International Journal of Multimedia and Ubiquitous Engineering
Vol. 9, No. 10 (2014), pp. 153-168
http://dx.doi.org/10.14257/ijmue.2014.9.10.16

# Efficient Divisible E-cash based on the P Signature

[1]Huo Yanling, [2]Wang Haibin, [3]Chai Xuguang and [4]Liu Xia

*[1,2,3]Department of Information Engineering, Xingtai Polytechnic College, Xingtai, Hebei, China*
*Department of Information Science and Technology, Xingtai University, Xingtai, Hebei, China*
*8076699@qq.com[1], 121780639 @qq.com[2], 709941311 @qq.com[3], 121780639@qq.com*

***Abstract***

*Divisible e-cash allows the user to withdraw a single divisible coin and spends the any sub-coins by dividing the value of the coin. The first divisible e-cash system in the standard model was proposed by Izabachene and Libert. However, the efficiency of the spending protocol and the deposit protocol is very low. In this paper, we construct an efficient divisible e-cash scheme without random oracle by using the Groth-Sahai (GS) proof system and bound accumulators. Our scheme is on-line and truly anonymous without a trusted third party. Comparing to Izabachene and Libert's work, we improve the efficiency of the spending protocol and deposit protocol by introducing a new generational algorithm. Moreover, the bank only needs to look up the coin's serial number in a table of previously spent coins. We give the NIZK proofs of bounded accumulator in the standard model. Some security properties of our scheme, such as anonymity, unforgeability and exculpability, are proved in the standard model.*

***Keywords:*** *Divisible E-cash; Binary Tree; Accumulator; P-signature; Groth-Sahai Proofs*

## 1. INTRODUCTION

Electronic cash (e-cash), introduced by Chaum, is an electronic analogue of physical money and has attracted a lot of researchers [5]. An e-cash system consists of three parties: the bank B, the user U and the merchant M. U withdraws an e-cash from B and spends the e-cash to M, and then M offers goods and services in exchange for e-cash. And last, M deposits the e-cash to B.

Divisible e-cash allows a user to efficiently withdraw a single divisible coin and spend this coin in several times by dividing the value of the coin. In order to obtain efficiency, all of the existed divisible e-cash schemes, with the exception of [15, 28], are constructed in random oracle model. Although Izabachene and Libert [28] propose the first divisible e-cash in the standard model, the efficiency of the spending protocol and deposit protocol is very low. In this paper, we construct an efficient divisible e-cash without random oracle.

Much research has been performed in the area of e-cash [1-2, 4-12, 15, 25-28]. Okamoto and Ohata proposed the first ideal untraceable electronic cash [8] using the cut-and-choose methodology and introduces some basic properties, *i.e.*, untraceability, transferability and divisibility. The cut-and-choose methodology causes low efficiency of Okamoto and Ohata's scheme. Pailles constructed a new protocol for e-cash [9] which develops the anonymity and the divisibility of the e-cash. Unfortunately, the bank has to perform a huge amount of computations. As for divisibility, Eng and Okamoto proposed a single-term divisible e-cash [11] which is not a practical divisible e-cash. Then Okamoto presented the first practical divisible e-cash [7] which was subsequently improved by Chan *et al.* [25]. However, the schemes mentioned above are linkable, since anyone can decide whether several spend come from the same coin. In 2000, Nakanishi and Sugiyama provided an unlinkable divisible electronic cash [6] by introducing a trusted third party.

The compact e-cash scheme [4] allows a user to withdraw a wallet containing 2L coins efficiently and satisfies all the security properties mentioned above. However, the number of the coins that the user wants must be chosen in the withdrawal protocol, and be spent one by one in the spending protocol.

The first anonymous divisible e-cash scheme was proposed by Canard and Gouget [10]. However, when a user spends a small number of coins, he has to prove the spending protocol is constructed correctly by non-interactive zero-knowledge proof of knowledge. This is well-known very costly. Au et al: constructed a divisible e-cash [1] from bounded accumulators. The efficiency of the computation and the storage is improved in the spending protocol. Unfortunately, it does not fulfill unforgeability.

In order to obtain unforgeability, Canard and Gouget proposed a divisible e-cash scheme [12]. However, the number of the accumulator is proportional to the number of the level of the binary tree in the withdrawal protocol.

All the security of above e-cash is proven in the random oracle model. Some results [21, 16] have shown that some schemes proven secure in the random oracle model, are not secure in the standard model. Belenkiy, Chase, Kohlweiss and Lysyanskaya [19] proposed a compact e-cash system with non-interactive spending in the standard model. This scheme is based on P-signature [13], simulatable verifiable random functions [23] and Groth-Sahai proofs systems [17]. Fuchsbauer *et al.* [20] constructed the first practical transferred constant-size fair e-cash in the standard model. However, each user has to keep in memory the data associated to all past transactions to prove her innocence in case of a fraud. Izabachene and Libert proposed the first divisible e-cash scheme [28] in the standard model. They used a different method to authenticate the spending path. Unfortunately, the communication complexity of the spending scheme is proportional to the level number of the spent node. Meanwhile, the computational workload of the bank depends on the number of previously received coins when it comes to check that the received coin does not constitute a double-spending.

Accumulators were firstly introduced by Benaloh and de Mare [29]. An accumulator allows aggregation of a large set of elements into one constant-size accumulator value. In 2005, Nguyen proposed a dynamic accumulator scheme from bilinear pairings. It allows elements in $Z_p\{-s\}$ for some prime p to be accumulated, where s is the master secret of the accumulator instance. Acar and Nguyen [30] give a delegable accumulator which is focus on the non-membership proofs. However, with the exception of Acar and Nguyen's accumulator, others are used in random oracle model. Acar and Nguyen's accumulator only focuses on the non-membership proofs. Therefore, we give the general correctness proof of accumulator in the standard model using the GS proof system [17].

Groth and Sahai constructed the first efficient non-interactive proof system [17] which considers a large class of statements over bilinear group. It is witness indistinguishable, i.e., any adversary cannot distinguish which witness is used by the user. The proof can be randomized to update the NIZK proof.

