Short E-Cash

Man Ho Au[1], Sherman S.M. Chow[2,⋆], and Willy Susilo[3]

[1] Department of Computer Science,
The University of Hong Kong, Pokfulam, Hong Kong
mhau@cs.hku.hk

[2] Department of Computer Science,
Courant Institute of Mathematical Sciences,
New York University, NY 10012, USA
schow@cs.nyu.edu

[3] Center for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong, Wollongong 2522, Australia
wsusilo@uow.edu.au

Abstract. We present a bandwidth-efficient off-line anonymous e-cash scheme with traceable coins. Once a user double-spends, his identity can be revealed and all his coins in the system can be traced, *without* resorting to TTP. For a security level comparable with 1024-bit standard RSA signature, the payment transcript size is only 512 bytes. Security of the proposed scheme is proven under the q-strong Diffie-Hellman assumption and the decisional linear assumption, in the random oracle model. The transcript size of our scheme can be further reduced to 192 bytes if external Diffie-Hellman assumption is made. Finally, we propose a variant such that there exists a TTP with the power to revoke the identity of a payee and trace all coins from the same user, which may be desirable when a malicious user is identified by some non-cryptographic means.

Keywords: E-cash, Coin-traceability, Bilinear Pairing.

1 Introduction

To conduct business transaction over the Internet, one of the ways to make payment is to use e-cash. The simplest model of an e-cash scheme involves three types of parties: *banks* B, *shops* S, and *customers* C. An e-cash scheme is a set of protocols which includes *withdrawal* (by C from B), *purchase* (by C to S) and *deposit* (by S to B). In the electronic world, all objects are represented by data; e-cash is by no means an exception. Special design can be incorporated in real cash to prevent counterfeiting, but it is easy to duplicate e-cash. Thus it is necessary to prevent a user from spending the same coin twice (*double-spending*).

Resembling real cash, it is desirable that the shop can accept a payment autonomously, without consult any other parties, possibly the bank. E-cash scheme

⋆ Corresponding author.

S. Maitra et al. (Eds.): INDOCRYPT 2005, LNCS 3797, pp. 332–346, 2005.

satisfying this property is described as an *off-line* one. The coins are most probably spent in two different shops when they are double-spent. It is kind of impossible for the shops to check for double-spending on their own. Instead, the bank checks for double-spending when the shops deposit the coins. Either the shops will get the real payment, or the bank will identify the double-spender. On the other hand, honest spenders cannot be slandered to have double spent (*exculpability*), and when the shops deposit the money from the payee, the bank should not be able to trace who the actual spender is (*anonymity)*.

Many e-cash systems allow the identification of double-spender have been proposed, but most of them rely on the existence of a trusted third party (TTP) to *revoke* the anonymity (so as to identify the double-spender) when double-spending occurs. The revocation is done probably with the help of a *database* maintained by the bank, where certain tracing information obtained during the withdrawal protocol are stored. This information is usually in an *encrypted* form that is believed to be decryptable by the TTP only.

Even though a secure e-cash system prevents the TTP from slandering an honest spender, the revocation feature gives the TTP an elusive power to revoke the anonymity of honest spender as well. To remove this high level of trust, an anonymous e-cash scheme should support owner-tracing without TTP. Identity of double spender should be revoked while the identity of honest user is always protected. To further punish the double spender, all coins spent (and possibly to be spent) by a cheating user can be linked while the withdrawals and payments of an honest user remains unlinkable. That is, certain information can be put in a blacklist so that the coin from the double-spenders can be recognized when it is spent. Moreover, such coin-tracing can only be (instead of trusted to be) performed after double-spending has occurred.

Recent proposal by Camenisch, Hohenberger and Lysyanskaya [8] supports traceability of owner and coin without a TTP. Moreover, their scheme (hereinafter referred as CHL scheme) has the distinctive feature that a user can withdraw more than one coin in a single withdrawal protocol, and these coins can be spent in an unlinkable manner. Put it in a more formal way, 2^ℓ coins can be withdrawn with the cost of $O(\ell \cdot k)$ instead of $O(2^\ell \cdot k)$, where k is a security parameter. As a result, a "compact electronic wallet" is made possible.

Our Contributions.

- We propose three short e-cash systems with different features:
  1. identification and coin-tracing of double-spender without TTP.
  2. even shorter payment transcript size.
  3. owner-tracing and coin-tracing of honest users with the help of a TTP.
- We reinvestigate the efficiency of the CHL scheme, which includes the bandwidth requirements in payment and deposit protocol, and also the bank's storage requirement. We compare it with our proposal for typical usage.

Organization. Next two sections discuss related works and technical preliminaries. We define our security model in Section 4. The constructions of the e-cash systems are presented in Section 5, accompanied by a comparison of our proposal with the CHL scheme. We conclude the paper in Section 6.

2 Related Work

To protect the benefit of the banks, e-cash should deter counterfeiting. A secure digital signature, being unforgeable, is a good candidate for implementing e-cash. The idea of blind signature was proposed in [11] to support untraceable payment system. The bank can sign on the information associated with the transaction in a blinded way without knowing the information about an individual's whereabouts and lifestyle. Beside, blind signature ensures *unlinkability*: even the bank is given the message/signature pair at later stage, it is impossible to recollect the corresponding invocation of signing protocol. However, the property that user can ask the bank to blindly sign any message is undesirable. Cut-and-choose methodology was applied in [12] such that the bank can ensure by statistical probability that the user has not presented a malformed message. But it is very inefficient by nature. Alternatively, later research work proposed using variations of blind signature scheme, such as restrictive blind signature [6] and partially blind signature [1], to prove a user has not breached security.

