Double-Blind Compact E-cash from Bilinear Map

Jiyang Chen[1,2], Bin Lian[1*], Yongjie Li[1], Jialin Cui[1], Ping Yu[1], Zhenyu Shu[1], and Jili Tao[1]
[1] NingboTech University
Ningbo, Zhejiang 315100, China
[e-mail: lianbin_a@163.com]
[2] College of information and electronic engineering, Zhejiang University
Hangzhou, Zhejiang 310058, China
[*]Corresponding author: Bin Lian

Received December 9, 2022; accepted April 5, 2022;
published April 30, 2022

Abstract

Compact E-cash is the first scheme which can withdraw 2^l coins within $\mathcal{O}(1)$ operations and then store them in $\mathcal{O}(l)$ bits. Because of its high efficiency, a lot of research has been carried out on its basis, but no previous research pay attention to the privacy of payees and in some cases, payees have the same privacy requirement as payers. We propose a double-blind compact E-cash scheme, which means that both the payer and the payee can keep anonymous while spending. In our scheme, the payer and the bank cannot determine whether the payees of two different transactions are the same one and connect the payee with transactions related to him, in this way, the privacy of the payee is protected. And our protocols disconnect the received coin from previous transaction, then, the coin can be transferred into an unspent coin which belongs to the payee. The proposed e-cash scheme is secure within y-DDHI and LRSW assumption.

Keywords: Compact E-cash, Bilinear map, zero-knowledge proof, CL Signatures, anonymity.

This research work is supported by National Natural Science Foundation of China (No: 61972350, 61872321, 62172356), Ningbo 2025 Major Project of Science and Technology Innovation (No: 2021Z109, 2019B10079, 2020Z021, 2021Z063, 2021Z010), Zhejiang Provincial Natural Science Foundation of China (No: LY17F020019, LQ20E070001, LY21F040004) and Natural Science Foundation of Ningbo (No: 2021J166).

http://doi.org/10.3837/tiis.2022.04.010 **ISSN : 1976-7277**

1. Introduction

In electronic cash(E-cash) schemes, users are allowed to withdraw coins from the bank, and then pay coins to merchants as the service fees. If the merchant wants to spend these coins, he needs to deposit them to the bank and then the bank credits them into his account. And a good electronic cash scheme requires the follow four properties:

1. **Balance**: The users cannot spend more coins than they have even cooperating with other users and merchants.

2. **Anonymity**: If the user is honest, then it is impossible for the bank to infer the user's identification even cooperating with any other malicious merchants and users.

3. **Identifying double-spenders:** The bank can identify the double-spender.

4. **Exculpability.** If the user is honest, then no one can frame him as a double-spender.

In compact E-cash scheme, a user can withdraw a wallet containing 2^l coins and store the wallet using $\mathcal{O}(l+k)$ bits, where k is a security parameter[1]. Compare to the best previously known scheme which requires at least $\mathcal{O}(2^l \cdot k)$ bits before, the storage efficiency of compact E-cash has made great progress. And many E-cash schemes have been put forward based on the compact E-cash. [2] introduces the bounded accumulator to construct the compact E-cash scheme. [3] introduces compact and batch spending to compact E-cash more practical and achieves spending multiple coins just in one protocol which makes compact E-cash more practical.[4] constructs a compact E-cash scheme without the random oracle. [5] presents an efficient and practical E-cash scheme based on an offline trusted third party (TTP). [6] constructs a new compact E-cash scheme with arbitrary wallet size using verifiable random functions and bounded accumulators. [7] achieves practical and complete tracing in compact E-cash scheme by designing new structure of knowledge proof and improves the complexity of withdrawing a wallet from $\mathcal{O}(k)$ to $\mathcal{O}(1)$. [8] introduces zero-knowledge proof with non-standard structure to solve the efficiency problem on coin-tracing.

Anonymity is an important research point in the field of E-cash and has attracted wide attention from the research community. [9] use a bounded accumulator with the classical binary tree to achieve full unlinkability and anonymity. [10] achieves full anonymity on divisible E-cash scheme without requiring a TTP.[11] constructs a multi-authority E-cash scheme based on blind threshold signature. [12] uses the malleable signatures to make coins transfer anonymously and safely.

However, none of the previous E-cash schemes consider the privacy protection for payees. And the privacy of payees needs to be considered in some cases. For example, when the payee of the coin is a person rather than a shop, he does not want anyone else to know how many coins he has received and other transaction.

In Bitcoin, the privacy protection for payees is realized and the privacy policy is to isolate the receipt and payment address from the physical identity of the holder [13]. However, all the transaction log is public, thus the only protection on user's privacy is the pseudonyms [14]. An increasing body of research shows that Bitcoin can be de-anonymized [15], which can make serious violation of user's privacy.

In view of the above problems, we introduce the concept of double-blind into compact E-cash, and protect the privacy of both users and merchants. Different from pseudonym in Bitcoin schemes, the signature from the bank belongs to the user and the merchant in our

scheme is randomized in every transaction such that the bank cannot link the two transactions where the payee and payer remain the same. And the merchant can transfer the coin he received previously into a new coin for next spending anonymously without its account.

In Section 2, we present the related preliminaries used in the scheme. We provide security model in Section 3. Our scheme is described in Section 4. Security of the proposed scheme is analyzed in Section 5 and we provide the system efficiency in Section 6. Finally, in Section 7, we put the conclusion of the paper.

2. Preliminaries

2.1. Complexity Assumptions

2.1.1. y-Decisional Diffie-Hellman Inversion(y-DDHI) Assumption [16], [17]

Randomly given a generator $g \in G$, where G is a group, whose order is a prime number q, the values $\left(g, g^x, \ldots, g^{(x^y)}\right)$ for random $x \in \mathbb{Z}_q$, and a value $R \in G$, it is hard to decide if $R = g^{1/x}$ or not.

2.1.2. LRSW Assumption [18]

$G = \langle g \rangle$ is a group. Let $X, Y \in G, X = g^x, Y = g^y$. Let $O_{X,Y}(\cdot)$ denote an oracle whose input is a value $m \epsilon \mathbb{Z}_q$, and outputs $A = (a, a^y, a^{x+mxy})$ where a is randomly chosen by the oracle. For all probabilistic polynomial time adversaries$\mathcal{A}$, the negligible function $v(k)$ defined as follows:

$$\Pr\left[(q, G, \mathcal{G}, g, \mathcal{g}, e) \leftarrow Setup\left(1^k\right); x \leftarrow \mathbb{Z}_q; y \leftarrow \mathbb{Z}_q; X = g^x; Y = g^y;\right.$$
$$(a, b, c, m) \leftarrow \mathcal{A}^{O_{X,Y}}(g, G, \mathcal{g}, \mathcal{G}, q, e, X, Y): m \notin Q \ \wedge m \in \mathbb{Z}_q \wedge m \neq 0 \wedge$$
$$\left. a \in G \wedge b = a^y \wedge c = a^{x+mxy}\right] = v(k) \quad (1)$$

Where Q is the set of queries that $\mathcal{A}$ made to $O_{X,Y}(\cdot)$, and $\mathcal{g} = e(g, g)$ is a generator of $\mathcal{G}$.

2.2. Bilinear Maps[16],[19]

$(q, g_1, h_1, G_1, g_2, h_2, G_2, G, e)$ are the generated parameters of a bilinear mapping with corresponding to the security parameter 1^k.The definition of bilinear maps and the exact meaning of the parameters are as follows:

1.G_1, G_2 are groups of prime order $q = \Theta\left(2^k\right)$;

2.each group has two generators where $G_1 = \langle g_1 \rangle = \langle h_1 \rangle$ and $G_2 = \langle g_2 \rangle = \langle h_2 \rangle$.

3.ψ is a isomorphism from $G_2$to G_1, where $\psi(g_2) = g_1$ and $\psi(h_2) = h_1$;

4.e is a bilinear map: $G_1 \times G_2 \rightarrow G$. The properties are as follow:

(**Bilinear**) for all $g_1 \in G_1, g_2 \in G_2$, and $a, b \in \mathbb{Z}_q$,$e\left({g_1}^a, {g_2}^b\right) = e(g_1, g_2)^{ab}$;

(**Non-degenerate**) if g_1 is a generator of $G_1$and g_2 is a generator of G_2, then $e(g_1, g_2)$ is a generator G.

