Lightweight Micro-Cash for the Internet

Wenbo Mao

Hewlett-Packard Laboratories
Bristol BS12 6QZ
United Kingdom

Abstract. We propose a micro-cash technique based on a one-time signature scheme: signing a message more than once leads to disclosure of the signer's private key. In addition to usual cash properties such as off-line bank for payment and spender's anonymity, the technique also provides a number of useful features. These include: identifying double spender with strong proof, cash revocable for identified double spender, independent of using tamper-resistant devices, coin sub-divisible to smaller denominations, and system simplicity in terms of small-sized data for cash representation as well as simple protocols for cash withdrawal, payment and deposit. We reason that these features support a lightweight cash system suitable for handling very low value payment transactions, such as information purchases on the Internet.
Keywords: Revocable cash for double spender, Internet electronic commerce.

1 Introduction

Today, the business potential of the Internet, particularly, of the world-wide-web applications, forms a new dimension in electronic commerce. It is believed that information purchases will form a big part of the activities in the Internet electronic commerce [33]. A typical nature of this form of commerce is to deal with large volume of low-value transactions. Usual price for a few information pages can be as low as several cents. Various techniques proposed for macro payments, e.g., [26, 33], are not suitable to be used here, as transaction fees may well exceed the value of payments. Furthermore, these techniques (including [33]) do not serve a proper purchaser's anonymity which can be an essentially important feature in information purchases. On the other hand, the vast diversity nature of the Internet information services means that the subscription-based services may not be very attractive to a large number of one-off viewers.

It is thus reasonable to consider facilitating information purchases on the Internet with a cash-like payment instrument.

Chaum's invention of blind signature techniques [7, 9] sets an important milestone for electronic commerce in cash-like transactions. After Chaum's original idea, the subject of electronic cash has widely been studied and many schemes proposed to tackle various unsolved problems (see e.g., [2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 17, 18, 19, 20, 22, 27, 28, 34]). Early schemes for off-line cash (e.g., Chaum, Fiat and Naor [14], Hayes [22], Okamoto and Ohta [28]) are notoriously inefficient as a result of using "cut-and-choose" techniques in cash withdrawal as well as in payment phases in order to thwart cheating. (Though the method of Okamoto and Ohta [28] includes a binary-tree technique for an efficient representation of coin division into smaller denominations.) Franklin and Yung first introduced a "line method" [19, 20] based on the Diffie-Hellman problem and set off a promising approach to avoidance of cut-and-choose (their scheme still uses cut-and-choose in withdrawal phase). More efficient "single-term" coins using no cut-and-choose were subsequently achieved by Brands [3] and by Ferguson [18]. Brands further generalised the line method into a "representation of Diffie-Hellman problem in groups of prime orders" [5] which can be used to design an electronic wallet with an observer nested in. Eng and Okamoto combined the Brands' representation problem with the binary-tree method for an improved efficiency of divisible coins [17].

Considering off-line cash with a decent anonymity service, an evident limitation in various previous schemes is the high degree of system complexity. Some of the above schemes, e.g., [14, 19, 20, 22, 28], require handling too large amount of data to be economically usable. Others, e.g., [3, 4, 5, 15, 17], critically rely on using a tamper-resistant device (a smartcard "observer" co-working with an electronic wallet) to protect the system secret. If we regard that the former kind of schemes have a high complexity in data processing, then the latter ones should be regarded to be expensive in hardware configuration: considering ubiquitous use of cash including on point sale, two co-working devices are necessary, a wallet to protect the personal secret and an observer to protect the system secret; further, the observer should meet a high standard of tamper-resistance as the holder has an incentive to open it.

In this paper we propose a micro-cash technique based on a new technique to solve the high cost problem. In addition to usual cash properties such as off-line bank for payment and spender's anonymity, the technique also provides a number of useful features. These include: identifying double spender with strong proof, cash revocable for identified double spender, independent of using tamper-resistant devices, coin sub-divisible to smaller denominations, and system simplicity in terms of small-sized data for cash representation as well as simple protocols for cash withdrawal, payment and deposit. We will reason that these features support a lightweight cash system suitable for handling very low value payment transactions.

Similar to all other off-line cash techniques, a double spender will be identified after a double spending has occurred. However, a unique and new feature in our after-the-fact identification method is that the identification is in terms of discovering the double spender's private key by the bank. Such a result of identification is effective to stop double spending: the bank can simply show the double spender's private key to the appropriate public-key certification authority (CA); the associated public-key certificate and the public key (needed for making payment) will be revoked instantly and unconditionally. Thus, double spending will be stopped within the cash mechanism itself rather than resorting to external forces. It is our understanding that every previous off-line cash scheme including those relying on tamper-resistant devices uses some unspecified external mechanisms to stop double spending (a tamper-resistant device only adds difficulty to double spending, once it is opened, double spending must be stopped by other mechanisms). Typically, the external mechanisms are police and the law court which are likely to be ineffective and too expensive for micro-cash payment systems.

The method proposed here also strongly deters double spending in the first place: a new certificate for an identified double spender can be made sufficiently expensive so that the price far outweighs the benefit of double spending low-value coins. Notice that the use of public key and certificate for payment does not mean that the spender's anonymity has to be compromised, as a certificate can be devised not to denote its holder's identity provided it is not abused.

