Single-Term Divisible Electronic Coins

Tony Eng[1] and Tatsuaki Okamoto[2]

[1] MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA
Email: tleng@theory.lcs.mit.edu
[2] NTT Laboratories
Nippon Telegraph and Telephone Corporation
1-2356, Take, Yokosuka-shi, Kanagawa-ken, 238-03 Japan
Email: okamoto@sucaba.ntt.jp

Abstract. In the literature, only one "divisible" off-line electronic cash scheme has been presented [OO91]. In this paper, we present the construction of more efficient "divisible" off-line electronic coin schemes that are "single-term". We examine some coin systems based on the "disposable authentication" paradigm [OO89], and show that a specific type of "disposable authenticated" coin system can be extended to handle divisible coins using our techniques.

1 Introduction

Recently, much research has been performed in the area of off-line electronic currency [Bra93, CFN88, DP92, Fer93, FY93, OO89, OO91]. Protocols have been proposed enabling consumers to withdraw "electronic coins" from a bank, and later spend these coins at a shop in an "off-line" manner. Here, off-line refers to the property that communication with a bank or authorized center is unnecessary during the payment protocol. A "divisible" coin worth some amount of money, say \$x, is a coin that can be spent several times as long as the sum total of all its the transactions does not exceed \$x.

So far, only Okamoto and Ohta have shown the construction of a divisible off-line electronic cash system in [OO91]. Brands touched upon the possibility of realizing divisibility with his scheme [Bra93], but did not achieve it in the sense of [OO91]. Pailles' divisible coin construction[Pai92] is inefficient. The communication complexity during payments (i.e., the size of a divisible coin) is linear in N where $N =$ (the total coin value)/(minimum divisible coin value). A system in which a coin worth \$367 consists 367 \$1 coins is a rather unwieldy and inefficient divisible cash system. In contrast, the communication complexity of payments in [OO91] is of the order of $\log N$. (This is the real reason why a binary tree is used in [OO91].)

However, there are two shortcomings of [OO91]: the amount of communication that occurs between the merchant (shop) and the bank, and the required

memory size of the database maintained by the bank. Since their scheme utilizes a cut-and-choose method, a paid/deposited coin consists of many terms (e.g., 40 terms), and hence, the resulting complexities can be quite large.

On the other hand, Brands [Bra93] and Ferguson [Fer93] have proposed "single term" coin systems, which do not utilize the cut-and-choose method, but coins in these schemes are not divisible.

In this paper, we exhibit a divisible off-line electronic coin system which resolves the efficiency problem of the Okamoto-Ohta scheme. Moreover, our new scheme is a single-term coin system. The communication complexity and memory requirements of our scheme are less than 1/10 of those of the Okamoto-Ohta scheme.

In fact, we will see later that our techniques can be applied to certain basic coin schemes that use the technique of "disposable authentication"[OO89] to catch double spenders.

The paper is organized as follows: Section 2 creates a basic framework from which we proceed to explain our protocols. In particular, this section includes notation, assumptions, a brief overview of our approach to realizing divisibility, and an informal description of the "key trick" (disposable authentication) used in our construction of divisible coins. Section 3 presents our single-term divisible cash scheme based on discrete logarithms. Section 4 examines the security and efficiency of the scheme proposed in Section 3, Section 5 shows a general construction based on disposable authentication, and Section 6 concludes this paper.

2 Basic Framework

2.1 Notations and Assumptions

Let p and q be large primes such that $q|(p-1)$. The notation $x \in_R X$ means that x is randomly and uniformly selected from X. Also, let $||$ denote concatenation and let $|x|$ be the length of the binary representation of x. If b is a bit, let $\overline{b}$ represent the negation of b. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{2|q|}$ be a polynomial-time computable one-way hash function.

For the remainder of this paper, let the root node of a binary tree be denoted by n_0. We will reference all nodes from the root in the following manner: consider the path from the root node to a given node, and let '0' represent a left branch and '1' represent a right branch. In this manner, the children of the root node are n_{00} and n_{01}, and their children are n_{000} and n_{001}, and n_{010} and n_{011} respectively. Note that there is no node with the label n_1.

2.2 Overview of Divisibility

We will adopt a binary tree approach as [OO91] did. Each coin of worth w is associated with a tree of $(1+\log w)$ levels and w leaves. The tree can be thought of as a collection of w paths of length $(1+\log w)$, each originating from the root and terminating in a leaf. Call these *routes*.

Each node of the tree represents a certain denomination. The root node is assigned a monetary value of w, and the value of all other nodes is found by halving the value of the node's parent. With this tree, we will show that for a single coin of worth w, it will be possible for a consumer to engage in several transactions, such that the sum total of the amounts of each transaction is less than or equal to w.

