Digital Coins based on Hash Chain

Khanh Quoc Nguyen, Yi Mu, Vijay Varadharajan
Distributed System and Network Security Research Unit
Computing Dept., University of Western Sydney, Nepean
Australia

Abstract

We propose three digital coin schemes for doing electronic transactions over the Internet. These schemes are based on one-way hash functions. The first scheme uses an ordinary one-way single hash function chain, which forms the basis of a simple micropayment scheme. The second scheme is based on a new technique called double-locked hash chain technique which overcomes some of the drawbacks found in single hash chain schemes. The third micropayment scheme presents a method to achieve client anonymity.

1 Introduction

The explosive growth of the Internet, coupled with its potential use as the core of the national and global information infrastructure, is creating a huge field of business opportunities. This includes a wide range of online services, ranging from home shopping to unforeseen network-based ventures. Activities can include accessing library books electronically from home, providing education electronically, viewing movies, news on demand, purchasing items from electronic catalogues, shared electronic white boards or more generally conducting business transactions electronically.

In carrying out electronic commerce over the Internet, it will be necessary to handle high volume of small transactions that are of low value; typically, each transaction's value may range from a few cents to a few dollars. With the current transaction fee structure, some of the existing credit-card base protocols, such as NetBill [9] and SET [10], do not seem to be suitable for supporting small or micropayments.

To support micropayments, a high degree of efficiency is required; otherwise the cost can exceed the value of the payments. There may also be a requirement for such transactions to be processed quickly given that there can be a large number of them. That is, the number of computations required per payment should be as minimum as possible. Recently, several micropayment schemes have been proposed; these include Millicent???, PayWord[7, 8] and NetCard [1]. Some of these protocols make use of hash functions instead of expensive public key operations and are based on decentralised validation of electronic payments at the vendor's server, thereby removing any unwanted communications. Nevertheless, they all are credit-based payment systems, so they might not be suitable for casual customers. Several cash based electronic payments have also been proposed in literature [3, 5, 6, 11, 14] but most of them are not suitable for micropayments for efficiency reasons.

In this paper, we study three cash-based micropayment protocols based on hash chains. The first scheme is proposed using a single one-way hash function chain, which forms a simple micropayment scheme. The second scheme is based on a new technique called double-locked hash chain technique which overcomes some of the drawbacks found in single hash chain schemes.

The third micropayment scheme presents a method to achieve client anonymity. These three protocols are useful in creating a platform for micropayments to be achieved in an Internet environment.

The paper is organised as follows. Section 2 outlines the micropayment environment and requirements for micropayment schemes. Section 3 introduces the notion of one-way hash chains. Sections 4 ,5 and 6 describe the three micropayment schemes respectively.

2 Micropayment Environment

In the usual manner, we will assume that there are three main parties namely, clients ($\mathcal{C}$), vendors ($\mathcal{V}$) and a financial institution (which we will refer to as the bank - $\mathcal{B}$) in the micropayment environment. The bank is responsible for managing accounts of both clients and vendors. It is the only party that can authorise digital coins for use in the system. The clients purchase services from the vendors using digital coins. The vendors deliver the requested services to the clients and deposit the digital coins at the bank. We also require a trusted public key authority TA who can sign public key certificates for all the parties involved.

A micropayment scheme is required to satisfy the following security requirements:

- *Efficiency:* Computations involved in the payment mechanisms should be as minimum as possible and be of low cost.
- *Hardware Independence:* No special physical device should be necessary to ensure system security.
- *Forgery:* It should not be economically feasible for an attacker to forge the electronic token; in other words, forging digital coins should be more expensive than buying them.
- *Off-line Payment:* The payment scheme should not require the bank to be involved on-line during any transaction between clients and vendors.
- *Multiple Transactions and Service Providers:* A client should be able to do a number of micropayment transactions with several different service providers on a single day.

There may be another requirement for electronic payments namely that of client anonymity. There may be several reasons for considering the anonymity feature to be optional. One such reason is the computational cost as it is likely to involve additional costs. .

3 Hash Coin Chain

Public key digital signature schemes have been widely used in electronic payment schemes as they help to minimise disputes that may arise. However public key schemes are computationally expensive; therefore, in certain circumstances, it may not be practical to require clients to sign each payment with public key signature schemes. Micropayment schemes such as the PayWord scheme make use of hash functions which can be done much faster compared to public key schemes.