The remainder of this paper is organised as follows. In Section 2 we introduce a one-time signature technique based on Schnorr's signature scheme. The special use of Schnorr's scheme forms the main security basis of our cash technique. In Section 3 the signature scheme will be applied to demonstrate the working principle of a simple cash coin. In Section 4 we analyse the security of the basic scheme. In Section 5 the basic scheme will be extended into one in which a coin unit can be divided into smaller denominations without adding space complexity. Using the extended scheme, a given payment amount can be made in one transaction without increasing the quantity of data flow. Finally, Section 6 summarizes the features of the electronic cash technique.

2 Schnorr's Digital Signature as a One-Time Signature Scheme

We use Schnorr's digital signature scheme [30, 31, 32] to describe a one-time signature scheme to be applied in our cash technique. We start by presenting Schnorr's original scheme[1]

In Schnorr's scheme, users in the whole system can share some public values as part of their public keys. First, choose two large primes, p and q where p is sufficiently large (512-1024 bits) such that the discrete logarithm problem in Z_p is intractable; q is also large (160 bits) and $q|(p-1)$. Then, choose a number $a \in Z_p$ of order q; namely, q is the smallest positive integer satisfying $a^q \equiv 1 \pmod{p}$. The number a can be computed as the $(p-1)/q$th power of a primitive element modulo p. It will be assumed that all parties in the system share these numbers. To generate a particular private/public key pair, Alice chooses a random number s for $0 < s < q$. This is her private key. Then she calculates

$$v := (a^{-s} \bmod p) \tag{1}$$

The result v is Alice's public key. Schnorr's scheme uses a secure one-way hash function. Let $h(.)$ denote such a function which maps from the integer space to

[1] The use of Schnorr's scheme is not necessary. Other schemes, such as ElGamal [16] and DSS [1], can also be used.

Z_q^* and symbol $\|$ be bit-string concatenation. To sign a message m, Alice picks a random number $r \in Z_q^*$ and does the following computations:

$$x := (a^r \bmod p) \tag{2}$$

$$e := h(m \parallel x) \tag{3}$$

$$y := ((r + se) \bmod q) \tag{4}$$

The signature on the message m is the pair (e, y). We will call the other two quantities r and x, *secret* and *public signature generators* (or SSG, PSG for short), respectively. To verify the signature, Bob computes:

$$z := (a^y v^e \bmod p) \tag{5}$$

and tests if e is equal to $h(m \parallel z)$. If the testing is true, Bob accepts the signature as valid.

Schnorr's signature scheme gets its security from the difficulty of calculating discrete logarithm. The difficulty means that the private key s cannot be easily derived from the public key v from their relation in (1). Similarly, the SSG r cannot be easily derived from the non-secret number x from their relation in (2) (x is equal to z available to the verifier). If SSG r can easily be discovered, then the private key s can easily be derived from (4).

Besides relying on the difficulty of discrete logarithm, the security of the one-way hash function $h(.)$ also plays an important role. The security is in terms of the infeasibility of inverting the function, and of finding two input values $x \neq x'$ such that $h(x) = h(x')$. When Bob verifies the signature, he knows from the property of the hash function that, without knowing Alice's private key, it is computationally infeasible to create the consistency among the numbers e, z and m which are related under the hash function used.

Note that the SSG r must be treated as one-time material. It must not be used more than once to generate different signatures. Assume that Alice has used an SSG r before to sign a message m and now she re-uses it to sign a different message m'. Let (e, y) and (e', y') be the respective signatures. Now that $m' \neq m$, from (3) and the property of the hash function we know with an overwhelming probability that $e' \not\equiv e \pmod q$. With these two signatures, Bob can compute Alice's private key s by subtracting two instances of (4) and obtain

$$s = (\frac{y - y'}{e - e'} \bmod q) \tag{6}$$

When creating a new signature, as long as Alice always choose a new SSG at uniformly random from Z_q^*, then subtraction of (4) will only result in

$$y - y' \equiv r - r' + s(e - e') \pmod q$$

Here the value $r - r' \not\equiv 0 \pmod q$ remains to be a secret that protects the private key s just in the same way as in Schnorr's scheme. More precisely, Alice should not use an SSG which is related to old SSG's in any known algorithmic way. As long as this precaution measure is taken, no computationally feasible method is known to derive the private key from different instances of signatures. In fact, the digital signature standard (DSS) proposed by NIST [1] uses essentially the same principle to protect the signer's private key.

We can employ the property illustrated in (6) to prevent a user from using certain data more than once. The idea is to let the user sign the data as a condition of using the data and the signature must be generated in such a way that the verifier can check whether the user has correctly complied with the signing procedure: to have used a specified PSG. If the signature is generated under challenge (e.g., to include time/location information), then the user cannot sign (use) the data more than once without disclosing her/his private key.

Electronic cash forms a good example of such data. A coin can be constructed to contain a PSG x. During the payment time Alice must sign the coin for the merchant to verify. The coin will not be accepted if Alice's signature is generated with using a wrong PSG (when the merchant sees z in (5) to be different from x in the coin) even if the signature is valid in the sense of the original Schnorr's scheme. If the signature also includes a timestamp and a merchant identity, then Alice cannot double spend a coin without identify herself since she cannot sign the coin more than once without disclosing her private key.

Theorem 1 *Let* $a > 1$ *and* $a^q = 1 \pmod p$. *Then* $r \not\equiv r' \pmod q$ *implies* $(a^r \bmod p) \neq (a^{r'} \bmod p)$.

