MIMICKING A COMPLETE LIFE-CYCLE OF FIAT CURRENCY IN ONE E-CASH SYSTEM

Peifang Ni[1, 2]

[1]TCA Laboratory, Institute of Software,
Chinese Academy of Sciences Beijing, China
[2]State Key Laboratory of Cryptology, Beijing, China

ABSTRACT

The electronic cash was introduced by Chaum in 1982 and now many e-cash systems have been proposed in order to mimic the fiat currency. Bitcoin provides us with an attractive way to construct a decentralized e-cash system. Ideally, we would like to make the system more practical, for example, the users can be able to transfer coins between each other multiple times and they can also withdraw arbitrary amount of coins rather than one or the predefined number, so that in the spend protocol the user can spend any amount of valid coins.

In this paper, we propose a provably secure and more practical e-cash system. Firstly, it can provide the anonymous transfer of coins between users, so that the merchant can spend the received coins further; secondly, the user can withdraw arbitrary amount of coins rather than the one or predefined number; thirdly, during the transfer of coins, the coins have a fixed size; finally, the fair exchange between the users can also be achieved.

KEYWORDS

E-cash, fiat currency, coin, anonymous, preventing double-Spending, fair exchange.

1. INTRODUCTION

This document describes, and is written to conform to, author guidelines for the journals of AIRCC series. It is prepared in Microsoft Word as a .doc document. Although other means of preparation are acceptable, final, camera-ready versions must conform to this layout. Microsoft Word terminology is used where appropriate in this document. Although formatting instructions may often appear daunting, the simplest approach is to use this template and insert headings and text into it as appropriate.

Bitcoin [1] is the most prominent cryptocurrencies, whose security does not rely on any single trusted third party and the transaction ledger is publicly available. But now the cash is still the most prevalent payment method in real life because that cash transaction is anonymous, transferable and it can prevent the double-spending attack. Furthermore, there exists no unfair exchange when the users exchange with physical cash. Here we focus on the construction of a more practical e-cash system [2] that can simulate the physical currency better. It is known that the proof of coins' validity is based on some personal message, which is easy to break the users' privacy. Moreover, in the e-cash system, the user withdraws coins from the bank then sends it to the merchant who must deposits it to the bank rather than spends it further. And in the trustless network, the unfairness occurs between users, e.g., the malicious payer or payee can refuse to sending the valid messages.

David C. Wyld et al. (Eds): AIFZ, SIGPRO, NWCOM, CLSB, MLNLP, DTMN, BDIOT, CSITY - 2022
pp. 41-52, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121604

Obviously, the ideal e-cash system should be equipped with the same security properties of the physical cash. There has been number of works aim to solve these problems, but these works usually solve several problems and also bring some new problems, for example, using *malleable signature* to achieve the transitivity of coins [3], so the users can get a valid signature of another message according to some transformation without communicating with the bank, but this scheme brings another problem that the size of the coin is increasing during the process of transfer.

1.1. Our Contributions

In this paper, we present a provably secure and more practical e-cash system that is closet to the physical cash。In this system, the only trusted component is the blockchain and we achieve that:

- **transitivity:** the valid coins can be used further by the payee;
- **arbitrary amount of coins:** the user can withdraw arbitrary amount of valid coins from the bank;
- **prevent double-spending:**any user cannot spend a same coin twice;
- **fair exchange:** none of users can cheat in the trustless network.

1.2. Related Work

A complete e-cash system should satisfies: anonymity, transitivity, divisibility, the ability to prevent double-spending attacks and fair exchange among users. Now we describe the related works in the following aspects.

Transitivity means that, in e-cash system, the merchant (or payee) can spend the received coin further without depositing it to the bank firstly. Okamoto and Oha[4,5] are the first to propose transferable e-cash, but they only provided the weak anonymity. Then in 1992 [6] proved that the size of transferable coins is increasing during the transfer process. Baldimtsi [3] showed us the first transferable e-cash system that satisfies all of the anonymity properties and provides us with a new efficient double-spending detection mechanism, but the size of coins is increasing during the transfer process. The works [7,8] proposed a fully anonymous and transferable e-cash scheme that satisfies all the security properties, which can prevent double-spending attack with the help of blockchain. However, the size of coins is still increasing during the transfer process, and it presents a noneffective solution for this problem that, when the size reaches the upper bound, the user can deposit it to the bank and then the bank declares that this coin is invalid.

Divisibility means that the user can withdraw a unique coin of value $\leq 2^n$ from the bank and spend it in several times to some distinct merchants. Compact e-cash allows the user to withdraw a wallet from the bank that contains 2^n coins in the withdraw protocol [9,10], however, this scheme only allow the user to spend one coin each time, which makes the scheme unpractical. Canard proposed the first efficient divisible e-cash system secure in the standard model [11]. And Tewari showed us the e-cash system that the user can withdraw arbitrary denomination of coin [7], which must be spent in one time.

