**SCIS 2008** The 2008 Symposium on Cryptography and Information Security
Miyazaki, Japan, Jan. 22-25, 2008
The Institute of Electronics, Information and Communication Engineers

# On the Security of an Unlinkable Off-line E-Cash System

Yosuke Tomii * Yoshifumi Manabe Tatsuaki Okamoto †

**Abstract**— This paper presents an off-line anonymous e-cash schemes, that is secure under the strong RSA assumption and the strong Diffie-Hellman (SDH) assumption. A user can withdraw a wallet containing $2^k$ coins, each of which she can spend unlinkably. The complexity of the withdrawal operation is $\mathcal{O}(k^4)$, the spend operation is $\mathcal{O}(k^3)$, where $k$ is security parameter. The user's wallet can be stored using $\mathcal{O}(k)$ bits. Our scheme also offers exculpability of users, that is, the bank can prove to third parties that a user has double-spent. Our scheme is secure in the random oracle model.

**Keywords:** e-cash

## 1 Introduction

### 1.1 Background

Electronic cash was proposed by Chaum [2][3], and has been extensively studied [4][5][6][7][8][9][10][11][12][13].

As a coin is represented by digital data, and it is easy to duplicate data, an electronic cash scheme requires a mechanism that prevents a user from spending the same coin twice (double-spending). There are two scenarios. In the *on-line* scenario, the bank is on-line in each transaction to ensure that no coin is spent twice, and each merchant must consult the bank before accepting a payment. In the *off-line* scenario, the merchant accepts a payment autonomously, and later submits the payment to the bank; the merchant is guaranteed that such a payment will be either honored by the bank, or will lead to the identification (and therefore punishment) of the double-spender.

In this paper, we give an off-line $2^l$-spendable unlinkable electronic cash scheme. Our paper's framework is based on [15] by Camenisch.

### 1.2 Our Result

This paper propose a new efficient unlinkable off-line electronic cash scheme secure in the random oracle model. The security proof of our scheme depends on the RSA assumption and the SDH assumption.

## 2 Preliminaries

### 2.1 Definition of Off-line E-Cash System

Our electronic cash scenario consists of three usual players: user $\mathcal{U}$, bank $\mathcal{B}$, and merchant $\mathcal{M}$; together with the algorithms: BKeygen,UKeygen, MKeygen, Withdraw,Spend,Deposit, Identify,Trace and VerifyOwnership.

- BKeygen is a key generation algorithm for bank $\mathcal{B}$. It takes as input $k$ bit security parameter, and outputs the key pair, $(\mathsf{pk_B}, \mathsf{sk_B})$.
- UKeygen is a key generation algorithm for

* Kyoto University, Department of Social Informatics
† NTT Labs, Nippon Telegraph and Telephone Corporation

user $\mathcal{U}$. It takes as input $k$ bit security parameter, and outputs the key pair, $(\mathsf{pk_U}, \mathsf{sk_U})$.

- Withdraw is a protocol between $\mathcal{U}$ and $\mathcal{B}$. $\mathcal{U}$ withdraws a $2^l$ unit wallet $\mathcal{W}$ with serial number $S$. $\mathcal{U}$ sends signature $\mathcal{Q}$ to $\mathcal{B}$. $\mathcal{B}$ records $\mathcal{Q}$ in database $\mathcal{D}$ to trace user's double spending some coin. $\mathcal{U}$ receives $\mathcal{B}$'s signature.

- Spend is a protocol between $\mathcal{U}$ and $\mathcal{M}$. $\mathcal{U}$ sends zero-knowledge proof of knowledge of $\mathcal{W}$ $\Phi$ to $\mathcal{M}$.

- Deposit is a protocol between $\mathcal{M}$ and $\mathcal{B}$. $\mathcal{M}$ sends $\Phi$ to $\mathcal{B}$. $\mathcal{B}$ verifies $\Phi$. If the coin has been received already, $\mathcal{B}$ rejects $\Phi$. Otherwise, $\mathcal{B}$ accepts it.

- Identify is an algorithm to find double-spender $\mathcal{U}'$ from double spent coin $\Phi_1$,$\Phi_2$.

- Trace is an algorithm to output evidence $\Pi$, which $\mathcal{B}$ computes from $\Phi_1$,$\Phi_2$ and $\mathcal{D}$, to be used in the VerifyOwnership step.

- VerifyOwnership is an algorithm to confirm that $\mathcal{U}'$ certainly spent coin $\Phi_1$,$\Phi_2$. Anyone can verify double spent coin via serial number $S$ and $\Pi$.

## 2.2 Definition of Security

### 2.2.1 Unforgeability

Adversary $\mathcal{A}$ is given the bank's public key $\mathsf{pk_B}$. First, $\mathcal{A}$ interacts with $\mathcal{B}$ $K$ times in the Withdraw protcol. $\mathcal{B}$ issues $2^L$ coins in each withdrawal. Second, $\mathcal{A}$ executes the Spend protocol with $\mathcal{M}$. Last, $\mathcal{M}$ executes the Deposit protocol with $\mathcal{B}$. $\mathcal{A}$ wins the game if $\mathcal{B}$ accepts $2^L K+1$ coins in the Deposit protocol. We define $\mathsf{Adv}_{\mathcal{A}}^{unforge}$ to be the probability that $\mathcal{A}$ wins the above game, taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{B}$.

### 2.2.2 Identification of double-spenders

Adversary $\mathcal{A}$ is given the bank's public key $\mathsf{pk_B}$. First, $\mathcal{A}$ interacts $K$ times with $\mathcal{B}$ in the Withdraw protcol. $\mathcal{B}$ issues $2^L$ coins in each withdrawal. Last, $\mathcal{A}$ executes the Spend protocol with $\mathcal{M}$. $\mathcal{A}$ wins the game if $\mathcal{M}$ accepts $2^L K + 1$ coins and $\mathcal{M}$ cannot output $\mathcal{A}$'s secret key. We define $\mathsf{Adv}_{\mathcal{A}}^{identify}$ to be the probability that $\mathcal{A}$ wins the above game, taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{M}$.