IL's E-cash Scheme. Izabachene and Libert [28] propose the first divisible e-cash in the standard model. Their construction relies on the classical binary tree approach [10] and the Groth-Sahai non-interactive proof systems [28]. To achieve divisibility without resorting to random oracles, they use the different method to authenticate the node corresponding to the spent divided coin in the tree. The construction of the binary tree is based on the classical binary tree, thus the communication cost of the spending phase is O(L-l) to spend a coin of value $2^l$, where $2^L$ is the value of the root node. The more they want to divide the wallet into small coins, the more expensive the spending phase is. To avoid the spending of an ancestor or a descendant of a spent node, they add a pair $(T_{j,1}, T_{j,2})$ in the coin for each node from the root node to the spent node. This makes the efficiency of detecting the double-spending is very low. The computation workload of the bank depends on the number of previously received coins when it comes to check that the user does not a double-spending. This problem is left as an open problem.

Overview of Our Scheme. We present an intuition on how our scheme is constructed. Each e-cash of monetary value $2^L$ is equipped with an L+1 level binary tree. Using a new algorithm, the user constructs the binary tree. To construct the binary tree, the user generates $2^L$ random numbers $s_i$ for the leaves node, where $i = 0, \cdots, 2^L - 1$. The keys of the leaves node are $K_{L+1,0} = g_0^{s_0}, K_{L+1,1} = g_1^{s_1}, \cdots, K_{L+1,2^L-2} = g_0^{s_{2^L-2}}, K_{L+1,2^L-1} = g_1^{s_{2^L-1}}$, where $g_0, g_1 \in G_1$. To be compatible with the bounded accumulator, the user computes the new keys using two hash functions HL and HR. The parent node of the leaves node is obtained by the multiplication of the keys of leaves node. The root node is generated by the multiplication of two direct child nodes. Then, the user accumulates the leaves node and all nodes into accumulator values $Acc_L$ and $Acc$ respectively. The user sends the accumulator values $Acc_L$ and $Acc$ to the bank. The bank signs $Acc_L$ and $Acc$. We assume that the user spends a coin of monetary value $2^l$ and the corresponding leaves node is $hk_{L+1,0}, \cdots, hk_{L+1,j}$. When the user wishes to spend the coin, he uses the corresponding leave nodes and computes the serial number $S = hk_{L+1,0} \cdot \cdots \cdot hk_{L+1,j}$, a security tag $T = pk_U \cdot \hat{e}(g, h^{l \cdot hk_{i,j} \cdot R \cdot s})$, where R is generated by the merchant, s is generated by the user and the bank together. The user submits S; T and proves to the merchant in zero knowledge manner that he is in possession of the bank's signature on

Acc, $Acc_L$, s, usk and S; T are correctly formed. If a user attempts to spend more than the coin of monetary value $2^L$, he will have to use the same S and identified.

That is almost our final solution. We use the bounded accumulator [2] to guarantee that the user accumulates the bounded nodes. To make sure that the monetary value of the spent node comes from the correct level, we borrow the idea from Belenkiy et al.'s scheme [19]. More precisely, the bank generates L P-signatures $\Sigma_0, \cdots, \Sigma_L$ on L coin indices. The user proves that the same commitment $C_L$ is used for the proof of T and $\Sigma_l$. Two problems remain, the first one being how be compatible with the Groth-Sahai proof system [17] to prove the spending path is correctly formed. The second problem is that there is no existing efficiency membership proof for the bounded accumulator in the standard model. We solve the first problem by constructing a new algorithm for the binary tree. We solve the second problem by making the membership proof of bounded accumulator in the standard model.

Our Contribution. Our construction of the divisible e-cash is a new binary tree generation algorithm, in combination with the use of a bounded accumulator [2] and GS proof system [17].We make the following contribution:

We use the nodes of a binary tree to represent the e-cash. In the tree-based constructions, one difficulty is for the user to efficiently prove that the spending path is well-formed. To solve the problem, efficiency of the Izabachene and Libert's scheme is very low. The last is our method. We firstly introduce a new algorithm to construct the binary tree. Then we identify the double-spender using the accumulator. In our new structure of the binary tree, each node of binary tree is constructed in one cyclic group, which would be compatible with Groth-Sahai toolbox [28]. By the accumulator, we efficiently prove the path connecting the spent node to the root is well-formed. Therefore, our scheme is more efficient in the spending protocol and the deposit protocol than Izabachene and Libert's scheme [28].

To use the bounded accumulator, we must prove the accumulator correctness of the bounded accumulator in the standard model. Therefore, we give the NIZK proof of the bounded accumulator using the GS proof system in the standard model. In order to prove the correctness of the spending, we use the technique [15] which the bank signatures n coin indices, *i.e.*, PSign(1), PSign(2), ⋯,PSign(n). Thus, we only need two accumulators to prove the correctness of the spending. However, the paper [12] needs L + 2 accumulators in the withdrawal protocol, to prove the correctness of the spending.

Our scheme is more efficient than Izabachene and Libert's work [28] in several metrics. Firstly, When

U spends a coin of monetary value $2^l$ in the spending protocol of Izabachene and Libert's scheme [28], U chooses an unspent node at level $L - l$ in the binary tree. It requires $O(L - l)$ group elements and multi-exponentiations. In contrast, only two accumulators are used in our scheme. Secondly, the merchant needs $O(L - l)$ pairings to verify the correctness of the path connecting the spent node to the root. However, we verify the correctness of the path with constant pairings in the spending protocol. Thirdly, in the deposit protocol of Izabachene and Libert's scheme [28], the bank must check and analyze all received coins to decide whether a double-spending had happened. In comparison, we only need to look up the coin's serial number in a table of previously spent coins.

Paper Outline. The rest of the paper is organized as follows. In Section 2 we present preliminaries on the various cryptographic tools and assumptions. Security model of divisible e-cash is presented in Section 3. We present our construction in Section 4 and its efficiency analysis in Section 5. In section 6, we give the security proof. Finally we conclude in Section 7.

