EFFICIENT OFFLINE ELECTRONIC CHECKS
(Extended Abstract)

David Chaum
Bert den Boer
Eugène van Heyst
Stig Mjølsnes
Adri Steenbeek

Centre for Mathematics and Computer Science
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

Abstract. Chaum, Fiat, and Naor proposed an offline check system [1], which has the advantage that the withdrawal and (anonymous) payment of a check are unlinkable. Here we present an improved protocol that saves 91% of the signatures, 41% of the other multiplications, 73% of the divisions, and 33% of the bit transmissions.

1. Introduction

In this paper we present a payment system that has the following advantages:

- the bank has only to give one signature per payment.
- the shop does not need online connection with the bank.
- during payment the payer has to do no computations.
- a payer can refund several checks at once, to keep the bank from learning the amount spent for each check.
- the withdrawal and payment of a check are unconditionally unlinkable, but if a check is spent twice, the identity of the cheater will be revealed with high probability.
- payments and refunds are unlinkable, except from that little that can be learned from the total number of each type of unspent denomination.
- refund and withdrawal can be made unlinkable.

In this paper we will only consider the withdrawal of one check. The payer (Alice) creates some candidates in a special way. The bank chooses randomly some of them, and the inner arguments of those selected candidates must be revealed by Alice (the "cut-and-choose" protocol). This is needed to prevent cheating users, or to be sure that there

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 294-301, 1990.

are not to many bad candidates left (i.e. candidates not created in the proper way). If more checks are withdrawn at the same time, less candidates have to be opened (to get the same security).

Because of the anonymous payment, checks must be bought for the maximal amount. Therefore the signature of the bank can be split by the user into two parts: one part is the check, which can be used for a payment; and the other is a refund part, which is used for refunding the unspent value.

2. Setting and Overview

The numbers which are used in this paper (such as the number of candidates in the denomination part, in the challenge part, or to be opened) *are examples*.

The underlaying scheme of this payment system, is an RSA scheme. As in the original paper [1], let f,g be two-argument, collision-free, one-way functions, with g such that if the first argument is fixed, the mapping is l-to-1 from the second argument onto the range. Let h be an injective one-way function with one argument and k an injective one-way function with 40 arguments. Let u_i be Alice's account number concatenated with a counter.

All calculations are modulo N, the factorization of which only the bank knows. In our formulas, we will omit the modular reduction. We use $\oplus$ to denote bitwise exclusive-or and $||$ for concatenation.

The check is a product of 20 terms, which are ordered in value: the first 10 are used for the amount to be spent in a shop (denomination part) and the last 10 to prevent the check from being spent twice (challenge part). These parameters can be varied, but the exponent then also needed to be changed (in order to prevent certain attacks). According to [2], we have the following table:

number of terms in denomination part +1	exponent
1,...,3	5
4,5	7
6,...,15	17
16,17	19
18,...,27	29

Three basic changes over [1] were made to improve efficiency :

(i) Alice initially sends to the bank *not* the candidates for the check, but the value of a

one-way function with the candidates as its arguments. Hence she avoids sending half of the candidates.

(ii) The terms in the check are ordered, so each can have almost the same root.

(iii) Alice sends a blinded product of the major and minor terms, so she needs only half as many blinding factors, half as much bandwidth is needed, and the bank makes only one signature. When she receives the signed check, though, she must do some calculations to separate the major and minor terms to separate the product of the signed minor terms for refund. Also an additional one-way function must be introduced for this.

3. Transactions

The payment system consists of three parties (bank, user Alice and shop) and four transactions: withdrawal, payment, deposit, and refund. Each of these transactions is a protocol between two of the three parties. The transactions do not need to be in this order; Alice can first refund a part of the check and later spend the rest of the check in a shop. Also the refund of a check can be done earlier than its deposit.

3.1. Check Withdrawal Transaction

(1) Alice first makes 40 candidates. For each, she chooses at random: r_i,a_i,b_i,c_i,d_i,e_i, ($1 \le i \le 40$) and computes:

$$x_i = g(a_i \| b_i, c_i),$$
$$y_i = g(a_i \oplus u_i, d_i),$$
$$M_i = f(x_i, y_i), \qquad \text{(called the } \textit{major term}\text{)}$$
$$m_i = h(g(b_i, e_i)), \qquad \text{(called the } \textit{minor term}\text{)}$$
$$\alpha_i = M_i^{3^{10}} \cdot m_i^{17} \cdot r_i^{17 \cdot 3^{10}} \qquad \text{(called the } \textit{candidate}\text{)}.$$

All computations can be made before connection with the bank, during which the hash value $k(\alpha_1, \alpha_2, \ldots, \alpha_{40})$ is sent.

(2) The bank splits the integers 1,...,40 randomly between two unordered partitions S_o and S_c, both of 20 elements. The partitioning is then sent to Alice.

(3) Alice orders the elements in the partition S_c by the M_i value of the corresponding candidate, forming the ordered set T_c. This set, the α_i's of the candidates in T_c, and the r_i,a_i,b_i,c_i,d_i,e_i of the candidates in S_0 are sent to the bank. The candidates in S_0 are said to be "opened".