### 2.2.3 Tracing double-spenders

Adversary $\mathcal{A}$ is given the bank's public key $\mathsf{pk_B}$. First, $\mathcal{A}$ interacts $K$ times with $\mathcal{B}$ in the Withdraw protcol. $\mathcal{B}$ issues $2^L$ coins in each withdrawal. Second, $\mathcal{A}$ executes the Spend protocol with $\mathcal{M}$. Last, $\mathcal{M}$ executes the Deposit protocol. $\mathcal{B}$ accepts $2^L K + 1$ coins and outputs evidence $\Pi$. $\mathcal{A}$ wins the game if $\mathcal{B}$ cannot output valid verify-ownership data. We define $\mathsf{Adv}_{\mathcal{A}}^{trace}$ to be the probability that $\mathcal{A}$ wins the above game, taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{M}$.

### 2.2.4 Anonymity of users

Adversary $\mathcal{A}$ sets a bank's secret and public key $\mathsf{pk_B}, \mathsf{sk_B}$. Honest users $\mathcal{U}_0$, $\mathcal{U}_1$ execute the withdraw protcol, and get wallet $\mathcal{W}_0$, $\mathcal{W}_1$, respectively.

One of $\mathcal{U}_0$ and $\mathcal{U}_1$ is now selected randomly, say $\mathcal{U}_b$. $\mathcal{U}_b$ executes the spend protocol. $\mathcal{A}$ outputs $b' = 0$ or 1.
$\mathsf{Adv}_{\mathcal{A}}^{Anonymity} := 2Pr[b = b'] - 1$

### 2.2.5 Exculpability

Adversary $\mathcal{A}$ sets $\mathsf{pk_B}, \mathsf{sk_B}$. An honest $\mathcal{U}$ executes withdraw and spend protocols as many times as $\mathcal{A}$ wishes.

$\mathcal{A}$ wins the game if $\mathcal{A}$ outputs $(S, \Pi)$ of user $\mathcal{U}$ such that
VerifyOwnership$(S, \Pi)$ returns accept.

## 2.3 Verifiable Encryption

In Section 4.2, we apply a technique by Camenisch and Damgard [14] for turning any semantically secure encryption scheme into a verifiable encryption scheme. A verifiable encryption scheme is a two-party protocol between a

prover and encryptor $\mathcal{U}$ and a verifier and receiver $\mathcal{B}$.

In the following, the verifiable encryption of a committed value is shown, in which ElGamal encryption is applied to the keys using bilinear maps.

・Encryption and Decryption

$\tilde{g} \in \mathbb{G}_1$ and $g, f, h \in \mathbb{G}_2$ are public data. $\mathcal{U}$ randomly chooses $u \in \mathbb{Z}_p^*$ and computes $e(\tilde{g}, g^u) = e(\tilde{g}, g)^u$. $(\mathsf{p_k}, \mathsf{s_k}) := (e(\tilde{g}, g)^u, g^u)$. Let $m$ be the plaintext and $c$ the cyphertext.

$$\begin{aligned} \text{Encrypt}: \quad & \mathcal{U} \text{ randomly chooses } k \in \mathbb{Z}_p^*. \\ & c := (c_1, c_2) = (\tilde{g}^k, {\mathsf{p_k}}^k m). \\ \text{Decrypt}: \quad & m = \frac{c_2}{e(c_1, g^u)} \end{aligned}$$

・A Verifiable Encryption Scheme

$A := \tilde{g}^u \tilde{f}^t \tilde{h}^s$ is a commitment to $s$. $E(s) := (\tilde{g}^k, {\mathsf{p_k}}^k s)$ is an encryption of $s$. $\mathcal{U}$ randomly chooses $r_1, r_2, r_3, r_4, k_1, k_2 \in \mathbb{Z}_p^*$. $\mathcal{U}$ computes

$$\begin{aligned} X &= \tilde{f}^{r_1} \tilde{g}^{r_2} \tilde{h}^{r_3} \\ c_{11} &= r_1 + u \bmod p \\ c_{12} &= r_2 + t \bmod p \\ c_{13} &= r_3 + s \bmod p \\ c_{21} &= r_1 + 2u \bmod p \\ c_{22} &= r_2 + 2t \bmod p \\ c_{23} &= r_3 + 2s \bmod p \\ e_1 &= (\tilde{g}^{k_1}, {\mathsf{p_k}}^{k_1}(c_{11}||c_{12}||c_{13})) \\ e_2 &= (\tilde{g}^{k_2}, {\mathsf{p_k}}^{k_2}(c_{21}||c_{22}||c_{23})) \ , \end{aligned}$$

and sends $\{X, e_1, e_2\}$ to $\mathcal{B}$.

$\mathcal{B}$ returns to $a = \{1 \text{ or } 2\}$ randomly.

$\mathcal{U}$ sends $\{c_{a1}, c_{a2}, c_{a3}, k_a\}$ to $\mathcal{V}$. Let $\bar{a} = \{1$ if $a = 2, 2$ if $a = 1\}$.

$\mathcal{B}$ verifies

$$\begin{aligned} e_a &= (\tilde{g}^{k_a}, {\mathsf{p_k}}^{k_a}(c_{a1}||c_{a2}||c_{a3})) \\ \tilde{f}^{c_{a1}} \tilde{g}^{c_{a2}} \tilde{h}^{c_{a3}} &= X A^a \\ E(s) &= (e_{\bar{a}}, c_{a1}||c_{a2}||c_{a3}) \end{aligned}$$

By decripting $e_{\bar{a}}$, B obtains $c_{\bar{a}1}$, $c_{\bar{a}2}$ and $c_{\bar{a}3}$, and calculates $s$ by $s := c_{23} - c_{13} \bmod p$.

This protocol is repeated $k$ times, and $\mathcal{U}$ succeds in cheating $\mathcal{B}$ with probability $\frac{1}{2^k}$.