Fair exchange between users means that, in the trustless network, the malicious user may refuse returning the corresponding message when he learns some knowledge. Constructing contracts on blockchain provides us with a feasible method [12,13] and the works [14,15,16] show us how bitcoin can be used in the area of *MPC*, where the fairness means that the dishonest users will pay a fine for the honest ones as a compensation. In the e-cash system the participants may face the following problem that shall the buyer pays the merchant first then the merchant sends the buyer

service or the other way around ? Is there a way that none of the parties can cheat the other one? The works [17,18] used blockchain to achieve the fair exchanges among users.

1.3. Organization

In section 2, we review some classical definitions and notations. Section 3 describes the Bitcoin simply. In section 4, we present our constructions of the e-cash system. And the security proof and conclusion are showed in section 5 and 6 respectively.

2. PRELIMINARIES

2.1. Assumptions

Our construction is based on the following assumptions:

- The only ``trusted component'' is the Blockchain that each user has access to it (denoted by $Ledger$), and if a transaction is on the $Ledger$ then it means that the transaction is provably valid. And the communication channel between users and $Ledger$ is secure;
- The amount of coins that the user can withdraw is limited;
- The coin has $l \geq 1$different denominations (to correspond with the physical cash, it can have other definitions)$Val_1 > Val_2 > \cdots Val_l$ and the bank holds the corresponding signature key pairs denoted as $(pk_1, sk_1), \dots, (pk_l, sk_l)$;
- Each denomination has a fixed prefix and the form is Value, serial number, signature, i.e., $\mathrm{Coin}_1 = (\mathrm{Val}_1, \mathrm{SN}_1, \sigma_1)$ means that the denomination of Coin_1 is Val_1, its serial number is SN_1 and it can be provably valid with σ_1 and pk_1;

2.2. E-Cash system

As the traditional e-cash system [19,20], our scheme consists of two basic parties, the user U (or the merchant M) and bank B, and the following algorithms:

- $ParamGen(1^{\lambda})$: on input security parameter λ, the parameter generation algorithm outputs the system public parameters par. We assume that par is the default input to the remaining algorithms;
- $KeyGen((1^{\lambda}))$: the key generation algorithm is executed by U and B respectively, and outputs (pk_U, sk_U) and (pk_B, sk_B) ,where $(pk_B, sk_B) =: \left(\left(pk'_B, sk'_B\right), \left(pk''_B, sk''_B\right)\right) = \left(\left(pk'_B, sk'_B\right); (pk_1, sk_1), \dots, (pk_l, sk_l)\right)$;
- $Register(B[pk_B, sk_B], U[pk_U, sk_U])$: the registration protocol allows U to have an account in the bank and, at the end, both parties output ok for success or $\perp$ for the failure of registration;
- $Withdraw(B\left[pk''_B, sk''_B\right], U[pk_U, sk_U])$: the withdraw protocol allows U to get the coins with value Val, which can be provable with the fixed prefix and the signature of B;
- $Spend(U_1\left[coin, pk_{U_1}, sk_{U_1}\right], U_2\left[coin, pk_{U_2}, sk_{U_2}\right])$: the spend protocol allows user U_1 with $coin$ to buy the U_2's service ω;
- $Deposit(U[coin, pk_U, sk_U], B[pk_B, sk_B])$: the deposit protocol allows the user U to deposit $coin$ to his account held by$B$.

2.3. Digital Signature Scheme

A digital signature scheme consists of a triple of probabilistic polynomial-time algorithms $(Gen, Sign, Ver)$ satisfying the followings:

- Key-generator algorithm Gen: on input 1^{λ}, Gen outputs (pk, sk);
- Signing algorithm $Sign$: on input secret key sk and a message $\alpha \in \{0,1\}^*$, $Sign$ outputs signature σ;
- (Deterministic) Verifying algorithm Ver: on input public key pk, message $\alpha \in \{0,1\}^*$ and signature σ, Ver returns $output \in \{accept, reject\}$;
- For each key pair (pk, sk) in the range of $Gen(1^{\lambda})$ and message $\alpha \in \{0,1\}^*$, the algorithms $Sign$ and Ver satisfy

$$\Pr\left[Ver\left(pk, \alpha, Sign(sk, \alpha)\right)\right] = 1$$

where the probability is taken over the internal coin tosses of $Sign$ and Ver.