2.3. Signature of Knowledge

2.3.1. statistically indistinguishable [20]

Let $L \in \{0,1\}^*$ be a language and let $\{A(x)\}_{x \in L}$ and $\{B(x)\}_{x \in L}$ denote two sets of random variables which can be indexed by $x \in L$. A and B are statistically indistinguishable if their statistical difference is negligible, or more specifically, if for every polynomial $p(\cdot)$ and for all

long $x \in L$ it holds that

$$\sum_{\alpha\epsilon\{0,1\}^*} |Pr(A(x) = \alpha) - Pr(B(x) = \alpha)| < \frac{1}{p(|x|)} \quad (2)$$

2.3.2. statistical zero-knowledge [21]

Given an interactive protocol (P, V), if there exists a probabilistic expected polynomial-time simulator S_V and two ensembles $\{[V, P](x)\}_{x\in L}$ and $\{S_V(x)\}_{x\in L}$ are statistical indistinguishable, then protocol (P, V) is statistical zero-knowledge.

2.3.3. signature of knowledge [22]

A pair $(c, s) \in \{0,1\}^{\ell} \times \mathbb{Z}_q$ satisfying $c = H(y||g||y^c g^s||m)$ is a signature of knowledge of the discrete logarithm which denotes y to the base g, on the message m and it's denoted as $SPK\{x: y = g^x\}(m)$.
$SPK\{x: y = g^x\}(m)$ can be computed if $x \equiv log_g y (mod\ q)$ is given by choosing a random $r \in \mathbb{Z}_q$ and computing c and s as: $c = H(g||y||t||m)$ and $s = r - cx\ (mod\ q)$. $t = g^r$ is the commitment to prove the knowledge of $x \equiv log_g y (mod\ q)$. And $H: \{0,1\}^* \rightarrow \{0,1\}^k$ is a strong collision-resistant hash function. Signature of knowledge is the non-interactive version of zero-knowledge proof protocol.

2.4. Pseudorandom Function [17]

Function $f \in F_n$ is defined by the tuple (G, q, g, s), where g is a generator of a group G whose order is a prime order q and s is a seed selected in $\mathbb{Z}_q$. For $x \in \mathbb{Z}_q$(except for $x \equiv -1\ mod\ q$), the function $f_{G,q,g,s}(\cdot)$, simply denoted as $f_{g,s}^{DY}(\cdot)$, is defined as $f_{g,s}^{DY}(x) = g^{1/(s+x+1)}$.

2.5. Pedersen Commitment Scheme [23]

G is a group with prime order q, and generators are $(g_0, \cdots, g_m)$. In order to commit the set of values $(v_1, \cdots, v_m) \in {\mathbb{Z}_q}^m$, the user picks a random $r \in \mathbb{Z}_q$ and sets $C = g_0^r \prod_{i=1}^m g_i^{v_i}$.

2.6. CL Signatures [18], [24]

The CL Signature scheme consists of two protocols:

1.The first protocol occurs between the user and the signer with (pk_S, sk_S). The public input of both parties is pk_S. And the secret input of the user are values $(v_1, \cdots, v_m; r)$ such that $C = PedCom(v_1, \cdots, v_m; r)$.The secret input of the signer is sk_S. During interacting, the signer cannot learn anything about the secret of the user. After the protocol, the user obtains the signer's signature on the values which is denoted as $\sigma_{pk_S}(v_1, \cdots, v_m)$.

2. The second protocol occurs between the user and the verifier to prove knowledge of a signature. The public input of both parties is pk_S and the secret input of the user are values $(v_1, \cdots, v_m; r)$ and the signature $\sigma_{pk_S}(v_1, \cdots, v_m)$. There is no secret input on the verifier's side. The verifier outputs whether the signature is valid or not.

The CL signature scheme's security is based on RSA group or bilinear maps.

3. Definition of Security

3.1. Syntax [1]

Suppose that P is a protocol occurring between A and B, then $P(A(x), B(y))$ denotes that A

inputs x, and B inputs y. There are three entries in our electronic cash scheme: user $\mathcal{U}$, bank $\mathcal{B}$, and merchant $\mathcal{M}$. The protocols and algorithms are defined as follow:

- $B/UKeyGen(1^k, params)$ is an algorithm for $\mathcal{B}$ to generate his key pair, and $UKeyGen(1^k, params)$ is the algorithm for $\mathcal{U}$ and $\mathcal{M}$ to generate key pair ($\mathcal{M}$ can be seen as a special user).
- $Sign(\mathcal{M}(sk_m, pk_B), \mathcal{B}(sk_B, pk_m))$ is a protocol whose outcome is a certificate for $\mathcal{M}$, which represents the legality of $\mathcal{M}$ as a payee.
- $Withdraw(\mathcal{U}(sk_u, pk_b, 2^l), \mathcal{B}(sk_b, pk_u, 2^l))$ is a protocol occurring between $\mathcal{U}$ and $\mathcal{B}$ which allows $\mathcal{U}$ to withdraw a wallet containing 2^ℓ coins from $\mathcal{B}$. $\mathcal{B}$ records a debit of 2^ℓ coins for $\mathcal{U}'$ account pk_{u}.
- $Spend(\mathcal{U}(W, pk_b, \sigma_m^r), \mathcal{M}(sk_m, pk_b, \sigma_m, \sigma_u^r))$ is a protocol occurring between $\mathcal{U}$ and $\mathcal{M}$ which enables $\mathcal{U}$ to spend a coin anonymously. $\mathcal{M}$ obtains (S, π) which denotes the serial number and the validity proof of the coin and $\mathcal{U}$ updates his wallet. Here, σ_m^r denotes the randomized signature.
- $Deposit(\mathcal{M}(sk_m, S, \pi, pk_b), \mathcal{B}(sk_b, \sigma_m^r))$ is a protocol occurring between $\mathcal{M}$ and $\mathcal{B}$ allowing $\mathcal{M}$ to deposit the coin he has received. After the protocol, $\mathcal{B}$ adds the coin to the spent coins list.
- $Transfer(\mathcal{M}(sk_m, S, \pi, pk_b), \mathcal{B}(sk_b, \sigma_m^r))$ is a protocol occurring between $\mathcal{M}$ and $\mathcal{B}$ allowing $\mathcal{M}$ to convert the coin he has received into a new coin belongs to him anonymously. After the protocol, $\mathcal{M}$ obtains a new wallet containing only one coin.
- $Identify(params, S, \pi_1, \pi_2)$ is an algorithm to identify the double-spender with S and two validity proofs, π_1 and π_2. The output of this algorithm consists of a public key pk_{u} and a proof $\prod_G$.
- $VerifyGuilt(S, pk_{\mathrm{u}}, \prod_G)$ is an algorithm to verify the proof $\prod_G$ to prove that the user whose public key is pk_{u} is a real double-spender.

3.2. Security Definitions

$Alg^{X-Y(\mathcal{A})}$ denotes an algorithm which has the access model X-Y to the adversary $\mathcal{A}$ with input x. For the more details about X-Y model we refer to [1]

$\mathcal{E}_{Prot,l}^{X-Y(\mathcal{A})}$denotes an extractor of the proof protocol $Prot$ for language l in the $X-Y$ model. For property formed $(params, auxext)$, the extractor will output a w in expected polynomial time such that $(x, w) \in l$ whenever the probability that the verifier accepts x in the $X-Y$ model, is non-negligible.

$\mathcal{S}_{Prot,l}^{X-Y(\mathcal{A})}$ denotes a simulator of the proof protocol $Prot$ for language l in the $X-Y$ model. When interacting with $\mathcal{A}$ in the $X-Y$ model, the zero-knowledge simulator $\mathcal{S}_{Prot,l}^{X-Y(\mathcal{A})}$ makes $\mathcal{A}$ unable to distinguish between the view generated by it and the view generated by a real user.

