In CARDIS 94: Proceedings of the First IFIP Smart Card Research and Advanced Application Conference, October 1994, pp. 89-100

A Fast Off-line Electronic Currency Protocol

Lei Tang* Doug Tygar†

Abstract

We present a new off-line protocol for electronic currency. The protocol uses the *RSA* public key system (we have an alternative version using the Rabin signature scheme.) The protocol is much faster than other electronic currency protocols but does not sacrifice security. The efficiency is achieved by using a new signature technique called the *restrictive randomized blind signature* (an extension to the *randomized blind signature* technique invented by Chaum) and by introducing a *partial untraceability* concept. Depending on whether the RSA or Rabin signature method is used, a customer need only perform at most four exponentiations or no exponentiations; the customer must also perform a small number of multiplications and compute one one-way function per transaction. This protocol appears to be very practical for a variety of applications, including Internet billing servers, smart card point of sale applications, electronic commerce, and other applications.

1 Introduction

Electronic currency has obvious and important applications in many proposed electronic services, such as electronic commerce. Electronic currency can be combined with techniques such as secure operating systems on tamper-proof hardware [18], cryptographic technology (blind signatures [3], zero-knowledge proofs [9] and secret sharing [16]), and contemporary systems software techniques (transactions [10] and fault tolerance methods [12]), to build secure and reliable commercial applications.

This paper describes the design of an electronic currency protocol that is much more efficient than previous protocols described in the literature. This protocol has a novel *partial untraceability* property. We assume the existence of a bank that issues electronic tokens, merchants which sell goods or services, and customer which purchase those good or services. Neither a bank nor a merchant can individually trace the flow of currency, and the bank is not aware of the identity of a merchant's customers. However, if it becomes necessary to audit the account (perhaps because abuse is suspected) then it is possible to trace a particular flow of specific electronic currency token by combining information from both the bank and the merchant. This trace information is limited in scope; if an electronic currency token is

*Supported by William Larier Mellon Fellowship granted by Graduate School of Industrial Administration of Carnegie Mellon Univ. Email: ltang@cs.cmu.edu

†School of Computer Science, Carnegie Mellon Univ. Email: tygar@cs.cmu.edu

traced, the process does not divulge information about transactions with other electronic currency tokens.

Ohta and Okamoto [13] give a list of desired properties for electronic currency systems:

1. Independence of physical requirements.
2. Unforgeable and uncopyable currency.
3. Untraceable purchases.
4. Off-line payments between the merchant and the customer.
5. Currency transferable from one customer to another.
6. Currency sub-dividable by a customer.

Uncopyability, unforgeability, and untraceability properties make the design of an electronic currency system challenging. Paper currency has physical properties that add security (it is difficult to copy or counterfeit dollar bills.) Electronic currency is only represented as a sequences of bytes. It is harder to prevent copying, counterfeiting, and tracking the flow of electronic currency.

Chaum introduced the *blind signature technique* [3] to solve the problem of untraceability. The basic idea of Chaum's blind signature is as follows: Let (e, n) and (d, n) be the *RSA* public key/secret key pair for a bank. When a customer wants an electronic token worth one unit, he chooses random numbers x and r and sends $C = x \cdot r^e \bmod n$ to the bank. After receiving C, the bank sends $D = C^d \bmod n$ to the customer. The customer computes the value of $x^d \bmod n$ from $D = x^d \cdot r \bmod n$ since the customer knows the value of r. When some merchant presents $(x, x^d \bmod n)$ to the bank, the bank can not determine which customer spent this token. Here the untraceability is achieved through the blind factor r generated randomly by the customer. We use the terminology *blind manipulation* to describe the technique of hiding values by multiplying them by random factors.

While blind signature techniques address very important privacy problems, and protect individuals from being traced by "big brother", it also introduces many problems in the case of abuse. For example, a perfect crime based on the blind signature is described in [17]. It is more startling to realize that if the secret key (d, n) of the bank is revealed (by a failure in system security or human security) then it would be impossible to trace counterfeiters. Such an attack could cripple the electronic currency system.

