Fair Multiple-bank E-cash in the Standard Model*

Jiangxiao Zhang[1], Yanwu Gao[1], Chunhui Feng[1], Hua Guo[2], Zhoujun Li[2]

[1] Mathematics and Information Technology Institute, Xingtai University, Xingtai, Hebei 054001, China

[2] State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China

Email: orange_0092008@163.com,gyanwu@163.com,
fch@xttc.edu.cn,hguo@buaa.edu.cn,zhoujun.li@263.net

Abstract. Multiple-bank e-cash (electronic cash) model allows users and merchants to open their accounts at different banks which are monitored by the Center Bank. Some multiple-bank e-cash systems were proposed in recent years. However, prior implementations of multiple-bank e-cash all require the random oracle model idealization in their security analysis. We know some schemes are secure in the random oracle model, but are trivially insecure under any instantiation of the oracle. In this paper, based on the automorphic blind signature, the Groth-Sahai proof system and a new group blind signature, we construct a fair multiple-bank e-cash scheme. The new scheme is proved secure in the standard model and provides the following functionalities, such as owner tracing, coin tracing, identification of the double spender and signer tracing. In order to sign two messages at once, we extend Ghadafi's group blind signature to a new group blind signature. The new signature scheme may be of independent interest.

Key words: Fair multiple-bank e-cash; Automorphic blind signature; Groth-Sahai proof; Group blind signature; Standard model.

1 Introduction

E-cash is the digital equivalent of regular money. Most proposed e-cash systems [2, 3, 4, 5, 6] in the literature have been developed on the assumption that users and merchants have their accounts at the same bank. But in the real world there exist the following model. Many banks issue real coin and the users open their account in different banks. The new model is multiple-bank model. To reflect real coin closer, many multiple-bank e-cash schemes [7, 8, 9, 10] have been proposed.

* This paper is the extended version of the paper [1] in CSS 2013.

A multiple-bank e-cash system should provide some basic security properties, e.g., the anonymity of users, the anonymity of banks, the transferability of coins and so on. Among them, anonymity of users and banks is also seen as a fundamental property of multiple-bank e-cash and is well-studied [8]. But anonymity can also be used for blackmailing or money laundering by criminals without revealing their identities. To make e-cash systems acceptable to government, a fair multiple-bank e-cash system should be proposed to recovering the criminal's identity.

Multiple-bank e-cash was firstly introduced by Lysyanskaya et al. [7]. In 2000, Jeong et al. constructed a multi-bank e-cash system [8] and analyzed the anonymity property. It provides both client anonymity and bank anonymity. In 2008, a multiple-bank e-cash [9] is proposed by Wang et al. However, it does not satisfy the unforgeable requirement. In order to obtain unforgeability, Chen et al. proposed an e-cash system [10] with multiple-bank.

The multiple-bank e-cash [7, 8, 9, 10] mentioned above are proposed in the random oracle model. Some results [11, 12] have shown that some schemes proven secure in the random oracle model, are not secure in the standard model. Groth and Sahai proof systems [13] are the most efficient non-interactive zero-knowledge (NIZK) proof systems for bilinear groups most heavily used in cryptographic constructions. However, these techniques used in the proposed multiple-bank e-cash [7, 8, 9, 10] do not appear compatible with the Groth-Sahai toolbox, we had to find other techniques to construct the coin. Thus, how to construct a fair multiple-bank e-cash and prove its security in the standard model is a challenging and meaningful problem.

In this paper, we try to solve the above problems by proposing a fair multiple-bank e-cash in the standard model. The new fair multiple-bank e-cash has the following functionalities, such as owner tracing, coin tracing, identification of the double spender and signer tracing (More details on these concepts are given in Section 2.3). Our contributions[1] are listed as follows.

1. To obtain the group blind signature which can sign two messages at once, we extend Ghadafi's group blind signature [14] to the group blind signature which can sign two messages at once.
2. Using the new group blind signature, the automorphic blind signature [15] and Groth-Sahai proof system, we construct a fair multiple-bank e-cash.

[1] Part of the approach has been published in our previous paper [1]. Compared to the work in [1], we have two enhancements in this paper: (1) We extend Ghadafi's group blind signature to the group blind signature which can sign two messages at once. (2) We achieve the fair properties, such as owner tracing, coin tracing, identification of the double spender and signer tracing.

3. We also modify and extend the model of the transferable e-cash to include the bank anonymity. Moreover, we give the security analysis of the new scheme in the standard model.

Paper Outline. The rest of the paper is organized as follows. In Section 2 we present preliminaries on the various cryptographic tools and assumptions. Definitions and the security properties of divisible e-cash are presented in Section 3. In Section 4, we give our new group blind signature. In Section 5, we present our construction and give efficiency analysis. We analyze the anonymity revocation in Section 6. In Section 7, We give the security proof. Finally we conclude in Section 8.

2 Preliminaries

This section introduces some preliminaries which will be used in this paper.

2.1 Mathematical Backgrounds

Our scheme uses the following mathematical backgrounds.

Bilinear Map. A pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ is a bilinear mapping from two group elements to a group element [13].

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_3$ are cyclic groups of prime order p. The elements G, H generate $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2$ is a non-degenerate bilinear map, so $\hat{e}(G, H)$ generates G_3 and for all $a, b \in \mathbb{Z}_n$ we have $\hat{e}(G^a, H^b) = \hat{e}(G, H)^{ab}$.
- We can efficiently compute group operations, compute the bilinear map and decide membership.

Diffie-Hellman pair. A pair $(X, Y) \in \mathbb{G}_1 \times \mathbb{G}_2$ is defined as a $Diffie-Hellman\ pair$ [16], if there exists $a \leftarrow \mathbb{Z}_p$ such that $X = G^a, Y = H^a$, where G, H generate $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We denote the set of $\mathcal{DH}$ pairs by $\mathcal{DH} = \{(G^a, H^a)|a \in \mathbb{Z}_p\}$.

2.2 Mathematical Assumptions

The unforgeability of the automorphic blind signature and our new group blind signature relies on the Asymmetric Weak Flexible Computational Diffie-Hellman (AWF-CDH) [17] and q-Asymmetric Double Hidden Strong Diffie-Hellman (q-ADH-SDH) [15]. The pseudo-randomness of Belenkiy, Chase, Kohlweiss and Lysyanskaya's (BCKL's) simulatable verifiable random functions (sVRF) is based on the q-Decisional Diffie-Hellman Inversion (q-DDHI) assumption [3]. The zero-knowledge of the Groth-Sahai proof system rests on the Symmetric External Diffie-Hellman (SXDH).

Definition 1 (SXDH). Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order generated by G_1 and G_2, respectively, and let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ be a bilinear map. The SXDH assumption states that given $(G_i, G_i^{a_i}, G_i^{b_i}, G_i^{c_i})$ for random $a_i, b_i \in \mathbb{Z}_p (i = 1, 2)$, it is hard to decide whether $c_i = a_i b_i$ or c_i is random.