3.2.1. Balance

Let m_1 denote the message sent from $\mathcal{U}$ to $\mathcal{B}$ which can identify himself and b_1 denote the state of the bank when m_1 received. The balance property requires that:

1. There exists an efficiently decidable language l_S and an extractor $\mathcal{E}_{Prot,l}^{X-Y(A)}(b_1, m_1)$ for all b_1 m_1, it extracts $w = (S_1, \dots, S_n, aux)$, where n denotes the capacity of the wallet. Such that (b_1, m_1, w)$\in l_s$ when the probability that the bank accepts in the $Withdraw$ or $Transfer$ protocol is non-negligible. The $Transfer$ protocol can be seen as a special variant of $Withdraw$ protocol.
2. On input $(params, pk_B)$, the adversary $\mathcal{A}$ plays the following game: $\mathcal{A}$ can execute $Withdraw$, $Transfer$ and $Deposit$ protocols with the bank as many times as he wishes (he can spend the coin to himself). Let $(b_1, i, m_1, I, w_i) \in l_S$ be the output of $\mathcal{E}_{Prot,l}^{X-Y(A)}(pk_i, b_{1,i}, m_{1,i})$ if the ith $Withdraw$ or $Transfer$ protocol. Recall that $w_i = (S_{i,1}, \dots, S_{i,n}, aux)$ is a set of n serial numbers belong to pk_i,where n is the capacity of the wallet. Let $A_f = \{S_{i,j} | 1 \le i \le f, 1 \le j \le n\}$ represents serial numbers that $\mathcal{A}$ get after f executions of $Withdraw$ or $Transfer$ protocol. If in some $Deposit$ or $Transfer$ protocol, the bank accepts a coin whose serial number $S \notin A_f$, then $\mathcal{A}$ wins. We require that $\mathcal{A}$ wins with only negligible probability.

3.2.2. Identification of double spenders

This property requires that no probabilistic polynomial-time adversary can win the following game with non-negligible probability:

On input pk_B, $\mathcal{A}$ can execute $Withdraw$,$Transfer$ and $Deposit$ protocols with bank as many times as he wishes. We have $w_i = (S_{i,1}, \dots, S_{i,n}, aux)$ is a set of n serial numbers belong to pk_i which we have defined in section 3.2.1. Let $A_f = \{S_{i,j} | 1 \le i \le f, 1 \le j \le n\}$ represents serial numbers that $\mathcal{A}$ get after f executions of $Withdraw$ or $Transfer$ protocol. If in some $Spend$ protocol, the bank, accepts a coin which was already recorded in the spent coin list, yet $Identify$ outputs pk_U and $\prod_G$ which is not accepted by $VerifyGuilt$, then $\mathcal{A}$ wins the game.

3.2.3. Anonymity of the user

Consider an adversary $\mathcal{A}$ with the bank's public key pk_B issues the following queries:

PK of I: $\mathcal{A}$ requests and received the public key of user $\mathcal{U}$, generated as (pk_u, sk_u).

Withdraw with I: $\mathcal{A}$ executes the $Withdraw$ protocol with $\mathcal{U}$:

$$Withdraw(\mathcal{U}(pk_B, sk_u, n), \mathcal{A}(state, n))$$

The user's output of jth query is W_j.

Spend from wallet j: $\mathcal{A}$ executes the Spend protocol from W_j:

$$Spend(\mathcal{U}(W_j), \mathcal{A}(state)).$$

We require that $\mathcal{A}$ cannot ask $\mathcal{U}$ to pay more than n coins from W_j and there exists a simulator $\mathcal{S}^{X-Y}(auxsim, \cdot)$ and the adversary $\mathcal{A}$ can distinguish which of the following game he is playing with only negligible advantage:

Game R: $\mathcal{A}$ issues the queries above to the real user $\mathcal{U}$.

Game I: $\mathcal{A}$ issues the PK and Withdraw queries to the real user $\mathcal{U}$, but in the Spend query, $\mathcal{A}$ interacts with the simulator rather than $\mathcal{U}$.

Recall that there are many users executes $Withdraw$ protocol with the adversary, and the Spend query is initiated by the user, so, the adversary cannot determine which user he interacts with.

3.2.4. Anonymity of the merchant

Here, we consider the anonymity of the merchant during the $Sign$, $Spend$ and $Transfer$ protocols. The adversary $\mathcal{A}$ with the bank's public key pk_B issues the following queries:

PK of M: $\mathcal{A}$ requests and receives the public key of merchant $\mathcal{M}$, generated as (pk_m, sk_m)

Sign with M: $\mathcal{A}$ executes the $Sign$ protocol with $\mathcal{M}$:

$$Sign(\mathcal{M}(pk_B, sk_m), \mathcal{A}(state))$$

We denote the merchant's output after the query by σ.

Spend with M: we suppose that $\mathcal{A}$ has a valid wallet W_j, and executes the Spend protocol with $\mathcal{M}$:

$$Spend(\mathcal{A}(W_j), \mathcal{M}(\sigma)).$$

Transfer: $\mathcal{A}$ executes the $Transfer$ protocol with the merchant $\mathcal{M}$:

$$Transfer(\mathcal{M}(pk_B, sk_m, \pi), \mathcal{A}(state))$$

No adversary $\mathcal{A}$ can distinguish which of the following game he is playing with non-negligible advantage:

Game R: $\mathcal{A}$ issues the queries above to the real user $\mathcal{M}$.

Game I: $\mathcal{A}$ issues the PK and Sign queries to the real merchant $\mathcal{M}$, but in Transfer and Spend query, $\mathcal{A}$ interacts with the simulator rather than $\mathcal{M}$.

3.2.5. Exculpability

Exculpability guarantees that no one can slander an honest user as a double-spender. We set an adversary $\mathcal{A}$ who launches the following attack against a user:

Setup: The bank $\mathcal{B}$ generates system parameters and the user $\mathcal{U}$ generates the key pair (pk_u, sk_u). The adversary gets the bank's public key pk_B.

Queries:

Withdraw wallet: $\mathcal{A}$ plays the role of $\mathcal{B}$ while executing $Withdraw$ protocol with $\mathcal{U}$.The output of $\mathcal{U}$ after $Withdraw$ is a wallet W_j. Recall that $Transfer$ can be seen as a special $Withdraw$ protocol.

Spend wallet: $\mathcal{A}$ plays the role of the merchant $\mathcal{M}$ while executing $Spend$ protocol with $\mathcal{U}$ whose input is W_j. We require that $\mathcal{A}$ cannot execute this query more than the capacity of W_j.

Success criterion: After above queries, $\mathcal{A}$ outputs $(S, \prod)$ and wins the game if $VerifyGuilt$ accepts.

The scheme guarantees exculpability if the adversary wins with negligible probability.

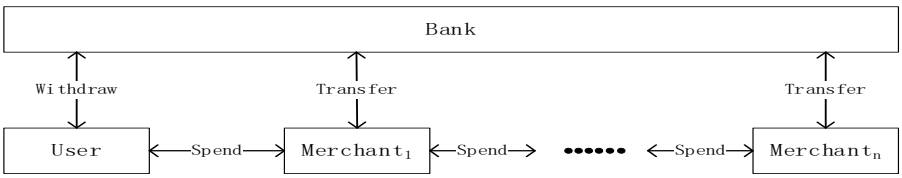

Fig. 1. System design block diagram

4. Double-Blind Compact E-cash based on bilinear map

4.1. The System Model

Our system model is illustrated in **Fig. 1**. Firstly, the user withdraws a wallet containing 2^l

coins and the bank records a debit for the user's account. Secondly, the user pays the merchant coins as the reward. After Transfer protocol, the merchant cuts off the connection between the coin he received and previous transactions for next spending. Before depositing this coin into his account, the final merchant could transfer the coin and pay the coin to himself for cutting off the link between the coin and previous transactions, so that no other entities can confirm whether the coin is from a special transaction.

4.2. Global parameters for the system

On input the security parameter 1^k, the system is initialized and the public parameters used are following:

$G_1 = \langle g_1 \rangle = \langle h_1 \rangle$ and $G_2 = \langle g_2 \rangle = \langle h_2 \rangle$, both have the same prime order $q = \Theta(2^k)$. These two groups are used to construct the Pedersen commitment scheme and CL signature scheme based on the bilinear map e: $G_1 \times G_1 \rightarrow G_2$.

$G = \langle g \rangle$ is a group whose order is a prime number and denoted as $q = \Theta(2^k)$. And the use of this group is to generate the serial number and the security tag of coins.

4.3. B/U Keygen Algorithm

In $BKeygen(1^k, \zeta)$,the bank generates a key pair (pk_B, sk_B) for CL signature, and the forms are as follows:

$\boldsymbol{sk_B}$: $x \leftarrow Z_q, y \leftarrow Z_q, z_i \leftarrow Z_q, 1 \leq i \leq 3$

$\boldsymbol{pk_B}$: $X = {g_1}^x, Y = {g_1}^y, Z_i = {g_i}^{z_i}, 1 \leq i \leq 3$

In $UKeygen(1^k, \zeta)$, $\mathcal{U}$ generates a unique key pair $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = ({g_1}^u, u)$ for random $u \epsilon \mathbb{Z}_q$.Recall that merchants can be seen as special users and also generates a unique key pair $(pk_{\mathcal{M}}, sk_{\mathcal{M}}) = ({g_1}^m, m)$ for a random $m \in \mathbb{Z}_q$.ζ denotes the necessary common parameters.