### 2.4 Committed Number Lies in an Interval

In Section 4.3, we apply a technique proposed by Boudot [1] to prove Committed Number:$J$ belongs to $[0, 2^k)$. This requires the strong RSA assumption.

### 2.5 Signature Scheme

In Sections 4.2, 4.3, 4.7, we apply a signature scheme proposed by Okamoto [16] to achieve anonymity and traceability. The signature scheme is existentially unforgeable against adaptive chosen message attacks.

・Key generation

Randomly select generators $g_2, u_2, v_2 \in \mathbb{G}_2$ and set $g_1 \leftarrow \phi(g_2)$, $u_1 \leftarrow \phi(u_2)$ and $v_1 \leftarrow \phi(v_2)$. Randomly select $x \in \mathbb{Z}_p^*$ and compute $w_2 \leftarrow {g_2}^x \in \mathbb{G}_2$. The public and secret keys are:

Public key : $g_1, g_2, w_2, u_2, v_2$,

Secret key : $x$

・Signature generation

Let $m \in \mathbb{Z}_p^*$ be the message to be signed. Signer $\mathcal{S}$ randomly selects $r$ and $s$ from $\mathbb{Z}_p^*$ and computes

$$\sigma \leftarrow ({g_1}^m u_1 {v_1}^s)^{\frac{1}{x+r}} \ .$$

$(\sigma, r, s)$ is the signature of $m$.

・Signature verification

Given public-key $(g_1, g_2, w_2, u_2, v_2)$, message $m$, and signature $(\sigma, r, s)$, check that $m, r, s \in \mathbb{Z}_p^*$, $\sigma \in \mathbb{G}_1$, $\sigma \neq 1$, and

$$e(\sigma, w_2 {g_2}^r) = e(g_1, {g_2}^m u_2 {v_2}^s) \ .$$

If they hold, the verification result is **valid**; otherwise the result is **invalid**.

・Security of signature

The signature scheme is secure against adaptive chosen message attacks in the standard model under the SDH-assumption.[16]

## 3 Assumptions

### 3.1 Strong RSA Assumption:

Given RSA module $n$ and random element $g \in {\mathcal{Z}^*}_n$, it is hard to compute $h \in {\mathcal{Z}^*}_n$ and

integer $e > 1$ such that $h^e \equiv g \bmod n$. Module $n$ has special form $pq$, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes. The S-RSA problem is defined as follows: given $n$ as input, output pair $(a, b)$
$\mathsf{Adv}_{S-RSA} := \Pr[n = ab]$
Adversary $\mathcal{A}(t, \epsilon)$-breaks S-RSA problem if $\mathcal{A}$ runs in time at most $t$ and $\mathsf{Adv}_{S-RSA}$ is at least $\epsilon$. The $(t, \epsilon)$-S-RSA assumption holds if no adversary $\mathcal{A}$ $(t, \epsilon)$-breaks the S-RSA problem.

### 3.2 Discrete Logarithm (DL) Assumption:

Given a large prime $p$ and random elements $f, g, h \in \mathcal{G}$ order $p$. It is hard to find $x, y, z$ that $f^x = g^y h^z$.
$\mathsf{Adv}_{DL} := \Pr[A(f, g, h) = (x, y, z : f^x = g^y h^z)]$
Adversary $\mathcal{A}(t, \epsilon)$-breaks DL problem if $\mathcal{A}$ runs in time at most $t$ and $\mathsf{Adv}_{DL}$ is at least $\epsilon$. The $(t, \epsilon)$-DL assumption holds if no adversary $\mathcal{A}$ $(t, \epsilon)$-breaks the DL problem.

### 3.3 Strong Diffie-Hellman (SDH) Assumption:

Let $(\mathcal{G}_1, \mathcal{G}_2)$ be bilinear groups. The $q$-SDH problem in $(\mathcal{G}_1, \mathcal{G}_2)$ is defined as follows: given the $(q+2)$-tuple $(g_1, g_2, g_2^x, \ldots, g_2^{x^q})$ as input, output pair $(g_1^{\frac{1}{x+c}}, c)$ where $c \in \mathcal{Z}_p{}^*$. Algorithm $\mathcal{A}$ has advantage, $\mathsf{Adv}_{SDH}(q)$, in solving $q$-SDH in $(g_1^{\frac{1}{x+c}}, c)$ if

$\mathsf{Adv}_{SDH}(q) \leftarrow \Pr[\mathcal{A}(g_1, g_2, g_2^x, \ldots, g_2^{x^q}) = (g_1^{\frac{1}{x+c}}, c)]$
Adversary $\mathcal{A}$ $(t, \epsilon)$-breaks the $q$-SDH problem if $\mathcal{A}$ runs in time at most $t$ and $\mathsf{Adv}_{SDH}(q)$ is at least $\epsilon$. The $(q, t, \epsilon)$-SDH assumption holds if no adversary $\mathcal{A}$ $(t, \epsilon)$-breaks the $q$-SDH problem.

### 3.4 External Diffie-Hellman Assumption (XDH):

Suppose a bilinear mapping $e$:$\mathcal{G}_1 \times \mathcal{G}_2 \rightarrow \mathcal{G}$. The XDH assmption states that the Decisional Diffie-Hellman (DDH) problem is hard in $\mathcal{G}_1$. This implies that there does not exist an efficiently computable isomorphism $\psi' : \mathcal{G}_1 \rightarrow \mathcal{G}_2$.
$\mathsf{Adv}_{XDH} := \Pr[A(g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_2, g \in \mathcal{G}) = ({g_1}^u : g = e(g_1, g_2)^u]$
Adversary $\mathcal{A}(t, \epsilon)$-breaks XDH problem if $\mathcal{A}$ runs in time at most $t$ and $\mathsf{Adv}_{XDH}$ is at least $\epsilon$. The $(t, \epsilon)$-XDH assumption holds if no adversary $\mathcal{A}$ $(t, \epsilon)$-breaks the XDH problem.

## 4 Proposed E-cash System

### 4.1 Key Generation

$H(x)$ is a collision-resistant hash function.