Group signatures, introduced by Chaum and Heyst [13], allow individual members to make signatures on behalf of the group. The identity of the actual signer is kept secret, but there is a TTP that can revoke this anonymity. Group signature also provides "another kind" of *unlinkability*, such that the signature produced by the same signer is unlinkable. These privacy-oriented properties (signer-anonymity and unlinkability) have been utilized in various e-cash proposals. The concept of "member" plays different roles in various e-cash proposal; for examples, the issuing banks [18], the payees who spend the coins [18, 19, 23, 25], and the coins themselves (referred as "group of coins" model) [10, 20].

The unlinkability of these signatures could be used maliciously, like money laundering and obtaining a ransom safely [27]. *Fair* e-cash system, suggested independently by [7] and [24], can detect the misuse by criminal when necessary. In fair blind signature [24] and group signature, a TTP can revoke the unlinkability and anonymity respectively. The existence of TTP is especially useful in designing fair e-cash systems. Examples include [25, 19, 23, 10].

For detection of double-spending, the idea of cut-and-choose can also help. However, many similar components are involved in the cash, which make the scheme inefficient. More efficient mechanism involves a single-term only, an example is the secret sharing line method in [14, 15]. The technique to realize this "single-term" property may vary in different schemes [6, 14, 15, 21].

In the "group of coins" model, double-spending detection mechanism can be achieved by compromising the unlinkability of signer-anonymous signatures. Some schemes exploited this idea implicitly. For example, the scheme in [10] incorporated a "linkability tag" to the underlying group signature scheme [2] to ensure the linkage of double-spent coins. As noted in [26], accusatory linking that outputs the identity of the double-spender is needed for offline e-cash system, or the cheater has already benefited by exchanging the double-spent coins with the goods or services before the coins are voided by the bank.

In addition to double-spending detection, it is beneficial to have the *coin-traceability*, such that all the coins withdrawn by a particular payee can be traced. Early fair e-cash systems either do not support coin tracing (e.g. [19] and [25]), rely on the online participation of a TTP (e.g. [7]), or rely on the offline presence of a TTP (e.g. [10] and [24]). Usually the TTP is overpowered. For examples, the TTP in [17] can trace the coins spent by any honest user, and the TTP in the linkable group signature extension of [26] can reveal the identity of any honest user. A new idea of coin-tracing is to do , the coin-tracing without a TTP: any party can trace the coins of the same payee once this payee double-spent [8]. The mechanism in [8] is efficient in the sense that one-by-one checking on spent coins is not necessary, in contract with the traceable signatures in [17].

Note that coin-traceability is different from double-spent coins detection. The later only applies on the coins spent by a double-spender, but the former notion has said nothing about it. For examples, the scheme in [22] and the e-cash system from the transaction escrow scheme in [16] support coin-tracing of *any* user.

3 Preliminaries

We review concepts related to bilinear pairings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic multiplicative groups of prime order p.
- g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- ψ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and $\psi(g_2) = g_1$.
- $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and a, $b \in \mathbb{Z}_p$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$.
- $\hat{e}(g_1, g_2) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be the same or different groups. We say that two groups ($\mathbb{G}_1$, $\mathbb{G}_2$) are a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, the isomorphism ψ and the bilinear mapping $\hat{e}$ are all efficiently computable.

Definition 1 (Decisional Diffie-Hellman). *The Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ is defined as follows: Given a quadruple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, decides whether $c = ab$. We say that the (t, ϵ)-DDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the DDH problem in $\mathbb{G}$.*

Definition 2 (Decisional Linear Diffie-Hellman). *The Decisional Linear Diffie-Hellman (DLDH) problem in $\mathbb{G}_1$ is defined as follows: Given a sextuple in the form of $(g_1, g_2, g_3, {g_1}^a, {g_2}^b, {g_3}^c) \in {\mathbb{G}_1}^6$, decides whether $c = a+b$. We say that the (t, ϵ)-DLDH assumption holds in $\mathbb{G}_1$ if no t-time algorithm has advantage at least ϵ in solving the DLDH problem in $\mathbb{G}_1$.*

DLDH problem is proposed and proven secure in the generic group model in [4].

Definition 3 (q-Strong Diffie-Hellman). *The q-Strong Diffie-Hellman (q-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: Given a $(q+2)$-tuple $(g_1, g_2,$*

$g_2^x, g_2^{x^2}, \cdots, g_2^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, *output a pair* (A, c) *such that* $A^{(x+c)} = g_1$ *where* $c \in \mathbb{Z}_p^*$. *We say that the* (q, t, ϵ)*-SDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if no t-time algorithm has advantage at least* ϵ *in solving the q-SDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$.

Again, q-SDH problem is proven secure in the generic group model [3].

Definition 4 (eXternal Diffie-Hellman). *The eXternal Diffie-Hellman (XDH) problem in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *is defined as solving the DDH problem in* $\mathbb{G}_1$ *given the follwing three efficient oracles*

1. *solving DDH problem in* $\mathbb{G}_2$,
2. *computing the isomorphism from* $\mathbb{G}_2$ *to* $\mathbb{G}_1$,
3. *and computing the bilinear mapping of groups* $\mathbb{G}_1 \times \mathbb{G}_2$ *to* $\mathbb{G}_T$.

We say that the (t, ϵ)*-XDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *if no t-time algorithm has advantage at least* ϵ *in solving the XDH problem in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

The above assumption implies that the isomorphism is computationally one-way, i.e. there does not efficient way to complete $\psi^{-1} : \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The discussion on the choice of elliptic curves which can make the above assumption hold can be found in [4]. In short, the bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ should be instantiated using the Weil or Tate pairing over MNT curves; but not supersingular curves.

4 Security Model of E-Cash System

4.1 Framework

An anonymous e-cash system consists of three parties: the bank, the user and the merchant, together with the following six algorithms.

