A New E-check System

Joseph K. Liu[1], **Sandy H. Wong**[2], **Duncan S. Wong**[3]
[1]Department of Information Engineering,
The Chinese University of Hong Kong, Hong Kong.

[2]Astri, Hong Kong.

[3]Department of Computer Science,
City University of Hong Kong, Hong Kong.

Abstract. We propose two e-check systems construction which can be regarded as the extensions of Fergerson's e-cash system [4]. The first one is as efficient as a single term e-cash system and supports partial unlinkability. The other one provides complete unlinkability with a more complex setting. Both of them do not require any trusted party and can be implemented efficiently.

Keywords: E-check, electronic commerce, electronic payment system

1 Introduction

E-check was introduced by Chaum, et al. [2, 3]. In [3], several offline cut-and-choose based e-check systems were proposed. The systems proposed in [1, 5] avoid using the cut-and-choose technique. However, [5] requires a trustee which knows the owner of each e-coin in the system even without double-spending. In this paper, we propose two e-check schemes by direct extension from an e-cash scheme [4]. The second scheme which provides complete linkability is as efficient as that in [1] in terms of computational complexity. The paper is organized as follows. In Sec. 2, the Single Term offline e-cash scheme proposed by Ferguson [4] is reviewed. This is followed by the proposed two e-check schemes in Sec. 3 and conclude the paper in Sec. 4.

2 Single Term Off-line Coins

It is an offline untraceable e-cash system without providing transferability [4]. The scheme is efficient and does not use the cut-and-choose methodology. Here we give a brief review of the scheme. Let n be the public RSA modulus of the bank and v be its public exponent. It is required that v is a reasonably large prime. Let g_a, g_b, g_c, h_b, h_c be publicly known integers such that $g_a, g_b, g_c \in \mathbb{Z}_n^*$ have large order and h_b, h_c are of order n in $GF(p)$, where $p-1$ is a multiple of n. Let U be an identity which is the concatenation of the user's identity and a unique coin number so that U is distinct for each e-coin. Let $f_1 : \{0,1\}^* \to \mathbb{Z}_v$ and $f_2 : \{0,1\}^* \to \mathbb{Z}_n^*$ be cryptographic hash functions.

2.1 Withdrawal Protocol

The withdrawal protocol consists of three parallel runs of the randomized blinded RSA signature scheme [4]. The user picks $c_1, a_1, b_1 \in_R \mathbb{Z}_n^*$, $\sigma, r, \phi \in_R \mathbb{Z}_v$, $\gamma, \alpha, \beta \in_R \mathbb{Z}_n$, and computes $G_c = \gamma^v c_1 {g_c}^\sigma \bmod n$, $G_a = \alpha^v a_1 {g_a}^r \bmod n$, and $G_b = \beta^v b_1 {g_b}^\phi \bmod n$. It sends $M_1 = (U, G_c, G_a, G_b)$ to the bank. For simplicity, we omit the notation of modular reduction in the rest of the paper when it gets clear from its context. The bank picks $c_2, a_2, b_2 \in_R \mathbb{Z}_n^*$ and sends $M_2 = ({h_c}^{c_2} \bmod p,\ a_2,\ {h_b}^{b_2} \bmod p)$ to the user. The user picks $t_1 \in_R \mathbb{Z}_v^*$ and computes $e_c = f_1({h_c}^{c_1 c_2}) - \sigma \bmod v$, $e_b = f_1({h_b}^{b_1 b_2}) - \phi \bmod v$, $a = (a_1 a_2 f_2(e_c, e_b))^{t_1} \bmod n$, and $e_a = \frac{1}{t_1} f_1(a) - r \bmod v$. It then sends $M_3 = (e_c, e_a, e_b)$ to the bank[1]. The user also signs (M_1, M_2, M_3) and sends the signature to the bank[2].