Definition 2 (AWF-CDH). Let $G_1 \leftarrow \mathbb{G}_1, G_2 \leftarrow \mathbb{G}_2$ and $a \leftarrow \mathbb{Z}_p$ be random. Given $(G_1, A = G_1^a, G_2)$, it is hard to output $(G_1^r, G_1^{ar}, G_2^r, G_2^{ar})$ with $r \neq 0$, i.e., a tuple (R, M, S, N) that satisfies

$$\hat{e}(A, S) = \hat{e}(M, G_2), \hat{e}(M, G_2) = \hat{e}(G_1, N), \hat{e}R, G_2 = \hat{e}(G_1, S).$$

Definition 3 (q-ADH-SDH). Let $G, F, K \leftarrow \mathbb{G}_1, H \leftarrow \mathbb{G}_2$ and $x, c_i, v_i \leftarrow Z_p$ be random. Given $(G, F, K, G^x; H, Y = H^x)$ and $(A_i = (K \cdot G^{v_i})^{\frac{1}{x+c_i}},$
$B_i = F^{c_i}, D_i = H^{c_i}, V_i = G^{v_i}, W_i = H^{v_i})$ for $1 \leq i$
$\leq q - 1$, it is hard to output a new tuple $(A = (K \cdot G^v)^{\frac{1}{x+c}}, B = F^c, D = H^c, V = G^v, W = H^v)$ with $(c, v) \neq (c_i, v_i)$ for all i. i.e., one that satisfies

$$\hat{e}(A, Y \cdot D) = \hat{e}(K \cdot V, H), \hat{e}(B, H) = \hat{e}(F, D), \hat{e}(V, H) = \hat{e}(G, W).$$

Definition 4 (q-DDHI). On input $g, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^q} \in \mathbb{G}$ for a random $\alpha \leftarrow \mathbb{Z}_p$, it is computationally infeasible to distinguish $g^{\frac{1}{\alpha}}$ from a random element of $\mathbb{G}$ with probability non-negligibly better than 1/2.

Note that our BCKL's sVRF requires that the q-DDHI assumption holds either in $\mathbb{G}_1$ or $\mathbb{G}_2$. Without loss of generality we fix this group to be $\mathbb{G}_1$.

2.3 Fair Properties.

A fair multiple-bank e-cash should provide the following properties.

- Owner tracing. The owner tracing is that the Center Bank can recover the identity of the coin in case of dispute.
- Coin tracing. The coin tracing is that the Center Bank can trace the coin from the record of the withdrawal protocol which is kept in the bank.
- Identification of the double spender. When a double-spending had happened, the bank can recover the identity of the double spender.
- Signer tracing. The signer tracing is that the Center Bank can recover the signer's identity in case of dispute.

2.4 Useful Tools

Now we revisit four useful tools: Groth-Sahai Proof, Automorphic Blind Signature, Ghadafi's Group Blind Signature and Strongly Simulatable Verifiable Random Functions (sVRF).

2.4.1 Groth-Sahai Proofs

Groth and Sahai [13] constructed an efficient non-interactive zero-knowledge proof systems. The class of equations for our proof system are pairing-product equations (PPE). A PPE is an equation of the form

$$\prod_{j=1}^{n} \hat{e}(A_j, Y_j) \prod_{i=1}^{m} \hat{e}(X_i, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} \hat{e}(X_i, Y_j)^{\gamma_{i,j}} = t_T,$$

where $X_1, \cdots, X_m \in \mathbb{G}_1$ and $Y_1, \cdots, Y_n \in \mathbb{G}_2$ are the secret variables and $A_i \in \mathbb{G}_1, B_i \in \mathbb{G}_2, \gamma_{i,j} \in \mathbb{Z}_p$ and $t_T \in \mathbb{G}_3$ are public constants. We use ε as a shorthand for the above pairing product equation.

Groth-Sahai proofs can be instantiated under different security assumptions but since as noted by [18] the most efficient Groth-Sahai proofs are those instantiated under the SXDH assumption, we will be focusing on this instantiation. Following the Ghadafi's definitions [14], the proof system consists of the algorithms $GS = (GSSetup, GSProve, GSVerify, GSExtract, GSSimSetup, GSSimProve)$. For ease of exposition, we also define the algorithm $GSPOK(crs, \{w_1, \ldots, w_i\}, \{\varepsilon_1 \wedge \ldots \wedge \varepsilon_j\})$ [14] which proves j multiple equations $\{\varepsilon_1 \wedge \ldots \wedge \varepsilon_j\}$ involving witness $(w_1, \ldots, w_i)$ and returns a vector of size j of Groth-Sahai proofs. We refer to Appendix A for the detailed descriptions of SXDH-based commitments, randomization of the SXDH-based commitments, SXDH-based Proof and randomization of the SXDH-based Proof.

Groth-Sahai Proof of Committing to Constants. Groth-Sahai proofs has many additional properties [16], of which we use the Lemma 5 in [16]. Note that we need to generate the Groth-Sahai proof of committing to the user's public key in our paper.

2.4.2 Automorphic Blind Signature

Abe et al. proposed an automorphic blind signature scheme [15] which is structure-preserving signatures. It allows a user to obtain signatures on messages hidden from the signer. The message signed may be one message or message vectors. We define the automorphic blind signature to sign one message as $ABSign()$ and the verification of the signature as $ABSVerify()$. We refer to Appendix B for the detailed descriptions of Abe et al.'s automorphic blind signature on one message and two messages.

Note that in the joining protocol of our paper, the group manager uses the automorphic blind signature on one message to issue a certificate.

2.4.3 Ghadafi's Group Blind Signature

Ghadafi constructed a group blind signature [14] which provides the dual privacy requirement. On the one hand, the signer can hide his identity and parts of the signature that could identify him. On the other hand, the user can hide the message and parts of the signature which could lead to a linkage between a signature and its sign request. Ghadafi presented two example instantiations. We use the first construction. We refer to Appendix C for the detailed descriptions of Ghadafi's group blind signature.

Note that the Ghadafi's group blind signature can only be used to sign a single message. Our multiple-bank need the bank to be able to sign two messages, thus we extend the construction to sign two messages at once in the Section 4. In order to extend the ghadafi's group blind signature to sign two messages, we employ the automorphic blind signature on message vectors.

2.4.4 BCKL's sVRF

BCKL's sVRF is the first efficient fully simulatable sVRF with a polynomial sized output domain. Belenkiy et al. revisit the simulatable verifiable random functions [3] to obtain the BCKL's sVRF construction. It will be in the bilinear group setting. The new construction would be compatible with the Groth-Sahai proof system. We refer to Appendix D for the detailed descriptions.

Note that to avoid forging the user's coin, we use BCKL's sVRF to construct the security tag of the coin.

3 Definitions for Fair Multiple-bank E-cash

Our model builds on the model for the transferable e-cash from [4]. We convert the anonymity of the model to that of uses and extend it to include the bank anonymity. The parties involved in a multiple-bank e-cash are: a Center Bank CB, a group manager GM, many banks B_i and users U_i. CB recovers the identity of the bank in case of dispute. GM controls which bank can join the group. Each bank can dispense coin. The users and merchants open their accounts in different bank. Note that merchants M are the special users and the opener is the Central Bank.

In the following, we firstly describe the algorithms for fair multiple-bank e-cash.

3.1 Algorithms

We represent a coin as $coin$, which its identity is Id. A fair multiple-bank e-cash system, denoted Π, is composed of the following procedures, where λ is a security parameter.