We can think of each node as being represented by a line, in such a way so that the consumer's identity is encoded in the line's parameters, and the lines of nodes belonging to the same route are related in some way. A node becomes "used" when the consumer reveals a point on the node's corresponding line and some information about the lines of ancestor nodes. During each payment, a subset of nodes (actually, a forest of trees) will become used; in particular, a set of previously unused/unspent nodes whose values sum up to the denomination being spent. After several transactions have occurred, in which parts of the same coin are spent, a collection of these used nodes will result.

Divisibility can then be implemented under the following invariant:

There is at most one "used" node on any route.

Having at most one used node on any route implies that the set of transactions involving the coin so far is legitimate and vice versa. Violation of spending rules will result in violation of this invariant. If the invariant is not upheld, then there are at least two used nodes along some route, and the defining line equation of one of these nodes can be recovered. Note that in our scheme, this also includes the case when the *same* node has been spent twice (as we will make use of different challenges to obtain two distinct points). Here, detection as well as identification of the perpetrator, will ensue.

2.3 Background Technique

We make use of "disposable authentication" techniques [OO89] to construct our divisible coin systems. By applying these techniques, we adapt a modified version of the Schnorr identification scheme (the Okamoto scheme [Oka92], and more generally the Brands scheme for "representation problem" [Bra93]) for our purposes.

Briefly, in [Oka92], a Prover (P) has a public key $m = {g_1}^{x_1} {g_2}^{x_2} \bmod p$ (the order of g_1 and g_2 is q in Z_p^*) and proves to a Verifier (V) that it knows private key (x_1, x_2) using the following protocol: First, P randomly selects (r_1, r_2) and calculates $\beta = {g_1}^{r_1} {g_2}^{r_2} \bmod p$. Then V returns a challenge message α, and P sends $y_1 = r_1 + \alpha x_1 \bmod q$ and $y_2 = r_2 + \alpha x_2 \bmod q$ to V who verifies that ${g_1}^{y_1} {g_2}^{y_2} \stackrel{?}{\equiv} \beta m^{\alpha} \pmod p$.

Here, (r_1, r_2) is chosen randomly, and the Verifier obtains no information regarding (x_1, x_2). Notice that if the choice of (r_1, r_2) is fixed for two different challenges, α and α', then the values of the secret (x_1, x_2) can be found.

Knowing this, we adapt this scheme as follows: we view an electronic coin as a public key m (that is blindly signed by the bank). The value used for (r_1, r_2)

will depend on the node of the binary tree that is being spent. Note that all (r_1, r_2)'s will be randomly correlated, and once decided upon, will be fixed in the bank's signature for a coin, and hence cannot be altered. In this respect, if the same node is spent twice, the same (r_1, r_2) value is used, and all information can be recovered.

3 Construction

In this section, we outline the various protocols in our single-term divisible electronic coin scheme.

When the customer wants to withdraw $\$x$ from the account, an electronic coin of worth $\$x$ is then obtained by executing the withdrawal protocol with the bank.

3.1 The Withdrawal Protocol

The protocols in our scheme are based upon those of Brands' scheme. In particular, our opening protocol has origins in Brands' withdrawal protocol.

Let p, q, g, g_1, g_2 be system parameters published by the bank where the order of g, g_1 and g_2 is q. Let $I = g_1^u \bmod p$ be the identity of a customer U and let u be U's secret key.

The Main Protocol Assume the consumer wishes to withdraw a divisible coin worth $w = 2^l$ dollars from his account at bank B, and assume also that B has a public key $h = g^x \bmod p$ which corresponds to $w = 2^l$ dollars[3], where x is B's secret key.

The withdrawal protocol (main protocol) is as follows:

1. U executes the precomputation stage (described later in Sections 3.1). As a result of this stage, U obtains $T = g_1^{r_{0,1}} g_2^{r_{0,2}} \bmod p$ (note $t_0 = (r_{0,1} || r_{0,2})$). Customer U sends his identity, $I = g_1^u \bmod p$, to bank B.
2. B subtracts $w = 2^l$ dollars from U's account balance. B forms the number $m = Ig_2 \bmod p$. B chooses $w \in_R Z_q$ and sends $z = m^x \bmod p$, $a = g^w \bmod p$, $b = m^w \bmod p$ to U.
3. U generates random numbers $s, t, v \in_R Z_q$, and calculates $m' = m^s \bmod p$ $(= g_1^{us} g_2^s \bmod p)$. U also computes $z' = z^s \bmod p$, $a' = a^t g^v \bmod p$, $b' = b^{st}(m')^v \bmod p$, $c' = \mathcal{H}(m', z', a', b', T)$, and $c = c'/t \bmod q$. U then sends c to B.
4. B responds with $r = xc + w \bmod q$.
5. U then calculates $r' = rt + v \bmod q$ after checking the validity of z, a, b and r by verifying that $m^r \equiv z^c b \bmod p$ and $g^r \equiv h^c a \bmod p$.