(4) The bank verifies that the hash of candidates revealed equals $k(\alpha_1, \alpha_2, \ldots, \alpha_{40})$ and that every element of the opened partition is correctly formed. (The opened partition can now be discarded by both parties.) The bank takes a random integer R and computes:

$$D = \prod_{i=1}^{10} (\alpha_{t(i)})^{\frac{1}{17 \cdot 3^{11-i}}} \cdot \prod_{i=11}^{20} (\alpha_{t(i)})^{\frac{1}{17}} \cdot R^{\frac{1}{10}},$$

where $t(i)$ is the i^{th} element of T_c; sends D and R to Alice; and diminishes Alice's account with the value of the check ($=2^{10}$-1). The bank also stores R with Alice's account number.

(5) Alice verifies the validity of the signature on the check by testing whether

$$D^{17 \cdot 3^{10}} \stackrel{?}{=} \left(\left(\left(\prod_{i=11}^{20} (\alpha_{t(i)}) \right)^3 \cdot \alpha_{t(10)} \right)^3 \cdots \right)^3 \cdot \alpha_{t(1)} R^{17}.$$

She unblinds the signature on the check and divides it into two parts:

$$C = \prod_{i=1}^{10} (M_{t(i)})^{\frac{3^{i-1}}{17}} \cdot \prod_{i=11}^{20} (M_{t(i)})^{\frac{3^{10}}{17}},$$

$$C' = R^{\frac{17}{10}} \cdot \prod_{i=1}^{10} (m_{t(i)})^{\frac{17}{3^{11-i}}},$$

where C is the signed check and C' is used to get a refund for any part of C that is unspent. These can be computed any time before payment.

3.2. Payment Transaction

Alice can use a check to pay in a shop. The first 10 terms of a check C (remember that a check is the product of 20 ordered elements) have denominations $2^9,2^8,..,2,1$ respectively. These must be in decreasing order, because the denominations in the refund part C' *must* be in decreasing order; otherwise, Alice can claim more refund with C', because from $e^{17/3^i}$ she can easily compute $e^{17/3^j}$, $(0 \le j \le i)$. Every amount smaller than 1024 can be paid with C; let $(w_1,\ldots,w_{10})$ be the binary representation of the amount of payment.

(1) Alice gives C to the shop.

(2) The shop generates a binary challenge-vector $(w_{11},\ldots,w_{20})$ and sends it to Alice.

(3) Alice gives a partial opening of the check C:

if w_i=1, she reveals the corresponding $a_i \| b_i, c_i, y_i$;

if w_i=0, she reveals x_i, $a_i \oplus u_i, d_i$.

(4) The shop verifies the partial opening, the check's signature, and the ordering of the M_i's.

3.3. Deposit Transaction

(1) The shop sends to the bank: C, the vector $\underline{w}=(w_1,..,w_{20})$ (amount and challenge vector), and the partial opening.

(2) The bank verifies the signature and the partial opening, just as the shop did.
(3) The bank stores the check C in its searchable (batch) list of spent checks, the corresponding partial opening a_i or $a_i \oplus u_i$ ($1 \le i \le 20$) in an archive list, and the revealed b_i's in its searchable (batch) list of revealed minor terms. The bank consults the searchable lists to be sure that no b_i is already refunded or that no check is spent twice (possibly by sorting them sometime later). When double spending of a check is found, the u_i can be reconstructed from any difference in the corresponding vectors $\underline{w}$ and $\underline{w}'$, thereby revealing the cheater's account number.

3.4. Refund Transaction

(1) After each payment, but before refunding, the minor terms of the checks are accumulated and $g(b_i,e_i)$ is computed for each $w_i=1$. Alice sends the bank the product of the C', the R's, and in addition the $g(b_i,e_i)$ for each denomination spent, and the b_i,e_i for the denominations not spent.
(2) The bank verifies the opened minor terms and their signature C' similar to the way checks are verified. The bank also verifies if the R's were stored with Alice's account number.
(3) The bank verifies that the b_i's are not listed and stores them on its list of revealed minor terms together with Alice's account number, to prevent their later use.

Notice that in case of refunding multiple checks, Alice keeps the bank from learning the amount spent for each check; only the total number of each type of unspent denomination is revealed.

Alice can cheat by ordering the set improperly. She can get the maximal profit by spending the 5 highest denominations and telling the bank that she has spent the 5 lowest: she can spend a check of value 2^{10}-1 for 2^{11}-2^6. But in [2] it is proven that Alice can not change the ordering of the denomination part, so this kind of cheating is not possible.

4. Storage

For one check Alice stores the following integers in her card computer:
before withdrawal: $r_i,a_i,b_i,c_i,d_i,e_i,[M_i,m_i,\alpha_i]$ for each candidate;
after withdrawal: C,C',R and $a_i,b_i,c_i,d_i,e_i,[x_i,y_i]$ for each unopened candidate;
after payment: C',R and $b_i,e_i,[g(b_i,e_i)]$ for each unopened candidate.
If the card computer has only a very limited amount of memory, it is possible not to

store all the integers for each candidate; for instance not the integers between the square brackets. But then before payment or refund, Alice has to compute these integers, in order to do no computations during payment or refund. The memory that comes available after a transaction can be used to store candidates as they are be generated in advance.