– `Setup`. On input an unary string 1^λ, where λ is a security parameter, the algorithm outputs a master secret key s and a list of publicly known system's parameter param. In an anonymous e-cash, the master secret key is owned by the bank which allows it to issue electronic coins.
– `User Setup`. On input of param, randomly outputs a key pair (pk, sk).
– `Withdrawal`. The user with input (pk, sk) withdraws a electronic coin from the bank. The bank responses with input s. After executing the protocol, the user obtains the coin c while the bank retains certain information τ_w which allows it to trace the user should this user double-spends some coin. The bank maintains a database for this trace information.
– `Payment`. The user with input c spends. The merchant response with input param. After the protocol the merchant obtains a transcript including a proof of validity π of the coin c, and possibly some auxiliary information aux, and outputs 0/1, depending whether the payment is accepted.
– `Deposit`. The merchant submits (π, aux) to the bank for deposit. The bank outputs 0/1, indicating whether the deposit is accepted. It is required whenever a honest merchant obtains (π, aux) by running the `Payment` protocol with some user, there is a guarantee that this coin will be accepted by the bank. The bank adds (π, aux) to the database of spent coins.

– `Owner tracing (of double-spender)`. Whenever a user double spent, this algorithm allows the bank to identify the double spender. Formally, on input two payment protocol transcripts from the same coin c, the algorithm outputs the public key pk of the owner of coin c.
– `Coin tracing (of double-spender)`. Whenever a user double spent, this algorithm allows the bank to publish some tracing information so that all spending of the same user are identified. Formally, on input two payment transcripts from the same coin c of the same owner pk, outputs a set of information $\{tag\}$ so that anyone with $\{tag\}$ can identify all coins from user (with public key pk) during the payment protocol.

We stress that the difference between fair e-cash and anonymous e-cash is that, in the former case, there exists a TTP which can revoke the anonymity of the coin and hence the privacy of the user. Whether this is desirable or not depends the application as the unconditional anonymity can be misused for illegal purposes such as money laundering or perfect blackmailing.

4.2 Security Definition

Security properties are described informally at first.

– *Correctness.* If an honest user runs `Withdrawal` with an honest bank and runs `Payment` with an honest merchant, the merchant accepts the coin. The merchant later runs `Deposit` with the bank, which will accept the coin.
– *Balance.* It means that no collusion of users and merchants can ever spend more coins than they withdrew. This is the most important property from the bank's point of view. We require that the adversary, after running q_u `Withdrawal` protocol with the bank, cannot run the `Deposit` protocol successfully with the bank for $q_u + 1$ times. A deposit query is successful if either (1) the bank accepts the coin or (2) the bank identifies the coin is being double-spent but is unable to identify the double spender[1].
– *Identification of double-spenders.* It is required that suppose a user double spent, he must be identified.
– *Tracing of double-spenders* It is required that if a user double spent, all of his other coins can be traced regardless of it is spent honestly or not.
– *Anonymity of users* Even when the bank cooperates with any coalition of users and merchants, cannot learn anything about an honest user's spending.
– *Exculpability* An honest user cannot be accused of having double spent.

We focus on *Balance* and *Anonymity*, the two most important requirements of e-cash system. The capabilities of an adversary $\mathcal{A}$ is modeled as oracles that answers the following queries from the adversary.

[1] It is assumed that the bank holds the responsibility to charge the double-spender, so the merchant is credited even if the coin has been identified to have been double-spent. An honest merchant may not be able to detect double-spending in an off-line anonymous e-cash system. Thus, condition (2) is included in the definition of balance.

– Withdrawal queries: $\mathcal{A}$ engages in the withdrawal protocol as user and obtains a valid coin.
– Payment queries: $\mathcal{A}$ engages in the deposit protocol as a merchant.
– Hash queries: $\mathcal{A}$ can ask for the values of the hash functions for any input.

We require that the answers from the oracles are indistinguishable from the view as perceived by an adversary in real world attack.

Balance. The following game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally defines the *Balance* property.

Definition 5 (Game Balance).

– (Initialization Phase.) *The challenger* $\mathcal{C}$ *takes a large security parameter* λ *and runs the* `Setup` *to generate a list of system's parameters* param *and also a master secret key* s*.* $\mathcal{C}$ *keeps* s *to itself and sends* param *to* $\mathcal{A}$*.*
– (Probing Phase.) *The adversary* $\mathcal{A}$ *can perform a polynomially bounded number of queries to the oracles in an adaptive manner.*

$\mathcal{A}$ wins the above game if the number of successful withdrawal queries plus payment queries is less than that of successful deposit queries. A deposit query is successful if either the bank accepts the deposit request or the bank identifies double-spent but is unable to identify the double spender. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Definition 6 (Balance). *An e-cash game is said to have the Balance property if no adversary has a non-negligible advantage in the game Balance.*

Anonymity. The following game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally defines the *anonymity of e-cash system.*

Definition 7 (Game Anonymity).

– (Initialization Phase.) *The challenger* $\mathcal{C}$ *takes a sufficiently large security parameter* λ *and runs the* `Setup` *to generate a list of system's parameters* param *and also the bank's secret key* s*.* $\mathcal{C}$ *gives* s *and* param *to* $\mathcal{A}$*.*
– (Challenge Phase.) *The adversary* $\mathcal{A}$ *runs the* `withdrawal` *protocol with* $\mathcal{C}$*. Then* $\mathcal{C}$ *runs* `deposit` *protocol with* $\mathcal{A}$ *acting as the bank.*
– (End Game Phase.) *The adversary* $\mathcal{A}$ *decides if the underlying coin of the two runs are the same.*

$\mathcal{A}$ wins the above game if it guesses correctly. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins minus $\frac{1}{2}$.

Definition 8 (Anonymity). *A e-cash scheme is anonymous if no adversary has a non-negligible advantage in the game Anonymity.*

5 Our Proposed E-Cash Systems

Global parameters for both systems. Let λ be the security parameter. $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair with computable isomorphism ψ as discussed. $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p of λ bits. $H : \{0,1\}^* \to \mathbb{Z}_p$ is a cryptographic hash function. We assume there exists a group $\mathbb{G}_p$ of order p where DDH is hard.

