0') is a certified public key, *i.e.*, that

$$c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'}(h_0 h_i')^{-c_0'}).$$

Since $\mathcal{U}$ and $\mathcal{U}^*$ compute c_0' according to $c_0' := \mathcal{H}(h_0 h_i', a_i', a_0 a_0^*)$, this follows from:

$$\begin{aligned}
a_0 a_0^* &= g_0^{r_0}(h_0 h_i)^{-c_0} g_0^{r_0^*}(h_0 h_i^*)^{-c_0^*} \\
&= g_0^{r_0 + r_0^*} h_0^{-(c_0 + c_0^*)} h_i^{-c_0} (h_i^*)^{-c_0^*} \\
&= g_0^{r_0 + r_0^*} h_0^{-(c_0 + c_0^*)} g_1^{-c_0 x_{1i}} g_1^{-c_0^* x_{1i}^*} \\
&\overset{(\star)}{=} g_0^{r_0'} h_0^{c_0'} g_1^{-(c_0 x_{1i} + c_0^* x_{1i}^*)} \\
&\overset{(\star\star)}{=} g_0^{r_0'} h_0^{c_0'} g_1^{-c_0' x_{1i}'} \\
&= g_0^{r_0'} (h_0 h_i')^{-c_0'}.
\end{aligned}$$

The substitution in $(\star)$ is allowed because

$$\begin{aligned}
c_0 + c_0^* &= (c_0' x_{1i}^* - c_0' x_{1i}')/(x_{1i}^* - x_{1i}) + (c_0' x_{1i}' - c_0' x_{1i})/(x_{1i}^* - x_{1i}) \\
&= (c_0' x_{1i}^* - c_0' x_{1i})/(x_{1i}^* - x_{1i}) \\
&= c_0' \bmod q,
\end{aligned}$$

and that in $(\star\star)$ because

$$\begin{aligned}
c_0 x_{1i} + c_0^* x_{1i}^* &= x_{1i}(c_0' x_{1i}^* - c_0' x_{1i}')/(x_{1i}^* - x_{1i}) + x_{1i}^*(c_0' x_{1i}' - c_0' x_{1i})/(x_{1i}^* - x_{1i}) \\
&= (x_{1i}^* c_0' x_{1i}' - x_{1i} c_0' x_{1i}')/(x_{1i}^* - x_{1i}) \\
&= c_0' x_{1i}' \bmod q.
\end{aligned}$$

□

To prevent unduly obscuring of the description of the attack, we have not incorporated the additional computations the attackers need to perform in order to obtain the certified key pair in a perfectly blind way.

Now, observe that both users in this attack must know their identification numbers, x_{1i} and x_{1i}^*, before returning their challenges to $\mathcal{B}$ in Steps 3 and 3* (for which they first need to break the tamper-resistance of their devices, if the assumptions and the conjecture in the previous section are true). The resulting certified key pair can be spent multiple times, because the procedure that is followed by the bank for tracing a double-spender results in x_{1i}'. (Note, though, that the bank will detect with overwhelming probability that the attack has been applied, since the probability is negligible that the identification number that is encoded in the certified key pair corresponds to some other, "honest" user).

The following simple modification to the withdrawal protocol is believed to suffice to make the withdrawal protocol immune to attacks such as these. Rather than computing for each *user* a random number h_i, the bank computes a random h_i for each *execution of the withdrawal protocol.* More specifically, $\mathcal{B}$ generates h_i of the form $h_i := g_1^{x_{1i}+y}$, where y is a randomly chosen "offset" in $\mathbb{Z}_q$ and x_{1i} is (as before) the secret key of $\mathcal{T}_i$. $\mathcal{B}$ sends h_i along with a_0 in Step 2 of the withdrawal protocol. In Step 4, $\mathcal{B}$ correspondingly computes its response according to $r_0 := c_0(x_{00} + x_{10}(x_{1i} + y)) + w_0 \bmod q$, and sends the offset y along with r_0 to $\mathcal{C}_i$. Note that $\mathcal{C}_i$ need not know $\log_{g_1} h_i$ in Step 3 of the withdrawal protocol, and so no other modifications to the withdrawal protocol are needed. In Step 3 of the payment protocol, $\mathcal{C}_i$ correspondingly computes r_{1i} as $r_{1i} := r_i + d'y + w_{1i} \bmod q$; no other modifications are needed.

The obvious purpose of this modification is to ensure that $\widehat{\mathcal{U}}_i$ cannot feasibly compute $\log_{g_1} h_i$ even if he has managed to break the tamper-resistance of his device.

It is easy to see that all the statements in the previous section remain valid. Partial proofs, based on the linearity of the determinant function and described in detail in part (ii) of [4], strongly suggest that Conjecture 11, and all other statements, apply to the modified cash system even when there is no restriction whatsoever on running executions of the withdrawal protocol in parallel.