Proof Assume $a^r = a^{r'} \pmod p$. We have $a^{r-r'} = 1 \pmod p$. Conjuncting this with $a^q = 1 \pmod p$, we reach $r \equiv r' \pmod q$. □

Theorem 1 insures that it is impossible for Alice to find two different SSG's $r \not\equiv r' \pmod q$ that map to the same PSG x. Being able to do so would allow Alice to cheat the bank since using (6) will not correctly reveal her private key, and also to cheat the merchant as if a "correct" PSG has been used.

Finally, we should point out that a digital signature can be produced without identifying the signer. In fact, a public-key can be certified to an anonymous holder in such a manner that even the key certification authority cannot link a certificate to its holder. Such a technique is reported in [25]. In the rest of this paper, we will always assume that Alice uses an anonymous public key certificate which does not contain her identity, even though the certificate will be denoted by $Cert_A$.

3 A Simple Cash Coin

We now devise a simple off-line electronic cash scheme. The scheme consists of three protocols: withdrawal, payment and deposit.

3.1 Withdrawal

Alice can withdraw a coin by running a withdrawal protocol with a bank. The bank need not be one in which she keeps an account.[2] The coin will be blindly signed by the bank to worth a specified value and this can be validated by any receiver if the signature is supported with a public-key certificate. Let B denote the bank; (K_B, N_B) be the bank's RSA public key; $f(.)$ be a secure one-way hash function, $x = (a^r \bmod p)$ be a PSG pre-computed by Alice and v be Alice's public key in Schnorr's scheme. Further, let b be a blinding factor in Chaum's blind signature technique that is chosen by Alice at uniformly random from Z_{N_B} (In RSA, there is no need to differentiate $Z^*_{N_B}$ and Z_{N_B} because the chance to have chosen a number in $Z_{N_B} \setminus Z^*_{N_B}$ is equivalent to have factorised N_B). The withdrawal protocol can be as follows.

Step 1. $A \to B$: *request*, $(b^{K_B} * f(x \parallel v)) \bmod N_B$

Step 2. $B \to A$: $(b^{K_B} * f(x \parallel v))^{K_B^{-1}} \bmod N_B$, $Cert_B$

The message "*request*" in Step 1 represents a (credit or debit) transaction request. It instructs the bank how to obtain money from Alice. For instance, it is a result of another fund transfer protocol (e.g., SET [26]). Let $L(x)$ be the length of bit string x. Considering that usually $L(f(.)) < L(N_B)$, in implementation $f(x \parallel v)$ can be replaced with a concatenation of itself for $\lfloor L(N_B)/L(f(.)) \rfloor$ times.

Upon receipt of the replied message from the bank, Alice can obtain her coin by dividing the blinding factor b into the first chunk. We will use *Coin* to denote the coin which has been blindly signed by the bank:

$$Coin = f(x \parallel v)^{K_B^{-1}} \bmod N_B$$

Notice that the withdrawal protocol does not go through any cut-and-choose style of cheating detection procedure because there is no need to do so. It is computationally infeasible for Alice to make more than one different coins out of one withdrawal transaction. Also in her own interest, Alice will not construct an invalid coin, such as one which encodes an invalid PSG x (replayed, or the matching SSG is not known by Alice), or an invalid public key v (uncertified, or not knowing the matching private key); or else the money is wasted.

3.2 Payment

When paying *Coin* to a merchant, Alice must sign a spending signature. The signature should include the merchant's identity and a timestamp stating the spending time. Let M be the merchant's identity, and *DateTime* be a timestamp.

[2] This is a useful feature not available in any previous cash scheme: cash can be withdrawn when the spender's bank is off-line (e.g., in a foreign country), or even the spender need not be any bank account holder (e.g., a child).

The message to be signed should be "$Coin, M, DateTime$". Following (3) and (4), the spending signature is a pair (e, y) where

$$e := h(Coin \mid\mid M \mid\mid DateTime \mid\mid x) \tag{7}$$

$$y := ((r + se) \bmod q) \tag{8}$$

and $x := (a^r \bmod p)$ as the PSG integrated in $Coin$. It suffices to use the following single step to specify the payment protocol:

$A \rightarrow M :\ Coin,\ M,\ DateTime,\ e,\ y,\ v,\ Cert_A,\ Cert_B$

Upon receipt of the payment, the merchant will validate the coin by verifying the bank's blind signature on $Coin$, and verify the spending signature. The verification of the spending signature goes as follows. The merchant should first validate the public key v and the supporting anonymous certificate $Cert_A$. In addition to the standard way of checking certificate, the merchant should also check if the public key has been revoked. A revoked key should appear in his local certificate revocation list (CRL) as a result of being periodically fed with a shorter CRL called Δ-CRL from the network directory services to update his local copy of CRL (see Section 12.6 of [23]). If v is properly backed by the anonymous certificate $Cert_A$ and is not in the CRL, the merchant will carry on to verify the spending signature. This is to compute z from a, v, e, y as in (5), and test if $h(Coin \mid\mid M \mid\mid DateTime \mid\mid z)$ is equal to e. If the testing passes, he should also check the correct use of the PSG x and the public key v. The correct use of these values is witnessed by hashing z and v into $f(x \mid\mid v)$ in $Coin$. Permitting the use of a wrong PSG or an invalid public key will put the merchant in trouble if Alice later double spends $Coin$ (see section 4.3).

The payment protocol has a trivially low computational complexity for Alice since to make a payment is merely to compute a hash function (7) and to give a dot on the line (8). Considering that in real application cash should be usable ubiquitously and in some environment (e.g., point of sale), a smartcard-like device should be used to protect trapdoor information for how to use cash (in this technique, this consists of the SSG r and the private key s). In a point-of-sale environment, payment should only be made by using such devices which in general have very limited computing capacity. The extremely low computational complexity for the customer to make payment is evidently desirable.

