HEWLETT PACKARD

# Blind Certification of Public Keys and Efficiently Revocable Electronic Cash: Secure Against Capable Attackers

Wenbo Mao
Networked Systems Department
HP Laboratories Bristol
HPL-96-134
September, 1996

electronic cash, blind certification of public keys, revocation, cryptography

Electronic cash should be revocable in order to nullify the effect of attacks mounted by a capable attacker (e.g. double spending by reverse-engineering tamper-resistant devices). For prudent engineering considerations, cash revocability forms a necessary complementary measure to using tamper-resistant devices, adding system security while lowering system cost by reducing the level of physical tamper resistance required.

We propose a revocable cash scheme. Cash revocation is achieved efficiently via withdrawing an attacker's ability to further use the system. In several ways the new scheme improves from previous ones (including trustee-based tracing techniques, e.g. [5, 14, 19]), that identify the attacker and revoke cash in terms of publishing a blacklist of coins misused or robbed. Firstly, the new scheme can stop the attacker from further using the system without need of physical confiscation of him/her or his/her devices (physical confiscation is difficult over the network). Secondly, it will achieve a better performance due to avoiding real-time checking through a potentially big coin revocation list. Thirdly, without using trustee, the system serves unconditional privacy for honest users and uses simpler protocols for transactions.

Internal Accession Date Only

# Blind Certification of Public Keys and Efficiently Revocable Electronic Cash: Secure Against Capable Attackers

Wenbo Mao
Hewlett-Packard Laboratories
Filton Road
Stoke Gifford
Bristol BS12 6QZ
United Kingdom
Telephone: +44 117 922 9528
Fax: +44 117 922 8924
Email: wm@hplb.hpl.hp.com

August 16, 1996

**Abstract**

Electronic cash should be revocable in order to nullify the effect of attacks mounted by a capable attacker (e.g., double spending by reverse-engineering tamper-resistant devices). For prudent engineering considerations, cash revocability forms a necessary complementary measure to using tamper-resistant devices, adding system security while lowering system cost by reducing the level of physical tamper resistance required.

We propose a revocable cash scheme. Cash revocation is achieved efficiently via withdrawing an attacker's ability to further use the system. In several ways the new scheme improves from previous ones (including trustee-based tracing techniques, e.g., [5, 14, 19]), that identify the attacker and revoke cash in terms of publishing a blacklist of coins misused or robbered. Firstly, the new scheme can stop the attacker from further using the system without need of physical confiscation of him/her or his/her devices (physical confiscation is difficult over the network). Secondly, it will achieve a better performance due to avoiding real-time checking through a potentially big coin revocation list. Thirdly, without using trustee, the system serves unconditional privacy for honest users and uses simpler protocols for transactions.

## 1 Introduction

In this paper we present an off-line cash scheme in which coins can be efficiently revoked (stopped from being spent) by using cryptographical means. Electronic cash should be revocable in order to nullify the effect of attacks mounted by a capable attacker (e.g., double spending by reverse engineering tamper-resistant devices, blackmailing the bank [25] or simply bank robbery). For prudent engineering considerations, cash revocability forms a necessary complementary measure to using tamper-resistant devices, adding system security while lowering system cost by

reducing the level of physical tamper resistance required. Further, with an effective revocation mechanism, stringent dependence on using tamper-resistant devices for prevention of double-spending can be relaxed, or even (for home PC users) the requirement of using such devices be dropped. The realistics of using no tamper-resistant devices in a similar electronic commerce application has been manifested by the recently emerging international standardisation approach to bankcard payment transactions, Secure Electronic Transaction (SET), developed by VISA and MasterCard [20]; the first phase of the SET will operate purely as a software system used by the consumers. Considering the unlikelihood in a near future to upgrade a large portion of home PCs with a cheap means to read tamper-resistant devices, independent of using such devices will be desirable allowing early availability of electronic cash to a large number of home PC users.