4.4. Overview of our scheme

we explain the main idea of each protocol to help readers to better understand our scheme in this section.

4.4.1. Withdraw Protocol

In this protocol, the user runs the CL Signature protocol with the bank for obtaining the signature on his secret values. Here, the original signature σ_u is the identification of the user, and in next protocols, we use randomized signature $\sigma_u{}'$ which is also valid but independent of σ_u to prove the validity of the user as a payer, such that others cannot determine whether two transactions are linkable.

4.4.2. Sign Protocol

In this protocol, the merchant runs the CL Signature protocol with the bank for obtaining the signature on his private key Here, the original signature σ_m is the certificate of the merchant, and in next protocols, we use randomized signature $\sigma_m{}'$ which is also valid but independent of σ_m to prove the validity of the merchant as a payee.

4.4.3. Spend Protocol

In order to protect the anonymity of both users and merchants while spending, the public keys of both two parties are not disclosed to each other. So, in order to prove the validity of the merchant as a payee and the validity of the user as a payer, we construct two zero-knowledge

proof.

In our scheme, there are two kinds of wallet, one contains 2^l coins from $Withdraw$, and the other contains only one coin from $Transfer$.Serial number and security tag of the two wallets construct similarly, so the user may spend the latter wallet as the former. To prevent this from happening, we change the structure of the latter wallet. And if the user is not honest, the merchant can detect it and reject this coin.

4.4.4. Deposit Protocol

The merchant sends the serial number and the validity proof of the coin to the bank and proves that he is the payee of the coin using the signature $\sigma_m{}'$ and sk_m. If the verification is passed and the bank confirms that the serial number has not been used before, then the bank accepts this coin and credits the merchant's account. And if the same serial number is received by the bank with same R for the second time, it indicates that the merchant sends the same coin twice; Otherwise, if R is different, the bank recognizes the user as a double-spender and runs $Identify$ algorithm to get his public key.

4.4.5. Transfer Protocol

The merchant sends the coin to the bank and proves that he is the payee of the coin using the signature $\sigma_m{}'$ and sk_m. If the verification is passed and the bank confirms that the serial number has not been used before, then the bank accepts this coin and the merchant can transfer the coin into a new coin whose value comes from the sent coin without the merchant's account.

4.5. Protocols and Algorithms of Our Schemes

In this section, we mainly talk about the interaction steps in each protocol. And more details are put in Appendix A.

4.5.1. Withdraw Protocol

In this protocol, the user's wallet containing 2^l coins is (sk_u, s, t, σ, J), where σ_u denotes the signature on (sk_u, s, t) from the bank and J is the coin counter with l bits. In the process of interaction, the bank cannot learn anything about (sk_u, s, t). The interactions between the user $\mathcal{U}$ and the bank $\mathcal{B}$ are following:

1. $\mathcal{U}$ proves the knowledge of sk_u to $\mathcal{B}$ for identifying himself.
2. $\mathcal{U}$ selects random value $s', t \in \mathbb{Z}_q$ and sends $\mathcal{B}$ a commitment

$$A' = PedCom(sk_u, s', t; r) = {g_1}^r {Z_1}^u {Z_2}^{s'} {Z_3}^t. \quad (3)$$

$\mathcal{B}$ sends a random $r' \in \mathbb{Z}_q$. $\mathcal{U}$ computes $s = s' + r'$. $\mathcal{U}$ and $\mathcal{B}$ respectively computes

$$A = {Z_2}^{r'} A' = PedCom(sk_u, r' + s', t; r) = PedCom(sk_u, s, t; r). \quad (4)$$

3. $\mathcal{U}$ runs the CL signature protocol with $\mathcal{B}$ and obtains $\mathcal{B}'$s signature on secret values which is:

$$\sigma_u = (a, a^y, a^x A^{\alpha xy}, A_1, A_2, A_3, B_1, B_2, B_3) \quad (5)$$

4. $\mathcal{U}$ saves the wallet $W = (sk_u, s, t, \sigma, J)$, and J is initialized to zero.
5. $\mathcal{B}$ records 2^l coins for the user's account pk_u.

4.5.2. Sign Protocol

The output of this protocol is a certificate δ_m, which represents the legitimacy of the merchant $\mathcal{M}$ as a payee. The interactions between the merchant $\mathcal{M}$ and the bank $\mathcal{B}$ are following:

1. $\mathcal{M}$ runs the CL signature protocol with $\mathcal{B}$ and obtains $\mathcal{B}'$s signature on sk_mwhich is:

$$\sigma_m = (a, a^y, a^x M^{\alpha xy}, A_1, B_1). \quad (6)$$

2. $\mathcal{M}$ saves the certificate $\delta_m = (sk_m, \sigma_m)$.

δ_m is the original certificate belonging to the merchant, and in the other protocols, we use the randomized certificate $\sigma_m{}'$ to prove the legitimacy of the merchant as a payee.

4.5.3. Spend Protocol

In this protocol, both the user and the merchant use the randomized signature to identify themselves, and the signatures is different after randomization in each transaction, so the link between each transaction is cut off. And the coin counter *J* is shown to the merchant in order to construct the proof of knowledge efficiently. The interactions between the user $\mathcal{U}$ and the merchant $\mathcal{M}$ are following:

1.$\mathcal{M}$ proves knowledge of sk_m to $\mathcal{U}$ with σ_m', where σ_m' denotes the randomized signature from $\mathcal{B}$.

2.$\mathcal{U}$ computes $R = H(\sigma_m || info)$, where $info \in \{0,1\}^*$ is provided by the merchant. $\mathcal{U}$ randomly selects $J \in [0, 2^l - 1]$ and *J* hasn't used before.

3.$\mathcal{U}$ sends a coin serial number and a security tag to the merchant:

$$S = F_{g,s}^{DY}(J) = g^{\frac{1}{J+s+1}}, \quad (7)$$

$$T = pkuF_{g,t}^{DY}(J)^R = g^{u+\frac{R}{J+t+1}} \quad (8)$$

4.$\mathcal{U}$ sends *J* and a ZKPOK Φ of (sk_u, s, t, σ) such that:

- $S = F_{g,s}^{DY}(J)$
- $T = pkuF_{g,t}^{DY}(J)^R$
- $VerifySig(pk_B, (sku, s, t), \sigma_u') = true$

5.If Φ verifies and $\mathcal{U}$ is honest, $\mathcal{M}$ accepts the coin $(S, (R, T, \Phi))$.

6.$\mathcal{U}$ records his counter *J* is used.

Recall that there are two kinds of wallet in our scheme, if $\mathcal{U}$ uses the wallet containing only one coin as the wallet containing 2^l coins, $\mathcal{M}$ will determine that the user is dishonest and reject this transaction.

4.5.4. Deposit Protocol

In this protocol, the interactions between the merchant $\mathcal{M}$ and the bank $\mathcal{B}$ are following:

1. $\mathcal{M}$ proves knowledge of sk_m to $\mathcal{B}$ with $\sigma_m{}'$.
2. $\mathcal{M}$ sends $\mathcal{B}$ the coin $(S, \pi = (R, T, \Phi))$ bound to $\sigma_m{}'$.
3. If Φ verifies and the pair (S, π) is not already in spent coin list (i.e., the coin is not used before), then $\mathcal{B}$ adds (S, π) to spent coin list and credits $\mathcal{M}'$account.

4.5.5. Transfer Protocol

$Transfer$ protocol can be seen as a special variation of $Withdraw$ protocol and the interactions between the merchant $\mathcal{M}$ and the bank $\mathcal{B}$ are following:

1. $\mathcal{M}$ proves knowledge of sk_m to $\mathcal{B}$ with $\sigma_m{}'$
2. $\mathcal{M}$ sends $\mathcal{B}$ the coin $(S, \pi = (R, T, \Phi))$ bound to $\sigma_m{}'$.
3. If Φ verifies and the pair (S, π) is not already in spent coin list (i.e., the coin is not used before), then $\mathcal{B}$ adds (S, π) to spent coin list and allows $\mathcal{M}$ to transfer the coin.
4. $\mathcal{M}$ selects random value $s', t \in \mathbb{Z}_q$ and sends $\mathcal{B}$ a commitment as shown in **(3)**, $\mathcal{B}$ sends a random $r' \in \mathbb{Z}_q$. $\mathcal{U}$ computes $s = s' + r'$. $\mathcal{U}$ and $\mathcal{B}$ respectively computes A as shown in

(4).