3.3 Deposit

In some time later, the merchant will redeem $Coin$ by depositing it to the bank:

$M \rightarrow B :\ Sign_M(Coin, M, DateTime, e, y, E_M(z)),\ Cert_M$

We call these data *coin-deposit*. Here, $Sign_M(.)$ denotes a digital signature of the merchant and $E_M(z)$ means that the merchant encrypts z using his own public key. (Usual implementations of digital signatures use a one-way hash function and so $Sign_M(\cdots, E_M(m))$ will not reveal the encrypted message m.) The merchant's signature on the coin-deposit means that the merchant has properly dealt with the data in the payment protocol and in the deposit protocol. The bank cannot alter the coin-deposit (e.g., to frame the merchant).

4 Analysis

Now we examine the security of the electronic cash scheme.

4.1 Anonymity

First, we assume that Alice does not double spend her coins. Her anonymity of using the coins will be protected. This is because no data in the payment and in the deposit protocols contains any information about her identity. In this paper we only assume that the public-key certification authority (CA) is a trusted anonymity server; it issues the anonymous certificate $Cert_A$ to Alice and is trusted not to identify Alice without a good reason. However, this trusted service is not necessary. A blind certification technique is reported in [25]. Using the technique, a certificate can be blindly issued to Alice such that after the issuing, the CA loses the linkage between Alice and $Cert_A$.

With the trusted anonymity service, collusion among all banks and all merchants will not identify the spender of a coin. If the blind certification technique [25] is used, then even adding a collusive CA will not be able to identify an honest spender.

4.2 Practical Unlinkability

A pragmatic unlinkability service is supplied. The service means that it is impractical for any party in the system to determine an anonymous spender's spending pattern. We will reason that in order to determine the spending pattern of an anonymous spender, there would have to be a large scale collusion between the bank and merchants in the system. Note that because money will eventually converge to the bank, the bank is in a better position than any merchant to partition a large number of coins. Our analyses in unlinkability will therefore be focused on the bank, with and without the help from merchants.

First of all, it is obvious that if public keys and/or the supporting anonymous certificates are deposited together with coins, then coins can be partitioned by public keys and/or anonymous certificates; all coins in the same partition are spent by the same person. Depositing public keys or anonymous certificates together with coins is regarded as collusion. Slightly less obvious is that the value z, if deposited, can also be used to derive the public key v. This is because of the following congruence:

$$v^e \equiv z/a^y \pmod{p} \tag{9}$$

Once v^e is known, it is easy to reveal v as

$$v := (v^{ed} \bmod p) \quad \text{where} \quad ed \equiv 1 \pmod{p-1} \tag{10}$$

Without giving these values to the bank, it is computationally infeasible for the bank to partition coins that have deposited. It suffices for a merchant to be non-collusive if he simply forgets the anonymous spender's public key, the anonymous certificate and the value z once the coin has been accepted. Each coin is a function of a one-time random PSG, so is each spending signature. Thus, in the absence of double-spending, no set of coin-deposit will give any information whatsoever about its relationship with other sets of coin-deposits. Brute-force searching through the public-key space, e.g., using a candidate public key v and (5) to get a candidate value z followed by checking if they can be hashed to $f(x \parallel v)$ in $Coin$ (since $z = x$), is intractable as the searching has to go through the vast space Z_p, unless the bank has acquired a sufficiently large number of public keys of the users in the system (which form a trivially small subset of the whole public-key space). However to collect public keys requires a large scale collusion among the banks and the merchants. Brute-force searching z for a matching $E_M(z)$, which would allow the computation of the associated public key using (7) and (8), is as infeasible as searching the public-key space, and the searching can also be thwarted by using randomised encryption in the coding of $E_M(z)$.

Finally we point out that even the bank has successfully collected data needed to investigate an anonymous person's spending pattern, the data are only good for knowing the person's spending *history*. Linking future coins requires further collusion. The necessity for maintaining a long-term collusion forms the foundation for us to claim the impracticability of the collusion, or in other words, that our technique gives unlinkability in practice.

4.3 Correctness

Now we look at the difficulty for various parties to defraud. Assume that the bank sees duplicated copies of $Coin$. This may be resulted from either (i) Alice's double spending, or (ii) the merchant's replay or depositing of bad data, or (iii) a collusion between Alice and the merchant.

Case (i) Alice double spends. There will be difference either in M, or in $DateTime$, or in both, and any of these will result in two pairs of spending signatures (e, y) and (e', y') where $e \not\equiv e' \pmod{q}$ (and hence $y \neq y'$) with an overwhelming probability. These two pairs will suffice the bank to discover Alice's private key s using (6), and further obtain her public key v from (1).

The bank can see the correctness of the revealed keys by re-verifying the two spending signatures as the merchants have done. Any incorrectness in the re-verification indicates either a fraudulent merchant, or a collusion between Alice and the merchant(s). These will be dealt with in Cases (ii) or (iii), respectively. Assume that the re-verification of the spending signatures passes. Now the bank can identify Alice by showing the revealed private/public key pair (s, v) to the appropriate CA. (The private key s can contain a sub-string that points uniquely to the CA which has issued the anonymous certificate to Alice.) Upon seeing the revealed key pair, the CA will revoke the public key v by publishing it onto the Δ-CRL. Alice's identity will also be revealed.