5.1 Short E-Cash

We present a short e-cash system that supports identification and coin tracing of double-spender without the need of a TTP. We require the user to verifiably encrypt the tracing information under his own public key during the withdrawal protocol, assuming PKI is deployed. By using technique in [6], secret key of the double-spender can be extracted, and thus tracing information can be decrypted.

- `Bank Setup`. The bank's public key is $bpk = (g_1, g_2, w, h_1, h_2, h_3, u, v, h, h_t)$ and the private key $bsk = \gamma$, generated as follows.
  1. Randomly generates generator $g_2 \in \mathbb{G}_2$ and sets $g_1 = \psi(g_2)$.
  2. Randomly selects $\gamma \in_R \mathbb{Z}_p^*$ and sets $w = {g_2}^{\gamma}$.
  3. Randomly selects generators $h_1,\ h_2, h_3, u, v \in_R \mathbb{G}_1$.
  4. Randomly selects generators h, h_t of $\mathbb{G}_p$.
- `User Setup`. Each user is equipped a discrete logarithm type of public and private key pair $(h^s, s) \in \mathbb{G}_p \times \mathbb{Z}_p^*$.
- `Withdrawal Protocol`. When a user with public key $y = h^s \in \mathbb{G}_p$ wants to withdraw money from the bank, the following protocol is executed.
  1. User selects $\bar{a}, \bar{b}$ such that $\bar{a}\bar{b} = s$, computes $\bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \in \mathbb{G}_1$, and a signature based on proofs of knowledge (SPK) Π_1 that $\bar{C}$ is correctly formed. User sends $(\bar{C}, \Pi_1)$ to the bank.
  2. The bank verifies that Π_1 is valid, randomly generates r and sends it back to the user.
  3. User then computes $a = \bar{a}r$, $b = \bar{b}r^{-1}$, $C = {h_1}^a {h_2}^b$, and computes the encryption R of ${h_t}^a$ and ${h_t}^b$ under its public key h^s for coin tracing. User sends to the bank C, R and SPK Π_2 that they are correctly formed.
  4. The bank verifies that Π_2 is valid, randomly selects $x \in_R \mathbb{Z}_p^*$ and computes $A = (g_1 C)^{\frac{1}{\gamma + x}} \in \mathbb{G}_1$. The bank sends (A, x) back to the user.
  5. The bank keeps (A, x, C, Π_2) in record and debits the user accordingly.
  6. User checks if the coin (A, x, a, b) satisfies $\hat{e}(A, w g_2^x) = \hat{e}(g_1 h_1^a h_2^b, g_2)$.

  The encryption and the proof Π_1 and Π_2 are shown in the appendix.
- `Payment Protocol`. Suppose the user spends the coin (A, x, a, b) to a merchant with the identity $I \in \{0,1\}^*$, the following protocol is executed.
  1. User randomly generates $\alpha, \beta \in_R \mathbb{Z}_p^*$, computes the auxiliary commitment $A_1 = u^{\alpha}$, $A_2 = v^{\beta}$, $A_3 = A{h_3}^{\alpha+\beta}$, and tracing information $B_1 = {h_t}^a$ and $B_2 = {h_t}^b$. $\{A_1, A_2, A_3\} \in \mathbb{G}_1$ and $\{B_1, B_2\} \in \mathbb{G}_p$.
  2. User computes two helper values $\delta_\alpha = x\alpha$ and $\delta_\beta = x\beta$.

3. User undertakes a proof of knowledge of values $(\alpha, \beta, x, a, b, \delta_\alpha, \delta_\beta)$ satisfying the relations: $A_1 = u^\alpha$, $A_2 = v^\beta$, $A_1^x = u^{\delta_\alpha}$, $A_2^x = v^{\delta_\beta}$, $B_1 = {h_t}^a$, $B_2 = {h_t}^b$, $\hat{e}(A_3, g_2)^x \hat{e}(h_3, g_2)^{-(\delta_\alpha+\delta_\beta)} \hat{e}(h_3, w)^{-(\alpha+\beta)} \hat{e}(h_1, g_2)^{-a} \hat{e}(h_2, g_2)^{-b} \frac{\hat{e}(g_1,g_2)}{\hat{e}(A_3,w)}$. This proof of knowledge proceeds as follow.
   - (*Auxiliary Commitment.*) User computes A_1, A_2, A_3, B_1, B_2 as above.
   - (*Commitment.*) User randomly selects r_α, r_β, r_x, r_a, r_b, r_{δ_α}, $r_{\delta_\beta} \in_R \mathbb{Z}_p^*$, computes $T_1 = u^{r_\alpha}$, $T_2 = v^{r_\beta}$, $T_3 = A_1^{r_x} u^{-r_{\delta_\alpha}}$, $T_4 = A_2^{r_x} v^{-r_{\delta_\beta}}$, $T_5 = \hat{e}(A_3, g_2)^{r_x} \hat{e}(h_3, g_2)^{-r_{\delta_\alpha}-r_{\delta_\beta}} \hat{e}(h_3, w)^{-r_\alpha-r_\beta} \hat{e}(h_1, g_2)^{-r_a} \hat{e}(h_2, g_2)^{-r_b}$, $T_6 = h_t^{(r_a)}$ and $T_7 = h_t^{(r_b)}$. T_1, T_2, T_3, T_4 are in $\mathbb{G}_1$, T_5 is in $\mathbb{G}_T$ and T_6, T_7 are in $\mathbb{G}_p$.
   - (*Challenge.*) Merchant sends the transaction information M to user. User computes $c = H(A_1, A_2, A_3, B_1, B_2, T_1, T_2, T_3, T_4, T_5, T_6, T_7, M, I)$.
   - (*Response.*) User computes $s_\alpha = r_\alpha - c\alpha$, $s_\beta = r_\beta - c\beta$, $s_x = r_x - cx$, $s_{\delta_\alpha} = r_{\delta_\alpha} - c\delta_\alpha$, $s_{\delta_\beta} = r_{\delta_\beta} - c\delta_\beta$, $s_a = r_a - ca$, $s_b = r_b - cb$ and $s_t = a - cb$. User sends (σ, c, s_t) to merchant, where $\sigma = (A_1, A_2, A_3, B_1, B_2, s_\alpha, s_\beta, s_x, s_a, s_b, s_{\delta_\alpha}, s_{\delta_\beta})$.
   - (*Verify.*) Merchant computes
     * $\tilde{T}_1 = A_1^c u^{s_\alpha}$, $\tilde{T}_2 = A_2^c v^{s_\beta}$, $\tilde{T}_3 = A_1^{s_x} u^{-s_{\delta_\alpha}}$, $\tilde{T}_4 = A_2^{s_x} v^{-s_{\delta_\beta}}$,
     * $\tilde{T}_5 = (\frac{\hat{e}(g_1,g_2)}{\hat{e}(A_3,w)})^c \frac{\hat{e}(A_3,g_2)^{s_x}}{\hat{e}(h_3,g_2)^{(s_{\delta_\alpha}+s_{\delta_\beta})} \hat{e}(h_3,w)^{(s_\alpha+s_\beta)} \hat{e}(h_1,g_2)^{s_a} \hat{e}(h_2,g_2)^{s_b}}$,
     * $\tilde{T}_6 = B_1^c {h_t}^{s_a}$, $\tilde{T}_7 = B_2^c {h_t}^{s_b}$.