**Bank:** Upon input of security parameter, $\mathcal{B}$ randomly generates
$\{g, f, h, v_b, w_b\} \in \mathbb{G}_2$ and sets $\tilde{g} \leftarrow \psi(g)$, $\tilde{f} \leftarrow \psi(f)$, $\tilde{h} \leftarrow \psi(h)$, $\tilde{v_b} \leftarrow \psi(v_b)$, $\tilde{w_b} \leftarrow \psi(w_b)$. $\mathcal{B}$ randomly selects $b \in \mathbb{Z}_p^*$ and computes $x_b \leftarrow g^b, y_b \leftarrow f^b, z_b \leftarrow h^b$. $\mathcal{B}$'s public key $\mathsf{p_{kB}}$ and secrets key $\mathsf{s_{kB}}$ are:
$\mathsf{p_{kB}} = \{g, f, h, v_b, w_b, x_b, y_b, z_b\}$, $\mathsf{s_{kB}} = \{b\}$.

**User:** $\mathcal{U}$ randomly selects $\{v_u, w_u\} \in \mathbb{G}_2, u \in \mathbb{Z}_p^*, \mu \in \mathbb{Z}_p^*$ and computes $x_u \leftarrow h^u$, $\tilde{v}_u \leftarrow v^u$ and $\tilde{w}_u \leftarrow w^u, \lambda = g^\mu$. $\mathcal{U}$'s public key $\mathsf{p_{kU}}$ and secret key $\mathsf{s_{kU}}$ are:
$\mathsf{p_{kU}} = \{g, f, h, v_u, w_u, x_u, e(\tilde{g}, g)^u, \lambda\}$, $\mathsf{s_{kU}} = \{u, g^u, \mu\}$.

### 4.2 Withdraw

1. $\mathcal{U}$ identifies himself to bank $\mathcal{B}$ by proving knowledge of $u.PK[u; x_u = h^u]$

2. $\mathcal{U}$ randomly selects $v, s', t \in \mathbb{Z}_p^*$. $\mathcal{U}$ sends $A' = \tilde{f}^u \tilde{g}^t \tilde{h}^{s'}$ to $\mathcal{B}$. $\mathcal{B}$ randomly selects $r' \in Z_p^*$, and sends it to $\mathcal{U}$. $\mathcal{U}$ sets $s = r' + s'$. $\mathcal{U}$ and $\mathcal{B}$ locally compute $A = \tilde{f}^u \tilde{g}^t \tilde{h}^s = A' \tilde{h}^{r'}$.

3. $\mathcal{U}$ and $\mathcal{B}$ execute the verifiable encryption protocol $k$ times. $\mathcal{U}$ randomly selects $\pi_i, \rho_i \in Z_p^*$. $\mathcal{U}$ computes signature $\{\sigma_u, \pi_i, \rho_i\}$ for $E(s)_i := g^{tk_{i\bar{a}}} || e(\tilde{g}, g)^{tk_{i\bar{a}}} s$.

$$\sigma_u := (\tilde{g}^{E(s)_i} v_u {w_u}^{\pi_i})^{\frac{1}{\rho_i + u}}$$

$\mathcal{B}$ verifies signature $\pi_i, \rho_i$ by

$$e(\sigma_u, x_u g^{\rho_u}) = e(\tilde{g}, g^{E(s)_i} v_u {w_u}^{\pi_i})$$

$\mathcal{B}$ accepts

$$Q = (Q_1, \ldots, Q_k) \ .$$
$$\Big(Q_i = (E(s)_i, \pi_i, \rho_i)\Big)$$

4. $\mathcal{B}$ randomly selects $r_1, r_2 \in Z_p^*$. $\mathcal{B}$ computes $\sigma_{\mathcal{B}} = (A\tilde{v_b}\tilde{w_b}^{r_1})^{\frac{1}{b+r_2}}$, and sends $\sigma := \{\sigma_{\mathcal{B}}, r_1, r_2\}$ to $\mathcal{U}$. $\mathcal{B}$ records the entry $(\mathsf{pkU}, Q, e(\tilde{g}, g)^u)$ in his database $\mathcal{D}$. $\mathcal{U}$ verifies signature $\sigma$ by

$$e(\sigma_{\mathcal{B}}, x_b y_b z_b (fgh)^{r_2}) = e(\tilde{f}\tilde{g}\tilde{h}, f^u g^t h^s v_b {w_b}^{r_1}) \ .$$

5. $\mathcal{U}$ saves the wallet $\mathcal{W} = (s, t, \sigma, J)$ , where $J$ is an $l$-bit counter initially set to zero.

### 4.3 Spend

1. $\mathcal{U}$ receives spending data $I$ including merchant information. $\mathcal{U}$ computes $R = H(I)$.

2. $\mathcal{U}$ sends $S = g^{\frac{1}{s+J}}, T = g^{u+\frac{R}{t+J}}$ to $\mathcal{M}$.

3. $\mathcal{U}$ chooses $R_1, \ldots, R_{13} \in Z_p^*$ randomly. $\mathcal{U}$ computes

$$\begin{aligned}
{\sigma_B}' &= {\sigma_B}^{\eta} \\
\alpha &= \{x_b y_b (fgh)^{r_2}\}^{\frac{\theta}{\eta}} \\
\beta &= \{f^u g^t h^s v_b {w_b}^{r_1}\}^{\theta} \\
X_u &= f^{R_1} z^{R_2} \\
X_s &= g^{R_3} z^{R_4} \\
X_t &= h^{R_5} z^{R_6} \\
X_J &= g^{R_7} z^{R_8} \\
X_\alpha &= x_b {y_b}^{R_9} (fgh)^{R_{10}} \\
X_{\beta_1} &= (fgh)^{R_{11}} \\
X_{\beta_2} &= g^{-R_5 R_{11}} f^{-R_3 R_{11}} h^{-R_1 R_{11}} v_b^{R_{12}} w_b^{R_{13}} \\
X_{\beta_3} &= g^{R_5} f^{R_3} h^{R_1} \\
X_S &= S^{R_3+R_7} \\
X_{T_1} &= T^{R_3+R_7} \\
X_{T_2} &= g^{R_5} \\
X_{T_3} &= g^{R_7+R_3} \\
X_{T_4} &= g^{R_5(R_3+R_7)}
\end{aligned}$$