5. $\mathcal{M}$ runs the CL signature protocol with $\mathcal{B}$ and obtains $\mathcal{B}$'s signature on secret values and the signature is:

$$\sigma_m = (a, a^y, a^x A^{\alpha xyl}, A_1, A_2, A_3, B_1, B_2, B_3) \quad (9)$$

6. $\mathcal{M}$ saves the wallet $W = (sk_m, s, t, \sigma_m)$, here the new wallet will only contain a coin, and the bank doesn't record the account for pk_m.

The structure of σ_m in this protocol is different from σ_u in $Withdraw$ protocol, we do this to distinguish the two kinds of wallet.

4.6. Identify Algorithm

Suppose (S, π_1) is in spent coin list for the number S. And the bank accepts a coin with (S, π_2) ,then the bank can identity the double-spender

$$pk = \left(\frac{{T_2}^{R_1}}{{T_1}^{R_2}}\right)^{(R_1 - R_2)^{-1}}. \quad (10)$$

Suppose coin S belongs to the user with $pk_u = g^u$, then each T_i constructed as (8). As the bank runs this algorithm with different R (i.e., $R_1 \neq R_2$), then the bank can compute the double-spender's public key:

$$\left(\frac{{T_2}^{R_1}}{{T_1}^{R_2}}\right)^{(R_1 - R_2)^{-1}} = \left(\frac{{g_3}^{uR_1 + R_1 R_2 \alpha}}{{g_3}^{uR_2 + R_1 R_2 \alpha}}\right)^{(R_1 - R_2)^{-1}} = {g_3}^{\frac{u(R_1 - R_2)}{(R_1 - R_2)}} = {g_3}^u = pk_u. \quad (11)$$

The output of this algorithm is a proof $\prod_G = (\pi_1, \pi_2)$ and public key of the double-spender pk_U.

4.7. VerifyGuilt Algorithm

Recall that everyone can call this algorithm to verify whether the user is a double-spender. Let another user $\mathcal{V}$ parse the proof $\prod_G$ as (π_1, π_2) and each π_i as (R_i, T_i, Φ_i), then run $Identify(params, S, \pi_1, \pi_2)$ and compare its output to pk_u given as input. If these values match, then $\mathcal{V}$ verifys each Φ_i with respect to (S, R_i, T_i). If all checks pass, $\mathcal{V}$ accepts that the user with pk_u is a double-spender; otherwise, deny that the user is a double-spender.

5. Security Analysis for the Proposed E-cash System

5.1. Balance

***Part* 1**. Let $\mathcal{A}$ be an adversary who executes f $Withdraw$ or $Transfer$ protocols with an extractor $\mathcal{E}$ acting as the bank. Suppose that the proof of knowledge is done interactively, and let $\mathcal{E}'$ denote the extractor of the proof of knowledge in CL signature, and for each $Withdraw$ protocol, $\mathcal{E}$ acts as an honest bank, except during the proof of knowledge where $\mathcal{E}$ runs the code of $\mathcal{E}'$ to extract (u, s, t). At the end of each $Withdraw$ execution, $\mathcal{E}$ outputs $(state_{\mathcal{E}}, A, w) \in l_S$, where A is a commitment computed by $PedCom(u, s, t; r)$. Let $A_f = \{S_{i,j} | 1 \leq i \leq f, 1 \leq j \leq n\}$ denote the set of serial numbers after f $Withdraw$ executions. And the proof of knowledge in the $Withdraw$ protocol is non-interactive, so both $\mathcal{E}$ and $\mathcal{E}'$ must be given control over the random oracle and their access model to $\mathcal{A}$ must be black-box.

***Part* 2**. Recall that A_f contains all the valid serial numbers produced by $\mathcal{A}$. Thus, if $\mathcal{A}$ wants to win the game, he must make $\mathcal{B}$ to accept a coin where $S \notin A_f$ with non-negligible probability. Now suppose $\mathcal{A}$ convinces $\mathcal{B}$ to accept the invalid coin (S, π) during $Deposit$ or $Transfer$ protocol. Then $\mathcal{A}$ must have concocted a false proof Φ that meets the following conditions: (1) $\mathcal{A}$ knows a signature from $\mathcal{B}$ on the values(u, s, t) corresponding to the invalid

coin and (2) that S and T are correctly formed. Due to the security of CL signature, we know case (1) only happens with negligible probability $v_1(k)$. Case (2) happens with negligible probability $v_2(k)$ due to the discrete logarithm assumption and the security of DYPRF[17]. Thus, $\mathcal{A}$ wins with only negligible probability $v_1(k) * v_2(k)$.

5.2. Identifying of double-spenders

We need to point out the case in which this property does not apply well. Suppose that the bank issues CL signatures on wallet secrets s_1 and s_2 belong to different users, such that $|s_1 - s_2| < 2^l$,the two users can both generate a valid spending proof for some coin $S = f_{g,s'}^{DY}(s - s' + 1) = g^{1/(s+1)}$ and there will be two valid coin in the bank's spent coin list as (S, π_1) and (S, π_2). However, the probability of that happening is $2^{l+1}/q$ which is negligible. We refer readers to [1] for more details about this.

As defined in balance, let $\mathcal{E}$ be the extractor interacting with $\mathcal{A}$ during the $Spend$ protocol to extract $A_f = \{S_{i,j} | 1 \le i \le f, 1 \le j \le n\}$. Now, suppose the merchant $\mathcal{M}$ has received two coins (S, π_1) and (S, π_2) for some $S \epsilon A_f$ and deposits them to the bank $\mathcal{B}$ and $\mathcal{B}$ parses each π_i as (R_i, T_i, Φ_i). Since $\mathcal{M}$ chooses R randomly while executing $Spend$ protocol with the user and $R_1 \neq R_2$ with high probability. If each $T_i = pk_{\mathcal{U}} f_{g,t}^{DY}(J + 1)^R$ has the same pk_u, J and t, then the algorithm $Identify$ (ζ, S, π_1, π_2) will recover pk_u successfully due to the correctness of the algorithm.

Since R_1, R_2 are computed by the hash function, and uniquely fixes T_1 and T_2 as the only valid security tags in the two transactions. To deviate from these tags during $Spend$, $\mathcal{A}$ must concoct the proof Φ which happens with negligible probability (which has been discussed in Balance).

Recall that $Identity$ can output the public key of double-spender pk_U along with proof $\prod_G$ that he has double-spent coin S. Since the essence of $VerifyGuilt$ is to run the algorithm $Identify$ again, and then compare the output and $pk_{\mathcal{U}}$ given as input and $VerifyGuilt$ will always accept the output of $Identify$.

5.3. Anonymity of users

The simulator $\mathcal{S}$ inputs the global parameters, some additional information $auxsim$, and capacity of a valid wallet n. $\mathcal{S}$ is controlled by the random oracle and the input-output model to the adversary $\mathcal{A}$. $\mathcal{S}$ executes $Spend$ protocol with $\mathcal{A}$ as follows:

1. $\mathcal{A}$ proves knowledge of his secret key sk to $\mathcal{S}$.
2. $\mathcal{S}$ receives $info \epsilon \{0,1\}^*$ and the transaction information, then computes $R \in \mathbb{Z}_q^*$.
3. $\mathcal{S}$ chooses values $u, s, t \epsilon \mathbb{Z}_q$ randomly and selects J within [0, n), then computes the serial number S and the security tag T.
4. $\mathcal{S}$ sends $\mathcal{A}$ the coin (S, π),where $\pi = (R, T, \Phi)$, and Φ is a simulated validity proof which is constructed as follows:
   (a)$A = PedCom(u)$ $B = PedCom(s)$, $C = PedCom(t)$ and a simulated signature proof that $\mathcal{S}$ knows the values (u, s, t) signed by the bank. ($\mathcal{S}$ invokes the simulator in CL Signature).
   (b) a proof Γ that S and T are computed correctly as **(7)** and **(8)**.

Observe that the invoked simulator handles the difficulty of $\mathcal{S}'s$ job.