[3] Different public key values can be used for different denominations. Note that the actual worth of a coin is nevertheless independent of the coin structure and the protocol definitions; this gives different schemes flexibility in handling the encoding of a coin's worth.

Now, as a result of this protocol, the consumer obtains the following quantities:

$$m',\ T,\ \text{sign}(m', T)$$

where $m' = (g_1^u g_2^s \bmod p)$ and $\text{sign}(m', T) = \{z', a', b', r'\}$. Note that verification of the signature entails checking that:

$$g^{r'} \stackrel{?}{\equiv} h^{c'} a' \pmod{p}, \tag{1}$$

$$m^{r'} \stackrel{?}{\equiv} (z')^{c'} b' \pmod{p}, \tag{2}$$

$$c' \stackrel{?}{=} \mathcal{H}(m', z', a', b', T). \tag{3}$$

Precomputation Stage As mentioned earlier, divisibility is implemented using a binary tree structure. The root of the tree represents the full value of a coin, and all descendants of the root denote various subdenominations of the coin.

We will associate seemingly random values with each node of the tree. In particular, we will first choose random values for the leaves of the tree, and based on these, compute values for all other internal nodes. In this manner, we proceed *up* the tree until we find a value for the root node. And it is this value that is fixed and encoded in the coin. A more detailed description of this process is as follows:

Part 1: Computation of t-values and r-values for Leaf Nodes

Assume the consumer wishes to withdraw a divisible coin worth $w = 2^l$ dollars. The consumer selects a random value e as a secret seed value.

For a node $n_{0j_1j_2\ldots j_v}$, $j_i \in \{0,1\}$, we denote its "t-value" by $t_{0j_1j_2\ldots j_v}$, and when $n_{0j_1j_2\ldots j_l}$ is a leaf of the tree, its t-value is defined as:

$$t_{0j_1j_2\ldots j_l} = \mathcal{H}(e||0j_1j_2\ldots j_l).$$

Suppose f_e is a pseudo-random function (e is an index to a family of pseudo-random functions) [GGM86], then the t-values given by $t_{0j_1j_2\ldots j_l} = f_e(0j_1j_2\ldots j_l)$ are theoretically secure (indistinguishable from random strings).

Now, from a node's t-value, we define its 2 "r-values" as

$$t_{0j_1j_2\ldots j_v} = (r_{0j_1j_2\ldots j_v,1}||r_{0j_1j_2\ldots j_v,2}) \quad \text{where} \quad r_{0j_1j_2\ldots j_v,i} \in \{0,1\}^{|q|}.$$

Note that there are other methods to efficiently calculate leaf node t-values from a seed value e, and that any of these can be used in our scheme if it produces leaf node t-values that are indistinguishable from pseudo-random strings.

Part 2: Computation of t-values and r-values for Internal Nodes

Now begins the process of computing t-values for all internal nodes. The t-value of an internal node $n_{0j_1j_2\ldots j_v}$ is dependent upon those of its children. So, if the t-values of its left and right children are respectively given by:

$$t_{0j_1j_2\ldots j_v0} = (r_{0j_1j_2\ldots j_v0,1}||r_{0j_1j_2\ldots j_v0,2}),$$

$$t_{0j_1j_2\ldots j_v1} = (r_{0j_1j_2\ldots j_v1,1}||r_{0j_1j_2\ldots j_v1,2}),$$

then, its t-value is found by computing:

$$t_{0j_1j_2\cdots j_v} = \mathcal{H}(\mathcal{H}(g_1^{r_{0j_1j_2\cdots j_v0,1}} g_2^{r_{0j_1j_2\cdots j_v0,2}} \bmod p) || \mathcal{H}(g_1^{r_{0j_1j_2\cdots j_v1,1}} g_2^{r_{0j_1j_2\cdots j_v1,2}} \bmod p)).$$

Using this formula, the consumer computes his way up the tree until he obtains t_0, the t-value of the root node. An example of this computation for a small sample tree is given in Subsection 3.4.