Chaum's original system did not protect against a customer making multiple copies of an electronic token and using it with multiple merchants. It is possible to solve this problem by requiring that the bank be on-line and participate in every transaction, but this raises difficult scaling and reliability issues. Using the cut-and-choose technique first introduced by Rabin [14], Chaum, Fiat, and Naor improved Chaum's original protocol by using an interactive protocol to determine if an electronic token was reused [4]. In the Chaum-Fiat-Naor protocol, many special values are computed by a customer and sent to the bank. The bank issues a cryptographic challenge, and the customer replies after performing additional computation. Unfortunately, the Chaum-Fiat-Naor protocol requires considerable computation and communication.

Ferguson [7] introduced a much faster protocol that achieved the same effect as the Chaum-Fiat-Naor protocol. The basic idea of Ferguson's protocol is that a customer receives two *RSA* signatures on two numbers generated by a bank, and then generates the electronic currency from these two signed values. Ferguson uses a new technique called *randomized blind signatures*, which he attributes to Chaum. He specifies that the randomized blind signature should have the following properties:

- The customer receives an *RSA* signature on a number of a special form, which he can not create by himself.
- The bank is sure that the number it signs was randomly chosen over a particular set.
- The bank receives no information regarding which signature the customer receives.

In this paper we simplify and extend Ferguson's approach by creating a new protocol; we use values of the form $af(a^e)$, where $a = a_1\sqrt[e]{a_2}$ with a_1 being generated randomly by the customer and a_2 being generated by the bank. (All the roots are computed over finite multiplicative groups.) If $e = 2$ we use the Rabin signature scheme; if $e > 2$ is an odd integer we use the *RSA* method.

We also introduce an new randomized blind signature scheme called *restrictive randomized blind signatures* to restrict a customer's use of blind manipulations. Restrictive randomized blind signatures restrict the user to encode his identity in a single signature. This identity is blind to a single agent in a system, but can be determined by cooperation of both a merchant and a bank. Our approach requires significantly less computation than Ferguson's. Our restrictive randomized blind signature technique also stops a broad class of chosen message (chosen plaintext) attacks by the customer. The customer receives a signature on a number of a special form, which he cannot create by himself.

Ferguson's protocol requires sixteen exponentiations by the customer to generate and check the numbers signed by the bank, three exponentiations by the bank to generate the signatures, a combined total of four one-way function computations, and a combined total of twenty-eight additional multiplications. We have two versions of our protocol; one uses the *RSA* signature method and one uses the Rabin signature method. In both versions, the bank performs three exponentiations before the withdrawal protocol. If we use *RSA* version of our protocol, we require one exponentiation by the bank to generate the signature, three exponentiations by the customer, one one-way function computation by the customer, and a combined total of seven additional multiplications. If we use the Rabin version of our protocol, we require no exponentiations by the customer, one one-way function computation, and a combined total of ten multiplications. (However, in the Rabin version of our protocol, we require the bank to compute square roots in modulo pq where p and q are large primes, and this puts a computational strain on the bank [1, 2].)

Pseudo-random number generation is also an expensive operation. Ferguson's protocol requires the customer to generate ten random numbers and the bank to generate four random numbers. In our protocol, the customer must generate three random numbers and the bank need not generate any random numbers.

Under normal circumstances, when a merchant cashes in an electronic token, the identity of a user who previously owned the token is completely hidden. However, if some user attempts to use a token twice, the bank can detect this. Under these circumstances, the bank asks a merchant for additional information, which allows it to detect the identity of the user. This protection of the user's identity is based on a version of Shamir's secret sharing protocol [16]. Here we keep customers' identities secret unless both a merchant and bank cooperate in tracing.

Section 2 describes some mathematical preliminaries. Section 3 describes the protocol. Using some common intractability assumptions, we demonstrate the security of our protocol against a class of chosen message attacks. Section 4 contains our conclusions.

2 Mathematical basis

First we define some notation.

$A||B$: Concatenation of two binary strings A and B.

Z_n^* : $\{a : a \in Z, 1 \leq a \leq n, \gcd(a,n) = 1\}$.