     Accepts if $c \stackrel{?}{=} H(A_1, A_2, A_3, B_1, B_2, \tilde{T}_1, \tilde{T}_2, \tilde{T}_3, \tilde{T}_4, \tilde{T}_5, \tilde{T}_6, \tilde{T}_7, M, I)$ and $B_1 \stackrel{?}{=} B_2^c {h_t}^{s_t}$ both hold and rejects otherwise.

– `Deposit Protocol`. The merchant with identity I sends the payment transcript (σ, c, s_t) and M to the bank. The bank verifies the payment transcript exactly as the merchant did. In addition, the bank has to verify that I is indeed the identity of the merchant and (M, σ) is not used before by that merchant. This is to prevent colluding users and merchants submitting double spent coin (which have completely identical transcript). The bank also checks for double-spending by searching if the (B_1, B_2) is already existing in some entry in the deposit database. If it is not found, (B_1, B_2, c, s_t) is stored and the payment is accepted as valid. Otherwise it is a doubly-spent coin.

– `Owner Tracing`. Let the two payment transcripts are (σ, c, s_t) and (σ', c', s_t'), the bank computes $\hat{b} = \frac{s_t - s_t'}{c' - c}$ and $\hat{a} = s_t + c\hat{b}$. The private key and the public key of the double-spender are $\hat{s} = \hat{a}\hat{b}$ and $\hat{y} = h^{\hat{s}}$ respectively.

– `Coin Tracing`. The bank decrypts the value ${h_t}^a$ and ${h_t}^b$ for all other coins issued to the double-spender by the exposed key pair.

5.2 Shorter E-Cash

We can further shorten our payment transcript to 192 bytes with the XDH assumption. We highlight the changes from the short e-cash system as follow.

– `Bank Setup`. Basically the same except the bank's public key is shortened to $bpk = (g_1, g_2, w, h_1, h_2, h, u, v)$.

– `User Setup`. Basically the same except the group $\mathbb{G}_p$ is replaced with $\mathbb{G}_1$.
– `Withdrawal Protocol`. The coin (A, x, a, b) is generated with the same mechanism and hence $A^{x+\gamma} = g_1 h_1^a h_2^b$ still holds. But the tracing information becomes $A_1 = u^a$ and $A_3 = u^b$. To accommodate the changes, we need a new SPK Π_3 instead of the original Π_2. Again Π_3 is shown in the appendix.
– `Payment Protocol`. User spends the coin (A, x, a, b) to a merchant with the identity $I \in \{0,1\}^*$ by executing the following protocol.
  1. User computes auxiliary commitment $A_1 = u^a$, $A_2 = Av^a$, $A_3 = u^b$ and a helper value $\delta = xa$.
  2. User undertakes a proof of knowledge of values (a, b, x, δ) satisfying $A_1 = u^a$, $A_1^x = u^\delta$, $A_3 = u^b$, $\hat{e}(A_2, g_2)^x \hat{e}(v, g_2)^{-\delta} \hat{e}(v, w)^{-a} \hat{e}(h_1, g_2)^{-a}$ $\hat{e}(h_2, g_2)^{-b} = \frac{\hat{e}(g_1, g_2)}{\hat{e}(A_2, w)}$. This proof of knowledge proceeds as follow.
     • (*Auxiliary Commitment.*) User computes A_1, A_2, A_3 as above.
     • (*Commitment.*) User randomly selects $r_a, r_b, r_x, r_\delta \in_R \mathbb{Z}_p^*$, computes
       ∗ $T_1 = u^{r_a}$, $T_2 = A_1^{r_x} u^{-r_\delta}$,
       ∗ $T_3 = \hat{e}(A_2, g_2)^{r_x} \hat{e}(v, g_2)^{-r_\delta} \hat{e}(v, w)^{-r_a} \hat{e}(h_1, g_2)^{-r_a} \hat{e}(h_2, g_2)^{-r_b}$.
     • (*Challenge.*) Merchant sends the transaction information $M \in \{0,1\}^*$ to user. User computes $c = H(A_1, A_2, A_3, T_1, T_2, T_3, M, I)$.
     • (*Response.*) User computes $s_a = r_a - ca$, $s_b = r_b - cb$, $s_x = r_x - cx$, $s_\delta = r_\delta - c\delta$ and $s_t = a - cb$. User sends (σ, c, s_t) to the merchant, where $\sigma = (A_1, A_2, A_3, s_a, s_b, s_x, s_\delta)$.
     • (*Verify.*) Merchant computes $\tilde{T_1} = A_1^c u^{s_a}$, $\tilde{T_2} = A_1^{s_x} u^{-s_\delta}$ and $\tilde{T_3} = (\frac{\hat{e}(g_1, g_2)}{\hat{e}(A_2, w)})^c \hat{e}(A_2, g_2)^{s_x} \hat{e}(v, g_2)^{-s_\delta} \hat{e}(v, w)^{-s_a} \hat{e}(h_1, g_2)^{-s_a} \hat{e}(h_2, g_2)^{-s_b}$. Accepts if both of $c \stackrel{?}{=} H(A_1, A_2, A_3, \tilde{T_1}, \tilde{T_2}, \tilde{T_3}, M, I)$ and $A_1 \stackrel{?}{=} A_3^c u^{s_t}$ hold, rejects otherwise.
– `Deposit Protocol`. Merchant sends the payment transcript (σ, c, s_t) to bank for deposit. In the enhanced protocol, double-spending is identified by the pair (A_1, A_3) (instead of (B_1, B_2)).
– `Owner Tracing`. Suppose the two transcripts are (σ, c, s_t) and (σ', c', s_t'), the bank computes $\hat{b} = \frac{s_t - s_t'}{c' - c}$ and $\hat{a} = s_t + c\hat{b}$. The private key and the public key of the double-spender are $\hat{s} = \hat{a}\hat{b}$ and $\hat{y} = h^{\hat{s}}$ respectively.
– `Coin Tracing`. The bank can decrypt the values u^a and u^b for all other coins issued to the double-spender for tracing.