We now explain why $\mathcal{A}$ is unable to distinguish between the output of $\mathcal{S}$ and that of a real user. First, $\mathcal{A}$ did not learn anything meaningful about (u, s, t) in the $Withdraw$ protocol due to the security of CL signature. Thus, $\mathcal{A}$ cannot distinguish between s and t chosen by $\mathcal{S}$ and

those chosen by the user. Due to security of the DYPRF[17], S and T generated by $\mathcal{S}$ are computationally indistinguishable from the elements in G, and thus, equally to those generated by real users with any valid value J. Finally, the only difference between $\mathcal{S}'s$ Φ and that of a real user is the simulated signature proof. However, we construct this proof by running CL Signature, so $\mathcal{A}$ distinguishes between Game R and Game I with negligible probability based on the bilinear map under the assumption called LRSW.

5.4. Anonymity of merchants

Suppose that the simulator has a received coin(S, π). We consider the anonymity of merchants in $Spend$, and $Transfer$ protocols. The Simulator executes $Spend$ and $Transfer$ with $\mathcal{A}$ as follows:

1.$\mathcal{S}$ randomly chooses $u \in \mathbb{Z}_q$, then sends the simulated signature and a simulated proof of knowledge from $\mathcal{B}$.
2. $\mathcal{S}$ sends an optional transaction string $info$.
3. $\mathcal{S}$ receives the coin (S, π).
4. $\mathcal{S}$ sends the simulated signature proof and the coin (S, π) to the bank.
5. $\mathcal{S}$ selects random value $s', t \in \mathbb{Z}_q$ and sends $\mathcal{B}$ a commitment **(3).** $\mathcal{B}$ sends a random $r' \in \mathbb{Z}_q$ and $\mathcal{S}$ computes $s = s' + r'$. $\mathcal{S}$ and $\mathcal{B}$ locally compute **(4)**.
6. $\mathcal{S}$ runs the CL signature protocol with $\mathcal{B}$ and obtains $\mathcal{B}$'s signature.
7. $\mathcal{S}$ saves the wallet $W = (sku, s, t, \sigma)$.

Observe above that the only difference between the output of $\mathcal{S}$ and that of a real merchant is the simulated proof. However, we ran the CL signatures to make the $\mathcal{A}$ can't distinguish the simulated one or the real one. So $\mathcal{A}$ distinguishes between Game R and Game I with only negligible probability under LRSW assumption.

5.5. Exclupability

The exclupability definition requires that no adversary could produce a serial number and a proof of guilt to slander an honest as a double-spender (i.e.,$VerifyGuilt$ accepts the serial number and the proof with negligible probability).

Recall that the proof $\prod = (\pi_1, \pi_2)$ for coins (S, π_1) and (S, π_2) is guilt of the user pk_u. And any valid π, involves the proof to prove the knowledge of the user's secret key sk_u with different randomized signature. If $\mathcal{A}$ wants to win, there are two situations need to be discussed as follows: (1)$\mathcal{A}$ is successful at producing a false $\prod$ or (2) if (S, π_1) and (S, π_2) are both valid coins of different users. In the first instance, $\mathcal{A}$ successfully concocts a false proof $\prod$ means that $\mathcal{A}$ can forge the underlying proof of knowledge which happens with only negligible probability; And in the second instance, that means (S, π_1) and (S, π_2) are registered to different users which we have discussed in section 5.2. And $VerifyGuilt$ will deny the user with pk_u as input is a double-spender, because the $Identify$ algorithm will recover a valid public key same as pk_u with negligible probability. Thus $\mathcal{A}$ wins with only negligible probability.

6. Efficiency Analysis

6.1. Storage Efficiency of E-Cash Systems

The storage cost of different E-Cash schemes are presented in this section. In order to put all schemes at the same security level, we require the order of the cyclic group G is 1024- bit in [1]

and [8]. The prime order p of G_1 and G_2 in bilinear map is 160-bit in [1], [25] and our scheme. we select $l = 10$ in [1], [8]and our scheme. For more intuitive comparison, we put total storage space of each protocol in **Table 1**.

6.2. Computation Efficiency of E-Cash Systems

Bilinear pairings and multi-based exponentiations of each protocol are listed in the **Table 2**, which are the main computation operations in E-cash scheme. In order to compare different schemes conveniently, slight computations are neglected uniformly. To evaluate the time required for each protocol, we test the time of bilinear map and single-based exponentiation using the JAVA programming language with the JPBC Library (jpbc-2.0.0). All tests are implemented on a laptop computer with the following features: (1) CPU: AMD Ryzen 5 4600U; (2) Physical Memory: 16GB; and (3) OS: Windows 10

Table 1. Storage Space Comparison

	Our scheme	[1]			[25]	[8]
		System I	System II	System III		
Withdraw [bit]	2730	5632	$6112 + 3lx$	6112	2048	5828
Sign [bit]	1760	-	-	-	-	-
Spend [bit]	10890	15488	$(21l + 12)x$	12032	12640	5190
Deposit [bit]	10890	15488	$(21l + 12)x$	12032	$12640+2^n \cdot 160$	5190
$Transfer$ [bit]	12970	-	-	-	-	-

l: the coin counter's bit quantity; **x**: in bilinear groups it is 160, and in RSA group, it is 1024; **n**: denotes how many parts one coin can be divided into. -: not available.

Table 2. Computation Cost Comparison

		Our scheme	[1]			[25]	[8]
			System I	System II	System III		
Withdraw	M	11	10	$8k + 10$	$8k + 10$	11	10
	P	0	0	$3k$	$3k$	7	0
Sign	M	3	-	-	-	-	-
	P	0	-	-	-	-	-
Spend	M	17	34	$72l + 58$	34	27	12
	P	20	0	0	0	27	0
Deposit	M	4	11	$21l + 17$	11	6	5
	P	7	0	0	0	$2^n + 14$	0
$Transfer$	M	14	-	-	-	-	-
	P	7	-	-	-	-	-

M: multi-based exponentiation operation; **P**: bilinear pairing maps; -: not available; **k**:is a system parameter, which denotes the cheating probability 2^{-k} at most; **n**: how many parts one coin can be divided into; ***l***: the coin counter's bit quantity.

We test the single-based exponentiation 30 times, and take the average 0.965 ms. And the bilinear map is also tested 30 times, the average value is about 7.608 ms. And if the algorithm

is well constructed, it will not take far more time to compute the multi-based exponentiation than the single-based exponentiation[8]. And normally a multi-based exponentiation takes only 10 more percent time if the multi-exponentiation multiplies up to 3 exponentiations. So, it can be evaluated that the time of one multi-based exponentiation operation is about 1.061ms. So, we can simply calculate the time of each protocol in our scheme without considering other negligible computations and communication time. The evaluated time of each protocol is in **Table 3**.

Table 3. Evaluated time of each protocol in our scheme

Protocol	Withdraw	Sign	Spend	Deposit	Transfer
Time(ms)	11.671	3.183	170.197	57.5	68.11

6.3. Comparison among E-Cash Systems

The comparison among the E-cash schemes is presented in the **Table 4**, and we can see that our scheme, [22] and three systems in [1] are all compact E-cash schemes and the computation complexity of wallet is all $\mathcal{O}(1)$ or $\mathcal{O}(k)$. Compared with [8] and System I and System II in [1], our scheme cannot trace double-spender's unspent coins efficiently, but its unique properties makes it also competitive which are:

1. the merchants can keep anonymous;
2. the coin received can be converted into a new coin anonymously without the merchant's account.

Table 4. The Function Comparison among E-Cash Schemes

Property	Our scheme	[1]			[25]	[8]
		System I	System II	System III		
Compact E-cash	Yes	Yes	Yes	Yes	No	Yes
Tracing double-spender	Yes	Yes	Yes	Yes	Yes	Yes
Tracing double-spender's coins	No	No	Yes	Yes	No	Yes
Users are anonymous	Yes	Yes	Yes	Yes	Yes	Yes
Merchants are anonymous	**Yes**	**No**	**No**	**No**	**No**	**No**
Anonymously transfer	**Yes**	**No**	**No**	**No**	**No**	**No**
Wallet's space complexity	$\mathcal{O}(l)$	$\mathcal{O}(l)$	$\mathcal{O}(l)$	$\mathcal{O}(l)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Computation complexity of withdrawing	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

-: not available; **k**: the security parameter, which means the cheating probability is 2^{-k} at most; **n**: how many parts which one coin can be divided into.