3.2 Payment

Payment consists of two stages: coin authentication and denomination revelation. During the coin authentication phase, the merchant verifies that the coin bear the bank's signature. During the second phase, the consumer reveals information about a certain set of nodes in the coin's binary tree representation depending on the denomination being spent. These stages are described in more detail as follows:

Coin Authentication The consumer supplies the merchant with m' ($= g_1^{us} g_2^s \bmod p$), T, $\text{sign}(m', T) = \{z', a', b', r'\}$. The merchant checks that $m' \neq 1 \bmod p$ (i.e. $s \neq 0 \bmod p-1$), and verifies the bank's signature of m' and T by computing c' and seeing that Equations (1), (2) and (3) hold.

Denomination Revelation To spend node $n_{0j_1j_2\cdots j_k}$:

1. The consumer reveals $n_{0j_1j_2\cdots j_k}$'s contribution to the t-value computation of its ancestor nodes:

$$\beta_{0j_1j_2\cdots j_k} = g_1^{r_{0j_1j_2\cdots j_k,1}} g_2^{r_{0j_1j_2\cdots j_k,2}} \bmod p.$$

2. Next, the consumer reveals:

$$\begin{aligned}
&\mathcal{H}(g_1^{r_{0j_1j_2\cdots\overline{j_k},1}} g_2^{r_{0j_1j_2\cdots\overline{j_k},2}} \bmod p),\\
&\mathcal{H}(g_1^{r_{0j_1j_2\cdots\overline{j_{k-1}},1}} g_2^{r_{0j_1j_2\cdots\overline{j_{k-1}},2}} \bmod p),\\
&\vdots\\
&\mathcal{H}(g_1^{r_{0j_1\overline{j_2},1}} g_2^{r_{0j_1\overline{j_2},2}} \bmod p),\\
&\mathcal{H}(g_1^{r_{0\overline{j_1},1}} g_2^{r_{0\overline{j_1},2}} \bmod p).
\end{aligned}$$

   This is all the information that is needed to compute the t-values for all ancestors of node $n_{0j_1j_2\cdots j_k}$.
3. Using the values from Steps 1 and 2, the merchant starts from $n_{0j_1j_2\cdots j_k}$ and proceeds up the tree, computing the t-values for each node from $n_{0j_1j_2\cdots j_{k-1}}$ to the root node inclusive, obtaining the root t-value t_0, from which the value for T and consequently, the signature of m' can be verified.

4. The merchant issues a challenge $\alpha \in Z_p^*$. Here, α is a function of the date, time, merchant identity, and other variants.
5. The consumer supplies the following responses:

$$y_1 = r_{0j_1j_2\ldots j_k,1} + \alpha(us) \bmod q, \tag{4}$$

$$y_2 = r_{0j_1j_2\ldots j_k,2} + \alpha s \bmod q. \tag{5}$$

6. The merchant checks that

$$(g_1)^{y_1}(g_2)^{y_2} \stackrel{?}{\equiv} \beta_{0j_1j_2\ldots j_k}(m')^{\alpha} \pmod p.$$

Once again, note that from this technique, the t-values (and hence r-values) of *all* direct ancestors of $n_{0j_1j_2\ldots j_k}$, including that of the root node, are revealed to the merchant (and bank).

Let the value of coin $(m', T, \text{sign(m', T)})$ be $\$2^l$ and assume customer U wishes to spend $\$x$ of it ($x \leq 2^l$). Let the $(l+1)$-bit binary representation of x be $b_1 b_2 \cdots b_{l+1}$ where $b_i \in \{0, 1\}$, and let $\nu = \#\{b_i \mid b_i = 1, 1 \leq i \leq l+1\}$ be the Hamming weight of x. If $b_i = 1$ (i.e., the i-th most significant bit is 1), a node of the i-th level is spent. Hence, ν represents the number of nodes that will become used as a result of this $\$x$ transaction.

For example, if $x =$ "0100100" then $l = 6, \nu = 2$ and n_{00} (a node of the 2nd level) and n_{01000} (a node of the 5th level) will become used and spent (assuming no nodes on their routes have been previously used). The following information is then given to the merchant:

$$\begin{aligned}
&\beta_{00} = g_1^{r_{00,1}} g_2^{r_{00,2}} \bmod p,\\
&\beta_{01000} = g_1^{r_{01000,1}} g_2^{r_{01000,2}} \bmod p,\\
&\mathcal{H}(g_1^{r_{01001,1}} g_2^{r_{01001,2}} \bmod p),\\
&\mathcal{H}(g_1^{r_{0101,1}} g_2^{r_{0101,2}} \bmod p),\\
&\mathcal{H}(g_1^{r_{011,1}} g_2^{r_{011,2}} \bmod p).
\end{aligned}$$

Therefore, there will be ν β values (from Step 1). Furthermore, if l' is the smallest value such that $b_i = 0$ for all $i > l'$ ($l' \leq l+1$), and if the ν spent nodes are optimally selected, then $(l' - \nu)$ hashed values are revealed in Step 2. In our example with $x =$ "0100100", two β values and three hashed values are revealed since $\nu = 2$ and $l' = -5$.