The bank computes $\overline{C} = G_c c_2 {g_c}^{e_c}$, $\overline{A} = G_a a_2 f_2(e_c, e_b) {g_a}^{e_a}$, and $\overline{B} = G_b b_2 {g_b}^{e_b}$ and selects $t_2 \in_R \mathbb{Z}_v^*$. It sends $(c_2, b_2, t_2, (\overline{C}^{t_2} \overline{A})^{1/v}, (\overline{C}^U \overline{B})^{1/v})$ to the user. The user computes $c = c_1 c_2$, $b = b_1 b_2$, $t = t_1 t_2 \bmod v$, $C = c {g_c}^{f_1({h_c}^c)}$, $A = a {g_a}^{f_1(a)}$, $B = b {g_b}^{f_1({h_b}^b)}$, $S_a = (\frac{(\overline{C}^{t_2} \overline{A})^{1/v}}{\gamma^{t_2} \alpha})^{t_1}$ and $S_b = \frac{(\overline{C}^U \overline{B})^{1/v}}{\gamma^U \beta}$, and checks whether ${S_a}^v \overset{?}{=} C^t A$ and ${S_b}^v \overset{?}{=} C^U B$. If these two equalities hold, it accepts. The user stores (a, b, c, t, S_a, S_b) as an e-coin. (a, b, c) are the *base numbers* of the coin.

2.2 Payment Protocol

To spend an e-coin (a, b, c, t, S_a, S_b), the user executes the following protocol with the shop. The user sends (a, b, c) to the shop. The shop randomly chooses a challenge x and sends it to the user. The user computes and sends $r = tx + U$ and $\mathcal{S} = (S_a)^x (S_b)$ to the shop. The shop computes $C = c {g_c}^{f_1({h_c}^c)}$, $A = a {g_a}^{f_1(a)}$, $B = b {g_b}^{f_1({h_b}^b)}$ and checks if $\mathcal{S}^v \overset{?}{=} C^r A^x B$. If the equality holds, the shop accepts the coin and stores $(a, b, c, x, r, \mathcal{S})$. Otherwise, it rejects. $(x, r, \mathcal{S})$ is a proof of the user's ownership to the

[1]Note that the exponents e_c, e_b and e_a are computed modulo v. Certain corrections in the final signature (S_a, S_b) are needed to make the blinding perfect. This is done by multiplying the final signature by a suitable powers of g_c, g_a and g_b [4]. Corrections are not shown in this paper.

[2]This corresponds to a signature of the user for all the data in the first three transmissions. It is used to protect the user against framing by the bank. We refer readers to [4] for detail.

e-coin with base number (a, b, c). Obviously, the user can only provide one proof in order to prevent from revealing its identity.

2.3 Deposit Protocol

To deposit an e-coin, it sends $(a, b, c, x, r, \mathcal{S})$ to the bank. The bank verifies the coin by following the steps below. Compute $C = cg_c{}^{f_1(h_c{}^c)}$, $A = ag_a{}^{f_1(a)}$ and $B = bg_b{}^{f_1(h_b{}^b)}$. Check if $\mathcal{S}^v \stackrel{?}{=} C^r A^x B$. If it is false, the bank rejects the deposit. Otherwise, it checks if (a, b, c) are already existed in its database. If yes, the bank rejects the deposit. Otherwise it accepts and stores (a, b, c) in its database and it credits the shop. Double-spending is detected if the bank finds the same triple (a, b, c) are already in its database. If the corresponding $(x, r, \mathcal{S})$ are the same as the ones stored in the database, the bank concludes that the shop is cheating. Otherwise, it concludes that the user double spends the coin. The identity of the user, U can be obtained easily by solving the two linear equations.

3 Our Proposed E-Checks

We present two e-check systems. The first one is highly efficient and supports partial unlinkability. The second one supports complete unlinkability.

3.1 E-Check I

We use the same notations as before. In this e-check system, there is a list of reasonable large prime numbers $(v_1, \cdots, v_k)$ as public exponents of the bank with v_i corresponding the value of $\$2^{i-1}$. Define that multiplying any set of v_i, $1 \le i \le k$, represents to the sum of their corresponding values. v_d denotes the public exponent of the bank representing $\$d$ such that

$$v_d = \prod_{i=1}^{k} v_i <d>_i \tag{1}$$

where $<d>_i$ denotes the value of the i-th least significant bit of d. For example, $<6>_1 = 0, <6>_2 = 1, <6>_3 = 1$. In this way, we can represent any amount up to $\$2^k - 1$.