Case (ii) The merchant replays data or deposits bad data. Because the merchant is unable to generate a valid spending signature using other people's coins, double depositing coins is confined to the following uninteresting scenario: the merchant simply replays all messages in the deposit protocol. It is easy for the bank to discover the replay and thereby only one instance of deposit will be redeemed.

Note that since the merchant is required to digitally sign each coin-deposit, depositing incorrect data containing gibberish as if they were "spending signatures" will lead to identifying the merchant as fraudulent. This is because, as long as a duplication of *Coin* is detected, the bank will demand the merchant to prove his honesty in depositing by decrypting $E_M(z)$ in the deposit and the result of decryption, z, will suffice the bank to re-verify the spending signature (using (9) and (10) to recover the public key v needed). We will see more about this in Case (iii).

Case (iii) Alice and the merchant collude. A collusion will make sense only if it does not lead to identifying Alice. Feasible ways to achieve this include that the merchant permits using incorrect public keys (uncertified or not matching v in *Coin*), or incorrect PSG's (not matching x in *Coin*). For instance, in the case of permitting the use of incorrect keys, a coin can be double spent by different people, or by the same person who holds different public keys (certified or not).

Firstly, we assume that the merchant permits the use of an uncertified public key; namely, the public key used in spending signature verification is not supported by a valid anonymous certificate. This collusion will be discovered because the correctly revealed private key (assuming it is in a valid format pointing to a known CA) will not lead to identification of a certificate holder from the CA's database. Such a public key will also be revoked (published in the CRL) to stop any further collusion. The merchant responsible will be identified (see below).

In other scenarios of collusion listed above, upon seeing duplication of *Coin*, the bank's computation using (6) will not reveal a correct private key, either. For instance, assume that two spending signatures (e_1, y_1) and (e_2, y_2) have been generated by two different key pairs where the private keys are s_1 and s_2, respectively. Then, using (6) will result in the following value:

$$s' = (\frac{s_1 e_1 - s_2 e_2}{e_1 - e_2} \bmod q)$$

Similarly, assume a merchant permits Alice to use an incorrect PSG which is mapped from a wrong SSG $r \not\equiv r' \pmod q$ where r' may or may not be a valid

SSG. Then (6) will disclose the following value:

$$s' = (\frac{s(e_1 - e_2) \pm (r - r')}{e_1 - e_2} \bmod q)$$

Other wrong forms of "private keys" can also be derived by mixed uses of wrong/good keys and wrong/good PSG's. Let v' be the matching "public key" computed from s' using (1). The bank will always re-verify the two spending signatures using the revealed public key v' as the merchant(s) have supposedly done during the two runs of the payment protocol. The re-verification will result in inconsistency; e.g., either s' is in an invalid format (does not point to a correct CA), or the two spending signatures $(e_{1,2}, y_{1,2})$ are incorrect regarding the verification key v' used, or the hashed value $f(z' \parallel v')$ does not match $f(x \parallel v)$ in *Coin*. (N.B. the re-verification excludes any possibility of mistakenly identifying an innocent user whose private key coincides with s' because even in such an extremely unlikely case, the "spending signatures" $(e_{1,2},\ y_{1,2})$ will be found to be incorrect when verified using v' as the "signatures" were not created by s' at all.)

In these situations, the two merchants (let them be M_1 and M_2) will be asked to decrypt $E_{M_1}(z)$ and $E_{M_2}(z)$, respectively, in order to prove their honesty. An honest merchant will be indicated by a z which can derive a public key v using (9) and (10) such that v is not in CRL, and using it the spending signature deposited by him can be re-verified as correct (using the same way as he has done during the payment time). The other merchant will be identified as fraudulent.

To this end, we see that Alice and the merchant cannot help each other to achieve double spending without identification. It is however interesting to point out that, as long as a coin is not to be double spent or double deposited, using invalid public keys or incorrect PSG's or even depositing gibberish spending signatures will not be detected since the bank will not and cannot verify the fake spending signature. Indeed, the bank need not be concerned with anything other than double spending.

Finally we point out that since each payment is signed by the merchant, the bank cannot frame the merchant by forging data.

5 Divisible Coin

In this section we will extend the basic scheme to one with which Alice can pay varied amount of moneies to various merchants who will be denoted as M_1, M_2, $\cdots$. The basic idea of the extension follows the Payword technique of Rivest and Shamir [29], or in a different topic, attributes to Lamport's original password identification technique [24] (also known as the S/Key technique [21]). It is to apply a secure one-way hash function, recursively, on a secret for a specified number of times. In the following three subsections we provide revised protocols for cash withdrawal, payment and deposit.

5.1 Withdrawal

To prepare withdrawal, Alice constructs a stick of n coins $C_0, C_1, C_2, \cdots, C_{n-1}$ by applying the hash function $f(.)$ recursively:

$$C_i = f(C_{i+1}) \text{ for } i = 0, 1, 2, \cdots, n-1 \tag{11}$$

where C_{n+1} is a secret random number (it is not a coin) chosen by Alice. She also chooses the first SSG r_1 and computes the respective PSG $x_1 := (a^{r_1} \bmod p)$, and creates

$$Top := C_0 \parallel f(x_1 \parallel v) \parallel n \tag{12}$$

The withdrawal protocol is similar to that for a single coin:

Step 1. $A \rightarrow B$: $request,\ (b^{K_B} * Top) \bmod N_B$

Step 2. $B \rightarrow A$: $(b^{K_B} * Top)^{K_B^{-1}} \bmod N_B,\ Cert_B$

Upon receipt of the replied message from the bank, Alice can obtain her coin stick by dividing the blinding factor b into the first chunk. We will use $Stick(n)$ to denote the coin stick blindly signed by the bank containing n coins:

$$Stick(n) = Top^{K_B^{-1}} \bmod N_B \tag{13}$$

Due to the one-way-ness of the hash function, the signature means that the bank has actually signed all of the n coins. The system can stipulate the bank's public key (K_B, N_B) to be only good for supporting a stick containing n coins. There will be no point for Alice to construct a longer stick $Stick(m)$ for $m > n$ since upon using (K_B, N_B) the merchant will not accept more than n coins from the stick. Thus, no cheating detection is needed still.