- ParamGen(1^λ) is run by some trusted third party (TTP) which takes as input 1^λ and outputs the public key *mbpk* for the fair multiple-bank e-cash system, the group manager's key pair (pk_{GM}, sk_{GM}) and the Center Bank's key pair (ck_{cb}, ek_{cb}).
- BKeyGen() is run by the banks B_i, to generate his pairs of personal secret/public key pairs $(bssk_i, bspk_i)$ and $(bgsk_i, bgpk_i)$. The former is used in the joining protocol. The latter is used for issuing the group blind signature in the withdrawal protocol. We assume that the public key is publicly accessible.
- UKeyGen() is run by the users U_i, to generate his pair of personal key pair (sk_{U_i}, pk_{U_i}). Note that the merchants M are special users.
- Join($B_i(mbpk, i, bssk_i)$,$GM(sk_{GM}, i, bspk_i)$) is an interactive protocol between a bank B_i and the group manager GM. B_i generates the signature sig_i on the second secret/public key $(bgsk_i, bgpk_i)$ using the first secret/public key $(bssk_i, bspk_i)$ and then sends sig_i and $bgpk_i$ to GM. GM generates the certificate $cert_i$ which is used for joining the group by B_i and then sends $cert_i$ to B_i. If successful, GM stores $bgpk_i$ and sig_i into the database reg_i. B_i becomes a member of the group and stores $(bgsk_i, bgpk_i)$ and $cert_i$ into the database gsk_i.
- Withdraw($U_i(mbpk, L)$,$B_i(bgsk_i, bgpk_i, pk_{U_i})$) is an interactive protocol between a user U_i and an anonymous bank B_i. If the protocol completes successfully, U_i obtains a coin *coin* of monetary value L. B_i does not learn what the coin was. U_i knows who issued the coin, but others only know that the coin is issued by the bank and does not know which bank issued the coin.
- Spend($U_i(coin, pk_M, sk_{U_i}, pk_{U_i}, ck_{cb})$,$M(sk_M, pk_M, mbpk)$) is an interactive protocol between a user U_i and a merchant M. If the protocol completes successfully, U_i obtains the corresponding serves. M obtains a coin *coin*.
- Deposit($M(coin, sk_M, pk_M, mbpk)$,$B_j(pk_M, DB, mbpk)$) is an interactive protocol between a merchant M and a bank B_j, where B_j may be B_i or not. If *coin* is not valid, B_j outputs $\perp$. Else, B_j checks whether the database DB contains a coin $coin'$ in which the serial number is the same as the one in *coin*. If DB contains $coin'$, B_j outputs $(coin, coin')$ and executes algorithm Identify. Else, B_j sends *coin* to Central Bank CB who adds *coin* to the database DB, and credits M's account. Note that DB is regularly updated by CB.
- Identify($coin, coin'$) is a deterministic algorithm executed by B_j. It outputs the public key pk_{U_i} and a proof τ_G.
- VerifyGuilt($coin, pk_{U_i}, \tau_G$) is a deterministic algorithm that can be executed by anyone. It outputs 1 if τ_G is correct, and 0 otherwise.

- Open($mbpk, ek_{cb}, reg_j, coin$) is a deterministic algorithm in which the Center Bank uses his extraction key ek_{cb} to recover the identity pk_{B_j} of the bank and produces a proof τ_S attesting to this claim.
- VerifySigner($mbpk, j, bspk_j, coin, \tau_S$) is a deterministic algorithm which inputs an index j and returns 1 if the coin $coin$ was signed by the bank B_j, and 0 otherwise.

In this paper, the Center Bank is a trust organizer. The Center Bank cannot recover an honest user's identity except that the bank supplies two double-spending coins. This is also the requirement of the fair e-cash [19].

3.2 Global Variables and Oracles

This section gives the adversary's means of interaction with his challenger in the security experiments of a fair multiple-bank e-cash system. Therefore, we introduce global variables and oracles.

The global lists are: HBL is a set of honest banks; CBL is a set of corrupt banks; HUL is a set of honest users; CUL is a set of corrupt users; SC is a set of coins supplied by the adversary in the withdrawal protocol; OC is a set of coins owned by the oracle. DC is a set of coins deposited; RU is a set of users who have received a coin from the adversary; SU is a set of users who have spent a coin to the adversary. The lists $HBL, CBL, HUL, CUL, RU, SU$ are empty at initialization. SC, OC, DC are 0 at initialization. The set of oracles the adversary has access to in the experiments are defined as follows:

- $AddB(i)$: The adversary can use this oracle to add an honest bank B_i to the group.
- $CrptB(i, bspk_i)$: The adversary can use this oracle to create a new corrupt bank B_i, where B_i's public key is $bspk_i$ and the secret key is $\perp$.
- $AddU(i)$: The adversary can use this oracle to add an honest user U_i.
- $CrptU(i, pk_{U_i})$: The adversary can use this oracle to create a new corrupt user U_i, where U_i's public key is pk_{U_i} and the secret key is $\perp$.
- $Join(i)$: The oracle plays the bank side to engage in the joining protocol with the honest group manager.
- $Issue(i)$: This oracle models the scenario that the adversary has corrupted the group manager. The adversary uses this oracle to engage in the joining protocol with an honest bank.
- $BWith(i)$: This oracle plays the bank side of a withdrawal protocol. The adversary supplies the coin to an honest user and updates SC by adding $(i, coin)$.

- $UWith(i)$: This oracle plays the user in a withdrawal protocol. The adversary obtains the coin and updates OC by adding $(i, coin)$.
- $Rcv(i)$: This oracle allows the adversary to receive a coin from an honest user U_i. The oracle plays M in the spending protocol. It updates the set OC by adding a new entry $(i, coin)$ and adds i to the set RU.
- $Spd(i)$: This oracle allows the adversary to spend a coin to an honest merchant M. The oracle plays the user U_i in the spending protocol.
- $BDepo()$: The merchant M uses the oracle to deposit a coin in the deposit protocol. This oracle gives the output of the deposit protocol and updates the set DC.
- $UDepo(j, coin)$: The bank B_j uses the oracle to accept a coin in the deposit protocol. This oracle plays the merchant M.
- $Open(m, \Sigma)$: This oracle allows the adversary to ask for signatures to be opened by revealing the identity of the bank B_j who signed them.
- $Idt(coin, coin')$: This oracle plays the role of the bank B_j in the Identify procedure.

3.3 Security Properties

In this section, we define the security properties by using a set of experiments.

3.3.1 Anonymity

Unlike the model of [3], ours adopts a indistinguishability-based formulation of anonymity. Anonymity includes the bank anonymity and the user anonymity. In the following, we give the formal definition of the bank anonymity and the user anonymity.

The bank anonymity guarantees that the adversary is unable to distinguish which bank produced a signature. We require that the adversary is given two banks of its choice, the adversary still cannot distinguish which of the two banks produced the signature. Formally, we have the following definitions based on the experiment given below. Note that b^* represents 0 or 1.

Definition 5 (Bank Anonymity). Let Π be a multiple-bank e-cash system. For an adversary $\mathcal{A}$ and $\lambda \leftarrow \mathbb{N}$, we let $Succ_{\Pi,\mathcal{A}}^{BAnon-b}(\lambda) = Pr[Exp_{\Pi,\mathcal{A}}^{BAnon-1}(\lambda) = 1] - Pr[Exp_{\Pi,\mathcal{A}}^{BAnon-0}(\lambda) = 1]$. Π is said to be bank anonymity if the function $Succ_{\Pi,\mathcal{A}}^{BAnon-b}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Experiment $Exp_{\Pi,\mathcal{A}}^{BAnon-b}(\lambda)$

- $(mbpk, pk_{GM}, sk_{GM}, bssk_i, bspk_i, ck_{cb}, ek_{cb}) \leftarrow AllGen(1^\lambda)$; $HBL := \phi$; $HUL := \phi$; $CBL := \phi$; $CUL := \phi$; $st := \phi$.
- $(B_0, B_1, st_1) \leftarrow \mathcal{A}(mbpk, pk_{GM} : CrptB, AddB, BWith, Join, Open, Rcv, Spd, UDepo, Idt)$.

- $(st_2, coin) \leftarrow Withdraw(\cdot, B_b)$.
- $b^* \leftarrow \mathcal{A}(st_2, coin : CrptB, AddB, BWith, Join, Open, Rcv, Spd, UDepo, Idt)$.
- Return b^*.

The user anonymity guarantees that the adversary, even helped by malicious users and banks, can not learn anything about a spending. We require that the adversary is given two users of its choice, the adversary still cannot distinguish which of the two users spends the coin to the adversary. Formally, we have the following definition based on the experiment given below.