## 2. Preliminaries

### 2.1 Mathematical Definitions and Assumptions

Definition 1. (Pairing). A pairing $\hat{e}: G_1 \times G_2 \rightarrow G_3$ is a bilinear mapping from two group elements to a group element [17].a. $G_1, G_2, G_3$ are cyclic groups of prime order p. The elements $g, h$ generate G1 and G2 respectively. b. $G_1 \times G_2$ is a non-degenerate bilinear map, so $\hat{e}(g, h)$ generates $G_3$ and for all a, $b \in Z_n$ we have $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.c. We can efficiently compute group operations, compute the bilinear map and decide membership.

The security of our construction is based on the following mathematical assumptions, namely Symmetric External Diffie-Hellman [17] and OMDL [3].Definition 2. (Symmetric External Diffie-Hellman). The Symmetric External Diffie-Hellman(SXDH) Assumption states that the DDH problem is hard in both $G_1, G_2$. It implies that there is no efficiently computable isomorphism from G2 to G1 or vice versa.

Definition 3. (One-More Discrete Logarithm Assumption). The one-more discrete logarithm assumption (OMDL) is defined as follows: on input $g, g^{x_1}, \cdots, g^{x_{n+1}}$ and dlogg(·) which is the oracle that takes input $y \epsilon G_1$ and returns its discrete log, namely x such that $y = g^x$. The adversary only can make the oracle queries to dlogg(·) at most n times, it is computationally infeasible to output the n+1 discrete logarithm.

### 2.2. Useful Tools

Groth-Sahai Proofs. Groth and Sahai [17] constructed the first NIZK proof systems. They proved a large class of statements in the context of groups with bilinear maps in the standard model. In order to prove the statement, the prover firstly commits to group elements. Then the prover produces the proofs and sends the commitments, the proofs and corresponding parameters to the verifier. And last the verifier verifies the correctness of the proof.

In this paper, SXDH-based commitments are used to commit to group elements. The simple description of SXDH-based Groth-Sahai commitments and Groth-Sahai proofs are given in the following.

AFG-commitment. Abe, Fuchsbauer and Groth (AFG) [24] proposed a trapdoor commitment scheme, which directly commits group elements in G (a message $m \in Z_p$) and can be combined with Pedersen commitments. The commitment is length-reducing, since the commitment to a tuple of messages yields a commitment consisting of a single target group element.

We only use the commitment to a message $m \in Z_p$. In the following, we only describe the AFG- commitment to a message $m \in Z_p$.

To commit a message $m \in Z_p$, the commitment keys ck = (G1,G2,G3, g,h, y), the opening $d = g^r$ are used to obtain the commitment $C = h^m Y^r$. And then we send the commitment C, the trapdoor opening D to the verifier. The verifier makes sure that the commitment is correct by verifying the equation

$\hat{e}(g, C)\hat{e}(d, y^{-1}) = \hat{e}(g, h^m)$.

A Multi-block P-Signature. A multi-block P-signature was introduced in [15]. It allows a user to sign a block of elements in a cyclic group G in the standard model. Suppose $C_{m_1}, \cdots, C_{m_n}$are the Groth-Sahai commits of $m_1, \cdots, m_n$ respectively. The P-signature allows a user to obtain a signature from the signer on the commitments of a block of messages $m_1, \cdots, m_n$. The signer learns nothing about $m_1, \cdots, m_n$ while he knows the commitments respectively. In the following, we simply describe the P-signature [15].

The public parameters are defined as (p,G1,G2,G3,ê,g,h, $params_{GS}$, ê(g,h)), where g and h are random elements of G1 and G2 respectively, and $params_{GS}$, is the Groth-Sahai common reference string.

The public key and the private key are defined to be pk = $(u, v = h^{\alpha}, \tilde{v} = g^{\alpha}, \{w_i, h^{\beta_i}, \widetilde{w} = g^{\beta_i}\}_{i=1}^{n},)$ and $sk = (\alpha, (\beta_1, \cdots, \beta_n))$, where u is the random element of G1 and$\alpha, \beta_1, \cdots, \beta_n$are the random elements of Zp.We define the NIZK proof $\pi_P$ of a multi-block P-signature for messages $m_1, m_2, \cdots, m_n$ as PSign($m_1, m_2, \cdots, m_n$).

## 2.3. Algorithms

The divisible e-cash needs three usual players, namely the user U, the merchant M and the bank B. It will include seven polynomial algorithms between them. The following give the specific algorithms.

1. ParamSetup($1^{\lambda}$) takes as input a security parameter $\lambda$ and outputs the public parameters params.

2. BKeyGen(params) is a probabilistic algorithm which outputs two key pairs (pkP,skP) for issuing coins and (pkC, skC) for signing coin indices. It also defines an empty database DB for later use.

3. KeyGen(params) is a probabilistic algorithm which outputs a user (merchant) key pair (skU,pkU)(resp.(skM, pkM)).

4. Withdraw(U(params, pkP,pkU, skU,L); B(params,pkU,pkP,skP)) is an interactive protocol between U and B that permits U withdraws a wallet $co_U$ of value $2^L$ from B. B debits U's account and stores a piece of tracing information Tr which can be later used to identify double-spenders.

5. Spend(U(params,pkP, pkC,pkM,skU,$co_U$, l);M(params,pkP,pkC,skM)) is a protocol that allows the user to spend a value $2^{l`}$ from the divisible wallet $co_U$ to M. The user outputs a updated wallet $co'_U$. The merchant obtains a coin coM.

6. Deposit(M(params,pkP,pkC,pkM, skM,coM); B(params,pkM,DB)) is a protocol that permits M to deposit a coin $co_M$to B. B outputs Ok or executes the double-spender identification. DB is a database which saves all coins users has spent.

7. Identify (params,pkP,pkC, $\mathrm{co}'_{\mathrm{M}}$ , $\mathrm{co}_{\mathrm{M}}$ ) is a algorithm which outputs a double-spender's public key pkU using the database DB and the two different coins $\mathrm{co}'_{\mathrm{M}}$, $\mathrm{co}_{\mathrm{M}}$.

### 2.4. Security Notions

A secure divisible e-cash scheme provides Anonymity, Unforgeability, Identification of double-spenders and Exculpability. We give the specific definitions as follows.

1. Anonymity. It guarantees that no coalition of banks and merchants can ever learn the spending habit of an honest user.