2.4. Public Encryption Scheme

A public encryption scheme consists of a triple of probabilistic polynomial-time algorithms $(EncGen, Enc, Dec)$ satisfying the followings:

- Key-generator algorithm $EncGen$: on input 1^{λ}, $EncGen$ outputs (sk, pk);
- Encryption algorithm Enc: on input public key pk and message $\alpha \in \{0,1\}^*$, Enc outputs a ciphertext $c \coloneqq Enc_{pk}(\alpha)$;
- Decryption algorithm Dec: On input private key sk, a ciphertext c, Dec outputs $\alpha' = Dec_{sk}(c)$.
- For each key pair (sk, pk) in the range of $EncGen\left(1^{\lambda}\right)$ and message $\alpha \in \{0,1\}^*$, the algorithms Enc and Dec satisfy:

$$\Pr\left[Dec_{sk}\left(Enc_{pk}(\alpha)\right) = \alpha\right] = 1$$

where the probability is taken over the internal coin tosses of algorithms Enc and Dec.

2.5. Zero-knowledge Proofs of Knowledge

In the $spend$ protocol, the user will use the ZKPoK to let the merchant believe that the $coin$ exactly belongs to him and the merchant uses $ZKPoK$ to let the user believe that he knows the service/witness ω.

A pair of interactive Turing machines $< P, V >$ is called an interactive proof system for a language L if machine V is polynomial-time and the following two conditions hold:

- Completeness: there exists a negligible function c such that for every $x \in L$,

$$\Pr[< P, V > (x) = 1] > 1 - c(|x|)$$

- Soundness: there exists a negligible function s such that for every $x \notin L$,

$$\Pr[< P, V > (x) = 1] < s(|x|)$$

$c(\cdot)$is called the completeness error and $s(\cdot)$ is the soundness error.

Zero-knowledge protocol is the interactive proof system with zero-knowledge property, which means that the prover can convince the verifier that some instance $x \in L$ without providing the verifier with any additional information beyond the fact.

A zero-knowledge protocol is called a zero-knowledge proof of knowledge if $L \in NP$ and for every prover P^* there exists a polynomial-time machine, called knowledge extractor, that can interact withP^*, and at the end it outputs x. We follow the requirement in [16], without loss of generality, the last two messages in the protocol are challenge x sent by the verifier and prover's response r. The extractor extracts x after being given transcripts of two accepting executions.

2.6. Security Properties

In this section we give the security properties by redefining these of [3]. Firstly, we show the $Oracles$ that the adversary $\mathcal{A}$ can interact in the security definitions.

- $Create(1^{\lambda})$ executes $(pk_U, sk_U) \leftarrow \text{UKeyGen}()$ and outputs pk_U;
- $BRegister(pk_U)$ plays the bank side in the $Register$ protocol and interacts with $\mathcal{A}$. If pk_U has been in the bank database, then abort; otherwise, $\mathcal{A}$ owns an account in bank B;
- $BWithdraw$ plays the bank side in the $Withdraw$ protocol and interacts with $\mathcal{A}$;
- $S\&R$ is the $Spend\ and\ Receive$ oracle that allows $\mathcal{A}$ to observe the process of $Spend$ protocol between honest users;
- $UReceive$ plays the $merchant$ side in the $Spend$ protocol and interacts with $\mathcal{A}$;
- $USpend$ plays the $user$ side in the $Spend$ protocol and interacts with $\mathcal{A}$;
- $UDeposit$ plays the $user$ side in the $Deposit$ protocol and interacts with $\mathcal{A}$.

$\boldsymbol{Unforgeability}$ means that the adversary $\mathcal{A}$ should not be able to forge a valid coin (the signature of bank) without communicating with bank or forges a valid coin according to the coins that he has owned so that to spend more coins than he has withdrew from the bank. We use the following experiment to define $\boldsymbol{Unforgeability}$.

Experiment $\boldsymbol{Expt}_{\mathcal{A}}^{unforg}(\lambda)$

- $par \leftarrow ParamGen\ (1^{\lambda})$;
- $\sigma \leftarrow \mathcal{A}^{\text{Creat,BRegister,BWith}}(par)$;
- let qW, qD be the amount of coins that $\mathcal{A}$ calls to $BWithdraw, BDeposit$ respectively.

If $(Ver_{pk_B}(\sigma) = 1) \wedge (qW < qD)$, then return 1; otherwise, return 0.

Definition 1 ($\boldsymbol{Unforgeability}$) An E-Cash system is unforgeable if for any probabilistic polynomial-time adversary $\mathcal{A}$, we have the $\boldsymbol{Expt}_{\mathcal{A}}^{unforg}(\lambda)$ defines as：

$$\Pr\left[\boldsymbol{Expt}_{\mathcal{A}}^{unforg}(\lambda) = 1\right] < negl(\lambda)$$

$\boldsymbol{Preventing\ double-spending}$ means that no user is able to spend a transaction twice. We use the following experiment to define $\boldsymbol{Preventing\ double-spending}$.

Experiment $\boldsymbol{Expt}_{\mathcal{A}}^{\mathrm{PreDS}}(\lambda)$

- $par \leftarrow ParamGen\ (1^{\lambda})$;
- $\mathcal{A}$ broadcasts two transactions to redeem the same transaction;

If these two transactions are confirmed in the $Ledger$, then return 1; otherwise, return 0.