$x^{1/e} \bmod n$: The e^{th} *RSA* root[1] of x mod n, i.e. the unique solution $s \in Z_n^*$ to $s^e = x \bmod n$ for $x \in Z_n^*$ and $e \in Z$ with $\gcd(e, \phi(n)) = 1$. Here ϕ is the Euler totient function. The e^{th} *RSA* root of x is also denoted as $\sqrt[e]{x}$.

$\oplus$: The XOR operation.

$a \in_R S$: An element a randomly selected from the set S under a uniform distribution.

$a \in_A S$: An element a selected from the set S arbitrarily by the selector.

generator : A generator g of a Z_q^* is an element of order $\phi(q)$.

We assume that the factorization problem is hard (that is not solvable in polynomial time.) Let m be an element in Z_n^*, where n is a product of two odd primes p and q. Rabin has proved in [15] that finding a square root of m in Z_n^* is equivalent to the factorization of n, where m is a quadratic residue in Z_n^*. When $e > 2$ is an odd number, finding the e^{th} (*RSA*) root of m in Z_n^* is at most as hard as the factorization of n. In this paper, we further assume that it is hard to compute the value of $\sqrt[e]{m}$ without knowing the factorization of n. We call this the *RSA assumption*. We call the pair $(m, \sqrt[e]{m})$ in Z_n^* the *RSA signature* on m, when $e > 2$ is an odd number, and the *Rabin signature*, when $e = 2$ and m is a square (quadratic residue) in Z_n^*.

We also assume the intractability of the *Discrete Logarithm Problem*: Let w be an odd prime, g be a generator of Z_w^*, and a be an element in Z_w^* satisfying $g^x = a \bmod w$. No probabilistic polynomial time algorithm can, on input p, q and a, reliably output a solution x.

[1]In the sequel, all operations related to the *RSA* signature procedure are to be taken modulo n.

3 Protocol

Using the above intractability assumptions, we first present our protocol. We introduce extended randomized blind signatures that protect the identities of customers unless multiple agents cooperate. The signatures multiply the identity of a customer by a random factor r, so no bank can directly trace the identity of a customer. However, the signatures also force the customer to encode his identity inside the token signed by the bank in such a way that if the customer reuses his token, the bank can detect this and begin a set of operations to trace the errant customer's identity. If it is necessary to trace a transaction, extended randomized blind signatures contain enough information to identify a customer. We then discuss the security of our system against a wide class of chosen message attacks.

3.1 Our Protocol

The bank chooses and makes public its *RSA* modulus n and public encryption key e. It also publish a one-way function f from Z_n^* to Z_n^*. f must also have the property $f(ab) \neq f(a)f(b)$ for any a, $b \in Z_n^*$. f is used to restrict customers; our protocol is constructed so that even if the bank knows a trapdoor in f that allows it to easily compute f^{-1}, the bank can not gain any advantage. Goldreich, Goldwasser and Micali describe a particular family of one-way functions in [8] which would satisfy our conditions. We believe that in practice one can also use simple and fast functions such as MD5 and preserve the security of our protocol. Let w be a prime larger than n, and let w' be a prime larger than $w||w$. The bank selects and publishes a pseudo-random one-way function h which maps $Z_{w'}^*$ to Z_n^*.

Before the customer (called C) and the bank (called B) begin the transaction, C must authenticate itself to B. This can be done by any standard authentication protocol, such as the zero-knowledge protocol designed by Feige, Fiat and Shamir [6].

B generates a u and v having the property that the knowledge of both u and v reveals C's identity. We generate u and v as follows: Assume that the identity of the customer C is denoted by a binary string s, for example, C's social security number. The bank generates a pseudo-random serial number s' for the token which can not be predicted by C. Then u, v can be defined as $u = s \oplus s'$ and $v = s'$. (It is important that w be larger than any possible value of u.) C's identity can be uniquely determined as $s = u \oplus v$ if both u and v are known. The bank also computes $a_2 = h(g^u \bmod w \,||\, g^v \bmod w)$. where g is a generator of Z_w^*. Since only the bank knows the factorization of n, only the bank can compute $\sqrt[e]{a_2}$ easily. The withdrawal protocol is shown in Figure 1.