The remainder of this paper is organised as follows. In the rest of this section we review the related work. In Section 2 we derive a method of issuing an anonymous certificate for being used in our electronic cash scheme. In Section 3 we describe the revokable electronic cash scheme with its security analysed in Section 4. Finally, Section 5 concludes the work of this paper.

### 1.1 Related work

Early proposals for realising off-line electronic cash [7, 8, 9, 10, 15, 16, 17, 21] provided inspirational insight in dealing with the issue of anonymity [6] and the problem of identifying double spender. In order to prevent cheating, these schemes rely on using large sized challenge-response terms during the time of withdrawing coin from the bank (some also require doing so in the payment time), and this is very inefficient. Effort was then turned to deriving more efficient schemes. The "line method" devised by Franklin and Yung [15, 16] started an important approach to avoiding using large challenge-response terms. More efficient "single-term" coins were later achieved by Brands [3, 4] and Ferguson [13]. Eng and Okamoto combined Brands' representation technique with a binary-tree method for a further improved efficiency of divisible coins [12]. In the schemes proposed by Okamoto and Ohta [21, 22], they introduced a notion of electronic license. An electronic license is an integer which can essentially be viewed as a certificate issued by the bank that entitles its customers to spend cash withdrawn from it. When a double spending is detected, the bank can factorise the integer in polynomial time and reveal the identity of the double spender. Using Ferguson's classification on anonymity [13], we understand that the scheme of Okamoto and Ohta should be regarded as serving a so-called *fake privacy*. (We will refer it to as *linkable payments* which means all payments made by the same person can be linked by the bank. See Section 4.2 for a more detailed description on linkable payments and the associated security problems.) In addition to the linkability problem, the construction of an electronic license has to go through a series of complex bit-commitment protocols with security based on several computational complexity assumptions on difficult problems, not all of them are well studied. Yacobi proposed another likable-payments scheme [26] based on using a special property of ElGamal signature scheme [11]: a double spender will be identified by means of exposing his/her private key as a result of being forced to replay a security parameter during double spending. The central idea of Yacobi's scheme is very similar to that of this paper (this was unawared to the author when writing an early version of this paper). Nevertheless,

several important differences between these two schemes should be observed: in our scheme, payments will not be linkable and cash withdrawal does not resort to a painful (phrase used by Yacobi, page 160 of [26], for extreme inefficiency) zero-knowledge proof procedure.

The schemes discussed above are not concerned of the issue of cash revocation. In most of them (e.g., [3, 4, 7, 8, 9, 10, 15, 16, 17]), a payment is essentially related to a unique public-key/private-key pair (different payments are related to different key pairs). Requirement on both off-lineness and anonymity means that it is impossible for these schemes to efficiently revoke an attacker's ability to further use the system even after the attacker has been identified for misuse. Since cryptography plays no role in stopping capable attackers, these schemes have to resort to external means such as police for the purpose. This is potentially dangerous as it is difficult to physically confiscate an attacker or his/her cash devices over the network (considering that the attacker can even propagate the knowledge of misuse to others). In addition to this danger, these schemes may also be expensive and can only have a limited population of users in the near future because of their absolute dependence on quality tamper resistance.

Recently, the issue of crime prevention has been considered in some cash schemes with trustee-based tracing mechanisms, [5, 14, 19]. In these schemes, trustee(s) are employed to escrow users' identities. Without co-operation between the bank and the trustee(s), it is computationally infeasible to compromise the users' anonymity, and it is further assumed that a co-operation requires presence of a warrant from the court. The use of trustee(s) is mainly for the purpose of detecting criminal activities such as money laundering or blackmailing banks [25]. In these schemes, after identification of an attacker, the required cash revocation still recourse to the conventional means of blacklisting all of the coins (public keys) misused or robbered. This is not an acceptable solution to cash revocation because while the system will suffer from a real-time performance penalty, it still allowing the attacker (identified) to continue (double) spending the coins not yet spent.