5.2 Payment

We begin with an informal description on the basic idea of how Alice pays coins to the first merchant M_1. After the informal description on the special case, we will specify the payment protocol in a general setting.

Assume that Alice is to pay i coins $(1 \leq i \leq n)$ to the first merchant M_1. The idea is that in addition to sending a signed payment (on $Stick(n)$), Alice should also disclose C_i in the stick to the merchant. The merchant can verify the validity of the i coins between the C_0 and C_i by recursively applying the hash function for i times, starting from C_i and finishing at C_0. To this end, the bank's blind signature on $Stick(n)$ can be verified (see (11, 12, 13)). However, this only tells the merchant the good structure of the coins. The merchant will only accept the coins provided that Alice has also correctly signed the spending signature on $Stick(n)$.

If $i \neq n$, then the coins in $Stick(n)$ has not been used up, and the merchant should make change. To let change be made, Alice should generate a second pair of SSG and PSG. Let them be r_2 and $x_2 := (a^{r_2} \bmod p)$ respectively. She sends the hashed value $f(x_2 \parallel v)$ to the merchant (can be sent together with the payment). These values together with C_i will allow the merchant to return change. To return change, the merchant M_1 generates and send back the following value which we will denote by $Stick(n-1)$:

$$Stick(n-i) := Sign_{M_1}(C_i, f(x_2 \parallel v), n-i)$$

A nice feature in Schnorr's scheme is that, the SSG, PSG pairs can be pre-computed before the signing time. Thus, there will be no problem for Alice to prepare these pairs for future use.

Now we describe the general setting. Assume Alice has spent j ($j < n$) coins with $k-1$ previous merchants M_1, M_2, $\cdots$, M_{k-1} (some or all of them may be the same merchant) and she now holds

$$Stick(n-j) = Sign_{M_{k-1}}(C_j, f(x_k \parallel v), n-j),\ Cert_{M_{k-1}}$$

which have been returned as change from the merchant M_{k-1} with whom Alice has shopped most recently. Under the general setting, we specify the payment protocol with which Alice pays i coins to the next merchant M_k for $k > 0$. These i coins are in $Stick(n-j)$. Note that in the above, we have informally described a special case where $j = 0$, $k = 1$ and $M_0 = B$.

Payment step

$$\begin{aligned} A \rightarrow M_k :\ & Stick(n),\ Stick(n-j),\ DateTime,\ e_k,\ y_k \\ & i,\ C_{j+i},\ f(x_{k+1} \parallel v),\ v,\ Cert_A,\ Cert_B,\ Cert_{M_{k-1}} \end{aligned}$$

Here

$$e_k = h(Stick(n-j) \parallel M_k \parallel DateTime \parallel x_k)$$

and

$$y_k = (r_k + se_k \bmod q)$$

Upon receipt of the message in **Payment step**, the merchant M_k will first validate the coins by applying the hash function for i times to see if he can start from C_{j+i} and reach C_j. Then after having checked the previous merchant's signature on $Stick(n-j)$, he can further apply the hash for another j times to reach C_0 followed by verifying the bank's blind signature on $Stick(n)$.

Assume the coins pass the validation, the merchant will verify the spending signature on $Stick(n-j)$. Analogous to (5), this is by computing z_k as follows:

$$z_k = (a^{y_k} v^{e_k} \bmod p)$$

and checking if the following equation holds:

$$e_k = h(Stick(n-j) \parallel M_k \parallel DateTime \parallel z_k)$$

Besides this, he must also check the spending signature has been generated using correct PSG x_k and public key v. If everything goes well, these i coins will be accepted.

If there are still unspent coins left (i.e., $n - j - i > 0$), there is a need to make change. In such a case, the merchant M_k should send the following message back to Alice:

Change step:

$$M_k \rightarrow A : \ \mathit{Stick}(n-j-i), \ \mathit{Cert}_{M_k}$$

where

$$\mathit{Stick}(n-j-i) = \mathit{Sign}_{M_k}(C_{j+i}, f(x_{k+1} \parallel v), n-j-i)$$

Note that although the scheme requires a merchant generate the integral combination between remaining coins and the (PSG, public-key) pair, this does *not* mean that the next merchant who is to be paid with the remaining coins has to trust the previous merchant. The signature merely indicates that the merchant has followed the protocol. Alice has freedom to choose any PSG she likes. It is purely Alice's interest to let each merchant combine a good PSG and the correct public key with the remaining coins.

In the next subsection, we will analyse the impossibility for any merchant to help Alice to spend more than n coins from $\mathit{Stick}(n)$ without being identified.