Thus, the consumer transfers $(\nu(|p|+2|q|)+2(l'-\nu)|q|)$ bits to the merchant. In the preliminary step, the consumer should give $(5|p|+|q|)$ bits to the merchant (for m', T and the signature).

3.3 Deposit

Deposit is as before; a transcript of payment is forwarded to the bank.

3.4 Example

To illustrate our technique, we will consider a coin of worth \$4 and show how to spend \$1 of it.

Before withdrawing the coin, we first select a value for e and compute the t-values for the leaves, finding them to be:

$$t_{000} = \mathcal{H}(e||000),\ t_{001} = \mathcal{H}(e||001),\ t_{010} = \mathcal{H}(e||010),\ t_{011} = \mathcal{H}(e||011).$$

To find the t-values of the other internal nodes, we compute the following:

$$\begin{aligned}
t_{0j_1j_2} &= (r_{0j_1j_2,1}||r_{0j_1j_2,2}), \quad \forall j_1, j_2 \in \{0,1\} \\
t_{01} &= \mathcal{H}(\mathcal{H}(g_1^{r_{010,1}} g_2^{r_{010,2}} \bmod p)||\mathcal{H}(g_1^{r_{011,1}} g_2^{r_{011,2}} \bmod p)), \\
&= (r_{01,1}||r_{01,2}). \\
t_{00} &= \mathcal{H}(\mathcal{H}(g_1^{r_{000,1}} g_2^{r_{000,2}} \bmod p)||\mathcal{H}(g_1^{r_{001,1}} g_2^{r_{001,2}} \bmod p)), \\
&= (r_{00,1}||r_{00,2}). \\
t_0 &= \mathcal{H}(\mathcal{H}(g_1^{r_{00,1}} g_2^{r_{00,2}} \bmod p)||\mathcal{H}(g_1^{r_{01,1}} g_2^{r_{01,2}} \bmod p)).
\end{aligned}$$

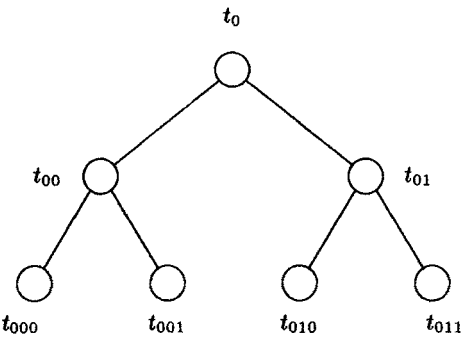

Fig. 1. Finding t-values of Internal Nodes

Figure 1 depicts this part of the computation for a sample tree corresponding to a coin worth \$4.

Having obtained the t-values for the root node n_0, we can find and use T in the computation of c' during the withdrawal protocol.

After obtaining a coin $g_1^{us} g_2^{s} \bmod p$ (unrecognizable by the bank because of blinding factor s), node n_{001} can be spent by revealing $g_1^{r_{001,1}} g_2^{r_{001,2}} \bmod p$.

We also need to reveal $\mathcal{H}(g_1^{r_{000,1}} g_2^{r_{000,2}} \bmod p)$ and $\mathcal{H}(g_1^{r_{01,1}} g_2^{r_{01,2}} \bmod p)$ so that the merchant(bank) can compute t_0 and verify T. Notice that in the process, the bank is able to find the value of t_{00} (and hence $r_{00,1}, r_{00,2}$) as well. Thus, if nodes n_0 or n_{00} are later spent (which would lead to double spending), then the resulting equations can be solved to find the identity of the cheater.

Finally, in response to the merchant's challenge α, we respond with:

$$y_1 = r_{001,1} + \alpha(us) \bmod q,$$
$$y_2 = r_{001,2} + \alpha s \bmod q.$$

Notice that when the same node is spent twice, we have a set of equations for two different challenges (since the r-values, u and s remain the same), and as a result, we can completely solve this system of equations for all the variables.