Note that in the modified system not only can a double-spender be traced, but also the execution of the withdrawal protocol in which the double-spent electronic coin has been issued. This suggests that we may be able to improve on the efficiency of the modification, and indeed we can. Since the purpose of the modification is merely to ensure that the user cannot compute $\log_{g_1} h_i$ *in real time*, before the challenge has to be returned in Step 3 of the withdrawal protocol, imposing an upper limit on the delay in time between sending out a_0 and receiving c_0 allows the bank to use random offsets y from a domain much smaller than $\mathbb{Z}_q$.

In order to avoid having to store one number (the offset) for each execution of the withdrawal protocol, the bank can encode a fixed number into the offsets that it generates in the withdrawal executions with respect to the same user. For instance, the least couple of bytes of y can be associated by $\mathcal{B}$ uniquely with $\mathcal{U}_i$. This unique number in effect plays the role that was played by the identification number in the original system.

7 Relation to Blind Signature Protocols

As mentioned already in the introduction, the withdrawal protocol of the new system is not a blind signature issuing protocol. Consider a triple consisting of a secret key, a matching public key, and a certificate on the public key. The user in the withdrawal protocol can completely blind the public key and the certificate, but not (part of) the secret key. If the certificate would be a public-key certificate, as is the case in all the references mentioned in the introduction, then the protocol would indeed be a particular case of an ordinary blind signature scheme (in which both the message and the signature can be blinded, see Chaum [8] and his later work); the public key is the message and the certificate is the signature on the message.

However, since the certificate in the presented system is a *secret-key* certificate, by definition it is *not* a signature on the public key. On the contrary, pairs consisting of a public key and a matching secret-key certificate can be generated by anyone with exactly the same probability distribution, as shown by the following proposition.

Proposition 17. *The certificates that are issued by $\overline{\mathcal{B}}$ are secret-key certificates.*

Proof. We construct a polynomial-time simulation algorithm A that generates certified public keys with the same probability as that by which they are generated in the withdrawal protocol. On input the public key $(q, g_0, g_1, h_0, \mathcal{H}(\cdot))$ of $\mathcal{B}$, A performs the following steps:

Step 1. Generate at random two numbers $x, t \in \mathbb{Z}_q$, and a number $a_i \in G_q$.
Step 2. Compute $h_i := h_0^{-1} g_0^x$, $c_0 := \mathcal{H}(h_0 h_i, a_i, g_0^t)$ and $r_0 := c_0 x + t \bmod q$.
Step 3. Output the pair (h_i, a_i), (c_0, r_0).

The output of A is a certified public key:

$$\begin{aligned} c_0 &= \mathcal{H}(h_0 h_i, a_i, g_0^t) \\ &= \mathcal{H}(h_0 h_i, a_i, g_0^{r_0 - c_0 x}) \\ &= \mathcal{H}(h_0 h_i, a_i, g_0^{r_0} (g_0^x)^{-c_0}) \\ &= \mathcal{H}(h_0 h_i, a_i, g_0^{r_0} (h_0 h_i)^{-c_0}). \end{aligned}$$

Since g_0 is a generator of G_q, and the numbers in Step 1 are chosen at random, the output distribution of A is identical to the distribution that applies when certified key pairs are issued by $\overline{\mathcal{B}}$. □

In the new approach the *secret* key is the message, and the certificate is the signature on the message. But the message is not blinded; it fact, it *cannot* be blinded.

This falsifies the popular belief that efficient privacy-protecting off-line electronic cash systems must be based on withdrawal protocols that are special instances of blind signature issuing protocols. In fact, as some hindsight will reveal, most of the presented protocol reductions were possible only because of the simulatability that is inherent to secret-key certificates.

8 Conclusion

It has been shown how to construct a privacy-protecting off-line electronic coin system based on a secret-key certificate scheme, rather than on a public-key certificate scheme. The new system improves significantly on the efficiency of the system in [3]; the computational effort for blinding an electronic coin is reduced by a factor of about two, and the on-line computational effort required to withdraw an electronic coin is reduced to what seems to be the absolute minimum achievable (one modular multiplication). Furthermore, several important claims have been proved by clean-cut simulations, by virtue of the simulatability of certified key pairs; such proofs are not known for any other "practical" privacy-protecting off-line cash system in the literature.

The presented cash system lends itself very well to practical implementation, as I showed in [5]. Practical optimizations, discussed in [5], ensure that the role of the tamper-resistant device can be implemented by a smart card with a standard 8-bit micro-processor, and that the bank needs to store in its deposit database no more than 50 bytes *per payment.* A description of how to extend the presented cash system such as to provide for electronic cheques, coins of different denominations and anonymous accounts, can be found in part (iv) of [4]. Furthermore, in part (v) of [4] it is shown how to generalize the applied techniques to general privacy-protecting credential mechanisms.