$$\begin{aligned}
Y_u &= f^u z^{R_u} \\
Y_s &= g^s z^{R_s} \\
Y_t &= h^t z^{R_t} \\
Y_J &= g^J z^{R_J} \\
\gamma &= H(I||X_s||X_J||X_t||X_u||X_\alpha||X_\beta \\
&\quad ||X_S||Y_s||Y_J||Y_t||Y_u) \\
C_u &= R_1 + \gamma u \bmod p \\
\tilde{C}_u &= R_2 + \gamma R_u \bmod p \\
C_t &= R_3 + \gamma t \bmod p \\
\tilde{C}_t &= R_4 + \gamma R_t \bmod p \\
C_s &= R_5 + \gamma s \bmod p \\
\tilde{C}_s &= R_6 + \gamma R_s \bmod p \\
C_J &= R_7 + \gamma J \bmod p \\
\tilde{C}_J &= R_8 + \gamma R_J \bmod p \\
C_\eta &= R_9 + \gamma \frac{\theta}{\eta} \bmod p \\
\tilde{C}_\eta &= R_{10} + \gamma r_2 \frac{\theta}{\eta} \bmod p \\
C_{\theta_1} &= R_{11} + \gamma\theta \bmod p \\
C_{\theta_2} &= R_{12} + \gamma^2\theta \bmod p \\
C_{\theta_3} &= R_{13} + \gamma^2 r_1 \theta \bmod p
\end{aligned}$$

$\mathcal{U}$ sends zero knowledge proof of knowledge $\Phi$:
$({\sigma_B}', \alpha, \beta, X_s, X_t, X_u, X_J, X_\beta, X_S, Y_s, Y_J, Y_t, Y_u, \gamma, C_s, \tilde{C}_s, C_t, \tilde{C}_t, C_u, \tilde{C}_u, C_J, \tilde{C}_J, C_\theta, \tilde{C}_\theta)$ to $\mathcal{M}$
.

$$\begin{aligned}
PK[(J, R_J') : \mathbf{Y_J} = \mathbf{g}^J \mathbf{h}^{R_{J'}} \bmod \mathbf{n} \\
\wedge\ Y_J = g^J h^{R_J} \wedge\ 0 \le J < 2^l] \ \ [1] \\
PK[s, R_s; Y_s = h^s z^{R_s}] \\
PK[t, R_t; Y_t = f^t z^{R_t}] \\
PK[u, R_u; Y_u = g^u z^{R_u}] \\
PK[J, R_J; Y_J = g^J z^{R_J}] \\
PK[R_9, R_{10}; X_\alpha = x_b {y_b}^{R_9} (gfh)^{R_{10}}] \\
PK[R_2, R_4, R_6, R_{11}, R_{12}, R_{13}; X_{\beta_1} = g^{R_{11}} \\
\wedge\ X_{\beta_2} = g^{(-R_6 R_{11} + R_4 R_{11} + R_2 R_{11}) v_b^{R_{12}} w_b^{R_{13}}} \\
\wedge\ X_{\beta_3} = g^{R_2+R_4+R_6}] \\
PK[J, s; S = g^{\frac{1}{s+J}}]
\end{aligned}$$

$$PK[u, t, J; T = g^{u+\frac{R}{t+J}}]$$

4. $\mathcal{M}$ verifies $\Phi$. $\mathcal{M}$ accepts the coin $\{S, T, \Phi, R, I\}$.

5. If $J > 2^l - 1$,$\mathcal{U}$ sets $J = J + 1$.

### 4.4 Deposit

$\mathcal{M}$ sends the coin $\{S, T, \Phi, R, I\}$ to $\mathcal{B}$. $\mathcal{B}$ verifies $\Phi$, and accepts the coin if the $(S, R)$ pair hasn't been spent.

### 4.5 Identify

From the two coins that have the same $S$ and different $R$, $\mathcal{B}$ computes $\mathsf{s_{kU}}$.
$\left(\frac{T_2 {R_1}}{T_1 {R_2}}\right)^{(R_1 - R_2)^{-1}} = g^u$.

### 4.6 Trace

$\mathcal{B}$ finds $e(\tilde{g}, g)^u = e(\tilde{g}^u, g)$. $\mathcal{B}$ searches for $e(\tilde{g}, g)^u$ in $\mathcal{D}$, $\mathcal{B}$ discovers double-spender's $\mathsf{p_{kU}}$. $\mathcal{B}$ recovers double spent coin $s, J_j, S_j = \tilde{g}^{\frac{1}{J_j + s}}$ from $\mathcal{D}$ . $\mathcal{B}$ outputs $\Pi := (s, J_j, g^u, \mathsf{p_{kU}}, Q_i)$.

### 4.7 Verify Ownership

Bank opens $\{S, J_j, s, g^u, E(s)_i, \sigma_u, \pi_i, \rho_i, \tilde{g}^{k_{\tilde{a}}}\}$ where $\{x_u, v_u, w_u, e(\tilde{g}, g)^u\}$ is user's public key data. Thus anyone can check that the user with $\mathsf{p_{kU}}$ is the owner of the coin with serial number $s$ by $S = g^{\frac{1}{J_j + s}}$, $e(\tilde{g}, g)^u = e(\tilde{g}, g^u)$, $E(s)_i = (\tilde{g}^{k_{i\tilde{a}}} || e(\tilde{g}, g)^u s)$, $e(\sigma_u, x_u g^{\rho_u}) = e(\tilde{g}, g^{E(s)_i} v_u {w_u}^{\pi_i})$.

## 5 Proof of Security

### 5.1 Unforgeability

Assume Adversary $\mathcal{A}$ is a $t$-time adversary whose $\mathsf{Adv}_{\mathcal{A}}^{unforge}$ is at least $\epsilon$. We will construct algorithm $\mathcal{D}$ that breaks the unforgeability of the underlying signature scheme running in time at most $t'$ with probability $\epsilon'$.