3.5 Modifications

Here we discuss three possible modifications to our scheme presented above:

1. Our scheme can be easily modified to handle the case when the customer's identity is given by $I = g_1^{u_1} g_2^{u_2} \bmod p$.
2. As in [OO89, OO91], the electronic cash in our proposed scheme can be easily modified into a scheme with an electronic license and electronic coins as follows: An electronic license (m', sign(m')) is issued by the bank to a customer during an "opening protocol". Here, sign(m')= $\{z', a', b', r'\}$, and $c' = \mathcal{H}(m', z', a', b')$. A coin worth \$$2^l$, ($T$, sign($T, m'$)), is withdrawn to a user through a "withdrawal protocol". Here, sign(T, m') is a blind signature of (T, m') by bank B. (Any blind signature scheme can be used here, such as the RSA blind signature [Cha85] or the Schnorr blind signature (Appendix B of [Oka92])).
3. Our proposed scheme can be modified to exhibit *transferability* by using approaches equivalent to those described in [OO89, OO91].

4 Evaluation

4.1 Security

In this subsection, we informally describe the security of our scheme. In our final paper, we will treat the security more formally.

We must ensure that incorrect spending of a coin results in the discovery of a consumer's identity, while legitimate transactions preserve his privacy. What constitutes incorrect spending? Violation of the aforementioned invariant (Section2.2) – namely, when any two nodes along the same path from a root to a leaf are spent.

Let n_a and n_b represent two used nodes that are situated on the same path and thus violate the above invariant. Assume without loss of generality that n_a

is the node farthest from the root. From the information revealed in Eqs. (4), (5) when n_a was spent (during Denomination Revelation), we can immediately find the t-value and r-values for n_b, and we can solve for s, u and hence the double spender's identity using the equations released for n_b.

When n_a and n_b are the same node, we have two sets of equations differing only in the two challenges, and we have seen that as part of the Schnorr scheme, s and u can likewise be recovered.

This strategy allows us to successfully identify the perpertrator in cases of double spending; we must also insure that the privacy of honest consumers are also protected.

Two totally unrelated nodes n_a and n_b do not lie on the same route (i.e. not the same node and not direct descendents of each other). The r-values of either of them cannot be recovered from their β values, which are of the form $g_1^{r_{n,1}} g_2^{r_{n,2}} \bmod p$, since that would entail solving for the representation problem, nor can either be computed from the opened t-values of other ancestor nodes since $\mathcal{H}$ is one-way. No useful equation relating the r's of node n_a to those of n_b is available for easy manipulation, so we essentially end up with 4 unknowns and 6 equations. Notice that for each additional unrelated node that is spent, 2 more equations are obtained, but 2 more unknowns are introduced, so as long as these nodes are unrelated, additional Denomination Revelation equations are of no use since the number of unknowns continue to outnumber the number equations in the resulting system.

4.2 Efficiency

For this section, let us assume that $|p| = 512$ bits, $|q| = 140$ bits, and \$1000 is an amount that is sufficient to meet a person's everyday needs, and moreover, is a quantity large enough to last him several days before needing to make another withdrawal. Assume a consumer is allowed to spend any amount from \$1000 to \$1 so that we only need to consider a tree of 11 levels.

We will see that the average amount of communication and the required memory size for one payment is approximately 0.91 Kbytes. Here, a single payment consumes an average of 5.5 nodes; hence $\nu = 5.5$ and $l' = 10.5$. (see the remark of Subsection 3.2) We can assume a slight modification to improve the efficiency as follows: Assume there are two hash functions $\mathcal{H}_1$, and $\mathcal{H}_2$ such that

$$t_{0j_1j_2\cdots j_v} = \mathcal{H}_1(\mathcal{H}_2(g_1^{r_{0j_1j_2\cdots j_v0,1}} g_2^{r_{0j_1j_2\cdots j_v0,2}} \bmod p)||\mathcal{H}_2(g_1^{r_{0j_1j_2\cdots j_v1,1}} g_2^{r_{0j_1j_2\cdots j_v1,2}} \bmod p)).$$

Here, $\mathcal{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^{2|q|}$, and $\mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{|q|}$. Therefore, the amount of data transferred to the merchant for one payment is $(5.5 \times (64 + 2 \times 17.5) + (10.5 - 5.5) \times 17.5) + 64 \times 4 + 17.5 = .9055$ Kbytes on average.

If little storage is available then spending a node necessitates the recalculation of its t-value, perhaps even starting from scratch with e. On the other hand, memorization of all t-values results in less computation (as relevant values are simply looked up and retrieved), but demands more in terms of storage

requirements. To make our scheme practical, we seek a balance between these computational and storage complexities.

We suggest the following: store the t-values for all nodes in the upper 7 levels of the tree, and compute any t-values of the lower 4 levels on a need-by-need basis. This requires about $2(2^7-1)|q|$ bits of storage for the upper 7 levels. This quantity is less than 4.5Kbytes.