The Withdrawal Protocol

1. C generates $r \in_R Z_n^*$, $x \in_R Z_n^*$ and $a_1 \in_R Z_n^*$. C sends B the number $r^e a_1 x \bmod n$.
2. B sends a_2 to C.
3. C computes $b = f(a_1^e a_2)x^{-1} \bmod n$ and sends it to C.
4. B computes $c = r^e a_1 x \cdot f(a_1^e a_2)x^{-1} \cdot \sqrt[e]{a_2} = r^e a_1 \sqrt[e]{a_2} f(a_1^e a_2) \bmod n$. B computes $\sqrt[e]{c} \bmod n$ and sends $\sqrt[e]{c}$, $\sqrt[e]{a_2}$, u, v to C.

$$\begin{tikzcd}
\textbf{Customer} & & \textbf{Bank} \\
r \in_R Z_n^* & & a_2 = h(g^u \bmod w \,||\, g^v \bmod w) \\
a_1 \in_R Z_n^* & & \\
x \in_R Z_n^* \arrow[rr, "r^e a_1 x"] & & \\
 & & a_2 \\
f(a_1^e a_2)x^{-1} & & \arrow[ll, "a_2"'] \\
{} \arrow[rr, "f(a_1^e a_2)x^{-1}"] & & {} \\
 & & c = r^e a_1 x \cdot f(a_1^e a_2)x^{-1} \cdot \sqrt[e]{a_2} \bmod n \\
a = a_1\sqrt[e]{a_2} & & \arrow[ll, "{\sqrt[e]{c},\ \sqrt[e]{a_2}}"', "{(u,v)}"] \\
af(a^e)\overset{?}{=}(\sqrt[e]{c}/r)^e & & \\
(\sqrt[e]{a_2})^e\overset{?}{=}a_2 & &
\end{tikzcd}$$

Figure 1: The Withdrawal Protocol

5. Let $a = a_1\sqrt[e]{a_2}$, C checks if $(\sqrt[e]{c}/r)^e = af(a^e)$, $(\sqrt[e]{a_2})^e = a_2$, and if (u,v) is in the proper form (namely $(u,\ v)$ contains his identity and must be unique). If all conditions above are satisfied, then C accepts the token $(a,\ \sqrt[e]{af(a^e)},\ a_2,\ \sqrt[e]{a_2},\ u,\ v)$.

Remarks

1. (u,v) should be different each time C receives an electronic token from B to prevent B from trying to cheat C by claiming that C reused the token. So C must check the uniqueness of the value $(u,\ v)$ in step 5.

2. If the computational power of C is limited, it is not necessary for C to check if $(\sqrt[e]{c}/r)^e = af(a^e)$, $(\sqrt[e]{a_2})^e = a_2$ and $a_2 = h(g^u \bmod w \,||\, g^v \bmod w) \bmod n$ hold. If the value of $\sqrt[e]{c}$ is not consistent with that of $r \cdot af(a^e)$, or $(\sqrt[e]{a_2})^e \neq a_2$, or the value of $(u,\ v)$ sent to C is not consistent with the value of $(u,\ v)$ used to generate a_2, the discrepancy will be detected in step 2 of the payment protocol described below. A bank would quickly ruin its credibility if it sent inconsistent signatures and numbers to a customer in step 5.

3. We multiply $r^e a_1 x \cdot f(a_1^e a_2)x^{-1}$ by $\sqrt[e]{a_2}$ in the final step. This prevents a broad class of chosen message attacks. If we did not perform this multiplication, then C could factor B's value of n when $e = 2$ using the following technique: C randomly generates y and sends y^2 to B. If B sends $\sqrt{a_2 y^2} \bmod n$ to C, with probability 1/2, C can determine $\sqrt{a_2}y'$ such that $\gcd(y - y', n) = p$ or q since C knows $\sqrt{a_2}$ at the last step.