1. (Input:)
   $\mathcal{D}$'s input $\{g_1, g_2, w_2, u_2, v_2\}$ is the public key of the signature. $\mathcal{D}$' selects $g, f, h$ such that $gfh = g_1$ and $\mathcal{D}$' sets $\{w_b = u_2, v_b = v_2, x_b y_b z_b = w_2(\text{random } x_b, y_b, z_b)\}$ as the public key of signature.

   $\mathcal{D}$ sets this input data as a bank's public key.

2. (Withdraw)
   $\mathcal{D}$ must simulate a e-cash system when using $\mathcal{A}$. However, because $\mathcal{D}$ cannot issue signatures, $\mathcal{D}$ asks a signing oracle to get bank's signature. While $\mathcal{A}$ executes the withdraw protocol $K$ times, $\mathcal{A}$ demands $K$ signatures for $\mathcal{D}$. Whenever $\mathcal{A}$ demands a signature, $\mathcal{D}$ asks the signing oracle. Let $\sigma_1, \sigma_2, \ldots, \sigma_K$ be the $K$ signatures $\mathcal{D}$ obtains from the signing oracle.

3. (Spend)
   We show that if $\mathcal{A}$ completes one spend protocol, $\mathcal{D}$ extracts one signature.

   $\mathcal{A}$ sends spending data information $I$ to the hash oracle to obtain a hash value. $\mathcal{D}$ simulates random oracle $H$. $\mathcal{D}$ sends random number $\gamma$ to $\mathcal{A}$.

   Once $\mathcal{D}$ gets a valid coin data, $\mathcal{D}$ resets $\mathcal{A}$ and changes its response to the hash oracle to $\gamma'$. Let $\{\gamma, C_\eta, C_{\theta 1}, \ldots\}$ be the spend coin data when $\mathcal{A}$ completes the spend protocol for the first time. Those values are defined in Chapter 4.3. Let $\{\gamma', {C_\eta}', {\tilde{C_{\theta 1}}}', \ldots\}$ be the spend coin data when $\mathcal{A}$ completes the spend protocol the second time.

   $\mathcal{A}$ randomizes the signature with $\eta, \theta$ to use the signature in the spend protocol. Let $\sigma_B$ be the original signature from the signing oracle and ${\sigma_B}'$ be a randomized signature by $\mathcal{A}$. $\mathcal{D}$ computes

$$C_\eta - {C_\eta}' = \frac{\theta}{\eta}(\gamma - \gamma')$$
$$C_{\theta_1} - {C_{\theta_1}}' = \theta(\gamma - \gamma'),$$

   obtains $\eta$, and calculates $\sigma_B$ because $\gamma - \gamma' \neq 0$. Consequently, $\sigma_B = ({\sigma_B}')^{\frac{1}{\eta}}$.

4. (Output)
   $\mathcal{A}$ completes the deposit protcol $2^L K + 1$ times using $2^L K$ coins, thus $\mathcal{D}$ extracts $2^L K + 1$ signatures.

   In contrast, $\mathcal{D}$ only received $K$ signatures from the signing oracle. $\mathcal{A}$ can use signature $2^L$ times per serial number $s$. $\mathcal{A}$ must use serial number $s$, which is unsigned from the bank, in at least one execution of the spend protocol. Thus $\mathcal{D}$ gets at least one signature that the singning oracle did not issue.

$\mathcal{D}$ selects random $i$ and $\mathcal{D}$ extracts only the $i$-th signature. We pay attention to the one spend protocol, $\mathcal{A}$ must use $s$ that is unsigned from the bank with probability greater than $\frac{1}{2^L K+1}$. From the assumption, $\mathcal{A}$ will complete the deposit protocol with probability $\epsilon$ at this time. However, $\mathcal{A}$ is not always completes the deposit protocol, probability of $\epsilon$ again when $\mathcal{D}$ resets $\mathcal{A}$. Thus we consider the probability that $\mathcal{A}$ in completing the deposit protocol with different $\gamma$.

**Heavy Low Lenma**

Assume $\mathcal{A}$ wins the unforgeability game with probability $\epsilon$. Let $\gamma$ be the hash value in the spend protocol. If $\mathcal{D}$ resets $\mathcal{A}$ and gives another hash value $\gamma'$, $\mathcal{A}$ in completing the same spend protocol with probability of at least $\frac{\epsilon^2}{4}$.

**Proof**

$\mathcal{A}$ in completing the deposit protocol with probability $\epsilon$ with random $\gamma$ and other random parameters. We show a random $\gamma$ and other random parameters, which $\mathcal{A}$ suceeds the deposit protocol probability $\frac{\epsilon}{2}$ as a defferent $\gamma$ and the same other parameters, are selected probability more than $\frac{1}{2}$. We define the heavy row that $\mathcal{A}$ in completing the deposit protocol with probability of more than $\frac{\epsilon}{2}$ using other fixed parameters. No heavy rows exist with probability of more than $\frac{\epsilon}{2}$. Therefore, heavy rows exist with probability of more than $\frac{\epsilon}{2}$. In other words, the random $\gamma$ and other random parameters that fallow $\mathcal{A}$ to complete in the deposit protocol belong to a heavy row with probability of more than $\frac{1}{2}$. Namely $\mathcal{A}$ suceeds the deposit protocol probability $\epsilon$, it belongs to heavy row probability more than $\frac{1}{2}$ and $\mathcal{A}$ suceeds the deposit protocol using differnt $\gamma$ probability more than $\frac{\epsilon}{2}$.

$t' = 2t$, $\epsilon' = \frac{\epsilon^2}{4(2^L K+1)}$. $2^L$ is coin size and a constant value. $K$ is a polynomial value.

### 5.2 Identification of double-spenders

Assume Adversary $\mathcal{A}$ is a $t$-time adversary whose $\mathsf{Adv}_{\mathcal{A}}^{identify}$ is at least $\epsilon$. We will construct algorithm $\mathcal{D}$ that breaks the underlying signature scheme running in time at most $t'$ with probability $\epsilon'$.