For example, the PayWord scheme (a credit-based scheme) [7] generates a chain of PayWords, $w_1, w_2, ..., w_n$, using hashing operations in the reverse order

$$w_i = h(w_{i+1}), \quad (i = n-1, n-2, ..., 0), \qquad (1)$$

The client then signs the root w_0 using his secret key of a public key system. The client sends the signed w_0 to the vendor and subsequently sends each PayWord to the vendor, starting from w_1. The verification of all the PayWords in the chain is based on Equation (1) and the root of the chain w_0. All PayWords are associated with the signed root via the hash function. This method ensures that all PayWords come from the client and no one else. The security of the PayWords is dependent on the difficulty of calculating the inverse of the hash function.

4 First Scheme

All parties, Client ($\mathcal{C}$), Vendor ($\mathcal{V}$), and Bank ($\mathcal{B}$), are assumed to have public-key capability. Let PK_X denote the party X's public key, SK_X be the corresponding secret key, $Cert_X$ be the corresponding public key certificate signed by a trusted authority such as the bank. Let $[...]_{SK_X}$ denote the signature using SK_X, and let $[...]_{PK_X}$ denote encryption using PK_X.

4.1 Withdraw Phase

Consider the case where the client $\mathcal{C}$ wishes to withdraw some n digital coins from the bank $\mathcal{B}$.

1. $\mathcal{C}$ generates a hash chain $\{c_i\}$, $(i = 0, 1, 2, \cdots, n)$, which is computed in reverse order recursively,
$$c_i = h(c_{i+1}), \quad (i = n-1, n-2, \cdots, 1, 0) \tag{2}$$
2. $\mathcal{C}$ requests to withdraw n cents by sending $[[n, c_0]_{PK_B}]_{SK_C}$ and $Cert_C$ to the bank.
3. $\mathcal{B}$, upon receiving the message, verifies $\mathcal{C}$'s certificate, validates $\mathcal{C}$'s signature, decrypts it with SK_B, and signs $h(n\|c_0)$ with its secret key SK_B to form
$$\mathcal{R} = [h(n\|c_0)]_{SK_B}. \tag{3}$$
$\mathcal{B}$ then sends $\mathcal{R}$ to $\mathcal{C}$ and deducts n cents from $\mathcal{C}$'s account.
4. $\mathcal{C}$ verifies $\mathcal{B}$'s signature on $\mathcal{R}$ and the withdrawal is completed.

Now C has n cents to spend.

4.2 Payment and Deposit Phases

For each hash chain, the client must start to spend the hash chain from the first coin c_1. Without the loss of generality, we assume that the client has spent $(j-1)$ coins and now wishes to spend the coins, $c_j, c_{j+1}, \ldots$ The payment protocol is as follows:

1. $\mathcal{C} \to \mathcal{V} : Cert_C, [c_0, n, \mathcal{R}]_{SK_C}$.
2. $\mathcal{V} \to \mathcal{C} : [ok, \mathcal{R}]_{SK_V}$.
3. $\mathcal{C} \to \mathcal{V} : (j, c_j), (j+1, c_{j+1}), ..., (k, c_k)$.

In Step 1, $\mathcal{C}$ sends its public key certificate and the payment commitment signed with SK_C to $\mathcal{V}$. In Step 2, $\mathcal{V}$ validates the information. If the checks are successful, V sends the ok token signed with SK_V to $\mathcal{C}$. In Step 3, $\mathcal{C}$ pays $\mathcal{V}$ $k-j$ coins. The validation of the coins is done as follows:

$$\overbrace{h(h(h(...h}^{k}(c_k)...))) \overset{?}{=} c_0.$$

At end of the day, the vendor can send the client's payment commitment $[c_0, n, \mathcal{R}]_{SK_C}$ and the first coin (j, c_j) and the last coin (k, c_k) to the bank. Upon validation of the coins, the bank deposits $(k-j)$ cents to the vendor's account. The client is allowed to spend the remaining of the chain using the same process.

The protocol is efficient as the number of public key operations are kept to a minimal level. To validate each coin, only one hash operation is needed.

Forging our coins is expensive. To spend each coin, one needs the bank signature on the first element (c_0) of the chain. All coins in the chain are linked to this element. So to forge electronic coins, one must either forge the bank signature on c_0 or be able to compute c_i from c_{i-1}. Given the assumption about strong hash functions, both these scenarios are assumed to be infeasible. Furthermore, since the client is not anonymous, s/he cannot double spend as the bank that has the full record of each spent coin chain.

5 Second Scheme

This scheme is based on double hash chains

$$\{c_0, c_1, \ldots, c_n; c'_0, c'_1, \ldots, c'_n; \mathcal{R}\} \tag{4}$$

where c_n and c'_n are integers chosen at random and $\mathcal{R} = h(n||c_0||c'_0)$ should be signed by the bank. "||" denotes bit-string concatenation. The hash chains are computed in terms of $c_i = h(c_{i+1})$, $c'_i = h(c'_{i+1})$, $(i = n-1, n-2, ..., 0)$.