7. Conclusion

After studying a lot of literature, we notice that no previous research pay attention to the privacy of merchants in E-cash system. So, we introduce the concept of double-blind to make both users and merchants anonymous. And based on the anonymity of merchants, the merchant can transfer the coin he received into a new coin for next spending without revealing the account to the bank. The proposed compact E-cash scheme satisfies Balance, Identifying double-spender, Anonymity of users and merchants and Exculpability under LRSW and y-DDHI assumption. Our solution to the privacy protection of merchants is also useful for other similar E-cash schemes, such as divisible E-cash and transferable E-cash.

Appendix

In the Appendix, we put the details about how to compute the signature and the two protocols $Spend$ and $Transfer$.

A.1 Signature Construction and Verification

A signature of the bank on values (u, s, t) contained in commitment $A = PedCom(u, s, t; r) = g_1^r Z_1^u Z_2^s Z_3^t$ consists of $(a, b, c, A_1, A_2, A_3, B_1, B_2, B_3)$ and the bank generates the signature as follow:

1. $\alpha \leftarrow Z_q, a = g_1^\alpha$.
2. For $1 \le i \le 3$, let $A_i = a^{z_i}$ and set $b = a^y$, for $1 \le i \le 3$, $B_i = A_i^y$.
3. $c = a^x M^{\alpha x y}$.

The zero-knowledge proof protocol for the signed secret values (u, s, t) are following:

1. The user $\mathcal{U}$ computes a blinded version of the signature σ as follows:

Choose $r, r' \in Z_q$ randomly and form $\tilde{\sigma} = \left(\tilde{a}, \{\widetilde{A_\iota}\}, \tilde{b}, \{\widetilde{B_\iota}\}, \tilde{c}\right)$ as follows:

$$\tilde{a} = a^r, \tilde{b} = b^r, \tilde{c} = c^r,$$
$$\widetilde{A_\iota} = A_i^r, \widetilde{B_\iota} = B_i^r \ \ for\ 1 \le i \le 3.$$

Then blind $\tilde{\boldsymbol{c}}$ and obtain $\hat{c}$ distributed independently: $\hat{c} = \tilde{c}^{r'}$.

Send $\left(\tilde{a}, \{\widetilde{A_\iota}\}, \tilde{b}, \{\widetilde{B_\iota}\}, \hat{c}\right)$ to the merchant $\mathcal{M}$.

2. Let $v_x, v_{xy}, v_{(xy,i)}\ i = 1,2,3\ and\ v_s$ be as follows:

$$v_x = e(X, \tilde{a}) \quad v_{xy} = e\left(X, \tilde{b}\right)$$
$$v_{(xy,i)} = e\left(X, \widetilde{B_\iota}\right) \quad v_s = e(g, \hat{c})$$

$\mathcal{U}$ and $\mathcal{M}$ compute the above equation locally and then $\mathcal{U}$ constructs and sends the following zero-knowledge proof to $\mathcal{M}$:

$$PK\{(\mu_0, \mu_1, \mu_2, \mu_3, \rho): (v_s)^\rho = v_x (v_{xy})^{\mu_0} \prod_{i=1}^{3} (V_{(xy,i)})^{\mu_i}\}$$

$\mathcal{V}$ accepts if the proof above verifies and the follows equation verifies:

(a)$\{\widetilde{A_\iota}\}$ were correctly formed:$e(\tilde{a}, Z_i) = e(g, \widetilde{A_\iota})\ 1 \le i \le 3$

(b) $\tilde{b}\ and\ \widetilde{B_\iota}\ were\ formed\ correctly$: $e(\tilde{a}, Y) = e\left(g, \tilde{b}\right)\ e\left(\widetilde{A_\iota}, Y\right) = e(g, \widetilde{B_\iota})\ \ 1 \le i \le 3$

The signature gets from the bank in $Sign$ protocol is (a, b, c, A_1, B_1) which is only on the merchant's private key.

A.2. Spend protocol

Recall that $\mathcal{U}$ obtains the signature $(a, b, c, A_1, A_2, A_3, B_1, B_2, B_3)$ from the bank on the value (u, s, t), and the merchant has obtained a signature on his sk_m.

In the $Spend$ protocol, not like that in the compact E-cash scheme, we send the value J to the verifier rather than prove the knowledge of it in the zero-knowledge proof. Recall that there are two kinds of wallet in the Spend protocol, suppose the wallet containing 2^l coins is W and the wallet containing only one coin is W'.

1. $\mathcal{M}$ sends $\mathcal{U}$ a string $info \epsilon \{0,1\}^*$ containing the transaction information and proves knowledge of sk_m to identify himself.

$$PK\{(\mu_0, \rho): (v_s)^\rho = v_x \left(v_{xy}\right)^{\mu_0} v_{(xy,1)}^{\mu_1}\}$$

2. $\mathcal{U}$ computes R from the hash function H.
3. $\mathcal{U}$ computes $S = g^{1/(J+s)}$ and $T = pk_U g^{R/(J+t)}$.
4. $\mathcal{U}$ chooses random value $r_A \in \mathcal{Z}_q$, and computes the commitment $A = g_1^u h_1^{r_A}$, The user

chooses $r, r' \epsilon \mathcal{Z}_q$.Note that $(\tilde{a}, \{\widetilde{A_i}\}, \tilde{b}, \{\widetilde{B_i}\}, \tilde{c})$ is also valid on (u, s, t) but statistically independent to the original signature.

5. $\mathcal{U}$ computes the proof of knowledge as follows:

$$PK\{(\mu_0, \mu_1, \mu_2, \mu_3, \rho, \gamma_1, \gamma_2, \lambda): (v_s)^\rho = v_x(v_{xy})^{\mu_0} \prod_{i=1}^{3} (V_{(xy,i)})^{\mu_i} \wedge A = g_1^{\mu_1} h_1^{\gamma_1}$$
$$\wedge g = S^{\mu_2} S^{J+1} \wedge 1 = A^{\mu_3} A^{J+1} (1/g_1)^{\lambda} h_1^{\gamma_2} \wedge g^R = T^{\mu_3} T^{J+1} (1/g_1)^{\lambda}\}$$

6. If the proof verifies, $\mathcal{M}$ accepts the coin.

7. $\mathcal{U}$ updates his counter J=J+1.

The steps above belong to the wallet W, as for the wallet W', the PK in the step 5 is as follow:

$$PK\{(\mu_0, \mu_1, \mu_2, \mu_3, \rho, \gamma_1, \gamma_2, \lambda): (v_s)^\rho = [v_x(v_{xy})^{\mu_0} \prod_{i=1}^{3} (V_{(xy,i)})^{\mu_i}]^l \wedge A = g_1^{\mu_1} h_1^{\gamma_1}$$
$$\wedge g = S^{\mu_2} S^{J+1} \wedge 1 = A^{\mu_3} A^{J+1} (1/g_1)^{\lambda} h_1^{\gamma_2} \wedge g^R = T^{\mu_3} T^{J+1} (1/g_1)^{\lambda}\}$$

We add the l to the signature to prevent the user spend the W' as the W. After the merchant receives the proof, he will verify whether the wallet is W'.

A.3. Transfer protocol

The $Transfer$ protocol allows the merchant convert the coin he has received into a new coin. The steps are follows:

1. The merchant sends the coin, and his blinded signature in the $Spend$ protocol. The bank check the cain and whether this coin is sent to the merchant.

2. The merchant identifies himself by proving knowledge of sk_m.And the signature sent to the user is blinded, so the bank and user cannot link it with the merchant's pk_m.

$$PK\{(\mu_0, \mu_1, \rho): (v_s)^\rho = v_x(v_{xy})^{\mu_0} v_{(xy,1)}^{\mu_1}\}$$

3. $\mathcal{U}$ selects random value $s', t \in \mathbb{Z}_q$ and sends $\mathcal{B}$ a commitment

$$A' = PedCom(sk_u, s', t; r) = {g_1}^r {Z_1}^u {Z_2}^{s'} {Z_3}^t.$$

$\mathcal{B}$ sends a random $r' \in \mathbb{Z}_q$. $\mathcal{U}$ computes $s = s' + r'$. $\mathcal{U}$ and $\mathcal{B}$ respectively computes

$$A = {Z_2}^{r'} A' = PedCom(sk_u, s' + r', t; r) = PedCom(sk_u, s, t; r).$$

4. $\mathcal{U}$ runs the CL signature protocol with $\mathcal{B}$ and obtains $\mathcal{B}$'s signature, the zero-knowledge sent to $\mathcal{B}$ is following:

$$PK\{(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4, \rho): (v_s)^\rho = v_x(v_{xy})^{\mu_0} v_{(xy,1)}^{\mu_1} \wedge A = {g_1}^{\mu_4} {Z_1}^{\mu_1} {Z_2}^{\mu_2} {Z_3}^{\mu_3}\}$$

This PK is to make sure that the bank signs on the merchant's $sk_m$and $\mathcal{U}$ obtains σ_m:

$$\sigma_m = (a, a^y, a^x A^{\alpha xyl}, A_1, A_2, A_3, B_1, B_2, B_3)$$

5. $\mathcal{U}$ saves the wallet $W = (sku, s, t, \sigma)$.