Definition 6 (User Anonymity). Let Π be a multiple-bank e-cash system. For an adversary $\mathcal{A}$ and $\lambda \leftarrow \mathbb{N}$, we let $Succ_{\Pi,\mathcal{A}}^{UAnon-b}(\lambda) = Pr[Exp_{\Pi,\mathcal{A}}^{UAnon-1}(\lambda) = 1] - Pr[Exp_{\Pi,\mathcal{A}}^{UAnon-0}(\lambda) = 1]$. Π is said to be user anonymity if the function $Succ_{\Pi,\mathcal{A}}^{UAnon-b}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Experiment $Exp_{\Pi,\mathcal{A}}^{UAnon-b}(\lambda)$

- $(mbpk, pk_{GM}, sk_{GM}, bssk_i, bspk_i, ck_{cb}, ek_{cb}) \leftarrow AllGen(1^\lambda); HBL := \phi; HUL := \phi; CBL := \phi; CUL := \phi; st := \phi$.
- $(U_0, U_1, st_1) \leftarrow \mathcal{A}(mbpk, pk_{GM} : CrptU, AddU, Issue, UWith, Rcv, Spd, UDepo, Idt)$.
- $(st_2, coin) \leftarrow Spend(U_b, \cdot)$
- $b^* \leftarrow \mathcal{A}(st_2, coin : CrptU, AddU, Issue, UWith, Rcv, Spd, UDepo, Idt)$.
- Return b^*.

3.3.2 Unforgeability

Unforgeability guarantees that no collection of users can ever spend more coins than they withdrew. Formally, we have the following definition based on the experiment given below.

Definition 7 (Unforgeability). Let Π be a multiple-bank e-cash system. For an adversary $\mathcal{A}$ and $\lambda \leftarrow \mathbb{N}$, we let $Succ_{\Pi,\mathcal{A}}^{unfor}(\lambda) = Pr[Exp_{\Pi,\mathcal{A}}^{unfor}(\lambda) = 1]$. Π is said to be unforgeability if the function $Succ_{\Pi,\mathcal{A}}^{unfor}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Experiment $Exp_{\Pi,\mathcal{A}}^{unfor}(\lambda)$

- $(mbpk, pk_{GM}, sk_{GM}, bssk_i, bspk_i, ck_{cb}, ek_{cb}) \leftarrow AllGen(1^\lambda); HBL := \phi; HUL := \phi; CBL := \phi; CUL := \phi; st := \phi; cont := true$.
- While ($cont$=true)do {

  $(cont, st) \leftarrow \mathcal{A}(st, mbpk : AddB, CrptB, Issue, BWith, Rcv, Spd, BDepo)$.

  Let q_W be the number of successful calls to $BWith$.

  Let q_D be the number of successful calls to $BDepo$.

  If $q_W \cdot L < q_D \cdot L$ then return 1;}

– Return $\perp$.

3.3.3 Identification of Double-spender

The identification of double-spender guarantees that no collection of users, collaborating with the merchant, can spend a coin twice without revealing one of their identities. Formally, we have the following definition based on the experiment given below.

Definition 8 (Identification of Double-spender). Let Π be a multiple-bank e-cash system. For an adversary $\mathcal{A}$ and $\lambda \leftarrow \mathbb{N}$, we let $Succ^{ident}_{\Pi,\mathcal{A}}(\lambda) = Pr[Exp^{ident}_{\Pi,\mathcal{A}}(\lambda) = 1]$. Π identifies double spenders if the function $Succ^{ident}_{\Pi,\mathcal{A}}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Experiment $Exp^{ident}_{\Pi,\mathcal{A}}(\lambda)$

– $(mbpk, pk_{GM}, sk_{GM}, bssk_i, bspk_i, ck_{cb}, ek_{cb}) \leftarrow AllGen(1^{\lambda})$; $HBL := \phi; HUL := \phi; CBL := \phi; CUL := \phi; st := \phi; cont := true.$

– While ($cont$=true)do {

$(st) \leftarrow \mathcal{A}(st, mbpk : AddB, CrptB, Issue, BWith, Rcv, Spd, BDepo, Idt)$.

If a call to $BDepo$ outputs $(coin^*)$ then $cont \leftarrow false.$}

– If $Identify(coin^*, ek_{cb}) = 0 \wedge VerifyGuilt(coin^*, pk_{i^*}, \tau_G) = 0$ then return 1.

– Return $\perp$.

3.3.4 Exculpability

The exculpability guarantees that the bank, even when colluding with malicious users, cannot falsely accuse honest users of having double-spent a coin. Formally, we have the following definition based on the experiment given below.

Definition 9 (Exculpability). Let Π be a multiple-bank e-cash system. For an adversary $\mathcal{A}$ and $\lambda \leftarrow \mathbb{N}$, we let $Succ^{excul}_{\Pi,\mathcal{A}}(\lambda) = Pr[Exp^{excul}_{\Pi,\mathcal{A}}(\lambda) = 1]$. Π identifies double spenders if the function $Succ^{excul}_{\Pi,\mathcal{A}}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Experiment $Exp^{excul}_{\Pi,\mathcal{A}}(\lambda)$

– $(mbpk, pk_{GM}, sk_{GM}, bssk_i, bspk_i, ck_{cb}, ek_{cb}) \leftarrow AllGen(1^{\lambda})$; $HBL := \phi; HUL := \phi; CBL := \phi; CUL := \phi; st := \phi; cont := true.$

– $(coin^*, i^*, \tau^*) \leftarrow \mathcal{A}(st, mbpk : AddU, CrpU, Join, UWith, Rcv, Spd, UDepo, Idt)$.

– If $Identify(coin^*) = (pk_{i^*}, \tau^*)$ and $VerifyGuilt(coin^*, pk_{i^*}, \tau^*) = 1$ and $sk_{i^*} \neq \perp$, return 1;

– Return $\perp$.

4 New Group Blind Signature.

Ghadafi constructed a group blind signature [14] which can only be used to sign one message. In Ghadafi's construction, he uses the automorphic blind signature [15] which only signs one message to construct the group blind signature. We extend the construction to sign two messages tusing the automorphic blind signature [15] which can sign two messages at once. We refer to the Appendix B for the automorphic blind signature on two messages.

We assume that the two messages (V, W) and (M, N) are signed by using the new group blind signature. Note that in our new group blind signature, the Center Bank is the opener in the Ghadafi's group blind signature. We define the new construction as $NGBS()$.

The new group blind signature consists of nine algorithms: *Group-Key-Generation*, *Bank-Key-Generation*, $Join$-*Protocol*, The-$First$-*Round*, The-*Second*-*Round*, *Obtain*, $Verify$, *Open* and $Judge$. *Group-Key-Generation*(1^{λ}) generates the group public key $gpk = (bgpp, F, K, T, crs_1, crs_2, pk_{GM})$. *Bank-Key-Generation*$(bgpp)$ generates two pairs of the secret/public key (sk_i, pk_i) and (ssk_i, spk_i). $Join$-*Protocol* allows the signer si_i to join a group and obtain the group certificate $cert_i$. The-$First$-*Round* allows the user U_i to blind the messages $(V, W), (M, N)$ to U and sends U and corresponding proof Ψ to si_i. The-*Second*-*Round* allows si_i to generate the pre-signature Ω and send Ω to U_i. *Obtain* allows U_i to compute a signature Σ. $Verify$ output accept if Σ is valid. *Open* allows the opener to recover si_i's identity. *Judge* allows anyone verify the correctness of *Open*. In the following, we give the construction of the new group blind signature.