Finally, a remark of more general interest. A large part of this paper is concentrated around the notion of polynomial-time reductions between protocols. Such reductions seem to hardly have been studied in the literature; only polynomial-time reductions between a protocol and an algorithm (which can be seen as a degenerate, "non-interactive," protocol) seem to be widely in use. The formalization of polynomial-time reductions between protocols (actually, cryptographic *schemes*, since a scheme also includes a key generation algorithm) may be a worthwhile area of investigation, since it can shed light on what criteria a suitable formal model (and terminology) for cryptographic protocols should meet. As a typical example of wrong protocol terminology, consider the widespread use of the terminology "blind signature;" the correct terminology in my opinion is "blind signature *issuing protocol*," where "blind" modifies "issuing protocol," not "signature;" there may be many protocols for issuing a signature, and some of these may have the property that they issue in a blind way (unlinkability of views). Furthermore, when describing polynomial-time reductions between cryptographic schemes we have to carefully state the constituent parts of a cryptographic scheme. For instance, reductions between cryptographic schemes require one key generation algorithm to be derived from another key generation algorithm; in general, this will require dissection of key generation algorithms into separate sub-algorithms (something that we have carefully avoided by assuming independent uniform probability distributions, in order not to make the descriptions of the simulated key generation for the presented proofs unduly complicated). As a part of this reduction between key generation algorithms, we must carefully state what we decide to call the public key. In light of this, the presented cash system seems very suitable for a study that aims at developing a formal model for cryptographic protocols.

References

1. Bellare, M., Goldreich, O., "On Defining Proofs of Knowledge," Advances in Cryptology – CRYPTO '92, Lecture Notes in Computer Science, no. 740, Springer-Verlag, pp. 390–420.
2. Bos, J., Chaum, D., "SmartCash: A Practical Electronic Payment System," Centrum voor Wiskunde en Informatica, Report CS-R9035, August 1990.
3. Brands, S., "Untraceable Off-Line Cash in Wallet with Observers," Advances in Cryptology – CRYPTO '93, Lecture Notes in Computer Science, no. 773, Springer-Verlag, pp. 302–318. An extended pre-print appeared as: "An efficient off-line electronic cash system based on the representation problem," Centrum voor Wiskunde en Informatica, Report CS-R9323, March 1993. Available by anonymous ftp from: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9323.ps.Z.
4. Brands, S., manuscript (1993). The following parts have been submitted for publication: (i) "Secret-Key Certificates," (ii) "Restrictive Blinding of Secret-Key Certificates," [(iii) is this paper], (iv) "Extensions of Off-Line Cash," and (v) "Privacy-protecting Digital Credentials Based on Restrictive Blinding."
5. Brands, S., "Off-line Cash Transfer by Smart Cards," Centrum voor Wiskunde en Informatica, Report CS-R9455, September 1994. Available by anonymous ftp from: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9455.ps.Z. Also in: Proceedings of the First Smart Card Research and Advanced Application Conference, France, October 1994, pp. 101–117.
6. Brickell, E., McCurley, K., "An Interactive Identification Scheme Based on Discrete Logarithms and Factoring," Journal of Cryptology, Vol. 5, No. 1 (1992), pp. 29–39.
7. Brickell, E., Gemmell, P., Kravitz, D., "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change," Submitted to the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '95), July 14, 1994.
8. Chaum, D., "Blind Signatures for Untraceable Payments," Advances in Cryptology – CRYPTO '82, Lecture Notes in Computer Science, Springer-Verlag, pp. 199–203.
9. Chaum, D., "Achieving Electronic Privacy," Scientific American, August 1992, pp. 96–101.
10. Chaum, D., Den Boer, B., Van Heijst, E., Mjolsnes, S., Steenbeek, A., "Efficient Offline Electronic Checks," Advances in Cryptology – EUROCRYPT '89, Lecture Notes in Computer Science, no. 434, Springer-Verlag, pp. 294–301.
11. Chaum, D., Fiat, A., Naor, M., "Untraceable electronic cash," Advances in Cryptology – CRYPTO '88, Lecture Notes in Computer Science, no. 403, Springer-Verlag, pp. 319–327.
12. Chaum, D., Pedersen, T., "Wallet databases with observers," Advances in Cryptology – CRYPTO '92, Lecture Notes in Computer Science, no. 740, Springer-Verlag, pp. 89–105.
13. Chaum, D., Pedersen, T., "Transferred Cash Grows in Size," Advances in Cryptology – EUROCRYPT '92, Lecture Notes in Computer Science, Springer-Verlag, pp. 357–367.
14. Chen, L., Damgard, I., Pedersen, T., "Parallel Divertibility of Proofs of Knowledge," Pre-proceedings of EUROCRYPT '94, pp. 137–150.
15. Cramer, R., Pedersen, T., "Improved Privacy in Wallets with Observers," Advances in Cryptology – EUROCRYPT '93, Lecture Notes in Computer Science, no. 765, Springer-Verlag, pp. 329–343.