5.3 Deposit

Later, the merchant M_k can redeem the i coins he has been paid from the bank B by depositing the following data:

$$\begin{aligned} M_k \rightarrow B : \ & \mathit{Stick}(n), \ \mathit{Stick}(n-j), \ \mathit{Stick}(n-j-i), \\ & \mathit{Sign}_{M_k}(M_k, \mathit{DateTime}, \ e_k, \ y_k, \ E_{M_k}(x_k)), \ \mathit{Cert}_{M_k}, \ \mathit{Cert}_{M_{k-1}} \end{aligned}$$

The certificate of the previous merchant M_{k-1} is needed in order to allow verification of his signature on $\mathit{Stick}(n-j)$ and thereby allow the current merchant M_k to correctly redeem i coins between $\mathit{Stick}(n-j)$ and $\mathit{Stick}(n-j-i)$.

Upon receipt of the coin-deposit message, the bank will check duplication of the coins. If any coin C_l for $j \leq l \leq j+i$ is found in the database, a fraud has been detected. The bank can differentiate double spending from double depositing, and deal with these frauds accordingly (see below). If everything is OK, it will credit the merchant the value of i coins. Data $\mathit{Stick}(n-j)$, $\mathit{Stick}(n-j-i)$, together with the signature and certificate of the merchant will be archived.

Property 2 *The merchant M_k can only get paid for coins between $\mathit{Stick}(n-j)$ and $\mathit{Stick}(n-j-i)$.*

Reasoning Firstly, M_k cannot get paid for coins above $\mathit{Stick}(n-j)$ because the previous merchants M_l for $l < k$ will claim them and whenever disputes occur between M_k and M_l, it can easily be checked that M_k does not have correct spending signatures from Alice. Secondly, M_k cannot claim any coins below $\mathit{Stick}(n-j-i)$ because the next merchant will claim them using the signature of M_k on the top of that stick. Similarly, no other merchants can claim coins between $Stick(n-j)$ and $Stick(n-j-i)$. □

Property 3 *No merchant is able to help Alice to spend more than n coins out of $Stick(n)$.*

Reasoning Assume that the merchant M_k helps Alice by making two different coin sticks which will be viewed by subsequent merchants as:

$$Stick(n-j-l) = Sign_{M_k}(C_{j+l}, f(x \parallel v), n-j-l)$$

$$Stick(n-j-m) = Sign_{M_k}(C_{j+m}, f(x' \parallel v), n-j-m)$$

where $x \neq x'$ and l may or may not be equal to m. The intention of this help is to let Alice use these different coin sticks and so she can spend more than n coins from $Stick(n)$. The collusion must not demand Alice make two different spending signatures on $Stick(n-j)$, otherwise it is a simple double spending of $Stick(n-j)$. The collusive merchant can only deposit either coins between $SticK(n-j)$ and $Stick(n-j-l)$, or those between $SticK(n-j)$ and $Stick(n-j-m)$, but not coins in both of the cases. In the first case of depositing, Alice cannot use $Stick(n-j-m)$ because the next merchant who deposits it will turn in the collusive merchant M_k. Samely, Alice cannot use $Stick(n-j-l)$ in the alternative case of depositing. Even if the collusive merchant does not deposit any coins, he will still be turned in as long as Alice uses the both sticks made by him. □

6 Conclusion

Finally we conclude the paper with a summary of the features of the electronic cash scheme.

An effective way to stop double spending. Double spending can be stopped within the cash mechanism. This is a unique feature that is not available in any previous off-line electronic cash schemes. After detection of a double spending, all the bank need to do is to revoke the double spender's public key. Since the identification of double spender is a strong proof, it is simple to achieve. This method of stopping double spending is cost effective.

Strong anonymity for the spender. The spender enjoys a decent anonymity service as long as she does not double spend. Collusion among banks and merchants will not lead to any computationally feasible way to compromise the anonymity. If the anonymous certificate is issued blindly, then even adding collusive CA's will not be able to identify an honest spender.

Independent of using tamper-resistant devices. It is not necessary to use tamper-resistant devices because there is no system secret need to be protected. Of course, in a point-of-sale environment, using tamper-resistant devices (e.g., smartcards) by the spender will undoubtedly be helpful in protecting the private key and in preventing accidental human errors. However, cheap devices suffice because there is no need to prevent the device holders from extracting data in the devices. (In fact, they should keep safe backup of the data.)

Coin sub-divisible to variant denominations. For instance, a typical coin stick can be $Stick(1,000)$ to worth 10 dollars with each coin in it to worth 1 cent. During spending, after having released a top coin with spending signature, Alice can then continuously release coins down to 1-cent refinement, no further signature on these subsequent coins is needed, until she feels enough services have been purchased. This why of payment is particularly suitable for web-based interactive information page purchase.

System simplicity. The protocols for cash withdrawal, payment and deposit are simple and the data size for coin representation is small. The payment protocol has an exceptionally low computational complexity for the spender because to compute hash function is efficient and to generate a spending signature is merely to release a dot in a line. These are attractive features for making point-of-sale payment using smartcards. Further, cash can easily be withdrawn from a foreign bank and usable by a non-bank-account holder.

We believe that the proposed electronic cash technique is readily workable and has potential to lead to a full-fledged electronic commerce for Internet information purchases.

Acknowledgements

Part of the work was completed when the author was participating in the research programme in Computer Security, Cryptography and Coding (the CCC Programme) at the Newton Institute, Cambridge University, April, 1996. Discussions with Tatsuaki Okamoto and Claus Schnorr during the CCC Programme were very interesting and helpful. The author would also like to thank Dipankar Gupta, Miranda Mowbray and anonymous reviewers for their helpful comments on an early version of this paper.