- The *Group-Key-Generation*, *Bank-Key-Generation*, $Join$-*Protocol*, *Open*, *Judge* and $Verify$ are the same as those of the Ghadafi's group blind signature. They can be found in Appendix C.
- The-$First$-*Round* $[U_i \rightarrow B_i]$. Choose $q_1, q_2 \leftarrow \mathbb{Z}_p$ and compute $Q_{11} = G^{q_1}$, $Q_{12} = H^{q_1}$,$Q_{21} = G^{q_2}$, $Q_{22} = H^{q_2}$, $U_1 = T_1^{q_1} \cdot V$ and $U_2 = T_2^{q_2} \cdot M$. U_i generates the proof Ψ_V and Ψ_M to the two messages (V, W) and (M, N). Finally, U sends (U_1, Ψ_V) and (U_2, Ψ_M) to B_i.
- The-*Second*-*Round* $[B_i \rightarrow U_i]$. If (crs_1, Ψ_V) and (crs_1, Ψ_M) are valid, B_i obtains the signatures $\sigma_0, \sigma_1, \sigma_2$ and σ_3 by using Abe et al.'s automorphic blind signature on two messages [16]. B_i also gives the proof for the correct of the certificate $\Omega_{cert_i} \leftarrow GSPOK(crs_2, \{cert_i\}, \{ABSSign(gpk, (S_1, S_2), cert_i) = 1\}$. The signatures $\sigma_0, \sigma_1, \sigma_2, \sigma_3$ and Ω_{cert_i} are a set of Groth-Sahai proofs of knowledge of values satisfying pairing product equations. By the Lemma 3 from [16], we know the Groth-Sahai proofs are homomorphic. We thus obtain the proof of the new group blind signature $\Omega = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \Omega_{cert_i}\}$. Finally, B_i sends $(R_{11}, R_{12}, R_{21}, R_{22}, \Omega)$ to U.

- *Obtain* $[U]$. If (crs_2, Ω) is valid and $\hat{e}(R_{11}, H) = \hat{e}(G, R_{12})$, $\hat{e}(R_{21}, H) = \hat{e}(G, R_{22})$, U computes $R'_{11} = R_{11} \cdot Q_{11}$, $R'_{12} = R_{12} \cdot Q_{12}$, $R'_{21} = R_{21} \cdot Q_{21}$ and $R'_{22} = R_{22} \cdot Q_{22}$. U re-randomizes Ω as Ω'. Then U gives the proof Ω' to the two messages (V, W) and (M, N). Finally, U obtains the signature $\Sigma = \Omega'$.

The unforgeability of the automorphic blind signature and the Ghadafi's group blind signature relies on the AWF-CDH and q-ADH-SDH. Our new group blind signature is obtained from the automorphic blind signature and the Ghadafi's group blind signature. Therefore, the new group blind signature is strongly unforgeable under q-ADH-SDH and AWF-CDH [14, 17].

5 Fair Multiple-Bank E-cash

Fair Multiple-bank e-cash allows users and merchants to open their accounts at different banks. It supplies the users anonymity and the banks anonymity. The Central Bank regularly updates the database DB. In the following, we give the details of this scheme.

5.1 General Description of the Scheme

We present an intuition on how our scheme is constructed. A coin is represented by a unique serial number S, where S is a Diffie-Hellman pair. Using the Diffie-Hellman pair, we can use the automorphic blind signature and new group blind signature.

The Fair multiple-bank e-cash scheme is composed of the joining protocol, the withdrawal protocol, the spending protocol, the deposit protocol, the identify procedures and the verify procedures. Before issuing the coin, the bank B_i firstly joins into a group for obtaining the certificate in the joining protocol. Then U_i withdraws a coin from B_i. U_i can spend the coin to a merchant M. Finally, M deposits the coin into B_j ($j = i$, or $j \neq i$). If B_j finds a double-spending of the coin, B_j executes the identify procedures. Anyone can verify the correctness of the double-spenders and the signer (bank). The framework map of our scheme is given in the following Figure 1.

The simple description on these protocols is given as follows.

- *The Joining Protocol*. B_i obtains the certificate for issuing the group blind signature from GM. B_i owns two pairs of secret/public key. The first is used for obtaining the certificate. The second is used for issuing the group blind signature. B_i firstly generates the signature on the second public key by using the first secret key. Then B_i sends the signature and the second public key to GM. Finally, GM generates the signature on the second public key and sends the signature (certificate) to B_i.

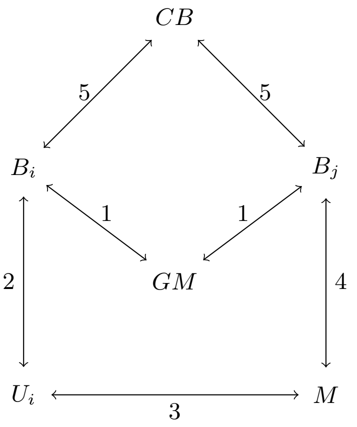

Fig. 1. Framework map.

(Note that: $i = j$ or $i \neq j$. 1 represents "the Joining Protocol". 2 represents "the Withdrawal Protocol". 3 represents "the spending Protocol". 4 represents "the Deposit Protocol". 5 represents "the Signer Tracing".)

- *The Withdrawal Protocol.* U_i withdraws a coin from B_i. U_i generates the serial number and the commitment of the serial number. Then U_i sends the commitments of the serial number and his public key to B_i. B_i verifies the correctness of the commitments. If they are correct, B_i generates the group blind signature on the commitments and sends the signature to U_i. U_i transforms the signature on the commitments to the signature on the serial number and the public key.
- *The Spending Protocol.* U_i spends a coin to M. M sends a random value to U_i. U_i firstly randomizes the commitment on U_i's public key and the signature into new commitment and signature. Meanwhile, U_i generates the security tag and the corresponding correctness proof. Then U_i sends the new coin to M. M verifies the correctness of the proof. If it is correct, M offers goods or services to B_i.
- *The Deposit Protocol.* M deposits the coin to B_j. B_j verifies the validity of the coin. If it is valid, B_j accepts the coin. Otherwise, B_j executes the following identify procedures.
- *The Identify procedures.* We assume that *coin* and *coin′* are two double-spending coins. B_j recovers the double-spender's public key by using the security tag in two double-spending coin.
- *The Verify procedures.* Anyone can verify the correctness of the double-spender and the signer by using the algorithms $VerifyGuilt$ and $VerifySigner$ respectively.

5.2 Setup

On input of 1^λ, then output is the public parameters of bilinear groups $bgpp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \hat{e}, G, H)$, where λ is the security parameter. We choose random elements $F, K, T \in \mathbb{G}_1$. On input $bgpp$ run the setup algorithms for Groth-Sahai proofs and return two reference strings crs_1, crs_2 and the corresponding extraction keys ek_1, ek_2. The two reference strings are used in constructing Groth-Sahai proofs used in the first round and the second round of the group blind signature. The Center Bank's commitment key and extraction key are $(ck_{cb} = crs_2, ek_{cb} = ek_2)$. Choose $s_{GM} \leftarrow \mathbb{Z}_p$ and output the key pair of group manager $(sk_{GM} = s_{GM}, pk_{GM} = (S_1 = G^{s_{GM}}, S_2 = H^{s_{GM}}))$. The public key of multiple-bank e-cash is $mbpk = (bgpp, F, K, T, crs_1, crs_2, pk_{GM})$. $\mathcal{H}$ is a collision-resistant hash function.

Each bank B_i chooses $s_{B_i}, s'_{B_i} \leftarrow \mathbb{Z}_p$ and creates two key pairs $(bssk_i = s_{B_i}, bspk_i = (S_1 = G^{s_{B_i}}, S_2 = H^{s_{B_i}}))$ and $(bgsk_i = s'_{B_i}, bgpk_i = (S_1 = G^{s'_{B_i}}, S_2 = H^{s'_{B_i}}))$. The first key pair is the bank's personal key pair. The second one is used for the group blind signature scheme. Each user U_i chooses $s_{U_i} \leftarrow \mathbb{Z}_p$ and generates the key pair $(sk_{U_i} = s_{U_i}, pk_{U_i} = (S_1 = G^{s_{U_i}}, S_2 = H^{s_{U_i}}))$. Each merchant M_i also creates the key pair $(sk_{M_i} = s_{M_i}, pk_{M_i} = (S_1 = G^{s_{M_i}}, S_2 = H^{s_{M_i}}))$.