2. Unforgeability. No coalition of users and merchants can deposit more coins than they have withdrawn from the bank.

3. Identification of double-spenders. It guarantees that coalition of users and merchants cannot be able to double-spend a coin with the same serial number or corresponding serial number.

4. Exculpability. No coalition of the banks and users can accuse a honest users from having double- spent a coin.

## 3. Construction of Divisible E-Cash

To construct our divisible e-cash, we firstly prove the membership proof of the bounded accumulators without random oracle. Secondly, a new binary tree generational algorithm is introduced. And last, we give the detailed construction of our divisible e-cash.

### 3.1. Bounded Accumulators

A bounded accumulator scheme was introduced in [2] as an accumulator with a limit s as the maximum number of elements that can be accumulated. For every element x in the bounded accumulator, there is a unique witness W which can prove the element x is accumulated into the accumulator. In this context, we give the membership proof of bounded accumulator using the GS proof system in the standard model. We obtain the following Theorem 1.

Theorem 1. The proof for the bounded accumulator $\pi_{\mathrm{Acc}}$, is a NIZK proof with perfect completeness, perfect soundness and composable witness-indistinguishability.

### 3.2. New Binary Tree Structure

We introduce a new generation algorithm of the binary tree. Each divisible coin of monetary value $2^{\mathrm{L}}$, is assigned to a binary tree of L+1 levels. The values of the leaves are the least, namely 1. Each of the leaves node is assigned a key denoted by $\mathrm{K}_{\mathrm{L+1,j}}$, where $0 \leq \mathrm{j} \leq 2^{\mathrm{L}} - 1$. Any other internal node corresponds to an amount of money which is exactly the twice amount of their corresponding child node values and also is assigned a key defined by $\mathrm{K}_{\mathrm{i,j}}$ . The root node lies in the 0th level and has the max value, namely $2^{\mathrm{L}}$. The corresponding key of the root node is $\mathrm{k}_{0,0}$.

Using the new generation algorithm, we construct the binary tree from the leaves node. Each internal node is obtained by the multiplication of its corresponding leaves node. Thus each internal node is independent of the others. Meanwhile we only supply two accumulators and corresponding proofs to merchant, to proof the spending is correct. When the user wants to spend a node, he supplies the keys of the spent node and the corresponding leaves node to the merchant. Although the key of the spent node is known by the merchant, the key of the spent node is generated using the leaves node which they are obtained by the random number. Therefore, the merchant cannot obtain any information from the key. The user generates the internal node only by the multiplication, so the efficiency of the construction of the binary tree is improved. The new binary tree is given in the above Figure 1.

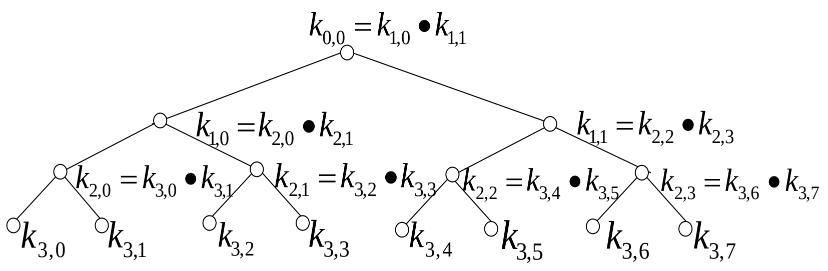
$k_{0,0} = k_{1,0} \bullet k_{1,1}$
$k_{1,0} = k_{2,0} \bullet k_{2,1}$
$k_{1,1} = k_{2,2} \bullet k_{2,3}$
$k_{2,0} = k_{3,0} \bullet k_{3,1}$
$k_{2,1} = k_{3,2} \bullet k_{3,3}$
$k_{2,2} = k_{3,4} \bullet k_{3,5}$
$k_{2,3} = k_{3,6} \bullet k_{3,7}$
$k_{3,0}$
$k_{3,1}$
$k_{3,2}$
$k_{3,3}$
$k_{3,4}$
$k_{3,5}$
$k_{3,6}$
$k_{3,7}$

**Figure 1. Construction of a Binary Tree (L=3)**

## 3.3. Construction

In this section, we describe our divisible e-cash construction in detail. Our divisible e-cash consists of the setup procedure, withdrawal protocol, spending protocol, deposit protocol and the double-spender identification. We construct it using P-signature, bounded accumulator and GS proof.

**3.3.1 Setup:** A divisible coin has a monetary value set to $2^L$. On input a security parameter $1^\lambda$ and a security prime number p. Let $\hat{e}: G_1 \times G_2 \to G_3$ is a bilinear map, where $|G_1| = |G_2| = |G_3| = p$. The elements hU,hM,hT $\in$ G2 and g,h generate G1 and G2 respectively. The bank randomly chooses $g_0, g_1 \in G_1$ whose discrete logarithms to the base g are unknown. Let $HL: \{0,1\}^* \to Z_p^*, HR: \{0,1\}^* \to Z_p^*$ and $HA: \{0,1\}^* \to Z_p^*$ be three secure cryptographic hash functions [1]. All these data compose the public parameters params.

The bank creates two key pairs (pkP,skP) for issuing wallets and (pkC,skC) for signing coin indices.The bank computes L P-signatures $\Sigma_1, \cdots, \Sigma_L$ on the coin indices $1,2,\cdots,L$ using skC, where $\Sigma_i = PSign(i), i \in 1, \cdots, L. U(resp. M)$ chooses skU = usk $\in Z_p^*$ (resp. skM = msk$\in Z_p^*$) as his private key and computes the public key pkU =ê(g, h)$^{usk}$ (resp. pkM =$h_M^{msk}$).