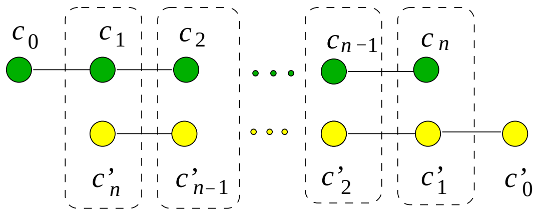

Figure 1: Illustration of the Second Scheme.

The second scheme is illustrated in Figure 1. Each digital coin in the double hash chain is represented by the pair of hash values (c_i, c'_{n-i}). The coin has the following features:

(1) knowing a coin (c_i, c'_{n-i}), one can readily validate it in terms of $\mathcal{R}$, and

(2) knowing a coin (c_i, c'_{n-i}), one cannot compute other coins

5.1 Withdraw Phase

In this phase, a client wishes to withdraw some n digital coins from the bank.

1. $\mathcal{C}$ generates double hash chains using Eq. (4).
2. $\mathcal{C}$ computes $\mathcal{R}^{PK_B}$. We have omitted the modulus for brevity. The modulus is a large composite number which is a product of large primes.

3. $\mathcal{C}$ requests to withdraw n cents by sending $[n, \mathcal{R}^{PK_B}]_{SK_C}$ and $Cert_C$ to the bank.

4. $\mathcal{B}$, upon receiving the message, verifies $\mathcal{C}$'s certificate using PK_{TA} and $\mathcal{C}$'s signature using PK_C, decrypts $\mathcal{R}^{PK_B}$ using SK_B, and then computes the signature $(n\mathcal{R})^{SK_B}$. $\mathcal{B}$ then sends it to $\mathcal{C}$, deducts n cents from $\mathcal{C}$'s account, and stores n and $\mathcal{R}$ (assuming each coin is worth one cent).

5. $\mathcal{C}$ then verifies $\mathcal{B}$'s signature and denotes

$$\mathcal{R}_0 \equiv (n\mathcal{R})^{SK_B}. \tag{5}$$

Now $\mathcal{C}$ has n cents to spend.

5.2 Payment and Deposit Phases

The payment protocol is as follows:

1. $\mathcal{C} \rightarrow \mathcal{V} : Cert_C, [c_0, c'_0, n, \mathcal{R}_0]_{SK_C}$.
2. $\mathcal{V} \rightarrow \mathcal{C} : [ok, \mathcal{R}_0]_{SK_V}$.
3. $\mathcal{C} \rightarrow \mathcal{V} : (1, c_1, c'_n), (2, c_2, c'_{n-1}), ..., (l, c_l, c'_{n-l+1})$.

In Step 1, $\mathcal{C}$ sends $\mathcal{V}$ its public key certificate and the payment commitment signed with SK_C. In Step 2, V validates the information. If these checks are successful, then $\mathcal{V}$ sends the *ok* token signed with SK_V to $\mathcal{C}$. In Step 3, $\mathcal{C}$ pays $\mathcal{V}$ a set of coins. The validation of the coins is done using

$$\overbrace{h(h(h(...h}^{l}(c_l)...))) = c_0, \quad \overbrace{h(h(h(...h}^{n}(c'_n)...))) = c'_0.$$

A current coin can be validated with the information of the previous coin. At end of the day, the vendor sends the client's payment commitment $[c_0, c'_0, n, \mathcal{R}_0]_{SK_C}$ and the last coin (l, c_l, c_{n-l+1}) to the bank. Upon validation of the coins, the bank deposits l cents to the vendor's account (assuming each coin is worth one cent).

The client can continue spending money with the same or a different vendor subsequently. The protocol for the same vendor is simple and hence is not included here. Now we consider how the client spends the remaining coins with a different vendor.

1. $\mathcal{C} \rightarrow \mathcal{V} : Cert_C, [c_0, c'_0, n, \mathcal{R}_0, (l+1, c_{l+1}, c'_{n-l})]_{SK_C}$.
2. $\mathcal{V} \rightarrow \mathcal{C} : [ok, l+1, \mathcal{R}_0]_{SK_V}$
3. $\mathcal{C} \rightarrow \mathcal{V} : (1, c_{l+1}, c'_{n-l}), (2, c_{l+2}, c'_{n-l-1}), ..., (n-l, c_n, c'_1)$.