5.3 The Joining Protocol

The joining protocol allows the bank to obtain a certificate from the group manager as described in the following Figure 2. In order to issue a coin, each bank firstly joins into the group whose manager is GM. Then the bank B_i obtains the certificate $cert_i$. Using the certificate and the key pair $(bgsk, bgpk)$, the bank issuing the coin. In the following, we give the details of the protocol.

1. $(B_i \rightarrow GM)$. The bank B_i generates the signature $sig_i = ABSign(bssk_i, bgpk_i)$. The signature is used to stop a corrupt bank from framing others. Then B_i sends $sig_i, bgpk_i = (S_1^{bg} = G^{s'_{B_i}}, S_2^{bg} = H^{s'_{B_i}})$ to the group manager GM.
2. $(GM \rightarrow B_i)$. GM checks whether the public key $bgpk_i$ has existed in the database DB_{pk}. If it is not, GM verifies $\hat{e}(S_1^{bg}, H) = \hat{e}(G, S_2^{bg})$ and $ABSverify(bspk_i, bgpk_i, sig_i) = 1$. If they are OK, GM generates the certificate $cert_i = ABSign(sk_{GM}, bgpk_i)$. Finally, GM stores $bgpk_i$ and $cert_i$ into reg_i and sends $cert_i$ to B_i.
3. B_i verifies the correctness of the certificate. If $ABSverify(pk_{GM}, bgpk_i, cert_i) = 1$, B_i stores $bgsk_i, bgpk_i$ and $cert_i$ into gsk_i.

B_i GM

$sig_i = ABSign(bssk_i, bgpk_i)$

$\xrightarrow{sig_i, bgpk_i}$

check $bgpk_i$ in DB_{pk}

verify $\hat{e}(S_1^{bg}, H) = \hat{e}(G, S_2^{bg})$

$ABSverify(bspk_i, bgpk_i, sig_i) = 1$

generate $cert_i$

$reg_i = \{bgpk_i, cert_i\}$

$\xleftarrow{cert_i}$

$ABSverify(pk_{GM}, bgpk_i, cert_i) = 1$

obtain $cert_i$

$gsk_i = \{bgsk_i, bgpk_i, cert_i\}$

Fig. 2. The joining protocol.

5.4 The Withdrawal Protocol

The withdrawal protocol allows U_i to withdraw a coin from B_i as described in the following Figure 3. It achieves the user anonymity and the bank anonymity. The bank anonymity guarantees that anyone does not know which bank issued the coin except U_i. The reason is that the group signature issued by the bank consists of the Groth-Sahai proofs and the proofs can be verified by the common reference string. The message m is a Diffie-Hellman pair (m_1, m_2). c_m includes the commitments to m_1, m_2 and the Pedersen commitment to m_1. $C_m^{ck_{cb}}$ is defined as the commitment to m using the commitment key ck_{cb}. In the following, we give the protocol in detail.

1. $(U_i \to B_i)$. U_i chooses $s_m \leftarrow \mathbb{Z}_p$ and generates the serial number $S = (G^{s_m}, H^{s_m})$. U_i also chooses $q_1, q_2 \leftarrow \mathbb{Z}_p$ and computes $Q_1 = G^{q_1}, Q_2 = H^{q_1}, Q_3 = G^{q_2}, Q_4 = H^{q_2}$. U_i picks at random nonces $\iota_1, \iota_2 \leftarrow \mathbb{Z}_p$. To hide the serial number, U_i generates the following commitments c_S by using the commitment key ck_{cb} and the correct proofs π_S. U_i also generates the commitment $c_{pk_{U_i}}$ and the correct proof $\pi_{pk_{U_i}}$ to U_i's public key $pk_{U_i} = (S_1 = G^{s_{U_i}}, S_2 = H^{s_{U_i}})$.

$$\begin{aligned} c_S &= (C_{G^{s_m}}^{ck_{cb}}, C_{H^{s_m}}^{ck_{cb}}, C_{Q_1}^{ck_{cb}}, C_{Q_2}^{ck_{cb}}, U_1 = T^{\iota_1} \cdot G^{s_m}), \\ \pi_S &\leftarrow GSPOK\{crs_2, \{G^{s_m}, H^{s_m}, Q_1, Q_2\}, \hat{e}(G^{s_m}, H) = \hat{e}(G, H^{s_m}) \wedge \hat{e}(Q_1, H) = \hat{e}(G, Q_2) \wedge \\ &\quad \hat{e}(T, Q_2) \cdot \hat{e}(G^{s_m}, H) = \hat{e}(U_1, H)\}, \end{aligned}$$

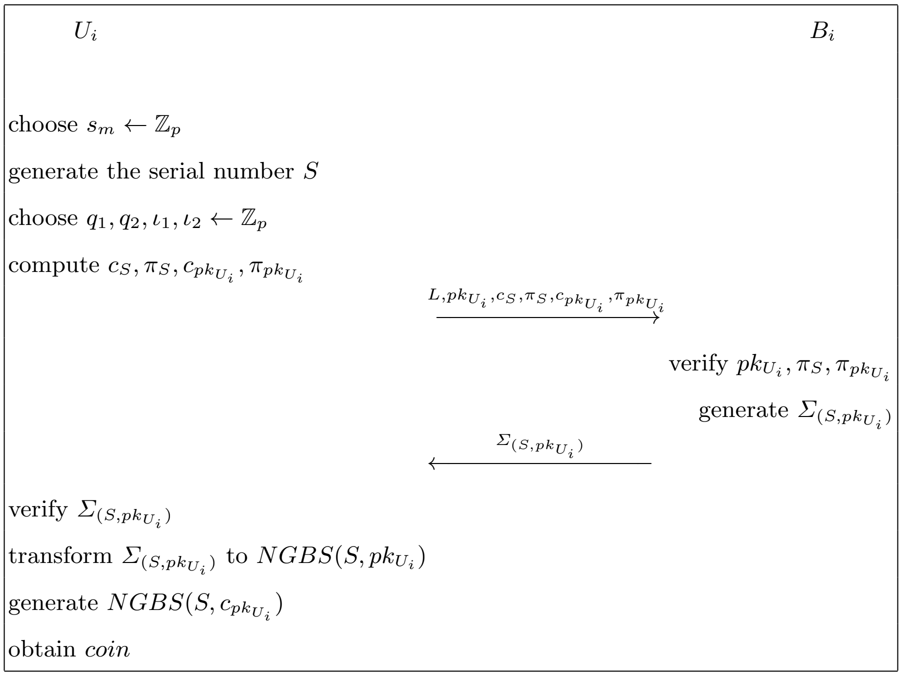
U_i
B_i
choose $s_m \leftarrow \mathbb{Z}_p$
generate the serial number S
choose $q_1, q_2, \iota_1, \iota_2 \leftarrow \mathbb{Z}_p$
compute $c_S, \pi_S, c_{pk_{U_i}}, \pi_{pk_{U_i}}$
$\xrightarrow{L, pk_{U_i}, c_S, \pi_S, c_{pk_{U_i}}, \pi_{pk_{U_i}}}$
verify $pk_{U_i}, \pi_S, \pi_{pk_{U_i}}$
generate $\Sigma_{(S,pk_{U_i})}$
$\xleftarrow{\Sigma_{(S,pk_{U_i})}}$
verify $\Sigma_{(S,pk_{U_i})}$
transform $\Sigma_{(S,pk_{U_i})}$ to $NGBS(S, pk_{U_i})$
generate $NGBS(S, c_{pk_{U_i}})$
obtain *coin*

Fig. 3. The withdrawal protocol.

$$\begin{aligned} c_{pk_{U_i}} &= (C^{ck_{cb}}_{G^{s_{U_i}}}, C^{ck_{cb}}_{H^{s_{U_i}}}, C^{ck_{cb}}_{Q_3}, C^{ck_{cb}}_{Q_4}, U_2 = T^{\iota_2} \cdot G^{s_{U_i}}), \\ \pi_{pk_{U_i}} &\leftarrow GSPOK\{crs_2, \{G^{s_{U_i}}, H^{s_{U_i}}, Q_3, Q_4\}, \hat{e}(G^{s_{U_i}}, H) = \hat{e}(G, H^{s_{U_i}}) \wedge \hat{e}(Q_3, H) = \hat{e}(G, Q_4) \wedge \\ &\quad \hat{e}(T, Q_4) \cdot \hat{e}(G^{s_{U_i}}, H) = \hat{e}(U_2, H)\}. \end{aligned}$$

Finally, U_i sends $\{L, pk_{U_i}, c_S, \pi_S, c_{pk_{U_i}}, \pi_{pk_{U_i}}\}$ to B_i, where L is the monetary value.