**3.3.2 Withdrawal Protocol:** The withdrawal protocol allows U to withdraw a coin with monetary value $2^L$ from the bank. At first, U computes all keys using the new binary tree algorithm which is presented in Figure 1. To be compatible with the bounded accumulator, the user obtains the new keys $hk_{L,0} = HL(k_{L,0}), hk_{L,1} = HR(k_{L,1}), \cdots, hk_{L,2^L-1} =$

$\mathrm{HR}(k_{L,2^L-1}), \cdots, hk_{L-1,0} = \mathrm{HL}(k_{L-1,0}), \cdots, hk_{1,0} = \mathrm{HL}(k_{1,0}), hk_{1,1} = \mathrm{HR}(k_{1,1})$, and the value of the root is $hk_{0,0} = \mathrm{HL}(k_{0,0})$. All keys and the new keys of the binary tree are stored in a table tr. And last, U generates $2^L - 1$ random numbers $s_i$ for $i = 0, \cdots, 2^L - 1$.

U computes two accumulators [1], one is the accumulation of $2^L$ random numbers $s_i \in Zp$ for $i = 0, \cdots, 2^L - 1$, namely $\mathrm{Acc}_L = u_0^{\prod_0^{2^L-1}(\alpha_1 + s_j)}$, where $\alpha_1$ is a public parameter of the bounder accumulator. This accumulator proves that the user uses the correct random numbers to construct the leaves node. And the other one is Acc which is the accumulator of all the new keys $hk_{0,0}, hk_{1,0}, hk_{1,1}, \cdots, hk_{L,0}, hk_{L,1}, \cdots, hk_{L,2^L-1}$ namely $\mathrm{Acc} = u_0^{\prod \alpha_1 + s_j}$, where $\alpha_1$is a public parameter of the bounder accumulator. This accumulator proves that the spent node and corresponding child nodes are correct. To be compatible with the P-signature, we compute the hash value of two accumulators, namely a = HA(Acc) and aL = HA($\mathrm{Acc}_L$). The bank produces the corresponding P-signatures $\sigma, \sigma_L$ on the messages (a, usk, s) and ($a_L$,s) respectively.

It is not necessary for the bank to check if the correct keys are accumulated since the bounded accumulator is bounded and we use the new generational algorithm. The main reason is given in the following, if the user uses the incorrect key to construct the binary tree or accumulates the incorrect values, he cannot complete the spending protocol, as the merchant can compute all the descendant keys of the node spent using the random elements si.

U's input is params, pkP,usk, $2^L$ and B's input is params,pkU, skC, $2^L$. The withdrawal protocol is described as follows.

1. U chooses at random $s', r_1, r_2, r_3, r_4 \in Z_p^*, d_1 = g^{r_1}, d_2 = g^{r_2}, d_3 = g^{r_3}, d_4 = g^{r_4}, y \in G_2$. U computes AFG commitments [24] $C_{s'} = h^{s'} y^{r_1}, C_a = h^a y^{r_2}, C_{a_L} = h^{a_L} y^{r_3}, C_{usk} = h_U^{usk} y^{r_4}$ . And then, the user gives the proofs [28] in zero-knowledge that he knows the opening to these values as follows.

$$\pi_{s'} = \{(s' : C_{s'}) : \hat{e}(g, C_{s'})\hat{e}(d_1, y^{-1}) = \hat{e}(g, h^{s'})\},$$
$$\pi_a = \{(a : C_a) : \hat{e}(g, C_a)\hat{e}(d_2, y^{-1}) = \hat{e}(g, h^a)\},$$
$$\pi_{a_L} = \{(a_L : C_{a_L}) : \hat{e}(g, C_{a_L})\hat{e}(d_3, y^{-1}) = \hat{e}(g, h^{a_L})\},$$
$$\pi_{usk} = \{(usk : C_{usk}) : \hat{e}(g, C_{usk})\hat{e}(d_4, y^{-1}) = \hat{e}(g, h^{usk})\}.$$

At last, U sends $\{d_1, d_2, d_3, d_4, y, C_{s'}, \pi_{s'}, C_a, \pi_a, C_{a_L}, \pi_{a_L}, C_{usk}, \pi_{usk}\}$ to B.

2. If the proofs verifies, B chooses at random $r'$ and computes $C_s' = y_1^{r_1 h^{r'}}$ and corresponding P-signatures $\sigma = \mathrm{PSign}(a, usk, s), \sigma_L = \mathrm{PSign}(a_L, s)$, where $r', r_1, x_1 \in Z_p^*, y_1 \in G_2$. And last, B sends $C_s', r', \sigma$ and $\sigma_L$ to the user.

3. U sets $s = s' + r'$, and updates commitment $C_s'$ into commitment Cs = $C_s' h^{r'}$. Then U verifies $\sigma$ and

$\sigma_L$. If the verifies are correct, U obtains the coin $co_U$= (a, aL,s,usk,, $\sigma, \sigma_L$), where a = HA(Acc); $a_L$ =HA($\mathrm{Acc}_L$).

**3.3.3. Spending Protocol:** We suppose that the user wants to spend a coin of monetary value $2^l$. The protocol works as follows:

1. M chooses at random $x_2, r \in Z_p^*, d_5 = g^r, y_2 \in G_2$, computes the AFG commitment [24] $C_{msk} = h_M^{msk} y_2^r$ and the proof [28] $\pi_{msk} = \{(msk: C_{msk}): \hat{e}(g, C_{msk})\hat{e}(d_5, y^{-1}) = \hat{e}(g, h^{msk})\}$.

And last, M sends (d5, y2,Cmsk, $\pi_{msk}$) to U. M computes R = H($2^l$||mpk||time), where $2^l$ is the monetary value of the spent coin and time is the current spending time.