Let us assume that the cost of $\mathcal{H}$ is negligible compared to that of modular exponentiation. Then, at most 4 nodes of the lower 4 levels are consumed during a given payment transaction. Therefore, (2^4-1) modular exponentiations need to be performed. If 1 modular exponentiation takes 50 ms to complete (assuming the usage of a 10 Kbps mod-exp chip or machine), then the payment protocol takes less than 1 second.

5 General Construction Based on Disposable Authentication

While the construction described in the first part of this paper is based on discrete logarithms, this section introduces a general construction method based on disposable authentication, and shows that a divisible coin system using this general framework can be constructed from Ferguson's scheme [Fer93].

5.1 General Construction

In this subsection, we show that any basic coin system based on a class of disposable authentication can be extended to become a divisible one. Let f, g, D, F, F' and G represent polynomial-time computable functions (in particular, functions f and g are one-way), and let H be a polynomial-time one-way hash function.

For simplicity, we assume that the underlying disposable authentication protocol is a three move protocol (P,V) in which P and V represent a prover and a verifier respectively. If m is some input common to both P and V, then (P,V) is P's proof to V that P knows some x such that $m = f(x)$.

Basically, P generates random string r, and sends $X = F(x, r)$. Then, V returns a random challenge E to P, and P sends $Y = D(x, r, E)$ to V, who checks the consistency of this conversation by computing $G(m, X, E, Y)$. For our extension to divisible coins later, we assume that $X = F(x, r)$ can be efficiently calculated from $m = f(x)$ and r (i.e., there exists an efficiently computable function F' such that $X = F(x, r) = F'(m, r)$).

Here, we construct a basic coin system based on the disposable authentication paradigm, using this three move protocol during the payment phase.

- (Withdrawal) A customer U selects u, sends $I = g(u)$ to B, and obtains a "restricted" blind signature for $m = f(x)$ from B, where x is a function of I. Here, "restricted" means that m is related to I such that if B knows x, B can calculate u, although B can get no linckage between m and I without x. Then, U generates a random string r and calculates X. This and m are

then sent to B and B releases a blind signature on $H(m, X)$ to U. A coin consists of m, X and B's signatures of m and $H(m, X)$.
Here, B knows that I is the identity of U, but because B issues a blind signature, B is not aware of the relationship between m and U and hence, will not recognize m as belonging to U later on. However, if U double spends a coin, then u can be found, and so u serves as a witness that U spent a coin twice.

- (Payment) Consumer U and merchant M enact the aforementioned basic three move protocol (U,M) when U wishes to spend coin X. Here, U sends m and X, along with B's signatures of these quantities to M, M checks their validity and challenges U with a quantity E, U replies with Y. If U uses the same coin (X) twice, B can then find x (using an approach identical to that of the knowledge extractor from the soundness property for interactive proofs of knowledge[OO89].) From x, B can proceed to find u and $I = g(u)$.

Now, we explain how to extend this coin system to handle divisible coins by applying the ideas from Sections 2.2 and 3.1. Here, Part 1 of the precomputation stage is exactly the same as that of our proposed scheme, but Part 2 is slightly different in that the r-values are implicitly included in the t-values; i.e., r-values are not used explicitly. So Part 2 becomes:

$$t_{0j_1j_2\ldots j_l} = \mathcal{H}(\mathcal{H}(F'(m, t_{0j_1j_2\ldots j_l0}))||\mathcal{H}(F'(m, t_{0j_1j_2\ldots j_l1}))),$$

$$\ldots\ldots$$

$$t_{00} = \mathcal{H}(\mathcal{H}(F'(m, t_{000}))||\mathcal{H}(F'(m, t_{001}))),$$

$$t_0 = \mathcal{H}(\mathcal{H}(F'(m, t_{00}))||\mathcal{H}(F'(m, t_{01}))),$$

$$T = F'(m, t_0).$$

So during withdrawal, $X = F(x, r)$ is of the form $F(x, t_{0j_1j_2\ldots j_l}) = F'(m, t_{0j_1j_2\ldots j_l})$ (i.e., $r = t_{0j_1j_2\ldots j_l}$). And when node $n_{0j_1j_2\ldots j_l}$ is spent, the relevant t-values and F' values are revealed so that t_0 can be found and the value of T can be verified.