4. B sends a_2 to C before B sends the value of $\sqrt[e]{a_2}$ to make sure that C encodes a_2 in the signature since C's identity is encoded in a_2. It is important for B to first send a_2 and

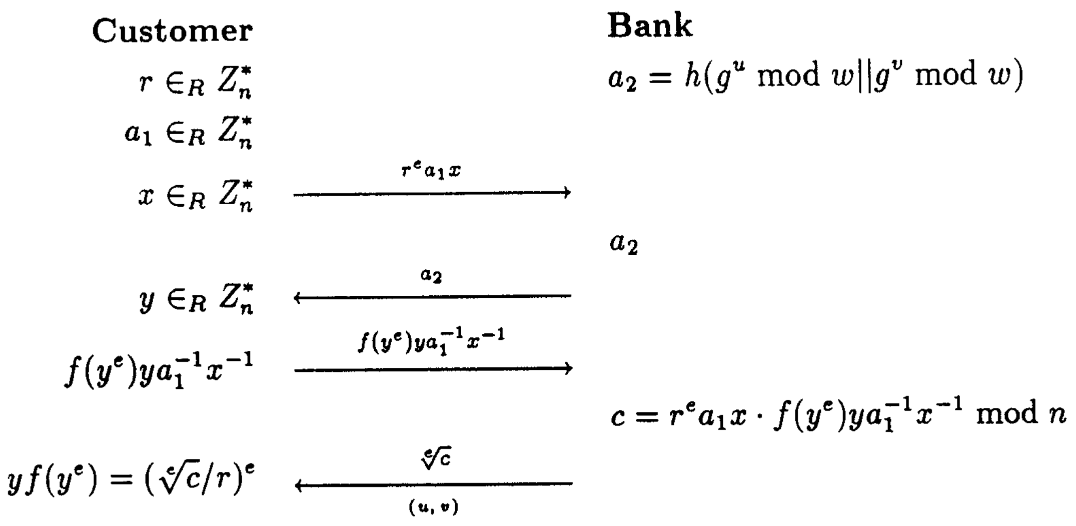
Customer
Bank
$r \in_R Z_n^*$
$a_1 \in_R Z_n^*$
$x \in_R Z_n^*$
$a_2 = h(g^u \bmod w || g^v \bmod w)$
$r^e a_1 x$
a_2
a_2
$y \in_R Z_n^*$
$f(y^e) y a_1^{-1} x^{-1}$
$f(y^e) y a_1^{-1} x^{-1}$
$c = r^e a_1 x \cdot f(y^e) y a_1^{-1} x^{-1} \bmod n$
$y f(y^e) = (\sqrt[e]{c}/r)^e$
$\sqrt[e]{c}$
(u, v)

Figure 2: How a customer may attempt to cheat the bank

then include $\sqrt[e]{a_2}$ in a later message's signature to C. If B sent a_2 to C in step 2, but did not include $\sqrt[e]{a_2}$ in the signature in step 4, then C could cheat using the method described in Figure 2. In that case, C's identity would not be encoded in the tokens he received from the bank. The token $(y, \sqrt[e]{yf(y^e)})$ would be completely untraceable after the bank issued it. Even if B could detect the reuse of this token, he would not be able to determine which customer reused this token. Note that y may be of the form z^e here. Similarly B sent C the value of $\sqrt[e]{a_2}$ before sending a_2, C could also cheat by using a similar method.

5. If the *RSA* signature scheme is used in step 4 (e is an odd prime), then $\sqrt[e]{x}$ is a simple exponentiation. If the Rabin signature is used ($e = 2$), and both $\gcd(p, c) = 1$ and $\gcd(q, c) = 1$ hold, then the square roots can be calculated as follows: first we compute $\sqrt{c} \bmod p$ and $\sqrt{c} \bmod q$ respectively using the algorithms described in [1] or [2], then we compute $\sqrt{c} \bmod n$ by the Chinese Remainder Theorem. Note that computation of square roots in a finite group is a moderately expensive operation.