2. If the proof $\pi_{msk}$ is correct, U also computes R = H($2^l$||mpk||time). U chooses a node at level $L-l$ which has not been spent and sends the corresponding random elements $s_{j2^l}, s_{j2^l+1}, \cdots, s_{(j+1)2^l-1}$ to M, and then obtains the serial number S = hki,j . U also chooses $r_U, x_3 \in Z_n, y_3 = h^{x_3}$ and computes a commitment $C_{K_U} = h^{usk} y_3^{r_U}$ to $K_U = h^{usk}$, where $pk_U = \hat{e}(g, h^{usk}) = \hat{e}(g,h)^{usk}$. Meanwhile, U computes the security tag $T = pk_U \hat{e}(g, h^{l \cdot hk_{i,j} \cdot R \cdot s})$. Note that in [12], the spent node serial number is represented by the serial number of its two child nodes. In our paper, the serial number is known by the merchant and the bank. The serial number is generated by multiplication of the corresponding leaves node which is obtained by the random number. Thus, it cannot supply any information to the adversary. To protect the public key of the user, we use the secret value s which are only known by U in the security tag. And last, U gives the following NIZK proof that the spent node is accumulated correctly and T is correctly formed.

$$\pi_{coin} \leftarrow NIZK\{((Acc: C_{Acc}), (w: C_w), (w_L: C_{w_L}), (Acc_L: C_{Acc_L}),$$
$$\{a_L: C_{a_L,i}\}_{i=1}^3 \wedge \{s: C_{s,i}\}_{i=1}^3 \wedge \{\sigma_{L,i}: C_{\sigma_L,i}\}_{i=1}^3 \wedge \{a: C_{a.i}\}_{i=1}^3, \wedge \{usk: C_{usk,i}\}_{i=1}^3 \wedge \{\sigma_i: C_{\sigma_i}\}_{i=1}^3$$
$$\wedge (s: C_s) \wedge (l: C_l) \wedge \{(l: C_l)_{i=1}^3\}:$$
$$\hat{e}(Acc, \hat{v}_0) = \hat{e}(W, \hat{v}_0 \prod_{j=1}^{l_2} \hat{v}_j^{p_j}) \wedge \sigma = \text{SPSign}(a, usk, s) \wedge$$
$$\hat{e}(Acc_L, \hat{v}_0) = \hat{e}(W_L, \hat{v}_0 \prod_{j=1}^{l_2} \hat{v}_j^{p_j}) \wedge \sigma_L = \text{SPSign}(a_L, s) \wedge$$
$$T = \hat{e}(g, h^{usk}) \cdot \hat{e}(g, h^{l \cdot hk_{i,j} \cdot R \cdot s}) \wedge \Sigma_l = PSign\{l\}\}.$$

The proof of accumulator is presented in Section 3.1. $\hat{e}(Acc, \hat{v}_0) = \hat{e}(W, \hat{v}_0 \prod_{j=1}^{l_2} \hat{v}_j^{p_j})$ proves that the spent node and the corresponding child node are correctly accumulated in Acc. $\hat{e}(Acc_L, \hat{v}_0) = \hat{e}(W_L, \hat{v}_0 \prod_{j=1}^{l_2} \hat{v}_j^{p_j})$ proves that the corresponding leaves node of the spent node are correctly accu-mulated in $Acc_L$. $\sigma = \text{SPSign}(a, usk, s)$ and $\sigma_L = \text{SPSign}(a_L, s)$ give the proofs that Acc and $Acc_L$ are signed by the bank. $T = \hat{e}(g, h^{usk}) \cdot \hat{e}(g, h^{l \cdot hk_{i,j} \cdot R \cdot s})$ proves that the security tag is correctly formed. $\Sigma_l = \text{PSign}\{l\}$ gives a proof of the ` is signed by the bank. l in the security tag is the same as the l signed in $\Sigma_l$. Thus the user proves that the spent node comes from the correct level.

3. M firstly verifies the structure of the binary tree is correct. M computes the corresponding leaves node of the spent node using the random elements

$s_{j2^l}, s_{j2^l+1}, \cdots, s_{(j+1)2^l-1}$ and the two hash functions HL and HR. M obtains keys $k_{j2^l}, k_{j2^l+1}, \cdots, k_{(j+1)2^l-1}$ and the new values

$hk_{L,j2^l}, hk_{L,j2^l+1}, \cdots, hk_{L,(j+1)2^l-1}$. Then M also computes the key of the spent node by executing the algorithm presented in Figure 2. And last, Mverifies the proof fiS. If these are correct, M obtains the coin $co_M = \{2^l, S, s_{j2^l}||s_{j2^l+1}||\cdots||s_{(j+1)2^l-1}, T, R, r_l, \pi_S\}$.

3.3.4 Deposit Protocol

M deposits a coin $co_M = \{2^l, S, K, T, R, r_l, \pi_S\}$ to B, where $K = s_{j2^l}||s_{j2^l+1}||\cdots||s_{(j+1)2^l-1}$. At first, B checks the proof $\pi_S$. If it is not correct, B rejects the deposit. Otherwise, B checks if these random elements $s_{j2^l}, s_{j2^l+1}, \cdots, s_{(j+1)2^l-1}$ are already in its database DB. If one of these random elements is already in the database, B executes the procedure of double-spender identification. Otherwise, B adds $2^l$ leaves node into the database. Then B checks whether R is fresh. If

R is fresh; B accepts the coin $co_M = \{2^l, S, K, T, R, r_l, \pi_S\}$, credits M's account. Otherwise, M deposits the coin twice. B refuses the deposit and warns M.

**3.3.5. Double-Spender Identification:** B obtains two coins $co_1 = \{2^{l_1}, S_1, K_1, T_1, R_1, r_{l_1}, \pi_{S_1}\}$ and $co_2 = \{2^{l_2}, S_2, K_2, T_2, R_2, r_{l_2}, \pi_{S_2}\}$. We describe the following two cases.

1. If U spends the same node, then $co_1 = \{2^{l_1}, S_1, K_1, T_1, R_1, r_{l_1}, \pi_{S_1}\}$ and $co_1' = \{2^{l_1}, S_2 = S_1, K_2 = K_1, T_2, R_2, r_{l_1}, \pi_{S_2}\}$. Thus, B computes $pk_U = (T_1^{R_2}/T_2^{R_1})^{1/R_2-R_1}$;

2. If U spends the different nodes, then $co_1 = \{2^{l_1}, S_1, K_1, T_1, R_1, r_{l_1}, \pi_{S_1}\}$ and $co_1' = \{2^{l_1}, S_2, K_2, T_2, R_2, r_{l_2}, \pi_{S_2}\}$. Without loss of generality, we assume $co_1$ includes $co_2$, so the leaves node of $co_1$ includes the leaves node of $co_2$. B obtains $l_3 = \frac{l_1}{l_2}$, S3 = S1/S2. B computes the public key $pk_U = ({T_1}^{R_2}/{T_2}^{R_1 \bullet S_3 \cdot l_3})^{\frac{1}{R_2 - R_1 \bullet S_3 \cdot l_3}}$.