Withdrawal Protocol Suppose a user wants to withdraw an e-check of maximum value $\$2^k - 1$. The withdrawal protocol is the same as Ferguson's one (Sec. 2.1) by having the public exponent $v = v_1 \cdot v_2 \cdots v_k$. Note that the maximum value of the e-check must be in the form of $\$2^i - 1$, for any $i > 1$. That is, all the bits of the maximum value of the e-check should be 1 in its binary representation. This ensures that the devaluation of $\overline{v_d}$ (first step of the Payment Protocol below) is always computable. Let the e-check be denoted as $K = (a, b, c, t, S_a, S_b)$ where (a, b, c) are the base numbers of the check.

Payment Protocol Suppose the user wants to spend $\$d$ to the shop, where $1 \le d \le 2^k - 1$. The corresponding public exponent of the bank is v_d which can be publicly computed from Eq. (1). First, the user 'devalues' the check from $\$2^k - 1$ to $\$d$. It proceeds as follows: The user computes $\overline{v_d} = v_1 \cdots v_k \text{ div } v_d$, and $S'_a = (S_a)^{\overline{v_d}}$, $S'_b = (S_b)^{\overline{v_d}}$. Note: div is normal division without taking modulo. Then he sends the base numbers of the check (a, b, c) to the shop. The shop randomly picks a challenge x and sends it to the user. The user computes $r = tx + U$, $\mathcal{S} = (S'_a)^x (S'_b)$ and sends $r, \mathcal{S}$ to the shop. The shop computes $C = cg_c{}^{f_1(h_c{}^c)}$, $A = ag_a{}^{f_1(a)}$, $B = bg_b{}^{f_1(h_b{}^b)}$ and checks whether $\mathcal{S}^{v_d} \stackrel{?}{=} C^r A^x B$. If it is true, the shop accepts and stores $(a, b, c, x, r, \mathcal{S}, d)$. Otherwise, it rejects.

Deposit and Refund Protocols The deposit protocol of our e-check system is the same as Ferguson's (Sec. 2.3), with the public exponent $v = v_d$. The user can refund the remaining $\$2^k - 1 - d$ from the bank by executing a refund protocol. The protocol is almost the same as the deposit protocol, except the checking of double spending. In the refund protocol, the user sends the used check-tuple $(a, b, c, x, r, \mathcal{S}, d)$ to the bank. The bank verifies user's ownership of the e-check by first carries out the steps similar to the payment protocol, namely it sends a challenge x' and obtains a response pair $(r', \mathcal{S}')$. Then it checks if the base numbers (a, b, c) are already in its database. If it exists and the amount is d, the bank refunds the remaining $\$2^k - 1 - d$ to the user and updates its database to record that the e-check has already been refunded.

Note that this part is not anonymous. The bank knows the identity of the user who asks for refund. The bank can also link the e-check which has already spent by the user in earlier time.

3.2 E-Check II

The scheme in the last section is linkable at the refund stage. We now propose another scheme which is completely unlinkable. In this scheme, the bank has only one public exponent v. Instead, we use different elements $g_{a_i} \in \mathbb{Z}_n^*, 1 \le i \le k$ of large order to represent different values of the e-check. Like the representation system in E-Check I, we use g_{a_i} to represent $\$2^{i-1}$. In this way, with k consecutive elements, the e-check has a maximum value of $\$2^k - 1$. We further use g_{a_0} to prevent a user from using the e-check twice or more. Thus g_{a_0} is included in the payment of an e-check regardless of the payment amount. E-Check II is similar to Ferguson's e-cash system. However, there are $k+1$ signatures in each e-check if its maximum value is $\$2^k - 1$, one is for embedding the identity of the user to prevent double-spending while the others are for composing the value of the e-check.

Withdrawal Protocol Suppose a user wants to withdraw an e-check of $\$2^k - 1$. Let $g_{a_0}, g_{a_1}, \cdots, g_{a_k}, g_b, g_c$ be public where $g_{a_0}, g_{a_1}, \cdots, g_{a_k}, g_b, g_c$ are of large order in $\mathbb{Z}_n^*$. The Withdrawal Protocol proceeds as follows.