6. We use three random values, a_1, r, x, to "blind" various portions of the electronic token granting process to the bank. The values a_1 is to disguise a_2 so that the bank can not determine the value of a_2 (and hence discover the identity of the customer) when the merchant presents the pair $(a, \sqrt[e]{af(a^e)})$ in the deposit protocol described below. As in the case of Chaum's original protocol [3], r disguises the particular token sent to a customer so that a bank can not trace the flow of the token through the electronic economy. Finally, x protects the value of $a = a_1 \sqrt[e]{a_2}$; because we assume that the bank has sufficient computational power to invert f, we need to use x to make sure that the value of a always remains secret.

Our restrictive randomized blind signature scheme satisfies the Ferguson's three properties listed on page 3. We can further show that our system is safe against a wide variety of chosen message attacks from the customer. (See Lemma 5 below.)

The Payment Protocol

1. C sends $e_1 = (a, \sqrt[e]{c}/r)$, $\sqrt[e]{a_2}$, $e_2 = (g^u \bmod w, g^v \bmod w)$ and (t, x) to the merchant. Here $x = id_v \,||\, time$; id_v is the identity of the merchant; $time$ is the time at which the transaction begins, $t = ux + v \bmod (w-1)$; and $x << w$.

2. The merchant checks if
$$g^t = (g^u \bmod w)^x (g^v \bmod w) \bmod w$$
and
$$(\sqrt[e]{c}/r)^e = af(a^e) \bmod n$$
and
$$h(g^u \bmod w \,||\, g^v \bmod w) = (\sqrt[e]{a_2})^e \bmod n$$
If so, the merchant accepts the token.

The Deposit Protocol

1. The merchant sends $(a, \sqrt[e]{af(a^e)})$ and (t, x) which he received from C to the bank. (Recall that we set $a = a_1 \sqrt[e]{a_2}$.)

2. The bank checks that $(\sqrt[e]{af(a^e)})^e = af(a^e)$ and uses t to detect reuse of tokens. (See remark 3 below).

Remarks

1. The merchant should not send $(g^u \bmod w, g^v \bmod w)$ to the bank, otherwise, the identity of the customer would be revealed since a_2 can be easily computed from $(g^u \bmod w, g^v \bmod w)$.

2. Since the merchant cannot determine (u, v) from $(g^u \bmod w, g^v \bmod w)$ by the intractability of the discrete logarithm problem, the identity of the customer is unknown to the merchant, unless the customer reveals it by other methods. On the other hand, the correctness of g^u and g^v sent to the merchant is checked in the final phase of step 2 of the payment protocol.

3. The bank should record all triples (a, t, x) sent to it by the merchants. If a merchant sends the bank $(a, \sqrt[e]{af(a^e)})$ where a is the same as the first element of some recorded triple (a', t', x'), the bank will ask the merchant to show it the value of $(g^u \bmod w, g^v \bmod w)$ to prevent the merchant from framing the customer by reusing his tokens.

   (This raises an important side issue. A dishonest bank might ask a merchant to show the value of $(g^u \bmod w, g^v \bmod w)$ under the pretense of tracing the identities of some customers. To prevent such a dishonest bank from cheating, we suggest that all spent token triples (a, t, x) be time-stamped, and then notarized in the sense of Haber and Stornetta [11]. Whenever a bank asks a merchant to show it the value of ($g^u \bmod w$, $g^v \bmod w$), it should present the merchant with the time stamp of the corresponding token (a, t, x). It is easy for a merchant to check if the bank has legitimately detected fraud or not.)

4. As was the case with previous electronic currency protocols, our protocol does not attempt to address the concern that a bank might attempt to cheat a customer by claiming that the customer has reused tokens. If this is a concern, then we can prevent this by having the customer sign all messages between him and the bank; if a dishonest bank attempts to cheat the customer, it will be unable to present the signed messages associated with the withdrawal.

3.2 Security Analysis

We do not know how to present a full formal proof of the security of any of the existing electronic currency protocols. However, we can formally prove the security of our protocols against a wide class of chosen message attacks. In this class of chosen message attacks, we assume that an adversary is restricted to basic arithmetic operations. Although this class of chosen message attack methods is very strong, we do not know whether an adversary armed with more complex functions might be able to break our system or not. Therefore, when we claim that our protocols are secure, we only mean that they are secure against all the attacks discussed below. (In practice, we believe that our use of a one-way function to precludes all chosen plaintext attacks known to us. However, we can not yet formally prove this.)