## 4. Efficiency Analysis

We analyze the efficiency of our scheme, Izabachene and Libert's scheme [28] and Canard and Gouget's scheme [12] from the following 5 aspects, namely the construction of the binary tree, the efficiency of the withdrawal protocol, the efficiency of the spending protocol, the efficiency of the deposit protocol and security model. According to [15], We know P-signature proofs for n messages need 8n + 12 elements ofG1, 8n + 10 elements of G2, and 32n + 44 pairings to verify. One ESS+ signature [2] needs the group element numbers of G1, G2 and Z*p are 2, 1 and 2 respectively.

We assume that C1 is the computation cost of the construction of the binary tree. C2 is the efficiency of the withdrawal protocol. C3 is the efficiency of the spending protocol. C4 is the efficiency of the deposit protocol. C5 is the security model. ME represents the number of multi-exponentiation. GE represents the number of the group elements. P represents the number of the pairings.

We will describe these in detail as follows:

The construction of the binary tree. When we construct the binary tree of monetary value 2L, Izabachene and Libert's scheme [28] does not need multi-exponentiations. The reason is that the node is represented by random number. Canard and Gouget's scheme [12] needs 2L+2-2 multi-exponentiations. However, we need 2Lmulti-exponentiations.

The efficiency of the withdrawal protocol. The withdrawal protocol of Izabachene and Libert's scheme [28] needs 80 group elements and the same number of multi-exponentiations. Canard and Gouget's scheme [12] needs L+2 Acc and L+2 Ess+ signatures. Thus it needs 6L+12 group elements and the samenumber of multi-exponentiations. Our scheme needs two P-signatures. Therefore, 151 group elements and151 multi-exponentiations are needed.

The eficiency of spending2l. In Izabachene and Libert's scheme [28], to prevent a user from double-spending, it provides some security tags and proofs of nodes from the root to the spent node. Therefore, the computational workload of the user is proportional to L-l. Thus, it needs 206+116(L-l) group elements and the same number of multi-exponentiations to compute. In order to verify the correctness, the merchant requires 412 + 206(L-l) pairings. The spending protocol of Canard and Gouget's scheme [12] needs 60 group elements, 70 multi-exponentiations and 28 pairings to verify.

On a contrary, our scheme computes 2L-l keys of leaves node, so 2L-l exponentiations are needed. However, our scheme only needs two accumulators that prove the correctness of the leaves node and the descendant node of spent node. Therefore, our scheme only needs 264 group elements and the same number of multi-exponentiations to compute. The merchant requires 520 pairings to verify the correctness.

The efficiency of the deposit protocol. In Izabachene and Libert's scheme [28], the bank needs to compare and analyze each node in DB, and decides whether the coin is valid. As for every node, the bank extracts the path from the root to the spent node, and decides if the corresponding node of the node happens a double-spending. We assume that there are k spent nodes in DB, and the spent node comes from the `th level. Therefore, every node has 2l+1-1 corresponding nodes. We check whether the2l+1-1 corresponding nodes had happened double-spending. Thus Izabachene and Libert's scheme [28]Efficient Divisible E-cash Without Random Oracle 11Needs O(k *(2l+1-1)). In contrary, our new scheme and Canard and Gouget's scheme [12] only need compare the coin's serial number with all serial numbers in DB to verify the double-spending. Therefore, we only need O (k).The security model. Canard and Gouget's scheme [12] is proven in the random oracle model.

However, our new scheme and Izabachene and Libert's scheme [28] are proven in the standard model. Based on the above analysis, we conclude that our scheme is more efficient than Izabachene and Libert's scheme [28] in the deposit protocol. In the spending protocol, our scheme is more efficient than Izabachene and Libert's scheme [28] except the computation of the leaves node. Although the efficiency of the withdrawal protocol is less efficient than Izabachene and Libert's scheme [28], the spending protocol is executed much more frequently than the withdrawal protocol. The efficiency of the spending protocol is less efficient that Canard and Gouget's scheme [12], but our scheme is proven in the standard model. Therefore, our scheme is much more desirable in practice. The comparison is given in Table 1.

**Table 1. Efficiency Comparison between Related Work and Our Proposal**

| Schemes Canard | | Gouget [12] Izabachene | Libert [28] | Ours |
|---|---|---|---|---|
| C1 | | $(2^{L+2}- 2)$ME | 0ME | $2^{L}$ME |
| C2 | | (6L + 12)GE<br>(6L + 12)ME | 80GE<br>80ME | 151GE<br>151ME |
| C3 | User | 60GE<br>70ME | 206 + 116(L-$l$)GE<br>206 + 116(L -$l$)ME | 264GE<br>$2^{L-l}$ME |
| | Merchant | 28P | 412 + 206(L-$l$)P | 520P |
| C4 | | O(k) | O(k *$(2^{l+1}- 1)$) | O(k) |
| C5 | | Random oracle model | Standard model | |

## 5. Security Analysis

Regarding the security of our construction, we have the following theorem. Theorem 2. In standard model, our divisible e-cash scheme fulfills (i) the anonymity under the SXDH assumption, zero-knowledge property of P-signature, and the soundness and indistinguishability of GS proofs; (ii) the unforgeability under the collision resistance of HA() and the assumptions that P-signature is unforgeable and the bounded accumulator scheme fulfills the bound property; (iii) the Identification of double-spenders under the soundness and indistinguishability of GS proofs and the unforgeability of the P-signature scheme; (iv) the exculpability under the soundness and indistinguishability of GS proofs and the one-more discrete logarithm assumption.

## 6. Conclusion

We proposed an efficient divisible e-cash scheme in standard model using the Groth-Sahai proof system. Firstly, a new generation algorithm of the binary tree was introduced in the paper. Thus, we improved the verifying efficiency of the merchant and the bank in the spending protocol and the deposit protocol respectively. Secondly, the membership proofs of bounded accumulator without random oracle are proven. Using the bounded accumulator, the computational efficiency of the user is relatively small in the spending protocol. And last, we analyzed the efficiency of our new scheme and made the security proof in the standard model.