After Step 1, the vendor should validate client's signature and all information embedded in the signature. This includes validating the coin $(l+1, c_{l+1}, c'_{n-l})$ using

$$\overbrace{h(h(h(...h}^{l+1}(c_{l+1})...))) = c_0, \quad \overbrace{h(h(h(...h}^{n-l}(c'_{n-l})...))) = c'_0,$$

and verifying the coin chain in terms of

$$\overbrace{h(h(h(...h}^{n-l-1}(c_n)...))) = c_{l+1}.$$

With (l, c_{l+1}, c'_{n-l}), the second vendor cannot produce the previous coins in the same chain which have been spent with the first vendor, since he cannot compute $c_{n-l+1}, c_{n-l+2}, ..., c_n$. The second vendor deposits the coins by sending the client's commitment $[c_0, c'_0, n, \mathcal{R}_0, (l+1, c_{l+1}, c'_{n-l})]_{SK_C}$ and the last coin $(n-l, c_n, c'_1)$ to the bank.

6 Third Scheme

Both of our previous schemes did not provide client anonymity; this scheme consider this issue. The basic technique used here involves Schnorr's one-time signature [12].

6.1 Anonymous certificate

If $\mathcal{C}$ wishes all of its payments to be anonymous, $\mathcal{C}$ contacts TA and asks for an anonymous certificate. Upon their agreement, $\mathcal{C}$ pays TA a small amount of money for the anonymous service and TA issues an anonymous certificate $[OK, g^U]_{SK_{TA}}$, where SK_{TA} is TA secret key and U is the identity of $\mathcal{C}$ registered with the TA. The certificate should also include some timestamp and expiry time but we will omit them here for convenience. TA must be trusted by $\mathcal{C}$. Alternatively, the client can obtain the anonymous certificate from the bank using zero-knowledge proof[13]. This might lead to some extra costs for the anonymous service due to the inefficiency of zero-knowledge proof.

6.2 Withdrawal Phase

If the client wishes to withdraw n coins, s/he chooses a random number c_n and computes $c_i = h(c_{i+1})$ for $\forall i \in \{1, \ldots, n-1\}$. For each c_i, $\mathcal{C}$ uses blind signature technique [4] to withdraw an anonymous coin from $\mathcal{B}$ using the following protocol:

1. $\mathcal{C}$ generates a random number x_i, computes $m_i = h(c_i \| g^{x_i})$ and sends m_i to $\mathcal{B}$.
2. $\mathcal{C}$ then uses blind signature technique [4] to obtain the bank signature on m_i by choosing a blind factor r_i and sending to $\mathcal{B}$ $t_i = r_i^{PK_B} \cdot m_i \bmod n$. $\mathcal{B}$ signs the value of t_i using the RSA scheme and returns the signature $t'_i = t_i^{SK_B}$. The client then removes the blind factor r_i to obtain $m'_i = t'_i / r_i = m_i^{SK_B}$,

After the withdrawal phase, $\mathcal{C}$ has each coin C_i in the form of $[h(c_i \| x'_i)]^{SK_B} \bmod n$, where $x'_i \equiv g^{x_i}$. This coin is unforgeable unless the factorization of n is known [5]. For each coin C_i, $\mathcal{C}$ stores $[c_i, x_i, x'_i, m'_i]$.

6.3 Payment and Deposit Phases

When the client wants to spend the chain $C_1, C_2, \ldots, C_n$, s/he must spend them in the order $C_1, C_2, \ldots, C_n$. Without the loss of generality, we assume that $\mathcal{C}$ has already spent all the coins $C_1, \ldots, C_{i-1}$ in some previous payments. Now if $\mathcal{C}$ wishes to pay some coins to $\mathcal{V}$, $\mathcal{C}$ must send them in the exact sequence $C_i, C_{i+1}, \cdots, C_j, \cdots$ as follows:

1. $\mathcal{V}$ generates a random challenge a and sends it to $\mathcal{C}$. This challenge should be unique for each transaction. For example, it can be computed as $a = h(\mathcal{V} \| Date \| Time)$.

2. $\mathcal{C}$ computes the response $b = x_i - Ua \bmod q$ to the challenge a and sends it along with $[OK, g^U]_{SK_{TA}}$, c_i, x_i', m_i' to $\mathcal{V}$, where g is a large prime number. The response b is the Schnorr's one-time signature on the message a and x_i is a one-time number. $\mathcal{V}$ accepts the coin if and only if the anonymous certificate is valid and C_i is signed by the bank and b is the client's Schnorr's one-time signature on C_i.

3. For every coin C_j, thereafter, $\mathcal{C}$ sends $[x_j, c_j, m_j']$ to $\mathcal{V}$. $\mathcal{V}$ accepts the coin C_j if the bank's signature on the coin is valid and the coin is indeed the next coin in the chain. This can be done by checking if $h(c_j) = c_{j-1}$.