16. Damgard, I., "Payment Systems and Credential Mechanisms With Provable Security Against Abuse by Individuals," Advances in Cryptology – CRYPTO '88, Lecture Notes in Computer Science, no. 403, Springer-Verlag, pp. 328–335.
17. D'Amiano, S., Di Crescenzo, G., "Methodology for digital money based on general cryptographic tools,"Pre-proceedings of EUROCRYPT '94, pp. 151–162.
18. De Santis, A., Persiano, G., "Communication Efficient Zero-Knowledge Proofs of Knowledge Without Interaction," Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 427–436.
19. Eng, T., Okamoto, T., "Single-Term Divisible Electronic Coins," Pre-proceedings of EUROCRYPT '94, pp. 311–323.
20. Feige, U., Shamir, A., "Witness Indistinguishable and Witness Hiding Protocols," Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing, 1990, pp. 416–426.
21. Feige, U., Fiat, A., Shamir, A., "Zero-Knowledge Proofs of Identity," Journal of Cryptology, Vol. 1, No. 2 (1988), pp. 77–94.
22. Fiat, A. and Shamir, A., "How to prove yourself: practical solutions to identification and signature problems," Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, Springer-Verlag, pp. 186-194.
23. Ferguson, N., "Single Term Off-Line Coins," Advances in Cryptology – EUROCRYPT '93, Lecture Notes in Computer Science, no. 765, Springer-Verlag, pp. 318–328.
24. Ferguson, N., "Extensions Of Single-Term Off-Line Coins," Advances in Cryptology – CRYPTO '93, Lecture Notes in Computer Science, no. 773, Springer-Verlag, pp. 292–301.
25. Franklin, M., Yung, M., "Secure and Efficient Off-Line Digital Money," Proceedings of ICALP '93, Lecture Notes in Computer Science, no. 700, Springer-Verlag, pp. 265–276.
26. Goldwasser, S., Micali, S., Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems," SIAM Journal on Computing, Vol. 18, No. (1989), pp. 186–208.
27. Guillou, L., Quisquater, J.-J., "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory," Advances in Cryptology – EUROCRYPT '88, Lecture Notes in Computer Science, no. 330, Springer-Verlag, pp. 123-128.
28. Hayes, B., "Anonymous One-Time Signatures and Flexible Untraceable Electronic Cash," Advances in Cryptology – AUSCRYPT '90, Springer-Verlag, pp. 294–305.
29. Hirschfeld, R., "Making Electronic Refunds Safer," Advances in Cryptology – CRYPTO '92, Lecture Notes in Computer Science, no. 740, Springer-Verlag.
30. Okamoto, T., "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes," Advances in Cryptology – CRYPTO '92, Lecture Notes in Computer Science, no. 740, Springer-Verlag, pp. 31–53.
31. Okamoto, T., Ohta, K., "Divertible Zero-Knowledge Interactive Proofs and Commutative Random Self-Reducibility," Advances in Cryptology – EUROCRYPT '89, Lecture Notes in Computer Science, no. 434, Springer-Verlag, pp. 481–496.
32. Okamoto, T., Ohta, K., "Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash," Advances in Cryptology – CRYPTO '89, Lecture Notes in Computer Science, no. 435, Springer-Ve rlag, pp. 481–496.
33. Okamoto, T., Ohta, K., "Universal Electronic Cash," Advances in Cryptology – CRYPTO '91, Lecture Notes in Computer Science, no. 576, Springer-Verlag, pp. 324–337.

34. Pfitzmann, B., Waidner, M., “How To Break and Repair A ‘Provably Secure’ Untraceable Payment System,” Advances in Cryptology – CRYPTO ’91, Lecture Notes in Computer Science, no. 576, Springer-Verlag, pp. 338–350.
35. Schnorr, C, “Efficient Signature Generation by Smart Cards,” Journal of Cryptology, Vol. 4, No. 3 (1991), pp. 161-174.
36. Van Antwerpen, H., “Electronic Cash,” Eindhoven University of Technology, master’s thesis, October 1990.
37. Veugen, T., “Some mathematical and computational aspects of electronic cash,” Eindhoven University of Technology, master’s thesis, November 1991.
38. Veugen, T., “The Security of an RSA-based Cut-and-choose Protocol,” Submitted for publication, September 15, 1993.
39. Yacobi, Y., “Efficient electronic money,” To appear in: Proceedings of AUSCRYPT ’94.