2. $(B_i \rightarrow U_i)$. B_i verifies the public key pk_{U_i}, π_S and $\pi_{pk_{U_i}}$. If $GSVerify(crs_2, \pi_S) = 1$ and $GSVerify(crs_2, \pi pk_{U_i}) = 1$, B_i generates the Groth-Sahai proof $\Sigma_{(S,pk_{U_i})}$ on c_S and $c_{pk_{U_i}}$ by using our new group blind signature.

   Finally, B_i sends $NGBS(S, pk_{U_i})$ to U_i.

3. U_i verifies $\Sigma_{(S,pk_{U_i})}$. If it is correct, U_i transforms $\Sigma_{(S,pk_{U_i})}$ to the group blind signature $NGBS(S, pk_{U_i})$ to S and pk_{U_i}. In the group blind signature, pk_{U_i} is a constant [14]. To hide the U_i's identity, U_i generates the group blind signature $NGBS(S, c_{pk_{U_i}})$ to S and $c_{pk_{U_i}}$ by using the Lemma 5 in [16]. Finally, U_i obtains the wallet $coin = \{S, L, c_{pk_{U_i}}, NGBS(S, c_{pk_{U_i}})\}$.

5.5 The Spending Protocol

The spending protocol allows U_i to spend a coin of monetary value L to the merchant M as described in the following Figure 4.

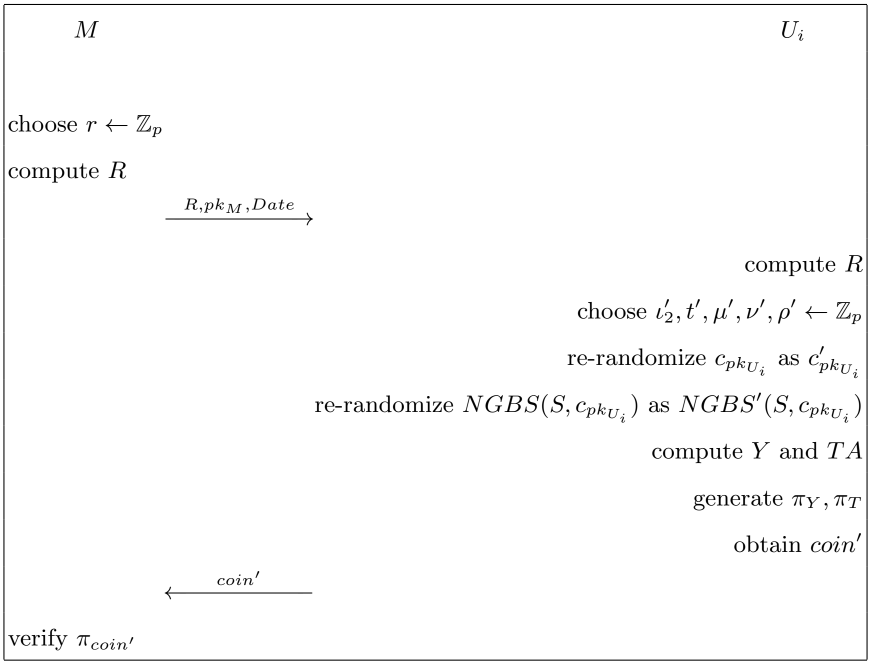
M
U_i
choose $r \leftarrow \mathbb{Z}_p$
compute R
$\xrightarrow{R, pk_M, Date}$
compute R
choose $\iota_2', t', \mu', \nu', \rho' \leftarrow \mathbb{Z}_p$
re-randomize $c_{pk_{U_i}}$ as $c'_{pk_{U_i}}$
re-randomize $NGBS(S, c_{pk_{U_i}})$ as $NGBS'(S, c_{pk_{U_i}})$
compute Y and TA
generate π_Y, π_T
obtain $coin'$
$\xleftarrow{coin'}$
verify $\pi_{coin'}$

Fig. 4. The spending protocol.

1. $(M \to U_i)$. M computes $R = \mathcal{H}(pk_M || Date)$ and sends $\{R, pk_M, Date\}$ to U_i.
2. $(U_i \to M)$. U_i also computes $R = \mathcal{H}(pk_M || Date)$. The commitment to U_i's public key is $c_{pk_{U_i}} = (C^{ck_{cb}}_{G^{s_{U_i}}}, C^{ck_{cb}}_{H^{s_{U_i}}}, C^{ck_{cb}}_{Q_3}, C^{ck_{cb}}_{Q_4}, U_2 = T^{\iota_2} \cdot G^{s_{U_i}})$. In order to hide U_i's public key, U_i chooses $\iota_2', t', \mu', \nu', \rho' \leftarrow \mathbb{Z}_p$ and randomizes $c_{pk_{U_i}}$ and $NGBS(S, c_{pk_{U_i}})$ by *RdCom* and *RdProve* [16] into $c'_{pk_{U_i}}$ and $NGBS'(S, c_{pk_{U_i}})$.
   U_i computes $Y = G^{\frac{1}{s_m + L}}$ and the security tag $TA = pk_{U_i} \cdot \hat{e}(Y, H^R)$, where L is the monetary value. Meanwhile, U_i gives the following NIZK proofs π_Y, π_T. π_Y gives a proof that $Y = G^{\frac{1}{s_m + L}}$ and s_m in Y is equal to s_m in S. π_T gives a proof that the security tag TA is correctly formed.

$$\pi_Y \leftarrow GSPOK\{crs_2, \{Y, s_m\}, \{\hat{e}(\phi_Y, H^{s_m} \cdot H^L) = 1_{\mathbb{G}_3} \wedge \hat{e}(Y/\phi_Y, H^\theta) = 1_{\mathbb{G}_3} \wedge \hat{e}(G, H^\theta) = \hat{e}(G, H)\}\},$$

$$\pi_T \leftarrow GSPOK\{crs_2, \{TA, Y\}, \{\phi_T = pk_{U_i} \cdot \hat{e}(\phi_{Y'}, H^R) \wedge \hat{e}(Y/\phi_{Y'}, H^\theta) = 1_{\mathbb{G}_3} \wedge \hat{e}(TA/\phi_T, H^\theta) = 1_{\mathbb{G}_3} \wedge \hat{e}(G, H^\theta) = \hat{e}(G, H)\}\},$$

where $\phi_Y, \phi_T, \phi_{Y'}$ and $\theta = 1$ are auxiliary variables [6].

Finally, U_i sends $coin' = \{S, L, R, c'_{pk_{U_i}}, TA, \pi_{coin'} = \{NGBS'(S, c_{pk_{U_i}}), \pi_Y, \pi_T\}\}$ to M.