Definition 2 ($\boldsymbol{Preventing\ double-spending}$)An E-Cash system can prevent double-spending if for any probabilistic polynomial-time adversary $\mathcal{A}$, we have the $\boldsymbol{Expt}_{\mathcal{A}}^{\mathrm{PreDS}}(\lambda)$ defines as：

$$\Pr\left[\boldsymbol{Expt}_{\mathcal{A}}^{\mathrm{PreDS}}(\lambda) = 1\right] < negl(\lambda)$$

$\boldsymbol{Anonymity}$. We define anonymity in three aspects as $\{Observe - then - Receive, Spend - then - Observe$ and $Spend - then - Receive$ [3] respectively.

$Observe - then - Receive$ means that the adversary $\mathcal{A}$ cannot link a coin that he receives as a bank during the $Deposit$ protocol or user during the $Spend$ protocol to a coin that he observed transitivity between two honest users before.

Experiment $\boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{OtR}}(\lambda)$

- $par \leftarrow ParamGen\ (1^{\lambda})$;
- $(coin_0, \mathrm{coin}_1, \delta) \leftarrow \mathcal{A}^{\mathrm{Create,S\ \&\ }R}(par)$;
- if $\delta = 1$ then simulate $USpend(coin_b)\}$ to $\mathcal{A}$; otherwise, simulate $UDeposit(coin_b)\}$ to $\mathcal{A}$ $(b \in \{0,1\})$;
- $b^* \leftarrow \mathcal{A}^{\mathrm{Create,S\ \&\ }R,USpend,UDeposit}(\mathrm{par})$;

If $b^* = b$,then return 1; otherwise, return 0.

$Spend - then - Receive$ means that the adversary $\mathcal{A}$ cannot link a coin that he spent as a user during the $Spend$ protocol to a coin that he receives during the $Spend$ protocol as a merchant or during the $Deposit$ protocol as a bank.

Experiment $\boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{StR}}(\lambda)$

- $par \leftarrow ParamGen\ (1^{\lambda})$;
- $(coin_0, \mathrm{coin}_1, \delta) \leftarrow \mathcal{A}^{\mathrm{Create,UReceive}}(par)$;
- if $\delta = 1$ then simulate $USpend(coin_b)\}$ to $\mathcal{A}$; otherwise, simulate $UDeposit(coin_b)\}$ to $\mathcal{A}$ $(b \in \{0,1\})$;
- $b^* \leftarrow \mathcal{A}^{\mathrm{Create,S\ \&\ }R}(\mathrm{par})$;

If $b^* = b$,then return 1; otherwise, return 0.

$Spend - then - Observe$ means that the adversary $\mathcal{A}$ can not link a coin that he spent as a user during the $Spend$ protocol to a coin that he observes the transitivity between two honest users.

Experiment $\boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{StO}}(\lambda)$

- $par \leftarrow ParamGen\ (1^{\lambda})$;
- $(coin_0, \mathrm{coin}_1) \leftarrow \mathcal{A}^{\mathrm{Create,UReceive}}(par)$;
- simulate $S\&R(coin_b)$;
- $b^* \leftarrow \mathcal{A}^{\mathrm{Create,UReceive,S\ \&\ R}}(\mathrm{par})$

If $b^* = b$,then return 1; otherwise, return 0.

Definition 3(Anonymity) An E-cash system is anonymous if for any probabilistic polynomial-time adversary $\mathcal{A}$, we have $\boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{OtR}}(\lambda), \boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{StR}}(\lambda)$ and $\boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{StO}}(\lambda)$ define as:

$$\Pr\left[\boldsymbol{Expt}_{\mathcal{A},b}^{\mathrm{OtR,StR,StO}}(\lambda) = 1\right] < negl(\lambda)$$

3. BITCOIN

Firstly, we recall the simplified version of bitcoin's transaction. Let $A = (A.pk, A.sk)$ be a key pair and the form of transaction that user A transfers the coin with value v to user B is as following:

$$T_x = (y, B.pk, v, \sigma)$$

where y is an index of the previous transaction $T_y (y = H(T_y)$ and $B.pk$ is the recipient of T_x, we also say that T_y is redeemed by T_x. The transaction T_x is valid if:

- $A.pk$ is the recipient of T_y;
- the value of T_y is at least v;
- transaction T_y has not been redeemed earlier;
- the signature σ of A is correct.