5.3 Short E-Cash with TTP

In some scenario, the law enforcing agency got the knowledge of a certain criminal by non-cryptographic means, and wants to stop this user from using his coins (which has already been withdrawn). This can be achieved by incorporating a TTP in our scheme for revoking identity and coin tracing of *all* users.

For our first proposed scheme, instead of having h_3, u, v generated fairly, the TTP selects ξ_1, ξ_2 such that $h_3 = u^{\xi_1} = v^{\xi_2}$. The TTP can revoke the identity of every spender by computing $A = A_3/(A_1^{\xi_1} A_2^{\xi_2})$ and identifying the spender from the withdrawal transcript. For the shorter version, TTP's private-public key pair is $(\xi, v = u^{\xi})$. To revoke the identity of the spender, TTP computes $A = A_2/(A_1^{\xi})$ for the bank to identify the spender from the withdrawal protocol.

Coin tracing can be achieved by requiring users to encrypt tracing information ($\{h_t^a, h_t^b\}$, or $\{u^a, u^b\}$ for the shorter version) under TTP's public key. In fact, coin tracing and owner tracing power can be held by different TTP, and each feature can be independently incorporated, by using different proofs in SPK. Due to space limitations, details can be found in the full paper.

5.4 Security Analysis

The security of our system is assured by the following theorems. Their proofs can be found in the full version of this paper. The security analysis of the shorter version goes in a similar way.

Theorem 1 (Balance). *Our proposed construction has the balance property under the q-SDH assumption, in the random oracle model.*

Theorem 2 (Anonymity). *Our proposed construction has the anonymity property under the DLDH assumption in $\mathbb{G}_1$ and DDH assumption in $\mathbb{G}_p$, in the random oracle model.*

5.5 Comparison with Compact E-Cash

We compare the bandwidth and the storage requirement of our scheme with the second scheme in [8] (which supports full coin-tracing). In the following comparison, we instantiate the CHL scheme with a 1024-bit RSA modulus. For our scheme, we take p be a 170-bit prime with the families of curves described in [5]. Using the standard point compression technique, each element in $\mathbb{G}_1$ is 171-bit. Each coin consists of one element in $\mathbb{G}_1$ and three elements in $\mathbb{Z}_p^*$. The coin size is thus 681 bits. Each payment transcript contains three elements in $\mathbb{G}_1$ (A_1, A_2, A_3), two elements in $\mathbb{G}_p$ (B_1, B_2) and nine element in $\mathbb{Z}_p^*$, making its length 512 bytes, if we assume elements in $\mathbb{G}_p$ is representable in 1024 bits. As for the shorter version, each payment transcript contains three elements in $\mathbb{G}_1$ (A_1, A_2, A_3) and six elements in $\mathbb{Z}_p^*$, making its length 192 bytes.

In the CHL scheme, the withdrawal protocol enables the user to withdraw 2^{ℓ} coins at a time. For the payment and deposit protocols, only one coin is processed each time. The space complexity of the withdrawal, payment and deposit transcript are all of order ℓ. In the payment protocol, the user needs to compute $7 + 9\ell$ auxiliary commitments together with $17 + 21\ell$ commitments during the SPKs, and the response takes about 20ℓ elements. The payment transcript size is about $(24 + 50\ell) \times 1024$ bits. Taking $\ell = 10$, spending one coin requires transmission bandwidth of $1024 \times (24 + 500)$ bits, i.e. around 60 Kilobytes. In our scheme, each payment transcript is of constant size 512 bytes. Our scheme's bandwidth requirement in payment is 100 times more efficient.