The user picks $b_1, c_1, a_{1_0}, a_{1_1}, \cdots, a_{1_k} \in_R \mathbb{Z}_n^*$, $\sigma, \phi, r_0, r_1, \ldots, r_k \in_R \mathbb{Z}_v$ and $\gamma, \beta, \alpha_0, \alpha_1, \cdots, \alpha_k \in_R \mathbb{Z}_n$. It then computes $G_b = \beta^v b_1 g_b{}^{\phi}$, $G_c = \gamma^v c_1 g_c{}^{\sigma}$ and $G_{a_i} = \alpha_i{}^v a_{1_i} g_{a_i}{}^{r_i}$, for $i = 0, \ldots, k$, and sends $M_1 = (U, G_b, G_c, G_{a_0}, \cdots, G_{a_k})$ to the bank.

The bank picks $b_2, c_2, a_{2_0}, a_{2_1}, \cdots, a_{2_k} \in_R \mathbb{Z}_n^*$ and sends $M_2 = (h_b{}^{b_2}, h_c{}^{c_2}, a_{2_0}, a_{2_1}, \cdots, a_{2_k})$ to the user.

The user picks $t_{1_0}, t_{1_1}, \cdots, t_{1_k} \in_R \mathbb{Z}_v^*$, computes $e_b = f_1({h_b}^{b_1 b_2}) - \phi \bmod v$, $e_c = f_1({h_c}^{c_1 c_2}) - \sigma \bmod v$, $a_i = (a_{1_i} a_{2_i} f_2(i, e_c, e_b))^{t_{1_i}}$, and $e_{a_i} = \frac{1}{t_{1_i}} f_1(a_i) - r_i \bmod v$, $0 \leq i \leq k$. He sends $M_3 = (e_b, e_c, e_{a_0}, e_{a_1}, \cdots, e_{a_k})$ to the bank.

The user also signs (M_1, M_2, M_3) and sends the signature to the bank. Note: refer to Sec. 2.1 for discussions. The bank computes $\overline{C} = G_c c_2 {g_c}^{e_c}$, $\overline{B} = G_b b_2 {g_b}^{e_b}$, $\overline{A_i} = G_{a_i} a_{2_i} f_2(i, e_c, e_b) {g_{a_i}}^{e_{a_i}}$, , $0 \leq i \leq k$. The bank selects $t_{2_0}, t_{2_1}, \ldots, t_{2_k} \in_R \mathbb{Z}_v^*$ and sends

$$c_2, b_2, \{t_{2_i}\}_{0 \leq i \leq k},\ \{(\overline{C}^{t_{2_i}} \overline{A_i})^{1/v}\}_{0 \leq i \leq k},\ (\overline{C}^U \overline{B})^{1/v}$$

to the user. The user computes $c = c_1 c_2$, $b = b_1 b_2$, $t_i = t_{1_i} t_{2_i} \bmod v$, $0 \leq i \leq k$, $B = b {g_b}^{f_1({h_b}^b)}$, $C = c {g_c}^{f_1({h_c}^c)}$, $A_i = a_i {g_{a_i}}^{f_1(a_i)}$, $0 \leq i \leq k$, $S_b = \frac{(\overline{C}^U \overline{B})^{1/v}}{\gamma^U \beta}$, $S_i = (\frac{\overline{C}^{t_{2_i}} \overline{A_i})^{1/v}}{\gamma^{t_{2_i}} \alpha_i})^{t_{1_i}}$, $0 \leq i \leq k$, and checks whether ${S_b}^v \stackrel{?}{=} C^U B$ and ${S_i}^v \stackrel{?}{=} C^{t_i} A_i$, for $i = 0, \ldots, k$. If all the equalities hold, he accepts. The user stores $(a_0, \cdots, a_k, b, c, t_0, \cdots, t_k, S_b, S_0, S_1, \cdots, S_k)$ for the payment of the e-check.

Payment Protocol Suppose the user wants to spend $\$2^j - 1$, for some $1 \leq j \leq k$, to the shop. The payment protocol proceeds as follows.