Cash revocability to be introduced in this paper is achieved from applying public-key cryptography in a standard way. Rather than relating each payment to a unique public-key/private-key pair, each user in our system will have a single pair of keys for spending her/his cash coins. A novel anonymity service will be introduced for certifying public key such that upon seeing the certificate, it is infeasible for any principal (including the certification authority who has issued the certificate) with bounded resources to identify the anonymous holder of the certificate as long as the certificate holder does not misuse the system. When a perpetration is detected, not just can the perpetrator be exposed algorithmically in polynomial time, but more importantly the perpetrator's ability to further use the system will be revoked by using the standard certificate revocation framework [18]. As we have mentioned in the above, the basic idea of double-spending detection in our method is similar to that of Yacobi's scheme. In fact, that scheme can easily be revised to include an efficient cash revocation feature. However, besides the added revocation feature, our scheme also improves from that of Yacobi in other two important aspects: unlinkable payments and single-term withdrawal transaction. Finally we point out that, compared with the trustee-based tracing schemes ([5, 14, 19]), our method is an improvement not only in terms of adding an efficient cash revocation mechanism, but also serving the honest users with unconditional anonymity; moreover, without using trustee, protocols for transactions will also become much simpler.

## 2 Blind Certification of Public keys

Similar to an ordinary certificate for an end-user's public key, a blind certificate is a digital signature created by an CA to combine a public key and its holder's identity. Nevertheless, the identity of a blind certificate's holder is concealed. Using Chaum's blind signature technique [6] in conjunction with cut-and-choose we can devise a protocol for the CA to issue a blind certificate to Alice for certifying her public key. Here we present a technique based on the RSA crypto-algorithm.

Let $(K_{CA}, N_{CA})$ be the RSA public key of the CA for certification purposes. To run *certificate issuing protocol*, Alice will prepare $J$ candidate private/public key pairs $(s_j, v_j)$ for $1 \leq j \leq J$. One pair will be chosen out of these $J$ candidates upon termination of a run of the protocol and the chosen key pair will be denoted as $(s, v)$. Further, let $E_s()$ be a symmetric encryption algorithm keyed by the secret key $s$ (here $s$ coincides to, or is truncated from Alice's private key, more details see below); $H()$ be a secure one-way hash function (e.g., SHA [2]); and $||$ be string concatenation. For $1 \leq j \leq J$, Alice will prepare and send the following $J$ pieces of data to the CA:

Step 1. $A \rightarrow CA:\ H(s_j),\ E_{s_j}(b_j),\ b_j^{K_{CA}} * (H(v_j, E_{s_j}(A)) \,||\, E_{s_j}(A)) (\mathrm{mod} N_{CA})$

Here $A$ denotes Alice's identity; each $b_j$ is called *blinding factor* which is randomly chosen from $Z^*_{N_{CA}}$. Upon receipt these data, the CA will randomly choose $1 \leq i \leq J$ and:

Step 2. $CA \rightarrow A:\ i$

Alice then discloses $J-1$ pairs $(s_j, v_j)$ for $1 \leq j \leq J$ and $j \neq i$:

Step 3. $A \rightarrow CA:\ s_j, v_j$

The CA can verify the correctness of the respective $J-1$ pieces of data received in Step 1, which are not indexed by $i$, to see if Alice has honestly encoded her identity in these data. If the checking passes, then the CA will assume that the remaining item (indexed by $i$) is also correct and will sign it. Alice has 1 in $J$ chance to succeed a cheating, i.e., to let the CA issue her a blind certificate which encodes a wrong identity. The probability to succeed the cheating is sufficiently small if $J$ is sufficiently large. Below we will further see that even if Alice does succeed a cheating, she will still be identified and the blind certificate will still be revoked once she uses the certificate dishonestly.