And there is also another condition that the transaction may have several ``inputs'' and we do not use this form in our construction so we will not describe it in details. Furthermore, we describe a more detailed version. In the real-world bitcoin system, the user has more flexibility in defining the conditions that how the transaction can be redeemed. The transaction T_y contains a description of a function (output-script) π_y whose output is Boolean and the transaction T_x that can redeem T_y if π_y evaluates to true with input T_x. The transaction is defined as:

$$T_x = (y, \pi_x, v, \sigma, t)$$

Where $[T_x] = (y, \pi_x, v)$ is the $body$ of T_x and σ is the witness that is used to make π_y evaluates to true with input T_x. the scripts π_x are written in the Bitcoin scripting language.The transaction T_x is valid if:

- time t is reached;
- $\pi_y([T_x], \sigma)$ is true;
- transaction T_y has not been redeemed before.

In our construction, we define the transaction that is sent by bank B as $T = (\wedge, \pi, v, \sigma)$ (the index of the transaction that T redeems is empty) and we also define the transaction that user U wants

to send to B to deposit the coins as $T = (H(T'), pk_B, \sigma)$, where T' is the transaction that be redeemed by T.

4. A MORE PRACTICAL E-CASH SYSTEM

In this section, we present the detailed construction of a more practical E-Cash system, which satisfies the properties of transitivity, anonymity, preventing double-spending, the size of coin is fixed during the transfer process.

For our construction we have the following assumptions: secure channels for all the communications so that an adversary cannot overhear or tamper with the transferred messages; in the $Spend$ protocol the user wants to buy the witness w from U_1, which satisfies $U's$ demand (i.e.: U_1 knows x that $f(x) = true$ which is harder to find x than verifying that $f(x) = true$ holds) with the value Val' and in the process that U proves knowledge of signatures and U_1 proves knowledge of the witness w will by executing a zero-knowledge proof of knowledge protocol with each other in the cut-and-choose technique, which is denoted as π.

Before giving the description of our construction, we show PoK of signature in details and PoK of witness w is similar.

- U_1 divides each signature $\sigma_{i,j}$ to n parts $\sigma_{i,j}^1, \dots, \sigma_{i,j}^n$ and each part is committed separately by computing $\tau_{i,j}^k := Commit(\sigma_{i,j}^k)$, where $i = 1, \dots, l, j \in \{1, \dots, m_1\} \cup \dots \cup \{1, \dots, m_l\}$ and $k = 1, \dots, n$, and sends the commitments $\tau_{i,j}^k$ to U_2;
- U_2 sends the set $J \subseteq \{(i, j, k): i = 1, \dots, l, j \in \{1, \dots, m_1\} \cup \dots \cup \{1, \dots, m_l\}, k = 1, \dots, n\}$, where the size of each J is smaller than n_0 so that U_2 cannot use any n_0-out-of-n parts to learn any knowledge of $\sigma_{i,j}$ and n_0 is predefined;
- U_1 opens the proper $\sigma_{i,j}$ according to subset J;
- U_2 verifies the openings.

Now, we present *a more practical E-Cash system*:

- $par \leftarrow ParamGen(1^\lambda)$: input the security parameter λ output the public parameter par and we assume that par is the default input to the remaining algorithms;
- $\{(pk_B, sk_B) \leftarrow BKeyGen(1^\lambda)$: the bank generates the key pairs for different denominations as $\left((pk'_B, sk'_B); (pk_1, sk_1), \dots, (pk_l, sk_l)\right) := (pk_B, sk_B) := ((pk'_B, sk'_B), (pk''_B, sk''_B))$;
- $\{(pk_U, sk_U) \leftarrow UKeyGen(1^\lambda)$: the user generates the key pairs as (pk_U, sk_U);
- Registration$(U[(pk_U, sk_U)], B[(pk_B, sk_B)]])$: the user U sends pk_U to bank B, if $pk_U \in DB$ (the database of the bank) then B outputs $\perp$; otherwise, B selects a nonce $nonce$, computes $ID_U = H(pk_U || nonce)$, adds (pk_U, ID_U) to DB and returns ID_U to U;
- $Withdraw(U[(pk_U, sk_U)], (pk'_U, sk'_U)], B[(pk''_B, sk''_B)])$: the user U selects a new key pair (pk'_U, sk'_U) and sends (pk'_U, ID_U, Val) where Val denotes the amount of coins that he will withdraw from B. B checks $U's$ account and its balance, returns $\perp$ if one of them is invalid; otherwise, B does the followings:
  - computes the number of coins with different denominations as $m_1, \dots, m_l$;
  - computes signatures for each coin: selects $s \in Z_p$ (where p is a prime) and computes $\sigma_{i,j} = Sign_{sk_i}(s||j||ID_U)$ ($\sigma_{i,j}$ is the signature of the j^{th} coin wit denomination Val_i), where $i = 1, \dots, l, j \in \{1, \dots, m_1\} \cup \dots \cup \{1, \dots, m_l\}$;

- prepares the transactions $T_{i,j}^{U} = (\wedge, \pi_{i,j}, Val_i, \sigma_{i,j}^{B})$, where $T_{i,j}^{U}$ means the j^{th} coin with denomination Val_i is transferred and $\pi_{i,j}(T'_{i,j}) = 1$ if $Verify_{pk'_U}\left(\left[T'_{i,j}, \sigma'_{i,j}, [T'_{i,j}]\right]\right) = 1$;
- broadcasts transactions $T_{i,j}^{U}$ and sends $Coin = (\sigma_{1,1} \dots \sigma_{1,m_1}, \dots, \sigma_{l,1} \dots \sigma_{l,m_l})$ to user U;
- deducts user $U's$ account.