At the end of the day, $\mathcal{V}$ deposits all the received coins at $\mathcal{B}$ using the challenge a. $\mathcal{B}$ goes through exactly the same verification process as $\mathcal{V}$ did in payment phase. If the checks are successful, then $\mathcal{B}$ pays $\mathcal{V}$ an equivalent value of money.

6.4 Discussion

Double Spending : Payment schemes that provide client anonymity are vulnerable to double-spending attacks. We will now show that double-spending cannot be achieved in this protocol. For convenience, let us refer to the first coin C_i as the **signed coin** and all other coins C_j as the **normal coins**.

In this scheme, the first double-spent coin(C_i) must be the first coin in at least one transaction, So there are only two possibilities: C_i is spent as either signed coin in the both transactions or as a signed coin in one transaction and a normal coin in another transaction.

- **Signed-Signed coin:** $\mathcal{C}$ spends C_i as a signed coin twice, i.e. for two different challenges a, a', B has $b = x_i - Ua \bmod q$, $b' = x_i - Ua' \bmod q$. B can easily find U by computing:
$$U = \frac{b - b'}{a' - a} \bmod q$$

- **Signed-Normal coin:** C spends C_i twice, once as a normal coin, the other as a signed coin. B therefore has a and $x_i - Ua$ from the signed coin, and x_i from the normal coin. These are sufficient to compute U.

So in either case, the value U can be computed. The evidence is *undeniable* because U is client's secret information, which is infeasible for the bank to compute unless the client had double-spent.

Anonymity: Client anonymity is protected against vendors and the bank. This is because, for all payment transactions, the client only shows his anonymous certificate. On the other hand, as the coins are blindly signed by the bank the bank cannot trace any particular coin to any client.

Coin Forgery: As each coin is signed by the bank, forging coins is equivalent of breaking RSA digital signatures.

Efficiency: Only one online computation is required per payment transaction. This is more efficient than currently known electronic payment schemes that allow client anonymity. One online computation that does not involve discrete exponential computations can be acceptable for micropayments.

References

[1] R.Anderson, C.Manifavas, C.Sutherland, "NetCard - A Practical Cash System", preprint , *Cambridge University*, 1996.

[2] Cybercash, "The Cybercash^{TM} system - how it works", *http:// www.cybercash.com/ cybercash/ cyber2.html.*

[3] S.Brands, " Untraceable off-line cash in wallet with observers", *Advances of Cryptology - CRYPTO '93 Proceedings*, Springer-Verlag,1994, pp.302-318.

[4] D.Chaum, " Security without Identification: Transaction systems to make Big Brother obsolete," *Communications of ACM, vol.28, no.10, pp.1030-1044, Oct.85.*

[5] D.Chaum, A.Fiat and M.Naor, "Untraceable electronic cash", in *Advances in Cryptology - CRYPTO '88 Proceedings*, pp.319-327,1990.

[6] N.T.Ferguson, " Single Term Off-Line Coins",*Advances in Cryptology - EUROCRYPT '93 Proceedings*, Springer-Verlag, 1994, pp.318-328

[7] R. L. Rivest and A. Shamir, "Payword and MicroMint: Two simple micropayment schemes," in *Proceedings of RSA '96 conference*, 1996. Available at:
<http://theory.lcs.mit.edu/rivest/>.

[8] Y. Mu, V. Varadharajan, and Y.-X. Lin "New Micropayment Schemes Based on PayWords," in *Proceedings of 2nd Australasian Conference on Information Security and Privacy* (ACISP '97), Lecture Notes in Computer Science 1270, pp283-293, Springer Verlag, 1997

[9] M.Sirbu, J.D. Tygar, " NetBill: An Internet Commerce System Optimised for Network Delivered Services", *Carnegie Mellon University Technical Report.*

[10] Secure Electronic Transaction, *http://www.visa.com/cgi-bin/vee/sf/set/intro.html.*

[11] T.Okamoto, K.Ohta, " Universal Electronic Cash", *Advances of Cryptology - CRYPTO '91 Proceedings*, Springer Verlag, 1991.

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

[13] W. Mao, " Blind Certification of Public Keys and Off-Line Electronic Cash", HP Laboratories Technical Report, HPL-96-71, May 1996.

[14] Y.Yacobi, "An efficient off-line cash", *Advances of Cryptology - Asiacrypt '94 Proceedings* , Springer-Verlag, 1994.

[15] R.Rivest, " Perspectives on Financial Cryptography", *Invited talk given at the Financial Cryptography '97 conference*, 1997.