The CA's signature on the third chunk of the remaining item will yield the following quantity (denoting $b_i$ by $b$ and $(s_i, v_i)$ by $(s, v)$):

$$(b_i^{K_{CA}} * (H(v_i, E_{s_i}(A)) \,||\, E_{s_i}(A)))^{K_{CA}^{-1}} (\mathrm{mod} N_{CA})$$
$$= b * (H(v, E_s(A)) \,||\, E_s(A))^{K_{CA}^{-1}} (\mathrm{mod} N_{CA})$$

Step 4. $CA \rightarrow A:\ b * (H(v, E_s(A)) \parallel E_s(A))^{K_{CA}^{-1}} (\mathrm{mod} N_{CA})$

By dividing $b$ into the data received, Alice obtains the following blind certificate:

$$Blind\_Cert_A = (H(v, E_s(A)) \parallel E_s(A))^{K_{CA}^{-1}} (\mathrm{mod} N_{CA})$$

The certificate and the public key $v$ can be validated using the CA's well-known public key. The CA will archive the following data:

$$Archive_A = H(s),\ A,\ E_s(b),\ b^{K_{CA}} * (H(v, E_s(A)) \parallel E_s(A)) (\mathrm{mod} N_{CA})$$

The property of the blind signature ensures that it is computationally infeasible to find any verifierable relationship between $Archive_A$ and $Blind\_Cert_A$ without knowing $s$. We assume that the CA is a competent party to use appropriate auditing measures to prevent the data $Archive_A$ from being deleted or altered.

The blind certification technique will be used in conjunction with a special signature scheme (to be introduced in the next section) to serve the following function: honestly using the system the user will be served with anonymity protected; abusing the system will lead to discovery of the user's private key $s$ and thereby her/his identity by decrypting the archived data. Upon identification, the matching public key and the certificate will be revoked promptly by publishing them in the CRL.

We should point out that although the cut-and-choose method used here only thwarts cheating in a linear stringency, this is adequate. This is because even if Alice does have succeeded a cheating against 1-$J$ odds by having encoded a wrong identity into a blind certificate, the success will not help her to gain a very clear advantage other than still being identified as a perpetrator. Upon abusing the system that leads to discovery of her private key, the matching public key and the certificate can be revoked regardless of whether or not the revealed identity matches the cleartext $A$ in $Archive_A$. Note that it is the revocation of the certificate and the public key that really matters. Alice cannot complain about the revocation (e.g. complaining the CA manipulating data) if she did have successfully cheated during the time of certificate issuing because a wrong identity has been encoded in the very blind certificate that she is holding.

For simplicity, we have omitted presentation of some implementation related information. For instance, a blind certificate should contain information pointing to the network directory services (e.g., the trust chains and the CRL locations); also, a private key $s$ should contain a substring of bits (e.g., two least significant bytes) to uniquely point to the CA who archives the data "$H(s), \cdots$" so that, with a revealed private key $s$, the dishonest user can quickly be identified from the right CA. Also we assume that the cryptographic codings contain properly redundant information needed to foil various forms of brute-force searching targeted at the private key $s$.

## 2.1 Schnorr's Digital Signature as a One-Time Signature Scheme

We know that after issuing a blind certificate to Alice, it becomes intractable for CA to link Alice with $Blind\_Cert_A$. So it is now possible for Alice to abuse her anonymity. Therefore a

blind certification technique should be combined with a means to counter misuse. Below we provide such a measure. The method is based on applying Schnorr's digital signature scheme [23, 24]. We start by presenting Schnorr's original scheme.

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 $Z_q^*$ 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:

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

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

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

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 $h(m \,\|\, z)$. 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 $e' \neq e \pmod{q}$. 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 $r - r' \neq 0 \pmod{q}$ 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.