## Acknowledgements

This work was supported by youth scientific research fund of hebei provincial of China (grant number QN201311603).

## References

[1] M. A. Au, W. Susilo and Y. Mu, “Practical anonymous divisible e-cash from bounded accumulators”, In Financial Cryptography'08, vol. 5143 of LNCS, (**2008**), pp. 287-301. Springer.
[2] M. A. Au, Q. Wu, W. Susilo and Y. Mu, “Compact e-cash from bounded accumulator”, In CT-RST'07, vol. 4377 of LNCS, (**2007**), pp. 178-195, Springer.
[3] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko, “The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme”, J. Cryptology, vol. 16, no. 3, (**2003**), pp. 185-215.
[4] J. Camenisch, S. Hohenberger and A. Lysyanskaya, “Compact e-cash”, In EUROCRYPT'05, (**2005**), pp. 302-321, Springer.
[5] D. Chaum, “Blind Signatures for Untraceable Payments”, In Advances in Cryptology: Proceedings of CRYP-TO'82, (**1983**), pp. 199-203, Plenum, New York, Springer.
[6] T. Nakanishi and Y. Sugiyama, “Unlinkable divisible electronic cash”, In ISW, (**2000**), pp. 121-134.
[7] T. Okamoto, “An eficient divisible electronic cash scheme”, In CRYPTO'95, (**1995**), pp. 438-451, Springer.
[8] T. Okamoto and K. Ohta, “Universal electronic cash”, In CRYPTO'91, (**1991**), pp. 324-337, Springer.
[9] J. C. Pailles, “New protocols for electronic money”, In ASIACRYPT'92, (**1992**), pp. 263-274, Springer.
[10] S. Canard and A. Gouget, “Divisible e-cash systems can be truly anonymous”, In EUROCRYPT'07, (**2007**), pp. 482-497, Springer.
[11] T. Eng and T. Okamoto, “Single-Term Divisible Electronic Coins”, In EUROCRYPT'94, vol. 950 of LNCS, (**1994**), pp. 306-319, Springer.
[12] S. Canard and A. Gouget, “Multiple Denominations in E-cash with Compact Transaction Data”, In Financial Cryptography'10, vol. 6052 of LNCS, (**2010**), pp. 82-97, Springer.
[13] M. Belenkiy, M. Chase, M. Kohlweiss and A. Lysyanskaya, “P-signatures and noninteractive anonymous credentials”, In TCC, volume 4948 of LNCS, (**2008**), pp. 356-374, Springer.
[14] J. Camenisch, R. Chaabouni and A. Shelat, “Eficient protocols for set membership and range proofs”, In ASIACRYPT’08, vol. 5350 of LNCS, (**2008**), pp. 234-252, Springer.
[15] M. Belenkiy, M. Chase, M. Kohlweiss,and A. Lysyanskaya, “Compact E-cash and simulatable VRFs revisited”, In PAIRING'09, vol. 5671 of LNCS, (**2009**), pp. 114-131.
[16] M. Bellare, A. Boldyreva and A. Palacio, “A uninstantiable random-oracle-model scheme for a hybrid-encryption problem”, In EUROCRYPT'04, vol. 3027 of LNCS, (**2004**), pp. 171-188, Springer.
[17] J. Groth and A. Sahai, “Efficient non-interactive proof systems for bilinear groups”, In Eurocrypt'08, vol. 4965 of LNCS, (**2008**), pp. 415-432, Springer.
[18] G. Fuchsbauer, “Commuting signatures and verifiable encryption”, In CRYPTO 2011, vol. 6632 of LNCS, pp. 224-245.
[19] M. Belenkiy, M. Chase, M. Kohlweiss and A. Lysyan- skaya, “Compact E-Cash and Simulatable VRFs Revisited”, Pairing'09, vol. 5671 of LNCS, (**2009**), pp. 114-131.
[20] G. Fuchsbauer, D. Pointcheval and D. Vergnaud, “Transferable constant-size fair e-cash”, CANS'09, vol. 5888 of LNCS, (**2009**), pp. 226-247.
[21] R. Canetti, O. Goldreich and S. Halevi, “The random oracle methodology, revisited”, In 30th AcM STOC, (**1998**), pp. 209-218, ACM Press.
[22] E. Ghadafi, N. P. smart and B. Warinschi, “Groth-Sahai proofs revisited”, In Public Key Cryptography, (**2008**), pp. 177-192.
[23] M. Chase and A. Lysyanskaya, “Simulatable vrfs with applications to multi-theorem nizk”, In CRYPTO'07, vol. 4622 of LNCS, (**2007**), pp. 101-115.
[24] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo, “Structure-preserving signatures and commitments to group elements”, In CRYPTO'10, vol. 6223 of LNCS, (**2010**), pp. 209-236, Springer.
[25] A. H. Chan, Y. Frankel and Y. Tsiounis, “Easy come-easy go divisible cash”, In EUROCRYPT'98, vol. 1403 of LNCS, (**1998**), pp. 561-575.
[26] G. Fuchsbauer, D. Pointcheval and D. Vergnaud, “Transferable constant-size fair e-cash”, In CANS'09, vol. 5888 of LNCS, (**2009**), pp. 226-247.
[27] O. Blazy, S. Canard, G. Fuchsbauer, A. Gouget, H. Sibert and J. Traore, “Achieving optimal anonymity in transferable e-cash with a judge”, In AFRICACRYPT'11, vol. 6737 of LNCS, (**2011**), pp. 206-223.
[28] M. Izabachene and B. Libert, “Divisible e-cash in the standard model”, In Pairing'12, vol. 7708 of LNCS, (**2012**), pp. 314-332.
[29] J. Cohen Benaloh and M. de Mare, “One-Way Accumulators: A decentralized alternative to digital signature”, In EUROCRYPT, vol. 765 of LNCS, (**1993**), pp. 274-285.

[30] T. Acar and L. Nguyen, "Revocation for Delegatable Anonymous Credentials", In PKC, vol. 6571 of LNCS, (**2011**) pp. 423-440.

## Author

**Huo Yanling**, she was born in Xian yang city, Shaanxi province of China in 1978; research direction: The application of virtual reality technology.