The withdrawal protocol of the CHL scheme require some more investigation. Without counting the verifiable encryption, the bandwidth required for withdrawing 2^ℓ coins is $(2 + 3\ell) \times 1024$ bits, which is very efficient per coin. However, the verifiable encryption is rather inefficient in itself. For a cheating probability lower than 2^k, the user is required to perform $2k$ encryptions while the bank must perform k encryptions. After this process, the bank needs to store all these $2k$ encryption transcripts later decryption. The verifiable encryption on s is to be performed with relative to the Pedersen commitment $A = {g_0}^r {g_1}^u {g_2}^s {g_i}^{t_i}$ for $i = 3$ to $3\ell{+}3$. Precisely speaking, k rounds of the verifiable encryption has to be done, with each round consisting of one commitment, two bilinear El Gamal encryptions, and $3\ell + 3$ responses (the $3\ell + 3$ term arise since the user has to proof that encryption on s is correctly formed with respect to A, which contains $3 + 3\ell$ exponents). Suppose each component is of size 1024-bit, the total transcript size is $(4 + 3\ell)/8$ kilobytes for each round, making the total transmission requirement of $k(4 + 3\ell)/8$ kilobytes.

A simple trick to simplify the computation is to compute another Pedersen commitment $B = g_0^{r'} {g_1}^s$, proved that the term s in both A and B are the same, and do the verifiable encryption with respect to B. In this case, each round is of size 5×1024 bits, and a total of $5k/8$ kilobytes for k rounds. After that, the bank need to store this $5k/8$ kilobytes of information for later decryption. Thus, bandwidth requirement for CHL's withdrawal protocol per coin (including verifiable encryption using the improved method) is $\frac{2+3\ell+5k}{8 \cdot 2^\ell}$ kilobytes.

For a cheating probability of $0.001 (k = 10)^2$ and taking $\ell = 10$, the average storage per coin required is 10 bytes, using the improved protocol. In our scheme, this kind of inefficient verifiable encryption is not needed with the help of SPK Π_2. A total of 873 bytes is required for each coin (the number of bits required by SPK Π_1, Π_2 is 1363 and 5627 bits respectively), and the bank only needs to store 512 bytes of the encrypted information for each coin.

In short, our scheme is about 50 times less efficient per coin in the withdrawal protocol, and 100 times more bandwidth efficient per coin during the payment protocol and the deposit protocol. Withdrawal can be done by a desktop while the payment may be done in a mobile device with lower computational power and storage. We believe that our scheme is an improvement over the CHL scheme.

6 Conclusion

Double spender tracing is important in an anonymous e-cash system. Coin tracing may be even more important as the bank can freeze the possible misbehavior of a double-spender. Most existing systems relies on the existence of an overpowered TTP, which may identify the spender of a coin and trace all the coins by a particular spender, even the spender is an honest one who never double-spend. Recently, Camenisch, Hohenberger and Lysyanskaya proposed an e-cash system with traceable coins [8]. Once a user double-spends, his identity can be revealed and all his coins in the system can be traced, *without* resorting to TTP. Their scheme is "compact" in the sense that a user can withdraw 2^ℓ coins in a

[2] It is worth noting that using $k = 10$ is in favor of the CHL scheme since the cheating probability of our scheme is $1/q$ with q being a 170-bit prime.

single withdrawal protocol with the cost of $O(\ell \cdot k)$, and the coins can be spent in an unlinkable manner. This result is theoretically very efficient. However, we identify that the bandwidth requirements in payment and deposit protocol, and the bank's storage, may not be efficient for realistic scenario.

In this paper, we present a bandwidth-efficient off-line anonymous e-cash scheme with traceable coins. For a security level comparable with 1024-bit standard RSA signature, the payment transcript size is only 512 bytes. Security of the proposed scheme is proven under the q-strong Diffie-Hellman assumption and the decisional linear assumption, in the random oracle model. The transcript size of our scheme can be further reduced to 192 bytes if external Diffie-Hellman assumption is made. To the best of authors' knowledge, it is the shortest e-cash system currently available. We also show how to incorporate a TTP that is responsible for the owner-tracing and coin-tracing, if such a TTP is desired.

References

1. Masayuki Abe and Eiichiro Fujisaki. How to Date Blind Signatures. *Advances in Cryptology - ASIACRYPT 1996*, *LNCS* 1163, pp. 244–251. Springer, 1996.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. *Advances in Cryptology - CRYPTO 2000*, *LNCS* 1880, pp. 255–270. Springer, 2000.
3. Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. *Advances in Cryptology - EUROCRYPT 2004*, *LNCS* 3027, pp. 56–73. Springer, 2004.
4. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. *Advances in Cryptology - CRYPTO 2004*, *LNCS* 3152, pp. 41–55. Springer, 2004.
5. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Advances in Cryptology - ASIACRYPT 2001*, *LNCS* 2248, pp. 514–532.
6. Stefan Brands. Untraceable Off-line Cash in Wallets with Observers. *Advances in Cryptology - CRYPTO '93*, *LNCS* 773, pp. 302–318. Springer, 1994.
7. Ernie Brickell, Peter Gemmell, and David Kravitz. Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change. *SODA '95: ACM-SIAM Symposium on Discrete Algorithms*, pp. 457–466. SIAM, 1995.
8. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-Cash. *Advances in Cryptology - EUROCRYPT 2005*, *LNCS* 3494, pp. 302–321. Springer.
9. Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups. *Advances in Cryptology - CRYPTO '97*, *LNCS* 1294, pp. 410–424.
10. Sébastien Canard and Jacques Traoré. On Fair E-cash Systems Based on Group Signature Schemes. *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, *LNCS* 2727, pp. 237–248. Springer, 2003.
11. David Chaum. Blind Signatures for Untraceable Payments. *Advances in Cryptology: CRYPTO '82*, pp. 199–203. Plenum, New York, 1983.
12. David Chaum, Amos Fiat, and Moni Naor. Untraceable Electronic Cash. *Advances in Cryptology - CRYPTO '88*, *LNCS* 403, pp. 319–327. Springer, 1990.
13. David Chaum and Eugène van Heyst. Group Signatures. *Advances in Cryptology - EUROCRYPT '91*, *LNCS* 547, pp. 257–265. Springer, 1991.
14. Niels Ferguson. Single Term Off-Line Coins. *Advances in Cryptology - EUROCRYPT '93*, *LNCS* 765, pp. 318–328. Springer, 1994.