Lemma 1 *If C follows the protocol and accepts a token, then the token must have originated with B.*

Proof. By our intractability assumptions, only B can calculate $\sqrt[e]{af(a^e)} \bmod n$ and $\sqrt[e]{a_2} \bmod n$ since only B knows how to factor n. □

Lemma 2 *C cannot generate a pair of numbers with the form* $(a, \sqrt[e]{af(a^e)} \bmod n)$ *by itself.*

Proof. This is equivalent to solving $x^e = yf(y^e) \bmod n$. But this equation can not be solved for y because f is a one-way function. On the other hand, it can not be solved for x by the RSA assumption.

Lemma 3 *Given some set S of RSA signatures, any new RSA signatures that can be computed from S are multiplicative combinations of elements of S.*

Proof. Evertse and Van Heyst have proved that this is true in [5].

Lemma 4 *C cannot construct a new token by multiplying old tokens given to him by B in the previous transactions.*

Proof. This follows immediately from property that $f(ab) \neq f(a)f(b)$ and Lemma 3. □

Lemma 5 *If C deviates from the withdrawal protocol by sending values in step 1 or step 3 to B which are inconsistent with the values of* r, a_1, x, *and* a_2, *and if the operations available to C are only multiplication and exponentiation, then C will not be able to get a new valid token.*

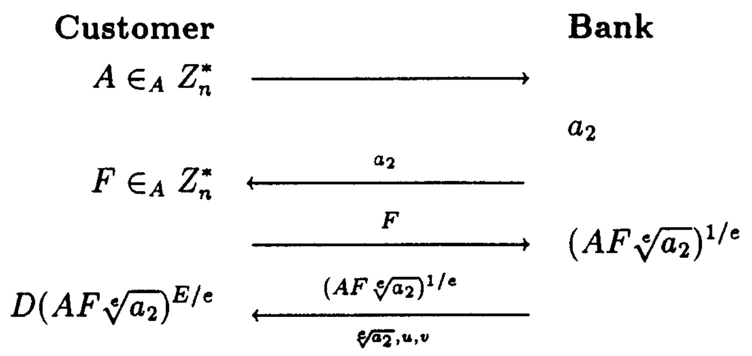
Customer
Bank
$A \in_A Z_n^*$
a_2
a_2
$F \in_A Z_n^*$
F
$(AF\sqrt[e]{a_2})^{1/e}$
$D(AF\sqrt[e]{a_2})^{E/e}$
$(AF\sqrt[e]{a_2})^{1/e}$
$\sqrt[e]{a_2}, u, v$

Figure 3: Chosen Message Attacks from a Customer

Proof. Suppose C attempts to deviate from the protocol. In step 1, C sends an arbitrary value A to B, and receives back a_2. In step 3, C sends a second arbitrary value F to B. Then C can compute any value of the form $D(AF\sqrt[e]{a_2})^{E/e}$. Figure 3 shows this scenario.

To have a valid signed token, C must have a pair of values K and $\sqrt[e]{Kf(K^e)}$. C can only generate values of the form $D(AF\sqrt[e]{a_2})^{E/e}$. This is equivalent to finding a solution to

$$D^e(AF\sqrt[e]{a_2})^E = Kf(K^e) \bmod n \tag{1}$$

Since C cannot invert f, C can only find a solution to the above equation by fixing a value for K, and solving for D, A, F, and E. Let H be equal to the value in equation 1.

Now, we prove that it is not possible to solve equation (1). We can rewrite the equation as

$$D^e = H(AF\sqrt[e]{a_2})^{-E} \bmod n.$$

H is fixed, and C must choose the values of A and F before C sees any value that includes $\sqrt[e]{a_2}$. Therefore, C can not successfully force or predict any value for $J = (AF\sqrt[e]{a_2}) \bmod n$.