5. Demo

Hans Beuze and Peter Sliepenbeek implemented this system on an Apple Macintosh, and Adri Steenbeek wrote the numerical part. Diskettes with this demo are available from the authors. There is also an earlier version available for an IBM pc.

6. Number representation

We suggest the following number representation:

u_i, a_i	:	32 bits,
b_i	:	128 bits,
c_i, d_i, r_i, e_i, N	:	512 bits,
f	:	$128\times128\rightarrow512$ bits,
g	:	$128\times512\rightarrow128$ bits,
h	:	$128\rightarrow512$ bits.

7. Improvements

7.1. Anonymous refund

In order to have anonymous refund, the payment system can be changed into the following way: in the withdrawal part, Alice chooses at random: $r_i, a_i, b_i, c_i, d_i, e_i, z_i$, $(1\le i\le 40)$ and computes:

$$x_i = g(a_i \| b_i \| z_i, c_i),$$
$$y_i = g(a_i \oplus u_i, d_i),$$
$$M_i = f(x_i, y_i),$$
$$m_i = h(g(b_i \| (z_i \oplus u_i), e_i)),$$
$$\alpha_i = M_i^{3^{10}} \cdot m_i^{17} \cdot r_i^{17 \cdot 3^{10}}.$$

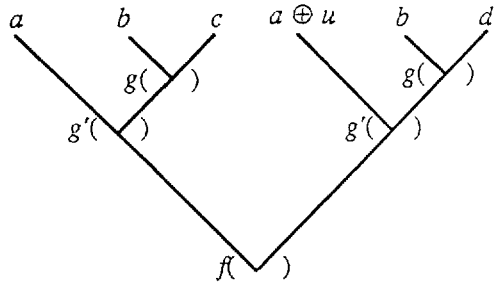
If Alice tries to spent a check twice or if she tries to ask refund for an already spent denomination, her identity will be revealed.

7.2. Combining challenge and denomination bits

We combine a challenge and a denomination bit into one term in the following way: in the withdrawal part Alice chooses at random: $r_i,a_i,b_i,c_i,d_i,e_i,z_i$, $(1 \le i \le 40)$ and computes:

$$x_i = g(b_i, c_i),$$
$$y_i = g(b_i, d_i),$$
$$v_i = g'(a_i, x_i),$$
$$w_i = g'(a_i \oplus u_i, y_i),$$
$$M_i = f(v_i, w_i),$$
$$m_i = h'(h(b_i, z_i)),$$
$$\alpha_i = M_i^{3^7} \cdot m_i^{17} \cdot r_i^{17 \cdot 3^7}.$$

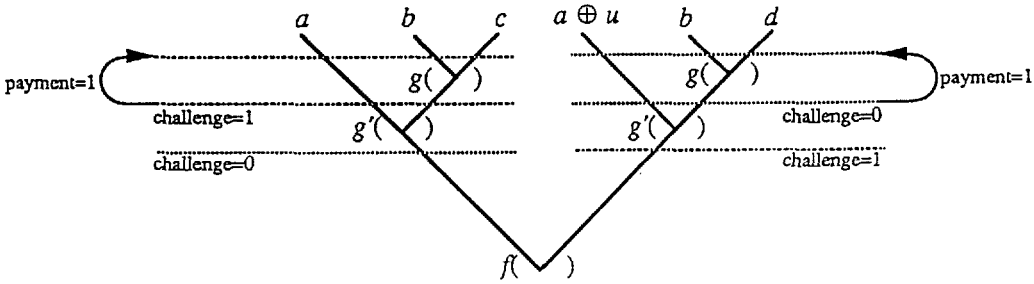
So the major term M can be expressed by the tree:



Let during the payment transaction $(k_1,\ldots,k_{10})$ be the binary representation of the amount of payment, and $(l_1,\ldots,l_{10})$ be the binary challenge-vector. Alice gives the following partial opening of the check:

l_i	k_i	revealed
0	0	$a_i \oplus u_i, v_i, y_i$
0	1	$a_i \oplus u_i, v_i, b_i, d_i$
1	0	a_i, w_i, x_i
1	1	a_i, w_i, b_i, c_i

The major term M can be expressed in the following tree, in which the dotted lines indicate the numbers revealed during the payment transaction: in each of the main subtrees, you start at a certain level (which depends on the challenge bit). If the payment bit is a one, then in one of the main subtrees, you go one level deeper.



References

[1] David Chaum, Amos Fiat, and Moni Naor, "Untraceable Electronic Cash", to appear in *Advances in Cryptology—Crypto '88*, Lecture Notes in Computer Science, Springer-Verlag.

[2] Jan-Hendrik Evertse and Eugène van Heyst, "Which RSA signatures can be computed from some given RSA signatures?", in preparation, 1989.