- Spend$\left(U_1\left[\text{coin}, \left(pk'_{U_1}, sk'_{U_1}\right)\right], U_2\left[\omega, \left(pk_{U_2}, sk_{U_2}\right)\right]\right)$:first we assume that U_1 wants to buy a witness ω from U_2, whose value is Val', so in this process U_1 transfers Val' to U_2 and get witness ω:

  - U_1 computes the number of coins with different denominations as $m_1, \dots, m_l$ and prepares the coins:

$$coin_{i,j} = (Val_i, SN_i, \sigma_{i,j})$$

  where $i = 1, \dots, l, j \in \{1, \dots, m_1\} \cup \dots \cup \{1, \dots, m_l\}$. And then U_1 computes $\pi = PoK(\sigma_{1,1} \dots \sigma_{1,m_1}, \dots, \sigma_{l,1} \dots \sigma_{l,m_l})$ and sends $coin = (coin_{1,1}, \dots, coin_{l,m_l}, \pi)$to U_2;
  - U_2verifies that U_1 owns $coin$ exactly, and then computes $h = H(\omega), \pi = PoK(\omega)$ and sends (h, π) to U_1;
  - U_1verifies (h, π) and prepares the following transactions:

$$\text{PutMoney}_{i,j}^{U_1} = (H\left(T_{i,j}^{U_1}\right), \pi'_{i,j}, Val_i, \sigma_{sk'_{U_1}}, t)$$

$$\text{ClaimMoney}_{i,j}^{U_1}(H\left(\text{PutMoney}_{i,j}^{U_1}\right), \pi_{i,j}^{U_1}, Val_i, \sigma_{sk'_{U_1}}, t')$$

  where$\pi'_{i,j} = ((\text{Verify}_{pk_{U_1}}(T_{i,j}) \wedge H(\omega) = h) \vee (\text{Verify}_{pk'_{U_1}}(T'_{i,j}) \wedge \pi_{i,j}^{U_1} = (\text{Verify}_{pk'_{U_1}}(T_{i,j})))$;
  - U_2 prepares the following transactions:

$$\text{ClaimMoney}_{i,j}^{U_2} = (H\left(\text{PutMoney}_{i,j}^{U_1}\right), \pi_{i,j}^{U_2}, Val_i, \sigma_{sk_{U_2}}, \omega, t_1)$$

  where $\pi_{i,j}^{U_2} = Verify_{pk_{U_2}}(T_{i,j})$;
  - U_1 broadcasts transactions $\text{PutMoney}_{1,1}^{U_1}, \dots, \text{PutMoney}_{l,m_l}^{U_1}$ to the $Ledger$ and computes $c = \text{Enc}_{pk_{U_2}}(\sigma_{1,1}, \dots, \sigma_{1,m_1}; \dots; \sigma_{l,1}, \dots, \sigma_{l,m_l})$.If not all the transactions appear on the $Ledger$ or U_2 does not receive the correct c in time t, then the protocol halts;
  - U_2 waits for all the transactions $\text{PutMoney}_{1,1}^{U_1}, \dots, \text{PutMoney}_{l,m_l}^{U_1}$ appearing on the $Ledger$,he broadcasts transactions $\text{ClaimMoney}_{1,1}^{U_2}, \dots, \text{ClaimMoney}_{l,m_l}^{U_2}$ to the $Ledger$,which will reveal the witness ω. And if none of these transactions appears on the $Ledger$ in time t_1 , then U_1 broadcasts transactions $\text{ClaimMoney}_{1,1}^{U_1}, \dots, \text{ClaimMoney}_{l,m_l}^{U_1}$to get his coins back.

- $Deposit(U[coin, (pk'_U, sk'_U)], B[pk_B, sk_B])$: before the $Deposit$ protocol starts, we assume that $Ledger$ contains transactions $T_i (i = 1, \dots, n_l)$ (we also say that the number of coins that U owns is n_1) and each transaction can be redeemed by U. First U prepares the following transactions whose recipient is bank B (pk_B), then B adds the amount of coins to $U's$ account.