1. ($\mathcal{A}$ breaks unforgeability)
   From difinition of identification, $\mathcal{A}$ withdraws a $2^l$ coin $K$ times and $\mathcal{A}$ spends $2^l K + 1$ coins.

   If two spending datas $\{T = g^{u+\frac{R}{t+J}}, R\}, \{T' = g^{u'+\frac{R'}{t'+J'}}, R'\}$ exist $u = u' \wedge t = t' \wedge s = s'$, $\mathcal{A}$ can compute user's public key $g^u$.

   $$\left(\frac{T'^R}{T^{R'}}\right)^{\frac{1}{R-R'}} = g^u.$$

   Thus two spending datas $\{T = g^{u+\frac{R}{t+J}}, R\}, \{T' = g^{u'+\frac{R'}{t'+J'}}, R'\}$ exist $\neg(u = u' \wedge t = t' \wedge s = s')$. It is show that $\mathcal{A}$ breaks unforgeability.

2. ($\mathcal{A}$ cheats verifiable encryption protocol)
   No more than $\frac{1}{2^k}$ probability $\mathcal{A}$ cheets the verifiable encryption protocol.

Because $\mathcal{A}$ can break unforgeability with probability $\epsilon - \frac{1}{2^k}$,

$$t' = 2t, \epsilon' = \frac{(\epsilon - \frac{1}{2^k})^2}{4(2^L K + 1)}.$$

$2^L$ is coin size and a constant value. $K$ is a polynomial value.

### 5.3 Tracing double-spenders

Assume Adversary $\mathcal{A}$ is a $t$-time adversary whose $\mathsf{Adv}_{\mathcal{A}}^{trace}$ is at least $\epsilon$. We will then construct algorithm $\mathcal{D}$ that breaks the DL assumption with $(t', \epsilon')$.

An informal outline of our proof is as follows: If $\mathcal{A}$ completes the verifiable encryption, $\mathcal{D}$ gets a DL-relation by resetting $\mathcal{A}$. If $\mathcal{A}$ completes the spend protocol, $\mathcal{D}$ gets another DL-relation by resetting $\mathcal{A}$. We show that if $\mathcal{A}$ breaks the trace security, $\mathcal{D}$ can obtain two different DL-relations.

Algorithm $\mathcal{D}$ is constructed as follows:

1. (Input:)
   $\mathcal{D}$'s input $(f, g, h)$ is 3 elements of DL assumption.

2. (Bank's key generation:)
   $\mathcal{D}$ sets $\tilde{f} = \psi(f), \tilde{g} = \psi(g), \tilde{h} = \psi(h)$. $\mathcal{D}$ randomly selects generators $v_b$ and $w_b$ and sets $\tilde{v_b} = \psi(v_b), \tilde{w_b} = \psi(w_b)$. $\mathcal{D}$ randomly selects $b \in \mathcal{Z}_p^*$ and computes $x_b = g^b, y_b = f^b, z_b = h^b$.

3. (Share $A$ in simulation withdraw protcol)
   $\mathcal{D}$ receives $A'$ from $\mathcal{A}$. $\mathcal{D}$ randomly selects $r' \in \mathcal{Z}_p^*$ and sends it to $\mathcal{A}$. Let $A = A'\tilde{h}^{r'}$.

4. (Verifiable encryption)
   First, $\mathcal{A}$ and $\mathcal{D}$ execute verifiable encryption $m$ times. At the m-th verifiable encryption cycle, $\mathcal{D}$ receives $X_m, e_{m1}, e_{m2}$, $\mathcal{D}$ sends bit $b_m = \{1 \ or \ 2\}$ to $\mathcal{A}$. $\mathcal{D}$ receives $m$-th verifiable encryption data: $c_{\{m,b_m,1\}}, c_{\{m,b_m,2\}}, c_{\{m,b_m,3\}}, k_m, \tau_m, s_{m1}, s_{m2}$.
   Second, $\mathcal{D}$ resets $\mathcal{A}$ and repeats verifiable encryption $m$ times. $\mathcal{A}$ is reset, so $\mathcal{A}$ returns the same $X_m, e_{m1}, e_{m2}$. $\mathcal{D}$ sends $\tilde{b}_m = \{1 \ or \ 2\}$ that $\tilde{b}_m = 3 - b_m$. Finally, $\mathcal{D}$ gets $c_{\{m,1,1\}}, c_{\{m,1,2\}}, c_{\{m,1,3\}}, c_{\{m,2,1\}}, c_{\{m,2,2\}}$ and $c_{\{m,2,3\}}$.
   $\mathcal{D}$ computes $\bar{u}_m = c_{\{m,2,1\}} - c_{\{m,1,1\}}$, $\bar{t}_m = c_{\{m,2,2\}} - c_{\{m,1,2\}}$ and $\bar{s}_m = c_{\{m,2,3\}} - c_{\{m,1,3\}}$.
   If verifiable encryption is successful,
   $$f^{\bar{u}_m} g^{\bar{t}_m} h^{\bar{s}_m} = A$$
   with probability $(1 - \frac{1}{2^m})$ because $\exists m, f^{c_{\{m,1,1\}}} g^{c_{\{m,1,2\}}} h^{c_{\{m,1,3\}}} = XA \wedge \ f^{c_{\{m,2,1\}}} g^{c_{\{m,2,2\}}} h^{c_{\{m,2,3\}}} = XA^2$. Let $\bar{u}, \bar{t}$ and $\bar{s}$ denote $\exists m$ above $\bar{u}_m, \bar{t}_m$ and $\bar{s}_m$.