Now, C must solve

$$D = \sqrt[e]{H}J^{-E/e} \bmod n.$$

C can only control the values of D and E in this equation. In particular, C can not compute the value of $\sqrt[e]{H}$ by the RSA assumption. □

Lemma 6 *C cannot reuse the token even if the merchant cooperates with him.*

Proof: Since the merchant sends $t = u(id_v||time) + v \bmod (w-1)$ to the bank, if C uses the token twice, (u, v) can be uniquely determined and the identity of C can be revealed. If the merchant sends the incorrect t and $(id_v||time)$ to the bank, since the bank keeps record of all triples (a, t, x), the bank will find that two identical signatures. By tracing the transaction (see Lemma 7), the bank can find the miscreant. (This is where we have used secret sharing [16].) □

Lemma 7 *If the merchant cooperates with the bank, then C's identity can be determined. If the merchant does not cooperate with the bank, then C's identity cannot be determined by the bank.*

Proof: If the merchant sends $(g^u \bmod w, g^v \bmod w)$ to the bank, the bank can compute a_2 and determine the identity of the customer. On the other hand, if the bank only knows $(ux + v \bmod (w-1), x)$, the bank can not determine C's identity. $\square$

4 Conclusions

Our electronic currency protocol is far more efficient then previous protocols. It also addresses an important social issue by allowing accounts to be audited by having the bank and the merchant cooperate, we feel that this is an important property for identifying sources of abuse. (This may arise if proper key management fails, for instance).

We have also shown that a wide class of chosen message attacks fail under our system by introducing the restrictive randomized blind signature technique. We believe that our protocol is practical for a number of applications.

References

[1] L. Adleman, K. Manders, and G. Miller. On taking Roots in Finite Fields. In *Proceedings of the 20th Annual Symposium on the Foundations of Computer Science*, 1979.

[2] E. Berlekamp. Factoring Polynomials over Large Finite Fields. *Mathematics of Computation*, Vol. 24, 1970.

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

[4] D. Chaum, A. Fiat, and M. Naor. Untraceable Eletronic Cash. In *Advances in Cryptology-Crypto88*, pages 319–327. Springer-Verlag, 1989.

[5] J. Evertse and E. Van Heyst. Which New RSA-signatures can be Computed from Certain Given RSA-signatures. *J. of Cryptology*, 5(1), 1992.

[6] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. of Cryptology*, 1(1), 1988.

[7] Niels Ferguson. Single Term Off-Line Coins. Technical Report TR CS-R9318, Centrum voor Wiskunde en Informatica, Amsterdam, 1993. (Also appeared in Advances in Cryptology-EuroCrypto'93).

[8] Goldreich, Goldwasser, and Micali. How to Construct Random Functions. In *25th Symposium on Foundations of Computer Science*, October 1984.

[9] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof System. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, 1985.

[10] J. Gray and A. Reuter. *Transaction Processing.* Morgan-Kaufmann, 1994.

[11] S. Haber and W. Stormetta. How to Time-Stamp a Digital Document. *J. of Cryptology*, 3(2), 1991.

[12] Butler W. Lampson. Reliable Messages and Connection Establishment. In Sape Mullender, editor, *Distributed Systems, 2nd Edition*. Addison-Wesley, 1993.

[13] T. Okamoto and K. Ohta. Universal Electronic Cash. In *Advances in Cryptology-Crypto91*. Springer-Verlag, 1992.

[14] M. Rabin. Digitalized signatures. In *Foundations of Secure Computation*. Academic Press, 1978.

[15] M. Rabin. Digital Signatures and Public Key Functions as Intractable as Factorization. Technical report, Laboratory for Computer Science, MIT, January 1979.

[16] Adi Shamir. How to Share a Secret. *Communications of ACM*, 22(11), 1979.

[17] S. Von Solms and D. Naccache. On Blind Signatures and Perfect Crimes. *Computer & Security*, 11(6), October 1992.

[18] D. Tygar and B. Yee. Dyad: A System for Using Physical Secure Coprocessors. Technical Report CMU-CS-91-140R, Carnegie Mellon University, 1992.