  - U prepares transactions $T'_1 = \left(H(T_1), pk_B, Val_1, \sigma_{[T'_1]}\right), \dots, T'_{n_1} = \left(H(T_{n_l}), pk_B, Val_l, \sigma_{[T'_l]}\right)$ and broadcasts them to the $Ledger$, and sends $c = Enc_{pk_B}(\sigma_1, \dots, \sigma_{n_l}, (pk_U, ID_U), (pk', sk'))$ to the bank B,where (pk', sk') is the key pair

that signs the $T'_1, \ldots, T'_n$, so that the adversary cannot disguise U to let the bank to add the coins to his account because he dose not know the correct key pair;

- Bwaits the transactions appear on the Ledgerand checks the received messages, if each of them is valid then adds the amount of coins to the $U's$ account, otherwise returns $\perp$.

5. SECURITY ANALYSIS

Theorem. Suppose the Digital Signature Scheme, Public Encryption Scheme, Zero-knowledge Proofs of Knowledge and the bitcoin blockchain protocol are secure, then our e-cash system satisfies unforgeability, anonymity, transferability, divisibility, preventing double-spending attacks and fair exchange between users.

Proof: ***(unforgeability)*** According to the definition of unforgeability (Section 2.6), we give the proof of unforgeability into two parts.

Part 1: Let the adversary $\mathcal{A}$ to execute $Create, Register$ and $Withdraw$ protocols with the $Oracle$, we denote it as simulator $\mathcal{S}$, without the knowledge of sk_B. By executing $Create$ and $Register$ protocols, the adversary $\mathcal{A}$ has an account in the bank's database DB. Then at the end of executing $Withdraw$ protocol with $\mathcal{S}$, the output of $\mathcal{S}$ is $coin = ((coin_1, \sigma_{1,1}, \ldots, \sigma_{1,.m_1}), \ldots, (coin_l, \sigma_{l,1}, \ldots, \sigma_{l,.m_l}))$. With the security of Digital Signature Scheme that, without the knowledge of signature key sk_B, $\mathcal{S}$ can give a valid signature $\mathcal{S}$ with negligible probability.

Part 2: Let the adversary $\mathcal{A}$ to execute $Create, Register, Withdraw$ and $Deposit$ protocols with the $Oracle$, we denote it as simulator $\mathcal{S}$ with the knowledge of sk_B, and gets the valid coins $coin = ((coin_1, \sigma_{1,1}, \ldots, \sigma_{1,.m_1}), \ldots, (coin_l, \sigma_{l,1}, \ldots, \sigma_{l,.m_l}))$ and the valid transactions that can be redeemed by $\mathcal{A}$. Thus if $\mathcal{A}$ succeeds at this game, then $\mathcal{A}$ must redeem the same transaction twice and both are confirmed on the $Ledger$ to accomplish the $Deposit$ protocol. With the security of Bitcoin system, $\mathcal{A}$ can redeem the same transaction more than once with negligible probability. So in this game, $\mathcal{A}$ succeeds with negligible probability.

(Preventing Double-Spending attack) Let the adversary $\mathcal{A}$ to execute $Create, Register, Withdraw$ and $Deposit$ protocols with the $Oracle$ to get the coins and transactions that can be redeemed by $\mathcal{A}$. Then $\mathcal{A}$ broadcasts two transactions that redeem a same transaction, so that the two transactions have the same ``head" and, with the security of Bitcoin system, the probability that these two transactions appear on the $Ledger$ is negligible. Thus in this game, $\mathcal{A}$ can succeed with negligible probability.

(Anonymity) In the definition of anonymity, we divide it into three parts OtR, StR and StO. For OtR, we assume that the users communicate with other with a secure channel, so in the $Spend$ protocol between two pairs of honest users (U_1, U_2) and (U_3, U_4), where U pays $coin_1$ to U_2, U_3 pays $coin_2$ to U_4 and $Val(coin_1) = Val(coin_2)$. Thus, what the adversary $\mathcal{A}$ can observe are $\text{View}_1 = \{(\text{PutMoney}^{U_1} = H(T^{U_1}), \pi_1, Val(coin_1), \sigma_1), \text{ClaimMoney}^{U_2} = (H(\text{PutMoney}^{U_1}), \pi_2, Val(coin_1), \sigma_2)\}$ and $\text{View}_2 = \{(\text{PutMoney}^{U_3} = H(T^{U_3}), \pi_3, Val(coin_2), \sigma_3), \text{ClaimMoney}^{U_4} = (H(\text{PutMoney}^{U_3}), \pi_4, Val(coin_2), \sigma_4)\}$. Then $\mathcal{A}$ communicates with $OracleUSpend(coin_1)$ or $UDeposit(coin_2)$ and the view of $\mathcal{A}$ is $\text{View}_3 = \{(\text{PutMoney}^{O} = H(T^{O}), \pi_5, Val(coin_1), \sigma_5), \text{ClaimMoney}^{\mathcal{A}} = (H(\text{PutMoney}^{O}), \pi_6, Val(coin_1), \sigma_6)$ or $\text{View}_4 = \{(\text{PutMoney}^{O} = H(T^{O}), pk_{\mathcal{A}}, Val(coin_2), \sigma_7)\}$. Because $\text{View}_1 = \text{View}_2$, so $\mathcal{A}$ cannot link View_3 or View_4 to View_1 or View_2. As a result, in this game, $\mathcal{A}$ can succeed with negligible probability.