## 3 Revocable Electronic Cash

We now apply the blind certification technique introduced in the previous section to devise the revocable electronic cash scheme. The cash 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.[1] 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. $A \rightarrow B$ : $request$, $(b^{K_B} * f(x \parallel v)) \bmod N_B$

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

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 [20]). 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:

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

In real application, the bank can publish a set of different public keys for being used to withdraw money for a given period of time (e.g., publishing 31 keys to use in a month and repeating the same keys in the next month). When a blackmail to the bank like the one in [25] occurs, the bank can revoke its own respective public keys and ask the honest customers to deposit coins up to the amount withdrawn which has of course been recorded by the bank.

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.

[1] 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).

In contrast, in Yacobi's scheme [26], coin withdrawal goes through a very inefficient procedure of zero-knowledge proof. Yacobi did not provide a concrete protocol for money withdrawal and we believe it is not a simple cut-and-choose method. We also think that the choice of using zero-knowledge proof is related to the linkable payments property in Yacobi's scheme. We will further discuss this issue in Section 4.2.

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

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

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

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:

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

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 certificate $Blind\_Cert_A$. 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 $\Delta$-CRL from the network directory services to update his local copy of CRL (see Section 12.6 of [18]). If $v$ is properly backed by the blind certificate $Blind\_Cert_A$ 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 $h(Coin \parallel M \parallel DateTime \parallel z)$ 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 $f(x \parallel v)$ 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 Strong 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. The anonymity service is in a strong sense in that, collusion among banks, all merchants and CA's will not result in any computationally feasible way to identify the spender of a coin.

### 4.2 Practical Unlinkability

Unlinkability is a notion stronger than anonymity. Ferguson differentiates the two notions by *privacy* and *fake privacy*, respectively [13]. Rigorously speaking, unlinkability means an inability to determine whether two coins have been spent by the same person (though without knowing the person's identity). A weaker meaning is that a person's spending pattern cannot easily be determined (usually by the bank). An unlinkability service (even in the weaker sense) is desirable because in reality a spender's identity may be accidentally disclosed outside the electronic cash mechanism, and without such a service, a person's spending history and future will be known once her/his identity is accidentally disclosed. We will reason that the blind certification technique allows us to achieve a practical quality of unlinkability: it requires an impractically large scale of collusion between the bank and merchants to determine a person's spending pattern. 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 blind certificates are deposited together with coins, then coins can be partitioned by public keys and/or certificates; all coins in the same partition are spent by the same person. Depositing public keys or blind

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

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

(If $e$ is an even number, i.e., $e = 2t$, then before computing (10), the bank can first find the square root of $v^e \bmod p$, and carries on square rooting until $t$ becomes an odd number that allows (10) to be computed.)

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 blind 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 $E_M(z)$.

Note that even the bank has successfully collected data needed to link an anonymous person's payments, 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.

In contrast, in Yacobi's scheme [26], all payments are linkable. We should point out that the linkability in there is not because of the merchant's deposit of the spender's public key together with the spending signature. It is because of a property of ElGamal signature scheme that will allow the bank to determine the spender's public key from the signature alone (an algorithm should be along the lines of (9) and (10) above). We believe that the linkability problem forms a contributive element to why coin withdrawal in Yacobi's scheme has had recourse to a painful zero-knowledge proof. Roughly speaking, in Yacobi's scheme, the merchant need not verify the authenticity of the spender's public key; the bank has done this during the withdrawal time and can re-verify it from the data deposited. Thus, the authenticity of the spender's public key that the bank has blessed during the withdrawal time has to be done in a zero-knowledge-proof way because the withdrawal protocol cannot be run anonymously.

### 4.3 Correctness in cash revocation

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) Double spending by Alice.** 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 substring of bits that points uniquely to the CA which has issued the blind certificate to Alice.) Upon seeing the revealed key pair, the CA will revoke the public key $v$ by publishing it onto the $\Delta$-CRL. Alice's identity will also be revealed.