References

[1] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact E-Cash," 060, 2005. Accessed: Apr. 10, 2021. [Online]. Available: http://eprint.iacr.org/2005/060

[2] M. H. Au, Q. Wu, W. Susilo, and Y. Mu, "Compact E-Cash from Bounded Accumulator," *Topics in Cryptology – CT-RSA 2007*, Berlin, Heidelberg, pp. 178–195, 2006. Article (CrossRef Link)

[3] M. H. Au, W. Susilo, and Y. Mu, "Practical Compact E-Cash," in *Proc. of Australasian Conference on Information Security and Privacy*, pp. 431–445, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007. Article (CrossRef Link)

[4] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, “Compact E-Cash and Simulatable VRFs Revisited,” in *Proc. of Pairing-Based Cryptography – Pairing 2009*, pp. 114–131, Third International Conference Palo Alto, CA, USA, August 12-14, 2009. Article (CrossRef Link)
[5] Q. Wang, “Compact k-spendable E-cash with anonymity control based offline TTP,” *International Journal of Innovative Computing, Information and Control*, 7(1), 459-469, 2011. Article (CrossRef Link)
[6] P. Märtens, “Practical Compact E-Cash with Arbitrary Wallet Size,” *Cryptology ePrint Archive*, 2015. Article (CrossRef Link)
[7] B. Lian, G. Chen, J. Cui, and D. He, “Compact E-Cash with Practical and Complete Tracing,” *KSII TIIS*, vol. 13, no. 7, pp. 3733-3755, Jul. 2019. Article (CrossRef Link)
[8] B. Lian, G. Chen, J. Cui, and M. Ma, “Compact E-Cash with Efficient Coin-Tracing,” *IEEE Trans. Dependable and Secure Comput.*, vol. 18, no. 1, pp. 220–234, Jan. 2021. Article (CrossRef Link)
[9] M. H. Au, W. Susilo, and Y. Mu, “Practical Anonymous Divisible E-Cash from Bounded Accumulators,” in *Proc. of International Conference on Financial Cryptography and Data Security*, pp. 287–301, 12th International Conference, FC 2008, Cozumel, Mexico, January 28-31, 2008. Article (CrossRef Link)
[10] S. Canard and A. Gouget, “Divisible E-Cash Systems Can Be Truly Anonymous,” in *Proc. of Advances in Cryptology - EUROCRYPT 2007*, pp. 482–497, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Article (CrossRef Link)
[11] W.-S. Juang and H.-T. Liaw, “A practical anonymous multi-authority e-cash scheme,” *Applied Mathematics and Computation*, vol. 147, no. 3, pp. 699–711, Jan. 2004. Article (CrossRef Link)
[12] F. Baldimtsi, M.Chase, G. Fuchsbauer, and M.Kohlweiss, “Anonymous Transferable E-Cash,” in *Proc. of Public-Key Cryptography -- PKC 2015*, pp. 101–124, 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 -- April 1, 2015. Article (CrossRef Link)
[13] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” *Decentralized Business Review*, 21260, 2008. Article (CrossRef Link).
[14] I. Miers, C. Garman, M. Green and A. D. Rubin, "Zerocoin: Anonymous Distributed E-Cash from Bitcoin," in *Proc. of 2013 IEEE Symposium on Security and Privacy*, pp. 397-411, 2013. Article (CrossRef Link)
[15] E. Ben Sasson et al., "Zerocash: Decentralized Anonymous Payments from Bitcoin," in *Proc. of 2014 IEEE Symposium on Security and Privacy*, pp. 459-474, 2014. Article (CrossRef Link)
[16] D. Boneh and X. Boyen, “Short Signatures Without Random Oracles,” in *Proc. of Advances in Cryptology - EUROCRYPT 2004*, pp. 56–73, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Article (CrossRef Link)
[17] Y. Dodis and A. Yampolskiy, “A Verifiable Random Function with Short Proofs and Keys,” in *Proc. of Public Key Cryptography - PKC 2005*, pp. 416–431, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005. Article (CrossRef Link)
[18] J. Camenisch and A. Lysyanskaya, “Signature Schemes and Anonymous Credentials from Bilinear Maps,” in *Proc. of Advances in Cryptology – CRYPTO 2004*, pp. 56–72, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Article (CrossRef Link)
[19] D. Boneh, B. Lynn, and H. Shacham, “Short Signatures from the Weil Pairing,” in *Proc. of Advances in Cryptology — ASIACRYPT 2001*, pp. 514-532, 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001. Article (CrossRef Link)
[20] J. Camenisch, “Group signature schemes and payment systems based on the discrete logarithm problem,” Ph.D. dissertation, Dept. Tech. Sci., ETH Zurich., Swiss, Switzerland 1998.

[21] F. Boudot, "Efficient Proofs that a Committed Number Lies in an Interval," in *Proc. of Advances in Cryptology — EUROCRYPT 2000*, pp. 431–444, International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000. Article (CrossRef Link)
[22] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A Practical and Provably Secure Coalition-Resistant Group Signature Scheme," in *Proc. of Advances in Cryptology — CRYPTO 2000*, pp. 255-270, 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000. Article (CrossRef Link)
[23] T. P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *Proc. of Advances in Cryptology — CRYPTO '91*, pp. 129–140, 1992. Article (CrossRef Link)
[24] J. Camenisch and A. Lysyanskaya, "A Signature Scheme with Efficient Protocols," in Security in Communication Networks, pp. 268–289, Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002. Article (CrossRef Link)
[25] S. Canard, D. Pointcheval, O. Sanders, and J. Traoré, "Divisible e-cash made practical," *IET Information Security*, vol. 10, no. 6, pp. 332–347, Nov. 2016. Article (CrossRef Link)

Jiyang Chen: received the B.S. degree in electronic information engineering from Zhejiang Sci-Tech University, Hangzhou, China, in 2020. He is currently a M.S. student in the college of information science and electronic engineering, Zhejiang University, Hangzhou, China. His current research interests include cryptographic protocols, secure multi-party computation, differential privacy and federal learning.

Bin Lian: received the M.S. degree in cryptography from Southwest Jiaotong University in 2005 and the Ph.D. degree in cryptography from Shanghai Jiao Tong University in 2015. He is currently an associate professor at NingboTech University. His current research interests include cryptographic protocols, information security system design, big data application security and industrial internet security.

Yongjie Li: received his B.S. and Ph.D. degree in electrical engineering from the College of Electrical Engineering, Zhejiang University, Hangzhou, China, in 2012 and 2017, respectively. He is currently a lecturer in the School of Information Science and Engineering, NingboTech University, Ningbo, China. His research interests include power system small-signal stability analysis and control, energy management of microgrids, and application of parallel computing.

Jialin Cui: received the M.S. degree in automation from Zhejiang University in 2005. He is currently an associate professor at NingboTech University. His current research interests include machine vision, artificial intelligence and application of artificial intelligence technology in information security.

Ping Yu: received the Ph.D. degree in mircoelectronics from Zhejiang University, Hangzhou, China, in 2013. He is currently with Ningbo Tech University, Ningbo, China. His current research interests include design, modeling, and implementation of trusted embed computation.

Zhenyu Shu: earned his PhD degree in 2010 at Zhejiang University, China. He is now working as a professor at NingboTech University. His research interests include computer graphics, digital geometry processing, and machine learning. He has published over 30 papers in international conferences or journals.

Jili Tao: received the B.Sc. degree in communication engineering and M.Sc. degree in traffic information engineering and control from Central South University, Changsha, China, in 2001 and 2004, respectively, and the Ph.D. degree in control science and control engineering from Zhejiang University, Hangzhou, China, in 2007. She is currently a Professor with the Ningbo Institute of Technology, Zhejiang University, Ningbo, China. Her research interests include intelligent optimization, modeling and its applications to electronic system design and control system design.