The same analysis method can be applied the to anonymity of StO and StR.

(Transferability, divisibility and fair exchange between users) The security analysis of transferability, divisibility and fair exchange can be reduced to the security of Bitcoin system. In this system, (1) the payee can use the coin further without depositing to the bank first by redeeming the transactions on the $Ledger$; (2) different coins have different denominations and each of them is transferred by an unique transactions so that user can spend arbitrary amount of coins; (3) the detailed form of bitcoin transaction allow us to define the conditions that the transactions can be redeemed. Thus, no user can cheat the others.

6. CONCLUSION

In this work, we develop a more practical E-Cash system that can be used to achieve unforgeability, preventing double-spending attacks, anonymity, transferability, divisibility, fixed size of coin during the transfer process and fair exchange between users. Our main contribution is to achieve the above security properties in one E-cash system. And the further work is to reduce complexity, e.g., in the Withdraw protocol, the bank signs every coin, so we assume that the amount of coins that the user can withdraw is limited and to achieve the real arbitrariness is the future works.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported in part by the National Key R&D Program of China (No. 2020YFB1005801) and in part by the National Natural Science Foundation of China (No. 62172396).

REFERENCES

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Consulted, 2009.
[2] Chaum D. Blind Signatures for Untraceable Payments[M]// Advances in Cryptology. Springer US, 1983:199-203.
[3] Baldimtsi F, Chase M, Fuchsbauer G, et al. Anonymous Transferable E-Cash[M]// Public-Key Cryptography -- PKC 2015. 2015:101-124.
[4] Okamoto T, Ohta K. Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash[C]// on Advances in Cryptology. Springer-Verlag New York, Inc. 1989:481-496.
[5] Okamoto T, Ohta K. Universal Electronic Cash[C]// Advances in Cryptology - CRYPTO '91, International Cryptology Conference, Santa Barbara, California, Usa, August 11-15, 1991, Proceedings. DBLP, 1991:324-337.
[6] Chaum D, Pedersen T. Transferable cash grows in size[J].
[7] Tewari H, Hughes A. Fully Anonymous Transferable Ecash[M]// Public-Key Cryptography -- PKC 2015. Springer Berlin Heidelberg, 2016:101-124.
[8] Märtens P. Practical compact e-cash with arbitrary wallet size[J]. Cryptology ePrint Archive, 2015.
[9] Man H A, Susilo W, Mu Y. Practical Compact E-Cash[M]// Information Security and Privacy. Springer Berlin Heidelberg, 2007:431--445.
[10] Camenisch J, Hohenberger S, Lysyanskaya A. Compact E-Cash[J]. Eurocrypt, 2005, 3494:566-566.
[11] Canard S, Pointcheval D, Sanders O, et al. Divisible E-Cash Made Practical[M]// Public-Key Cryptography -- PKC 2015. Springer Berlin Heidelberg, 2015:77-100.
[12] Heilman E, Baldimtsi F, Goldberg S. Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions[M]// Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2016.
[13] Christidis K, Devetsikiotis M. Blockchains and Smart Contracts for the Internet of Things[J]. IEEE Access, 2016, 4:2292-2303.
[14] Kumaresan R, Bentov I. Amortizing Secure Computation with Penalties[C]// ACM Sigsac Conference on Computer and Communications Security. ACM, 2016:418-429.

[15] Kumaresan R, Moran T, Bentov I. How to Use Bitcoin to Play Decentralized Poker[C]// ACM Sigsac Conference on Computer and Communications Security. ACM, 2015:195-206.
[16] Kumaresan R, Bentov I. How to Use Bitcoin to Incentivize Correct Computations[C]// ACM Sigsac Conference on Computer and Communications Security. ACM, 2014:30-41.
[17] Banasik W, Dziembowski S, Malinowski D. Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts[M]// Computer Security – ESORICS 2016. 2016.
[18] Andrychowicz M, Dziembowski S, Malinowski D, et al. Secure multiparty computations on Bitcoin[C]// Security and Privacy. IEEE, 2014:443-458.
[19] Srivastava, Shweta, and V. Saraswat. "E-Cash Payment Protocols." International Journal on Computer Science Engineering 4.9(2012).
[20] Anand R S, Madhavan C E V. An Online, Transferable E-Cash Payment System[M]// Progress in Cryptology —INDOCRYPT 2000. 2000:93-103.

AUTHORS

Peifang Ni received the Ph.D. degree in the Institute of Information Engineering, Chinese Academy of Sciences, in 2020. She is currently a postdoctoral with the Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences. Her major research interests include applied cryptography, security protocol and blockchain consensus.