5. (Spend)
   First, $\mathcal{A}$ sends spending data information $I$ to the hash oracle. $\mathcal{D}$ simulates random oracle $H$. $\mathcal{D}$ sends ramdom number $\gamma$ as $H(I||\ldots)$ to $\mathcal{A}$.
   $\mathcal{D}$ receives $\{\gamma, C_u, C_t, C_s, \ldots\}$.
   Second, $\mathcal{D}$ resets $\mathcal{A}$ and repeats the spend protcol again. $\mathcal{A}$ is reset, so $\mathcal{A}$ returns the same value $\{I, \ldots\}$. $\mathcal{D}$ sends another random number $\gamma'$ as $H(I||\ldots)$ to $\mathcal{A}$.
   $\mathcal{D}$ receives $\{\gamma', C_u', C_t', C_s', \ldots\}$.
   $\mathcal{D}$ computes
   $$u = \frac{C_u - C_u'}{\gamma - \gamma'}$$
   $$t = \frac{C_t - C_t'}{\gamma - \gamma'}$$
   $$s = \frac{C_s - C_s'}{\gamma - \gamma'} \ .$$
   Since $\mathcal{A}$ succeeds in spending, $(u, t, s)$ satisfies $A = f^u g^t h^s$ with probability $\epsilon$.

6. (Trace)
   If $(\bar{u}, \bar{t}, \bar{s}) = (u, t, s)$, $\mathcal{D}$ gets correct value $g^t$ in the identify phase. Thus the bank can find the withdraw data to search the database. $s$ is also correct, so $Q$ is an evidence of double-spending.

7. (Output)
   $\mathcal{D}$ finds discrete logarithm relation, $f^u g^t h^s = f^{u'} g^{t'} h^{s'}$.
   $$h = f^{\frac{\bar{u}-u}{s-\bar{s}}} g^{\frac{\bar{t}-t}{s-\bar{s}}}$$

$\mathcal{D}$ outputs

$$\begin{aligned} x &= \frac{\bar{u}-u}{s-\bar{s}} \\ y &= 1 \\ z &= \frac{\bar{t}-t}{s-\bar{s}} \ . \end{aligned}$$

$\mathcal{D}$ gets the DL relation $t' = 3t + cmT$, $\epsilon' = \epsilon(1 - \frac{1}{2^m})$. $c$ is a constant value, where T is the time to calculate an exponential.

### 5.4 Anonymity of users

Let $A_1 = d^{y_1} f^{u_1} g^{t_1} t^{s_1}$ be a share between $\mathcal{A}$ and $\mathcal{U}_1$ and $A_2 = d^{y_2} f^{u_2} g^{t_2} t^{s_2}$ be a share between $\mathcal{A}$ and $\mathcal{U}_2$.

$d^{y_1}$ and $d^{y_2}$ are randomize numbers, $\forall y_1, u_1, t_1, s_1, u_2, t_2, s_2, \exists y_2, A_1 = A_2$. In other words, no one distinguish $A_1$ of random select $y_1$ from $A_2$ of random select $y_2$. Thus this protocol is information-theoretically secure.

### 5.5 Exculpability

Assume Adversary $\mathcal{A}$ is a $t$-time adversary whose $\mathsf{Adv}_{\mathcal{A}}^{exculpability}$ is at least $\epsilon$. We will then construct algorithm $\mathcal{D}$ that breaks the XDH assumption with $(t', \epsilon')$.

$\mathcal{A}$ has to output $g^u$ from $e(g, \tilde{g})^u$. If such $\mathcal{A}$ exists, it breaks the XDH assumption. Thus $\mathcal{D}$ gives $\mathcal{A}$ $e(g, \tilde{g})^u$ and outputs $g^u$.

$t' = t, \epsilon' = \epsilon$

## 6 Conclution

This paper proposes two off-line anonymous e-cash schemes. One is an efficient off-line e-cash schemes that is secure under the strong RSA assumption, the strong Diffie-Hellman (SDH) assumption and External Diffie-Hellman (XDH) assumption with the random oracle model. This scheme is more efficient than the "Compact E-Cash" proposed by Jan Camenisch, Susan Hohenberger and Anna Lysyanskaya, and is removed non-standard assumption such as Sum-Free Decisional Diffie-Hellman assumption using "Efficient Blind and Partially Blind Signature Without Random Oracles" proposed by Tatsuaki Okamoto.

## References


[1] Fabrice Boudot, "Efficient Proofs that a Committed Number Lies in an Interval", Eurocrypt 2000, 2000.

[2] David Chaum, "Blind signature for untraceable payments", Crypto '82, 1982.

[3] David Chaum, "Blind signature systems", Crypto '83, 1983.

[4] David Chaum, Amos Fiat, Moni Naor "Untraceable electronic cash", Crypto '88, 1988.

[5] Matthew Franklin, Moti Yung, "Towards provably secure efficient electronic cash", Asiacrypt '96, 1996.

[6] David Chaum, Torben Pryds Pedersen "Transferred cash grows in size", Eurocrypt '92, 1992.

[7] Stefan Brands, "An efficient off-line electronic cash system based on the representation problem", CS-R9323, 1993.

[8] Jan L. Camenisch, Jean-Marc Piveteau, Markus A. Stadler, "Blind signature based on the discrete logaritm problem", Eurocrypt '94, 1994.

[9] Stefan Brands, "Rapid demonstration of linear rations connected by boolean operators", Crypto '93, 1993.

[10] Markus A. Stadler, Jean-Marc Piveteau, Jan L. Camenisch, "Fair blind signature", Eurocrypt '95, 1995.

[11] Yair Frankel, Yiannis Tsiounis, Moti Yung, "Indirect discourse proofs", Asiacrypt '96, 1996.

[12] Yiannis S. Tsiounis, "Efficient Electronic Cash", PhD thesis, 1997.

[13] Mihir Bellare, Adriana Palacio, "GQ and Schnorr Identification Schemes:Proofs of Security against Impersonation under Active and Concurrent Attacks", Crypto '02, 2002.

[14] Jan Camenish, Ivan Damgard, "Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes", Asiacrypt 2000, 2000.

[15] Jan Camenish, Susan Hohenberger, Anna Lysyanskaya, "Compact E-Cash", Eurocrypt 2005, 2006.

[16] Tatsuaki Okamoto, "Efficient Blind and Partially Blind Signatures Without Random Oracle", TCC 2006, 2006.