**Case (ii) Data replay or depositing bad data by the merchant.** 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) Collusion between Alice and the merchant.** A collusion can make sense only if it does not lead to identifying Alice and revoking her cash. 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 blind 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 Conclusion

We conclude the paper with a summary of the features of the electronic cash scheme using the blind certification technique.

**Revocable cash.** Double spending can be stopped within the cash mechanism. Cryptographical means for cash revocability is a unique feature not available in any previous off-line electronic cash schemes. With this feature, the dependence on using physical devices with tamper-resistant property can actually be dropped and so lightweight cash systems can readily be developed and used over the Internet. Of course, in a point-of-sale environment, using such physical devices by the spender will undoubtedly be helpful in protecting 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 within. (In fact, they should keep safe backup of the data.) From prudent engineering considerations, cash revocability forms a necessary complementary measure to using physical devices.

**Strong anonymity for the spender.** As long as the spender does not double spend, no party in the system can have any computationally feasible way to identify who has spent the money. The anonymity service is strong in that, collusion among all parties will not achieve a computationally feasible method to compromise the quality of the service.

**Practical unlinkability.** The bank does not have a computationally feasible way to link payments nor determine the spending pattern of an anonymous person. This is true in a weak sense in that collusion among the bank and merchants will achieve coin linkage. However, collusion of a large scale (proportional to the volume of coin-deposit) is required in order to collect useful information. Furthermore, data gathered in the history are not useful for linking coins to be deposited in the future and further collusion is necessary for that purpose.

**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: to compute hash functions is efficient, and to generate a spending signature has a linear time complexity (merely to release a dot in a line). These are attractive features for making point-of-sale payment using cheap smartcards. Further, cash can easily be withdrawn from a foreign bank.

## 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 for their helpful comments on early versions of this paper.

# References


[1] Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). Federal Register, v.56, n.169, August 1991.

[2] NIST FIPS PUB 180, Secure hash standard. National Institute of Standards and Technology, U.S. Department of Commerce, DRAFT, April 1993.

[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] E. Brickell, P. Gemmell, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making anonymous change. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1995 pages 457–466.

[6] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology — Proceedings of Crypto'82*, pages 199–203. Plenum Press, 1983.

[7] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[8] D. Chaum. Privacy protected payments: Unconditional payer and/or payee untraceability. In *Smartcard 2000*. North Holland, 1989.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] N. Ferguson. Single term off-line coins. In *Advances in Cryptology — Proceedings of EUROCRYPT'93 (LNCS 765)*, pages 318–328. Springer-Verlag, 1994.

[14] E. Fujisaki and T. Okamoto. Practical escrow cash systems. In *Proceedings of 1996 Cambridge Workshop on Security Protocols*, to appear LNCS Springer-Verlag. April, 1996.

[15] M. Franklin and M. Yung. Towards provably secure efficient electronic cash. Technical Report: TR CUCS-018-92, April 1992.

[16] 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.

[17] 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.

[18] 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.

[19] M. Jakobsson and M. Yung. Revokable and versatile electronic money. In *3rd ACM Conference on Computer and Communication Security.* March, 1996.

[20] MasterCard and Visa Secure Electronic Transaction (SET) (see, e.g., `http://www.visa.com/`), February 1996.

[21] T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology — Proceedings of CRYPTO'91 (LNCS 576)*, pages 324–337. Springer-Verlag, 1992.

[22] T. Okamoto. An efficient divisible electronic cash scheme. In *Advances in Cryptology — Proceedings of CRYPTO'95 (LNCS 950)*, pages 438–451. Springer-Verlag, 1995.

[23] 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.

[24] C.P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[25] S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computer and Security*, 11 (1992), pages 581–583.

[26] Y. Yacobi. Efficient electronic money (extended abstract). In *Advances in Cryptology — Proceedings of Asiacrypt'94 (LNCS 917)*, pages 153–163. Springer-Verlag, 1995.