The user sends $b, c, a_0, \ldots, a_j$ to the shop. The shop selects a challenge number x and sends it to the user. The user computes $r_i = t_i x + U$ and $\mathcal{S}'_i = (S_b)(S_i)^x$, and sends $(r_i, \mathcal{S}'_i)$, $0 \leq i \leq j$, to the shop. The shop computes

$$\begin{aligned} C &= c {g_c}^{f_1({h_c}^c)} \\ B &= b {g_b}^{f_1({h_b}^b)} \\ A_i &= a_i {g_{a_i}}^{f_1(a_i)}, 0 \leq i \leq j, \end{aligned}$$

and checks whether ${\mathcal{S}'_i}^v \stackrel{?}{=} C^{r_i} {A_i}^x B$ for $0 \leq i \leq j$. If all the equalities hold, the shop accepts and stores $(a_0, \cdots, a_j, b, c, x, r_0, \cdots, r_j, \mathcal{S}'_0, \cdots, \mathcal{S}'_j)$. Otherwise, it rejects.

Deposit Protocol The deposit protocol is constructed in its natural way. When the shop deposits the e-check, it sends the check-tuple

$$(a_0, \cdots, a_j, b, c, x, r_0, \cdots, r_j, \mathcal{S}'_0, \cdots, \mathcal{S}'_j)$$

to the bank. The bank verifies of the tuple as follows.

Compute $C = c {g_c}^{f({h_c}^c)}$, $B = b {g_b}^{f({h_b}^b)}$, $A_i = a_i {g_{a_i}}^{f(a_i)}$, for $i = 0, \cdots, j$. Check whether ${\mathcal{S}'_i}^v \stackrel{?}{=} C^{r_i} {A_i}^x B$, $0 \leq i \leq j$. If not all equal, the bank rejects the deposit. Check whether the same values of (a_0, b, c) already exist in its database. If yes, the bank rejects the deposit and the double-spender can easily be found. Otherwise, it accepts and credits the shop.

Refund Protocol If the user wants to refund the remaining amount of the e-check, that is, $\$2^k - 1 - (2^j - 1) = \$2^k - 2^j$, he has to inform the bank his account number and his identity U for the refund purpose and execute the following steps.

The user sends $U, a_{j+1}, \cdots, a_k$ and $t_{j+1}, \cdots, t_k$ to the bank. The bank retrieves $\overline{B}, \overline{C}$ from the withdrawal record. The bank checks if any of $a_{j+1}, \cdots, a_k$ are already in the database. If yes, it rejects. Otherwise, the bank selects a challenge number x and sends it to the user. The bank also computes $r_i = t_i x + U$, for $j + 1 \leq i \leq k$. The user computes $r_i = t_i x + U$ and $\mathcal{S}''_i = (S_b)(S_i)^x \gamma^{r_i} \beta$, and sends $\mathcal{S}''_i$, $j + 1 \leq i \leq k$, to the bank. The bank computes $A_i = a_i {g_{a_i}}^{f(a_i)}$ and checks whether ${\mathcal{S}''_i}^v \stackrel{?}{=} \overline{C}^{r_i} {A_i}^x \overline{B}$ for $j + 1 \leq i \leq k$. If not all of them are equal, the bank rejects. Otherwise, the bank records that the e-check has been refunded in its database and refunds $\$2^k - 2^j$ to the user.

4 Conclusion

We have proposed two e-check systems. One is almost as efficient as a single term e-cash such as [4] with partial unlinkability only. The other one provides complete unlinkability with a more complex setting. We believe these e-check systems can be implemented practically.

References

[1] S. Brands. An Efficient Off-line Electronic Cash System based on the Representation Problem. *Technical Report CS-R9323*, CWI, April, 1993.

[2] D. Chaum. Online Cash Checks. *Proc. EUROCRYPT 89*, LNCS 0403, pp. 288-293. Springer-Verlag, 1990.

[3] D. Chaum, B. den Boer, E. van Heyst, S. Mjolsnes and A. Steenbeek. Efficient offline electronic checks. *Proc. EUROCRYPT 89*, LNCS 0403, pp. 294-301. Springer-Verlag, 1990.

[4] N. Ferguson. Single Term Off-line Coins. *Proc. EUROCRYPT 93*, LNCS 0765, pp. 318-328. Springer-Verlag, 1993.

[5] A. de Solages and J. Traore. An Efficient Fair Offline Electronic Cash System with Extensions to Checks and Wallets with Observers. *Financial Cryptography 1998*, LNCS 1465, pp. 275-295. Springer-Verlag, 1998.