References

1. Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). Federal Register, v.56, n.169, August 1991.
2. J.-P. Boly et al. The ESPRIT Project CAFE — High Security Digital Payment Systems. In *Computer Security — ESORICS'94 (LNCS 875)*, pages 217–230. Springer-Verlag, 1994.
3. S. Brands. Untraceable off-line cash in wallet with observers. In *Advances in Cryptology — Proceedings of CRYPTO'93 (LNCS 773)*, pages 302–318. Springer-Verlag, 1993.

4. S. Brands. Electronic cash on the internet. In *Proceedings of the Internet Society 1995 Symposium on Network and Distributed System Security*, 1995.
5. S. Brands. Off-line electronic cash based on secret-key certificates. Technical Report: CS-R9506, 1995.
6. J. Camenisch, J-M. Piveteau, and Stadler M. An efficient electronic payment system protecting privacy. In *Computer Security — ESORICS'94, (LNCS 875)*, pages 207–215. Springer-Verlag, 1994.
7. D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology — Proceedings of Crypto'82*, pages 199–203. Plenum Press, 1983.
8. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
9. D. Chaum. Blind signatures systems. U.S. Patent No 4,759,063, July 1988.
10. D. Chaum. Privacy protected payments: Unconditional payer and/or payee untraceability. In *Smartcard 2000.* North Holland, 1989.
11. D. Chaum. Online cash checks. In *Advances in Cryptology — Proceedings of EUROCRYPT'89 (LNCS 434)*, pages 288–293. Springer-Verlag, 1990.
12. D. Chaum. Achieving electronic privacy. *Scientific American*, pages 96–101, August 1992.
13. D. Chaum, B. den Boer, E. van Heyst, S. Mjolsnes, and A. Steenbeek. Efficient offline electronic checks. In *Advances in Cryptology — Proceedings of EUROCRYPT'89 (LNCS 434)*, pages 294–301. Springer-Verlag, 1990.
14. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology — Proceedings of CRYPTO'88 (LNCS 403)*, pages 319–327. Springer-Verlag, 1990.
15. D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology — Proceedings of CRYPTO'92 (LNCS 740)*, pages 89–105. Springer-Verlag, 1992.
16. T. ElGamal. A public-key Cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — Proceedings of CRYPTO'84 (LNCS 196)*, pages 10–18. Springer-Verlag, 1985.
17. T. Eng and T. Okamoto. Single-term divisible electronic coins. In *Advances in Cryptology — Proceedings of EUEOCRYPT'94 (LNCS 950)*, pages 306–319. Springer-Verlag, 1995.
18. N. Ferguson. Single term off-line coins. In *Advances in Cryptology — Proceedings of EUROCRYPT'93 (LNCS 765)*, pages 318–328. Springer-Verlag, 1994.
19. M. Franklin and M. Yung. Towards provably secure efficient electronic cash. Technical Report: TR CUCS-018-92, April 1992.
20. M. Franklin and M. Yung. Secure and efficient off-line digital money. In *Proceedings of ICALP'93, (LNCS 700)*, pages 265–276. Springer-Verlag, 1993.
21. N.M. Haller. The S/Key one-time password system. `http://ftp.cert.dfn.de/pub/tools/password/SKey/`.
22. B.. Hayes. Anonymous one-time signatures and flexible untraceable electronic cash. In *Advances in Cryptology — Proceedings of AUSCRYPT'90 (LNCS 453)*, pages 294–305. Springer-Verlag, 1990.
23. ITU/ISO/IEC. Draft Amendment 1 to ITU Rec. X.509 (1993) — ISO/IEC 9594-8: Information Technology — Open Systems Interconnection — The Directory: Authentication Framework, Amendment 1: Certificate Extensions. ISO/IEC JTC 1/SC 21/WG 4 and ITU-T Q 15/7 Collaborative Editing Meeting on the Directory, Ottawa, Canada, July 1995.

24. L. Lamport. Password identification with insecure communications. *Communications of the ACM*, 24(11):770–772, 1981.
25. W. Mao. Blind Certification of Public Keys and Off-Line Electronic Cash. HP Laboratories Technical Report, HPL-96-71, May 1996.
26. MasterCard and Visa Secure Electronic Transaction (SET) (see, e.g., `http://www.visa.com/`), February 1996.
27. G. Medvinsky and B.C. Neuman. NetCash: A design for practical electronic currency on the Internet. In *Proceedings of First ACM Conference on Computer and Communications Security*, pages 102–196. ACM Press, 1993.
28. T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology — Proceedings of CRYPTO'91 (LNCS 576)*, pages 324–337. Springer-Verlag, 1992.
29. R.L. Rivest and A. Shamir. Payword and micromint: Two simple micropayment schemes. `http://theory.lcs.mit.edu/~rivest/publications.html`, December 1995.
30. C.P. Schnorr. Efficient signature generation for smart cards. In *Advances in Cryptology — Proceedings of CRYPTO'89 (LNCS 435)*, pages 239–252. Springer-Verlag, 1990.
31. C.P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
32. C.P. Schnorr. A method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system. U.S. Patent No. 4,995,082, February 1991.
33. M. Sirbu and J.D. Tygar. NetBill: An Internet Commerce System. `http://www.ini.cmu.edu/netbill/CompCon.html`.
34. UK banks introduce Mondex, the cashless cash card. Newsbytes News Network (also see `http://www.mondex.com/`), January 1993.

This article was processed using the LaTeX macro package with LLNCS style