5.2 Divisible Coins Based on Ferguson's Scheme

Using the same notation as [Fer93], we modify Ferguson's scheme to realize a divisible coin system. Customer Alice gets two RSA signatures from the bank: $(C^U A)^{1/v}$ and $(CB)^{1/v}$. When Alice wants to withdraw a coin from the bank, she selects $k \in_R Z_v$ and $R \in_R Z_n$, and sends $X = H(\beta, A, B, C)R^v$ to the bank, where H is a polynomial-time one-way hash function and $\beta = B^k$, and the bank returns $X^{1/v}$ to Alice. From this, Alice obtains $H(\beta, A, B, C)^{1/v}$. When Alice wants to pay at a shop, she reveals β along with $H(\beta, A, B, C)^{1/v}$, a, b, c. The shop calculates and verifies $A = f_a(a), B = f_b(b), C = f_c(c)$, checks the validity of $H(\beta, A, B, C)^{1/v}$, and replies with a challenge x. Finally Alice sends $r = k + Ux \bmod v$ and the signature $(C^r A^x \beta)^{1/v}$ which she can easily compute from the two signatures she got from the bank.

The original Ferguson scheme is basically a two move (challenge and response) protocol. Roughly speaking, this corresponds to a modified three move version of Ferguson's scheme, in which the disposable authentication technique is employed.

We can easily apply our technique to realize divisibility for this modified version of Ferguson's scheme in such a way so that: Part 1 of the precomputation stage is exactly the same as before. In Part 2, the t-values are equivalent to r-values (i.e., no distinction is needed between t-values and r-values), and

$$t_{0j_1j_2\ldots j_l} = \mathcal{H}(\mathcal{H}(B^{t_{0j_1j_2\cdots j_l0}})||\mathcal{H}(B^{t_{0j_1j_2\cdots j_l1}})).$$

When node $n_{0j_1j_2\ldots j_l}$ is spent, $\beta_{0j_1j_2\ldots j_l} = B^{t_{0j_1j_2\ldots j_l}}$ (i.e., $k = t_{0j_1j_2\ldots j_l}$). The linkage between an electronic coin and $T = B^{t_0}$ is done in a manner similar to that Section 3.1.

6 Conclusions

We have presented here, the first single-term divisible coin system, that is more efficient than that of [OO91]. In addition, we show that a specific type of "disposable authenticated" coin system can be extended to handle divisible coins using our techniques – more specifically, such coin systems are characterized by the presence of a randomized term (e.g., $F(x, r)$ in subsection 5.1) that can be expressed as a function (e.g., F') of a public value (e.g., m) and a random variable (e.g. r). Finally, we use our techniques to transform Ferguson's electronic coin scheme into a version that handles divisible coins.

Acknowledgments

The authors wish to thank anonymous referees for their helpful suggestions, and are grateful to Stefan Brands for his kind and valuable comments. We also thank Kazuo Ohta for valuable discussions with him in the earlier stages of our work.

References

[Bra93] Brands, S., "Untraceable Off-line Cash in Wallet with Observers", Proceedings of the Crypto 93, pp.302–318 (1994).

[Cha85] Chaum, D., "Security without Identification: Transaction Systems to Make Big Brother Obsolete," Comm. of the ACM, 28, 10, pp.1030–1044 (1985).

[CFN88] Chaum, D., Fiat, A., and Naor, M., "Untraceable Electronic Cash," Proceedings of the Crypto 88, pp.319-327 (1990).

[DP92] De Santis, A. and Persiano, G., "Communication Efficient Zero-Knowledge Proofs of Knowledge (with Applications to Electronic Cash)" Proceedings of STACS 92, pp. 449-460 (1992).

[Fer93] Ferguson, N., "Single Term Off-line Coins", Proceedings of the Eurocrypt 93, pp.318–328 (1994).

[FY93] Franklin, M. and Yung, M., "Secure and Efficient Off-Line Digital Money", Proceedings of ICALP 93, pp. 449-460 (1993).

[GGM86] Goldreich, O., Goldwasser, S., and Micali, S., "How to Construct Random Functions," Journal of ACM, Vol.33, No.4 (1986).

[Oka92] Okamoto, T., "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes", Proceedings of the Crypto 92, pp. 31-53 (1993).

[OO89] Okamoto, T., and Ohta, K., "Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash", Proceedings of the Crypto 89, pp. 481-496 (1990).

[OO91] Okamoto, T., and Ohta, K., "Universal Electronic Cash", Proceedings of the Crypto 91, pp. 324-337 (1992).

[Pai92] J.C. Pailles, "New Protocols for Electronic Money", Proceedings of the Auscrypt 92, pp. 263–274 (1993).

[Sch91] Schnorr, C.P., "Efficient Signature Generation by Smart Cards", Journal of Cryptology, Vol. 4, No. 3, pp.161-174 (1991).