15. Matthew K. Franklin and Moti Yung. Secure and Efficient Off-Line Digital Money (Extended Abstract). *Automata, Languages and Programming, 20th International Colloquium, ICALP '93*, *LNCS* 700, pp. 265–276. Springer, 1993.
16. Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing Big Brother: An Abuse-Resilient Transaction Escrow Scheme. *Advances in Cryptology - EUROCRYPT 2004*, *LNCS* 3027, pp. 590–608. Springer, 2004.
17. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable Signatures. *Advances in Cryptology - EUROCRYPT 2004*, *LNCS* 3027, pp. 571–589. Springer, 2004.
18. Anna Lysyanskaya and Zulfikar Ramzan. Group Blind Digital Signatures: A Scalable Solution to Electronic Cash. *Financial Cryptography 98*, *LNCS* 1465, pp. 184–197. Springer, 1998.
19. Greg Maitland and Colin Boyd. Fair Electronic Cash Based on a Group Signature Scheme. *Information and Communications Security, 3rd International Conference, ICICS 2001*, *LNCS* 2229, pp. 461–465. Springer, 2001.
20. Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable Electronic Coupon Protocol with Anonymity Control. *Information Security, Second International Workshop, ISW'99*, *LNCS* 1729, pp. 37–46. Springer, 1999.
21. Tatsuaki Okamoto. An Efficient Divisible Electronic Cash Scheme. *Advances in Cryptology - CRYPTO '95*, *LNCS* 963, pp. 438–451. Springer, 1995.
22. Tatsuaki Okamoto and Kazuo Ohta. Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash. *Advances in Cryptology - CRYPTO '89*, *LNCS* 435, pp. 481–496. Springer, 1990.
23. Weidong Qiu, Kefei Chen, and Dawu Gu. A New Offline Privacy Protecting E-cash System with Revokable Anonymity. *Information Security, 5th International Conference, ISC 2002*, *LNCS* 2433, pp. 177–190. Springer, 2002.
24. Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. Fair Blind Signatures. *Advances in Cryptology - EUROCRYPT '95*, *LNCS* 921, pp. 209–219. Springer.
25. Jacques Traoré. Group Signatures and Their Relevance to Privacy-Protecting Off-Line Electronic Cash Systems. *Information Security and Privacy, 4th Australasian Conference, ACISP'99*, *LNCS* 1587, pp. 228–243. Springer, 1999.
26. Patrick P. Tsang and Victor K. Wei. Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation. *Information Security Practice and Experience, First International Conference, ISPEC 2005*, *LNCS* 3439, pp. 48–60. Springer, 2005.
27. Sebastiaan von Solms and David Naccache. On Blind Signatures and Perfect Crimes. *Computer Security*, 11(6):581–583, 1992.

A Signature Knowledge of Representation

A signature of knowledge allows a signer to prove the knowledge of a secret with respect to some public information non-interactively by tying his knowledge of a secret to a message begin signed. Following the notion in [9], we called these signature "Signatures based on Proofs of Knowledge" (SPK).

As an example, we denote the zero-knowledge proof of the discrete logarithm of y by $SPK\{(x) : y = g^x\}(M)$, where M is the hash value of the commitment.

The SPK Π_1, Π_2 and Π_3 used in our proposal are shown below.

$\Pi_1 = \mathrm{SPK}\{(\bar{a}, \bar{b}, s, r_1, \delta) : \bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \bigwedge A_1 = h_1^{r_1} h_2^{\bar{a}} \bigwedge A_1^{\bar{b}} = h_1^{\delta} h_2^{s} \bigwedge y = h^s\}(M)$ where $M = H(\bar{C}, A_1, y, h_1, h_2,)$.

For Π_2, first compute $A_1 = \bar{C}^r$ and $A_2 = \bar{C}^{r^{-1}}$ and execute the following SPK: $\Pi_2 = \mathrm{SPK}\{(\bar{a}, \bar{b}, a, b, \delta_a, \delta_b, t_a, t_b) : \bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \bigwedge A_1 = h_1^{a} h_2^{\delta_b} \bigwedge A_2 =$

$h_1^{\delta_a} h_2^b \bigwedge C = h_1^a h_2^b \bigwedge R_1 = h^{t_a} \bigwedge R_2 = y^{t_a} h_t^a \bigwedge R_3 = h^{t_b} \bigwedge y^{t_b} h_t^b\}(M)$, where $M = H(\bar{C}, A_1, A_2, C, R_1, R_2, R_3, R_4, h_1, h_2, y)$. Note that R_1,R_2,R_3,R_4 is the encryption of h_t^a and h_t^b under the public key $y = h^s$.

For Π_3, first compute $A_1 = \bar{C}^r$ and $A_2 = \bar{C}^{r^{-1}}$ and execute the following SPK: $\Pi_3 = \mathrm{SPK}\{(\bar{a}, \bar{b}, \delta_a, \delta_b,\ a, b, s) : \bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \bigwedge A_1 = h_1^a h_2^{\delta_b} \bigwedge A_2 = h_1^{\delta_a} h_2^b \bigwedge C = h_1^a h_2^b \bigwedge y = h^s \bigwedge R_1 = h^{t_1} \bigwedge R_2 = y^{t_1} u^a \bigwedge R_3 = h^{t_2} \bigwedge R_4 = y^{t_2} u^b\}(M)$, where $M = H(\bar{C}, C, A_1, A_2, R_1, R_2, R_3, R_4, y, h_1, h_2, u, v)$. Note that R_1, R_2, R_3, R_4 is the encryption of u^a and v^b under the public key $y = h^s$.