3. M verifies the proofs. If they are correct, M saves $coin'$ and offers goods and services to U_i.

5.6 The Deposit Protocol

M can deposit the coin to any bank. We assume that M has an account in B_j, where B_j may be B_i or not. When M wants to deposit a coin $coin'$ to B_j, M just sends $coin'$ to B_j. B_j checks the validity of $\pi_{coin'}$ and the consistency with S. If $coin'$ is not a valid coin, B_j rejects the deposit. Else, B_ij checks if there is already the serial number S in the database. If S is not found in the database, then B_j accepts the deposit of the coin $coin'$, credits the M's account and adds $coin'$ in the database. Else, there is an entry $coin'' = \{S, L, R', c''_{pk_{U_i}}, TA', \pi'_{coin'}\}$ in the database. Then, B_j checks the freshness of R in $coin'$ compared to $coin''$. If it is not fresh, M is a cheat and B_j refused the deposit. If R is fresh, B_j accepts the deposit of the $coin'$, credits the M's account and add $(coin', coin'')$ to the list of double spenders. For recovering the identity of double spender, B_j executes the **Identify** algorithm.

5.7 Identify

The Identify algorithm makes sure that when a double-spending is found, B_j recovers the identity of double spender. The description of the Identify algorithm is as follow.

B_j knows two coins $coin_1 = \{S, L, R_1, \{c'_{pk_{U_i}}\}_1, TA_1, \pi_1\}$ and $coin_2 = \{S, L, R_2, \{c'_{pk_{U_i}}\}_2, TA_2, \pi_2\}$. Therefore, B_j directly recovers the public key pk_{U_i} by computing $(TA_1^{R_2}/TA_2^{R_1})^{\frac{1}{R_2 - R_1}}$.

5.8 Verify

Anyone can verify the correctness of the double spenders and the signer (bank). In order to verify the correctness of the double spenders, anyone executes the algorithm $VerifyGuilt$. One can parse the $coin_1$ and $coin_2$ as $(S, L, R_1, \{c'_{pk_{U_i}}\}_1, TA_1, \pi_1)$ and $(S, L, R_2, \{c'_{pk_{U_i}}\}_2, TA_2, \pi_2)$ and next run Identify on these values. If the algorithm Identify returns a public key, then one can check if π_1 is consistent with $(S, L, R_1, \{c'_{pk_{U_i}}\}_1, TA_1)$ and if π_2 is consistent with $(S, L, R_2, \{c'_{pk_{U_i}}\}_2, TA_2)$.

In order to verify the correctness of the signer (bank) who is opened by Center Bank, anyone executes the algorithm $VerifySigner$. The input of the algorithm is $(mbpk, j, bspk_j, m, \pi_m, \tau_S)$. After verifying the correctness of π_m, anyone can check if the signature is signed by the bank.

5.9 Efficiency Analysis

We compare the efficiency of the multiple-bank e-cash [10, 8] with our scheme and analyze the security model of them in the following Table 1. It is somehow hard to quantify the exact cost of the spending protocol in [8] as the instantiation of the SKREP is very complex. We thus simplify the comparison by stating the total multi-exponentiations needed.

We give the following definitions. ABS and ABV represent the computation complexity of the automorphic blind signature and the verification of the automorphic blind signature. $NGBS$ and $NGBV$ represent the computation complexity of the new group blind signature and its verification. GSP and GSV represent the computation complexity of the Groth-Sahai proof and its verification. E represents the computation complexity of the verification of a pairing product equation. EXP represents a modular exponentiation. REC and REP represent the computation complexity of the re-randomization of the Groth-Sahai commitments and proofs. C and $\mathit{\Pi}$ represent the computation complexity of the Groth-Sahai commitments and proofs. Assuming that Groth-Sahai proofs are instantiated in the SXDH setting, we get the following efficiency.

We assume that C_1 is the computation cost of the joining protocol. C_2 is the efficiency of the withdrawal protocol. C_3 is the efficiency of the spending protocol. C_4 is the efficiency of the deposit protocol. C_5 is the security model. ME represents the number of multi-exponentiation. ROM represents random oracle model. SM represents standard model.

The efficiency of the Joining Protocol. In our joining protocol, the bank B_i needs to generate and verify an automorphic blind signature. Thus, B_i needs $(1ABS + 1ABV) = 17ME$. GM needs to verify a pairing product equation, and generate and verify an automorphic blind signature. Thus, GM needs $(1E + 1ABS + 1ABV) = 17ME$. In Chen's scheme [10], B_i and GM need $62ME$ and $273ME$ respectively. In Jeong's scheme [8], B_i and GM need $11ME$ and $12ME$ respectively.

The efficiency of the Withdrawal Protocol. In our withdrawal protocol, U_i generates two Groth-Sahai commitments and proofs, does two new group blind signature verifications, verifies two pairing product equations and re-randomizes two Groth-Sahai commitments and proofs. So U_i needs $2(C+\mathit{\Pi}+NGBV+GSV+E+RE) = 36ME$. The bank B_i needs to verify two Groth-sahai proofs and does two new group blind signatures. Thus, B_i needs $2(GSV + NGBS) = 17ME$. In Chen's scheme [10], B_i and U_i need $91ME$ and $71ME$ respectively. In Jeong's scheme [8], B_i and GM need $10ME$ and $19ME$ respectively.

The efficiency of the Spending Protocol. In our spending protocol, U_i re-randomizes one Groth-Sahai commitment and proof, generates Y and the security tag T and two correct proofs. Thus,

U_i needs $2(REC + REP + EXP + GSP) = 26ME$. M verifies the correctness of the coin. M needs to verify two Groth-Sahai proofs and new group blind signatures. Thus, M needs $2(GSV + NGBV) = 101ME$. In Chen's scheme [10], M and U_i need $580ME$ and $61ME$ respectively. In Jeong's scheme [8], M and U_i need $12ME$ and $11ME$ respectively.

The efficiency of the Deposit Protocol. In our deposit protocol, M deposits the coin to B_i. B_i verifies the correctness of the coin. Thus, B_i also needs $2(GSV + NGBV) = 76ME$. In Chen's scheme [10], B_i needs $285ME$. In Jeong's scheme [8], B_i needs $12ME$.

The security model. Our scheme is proven secure in the standard model. Chen's scheme [10] and Jeong's scheme [8] are proven secure in the random oracle model.

Table 1. Efficiency comparison between related work and our scheme.

	C_1		C_2		C_3		C_4		C_5
[10]	B_i	$62ME$	B_i	$91ME$	M	$580ME$	B_i	$285ME$	ROM
	GM	$273ME$	U_i	$71ME$	U_i	$61ME$			
[8]	B_i	$11ME$	B_i	$10ME$	M	$12ME$	B_i	$12ME$	ROM
	GM	$12ME$	U_i	$19ME$	U_i	$11ME$			
Ours	B_i	$17ME$	B_i	$17ME$	M	$101ME$	B_i	$76ME$	SM
	GM	$17ME$	U_i	$36ME$	U_i	$26ME$			

Based on the above analysis, the number of multi-exponentiation in our scheme is less than one in Chen's scheme [10], but more than Jeong's scheme [8]. However, our scheme is proven secure in the standard model. We know that the scheme proven secure in the standard model is more securer than one proven secure in the random oracle model. Therefore, our scheme is more secure.

6 Anonymity Revocation

The fair e-cash system provides three functionalities, i.e., owner tracing, coin tracing and identification of the double spender. To obtain the signer bank's identity in the multiple-bank e-cash, we also supplies the signer tracing. The identification of the double spender is given in the section 5.7. In this section, the coin is $coin = \{S, L, c_{pk_{U_i}}, NGBS(S, c_{pk_{U_i}})\}$ in the withdrawal protocol, and the coin is $coin' = \{S, L, R, c'_{pk_{U_i}}, TA, \pi_{coin'}\}$ in the spending protocol. When U_i withdraws a coin from B_i, B_i obtains $\{c_S, c_{pk_{U_i}}\}$ from U_i. In the following, we give the owner tracing, coin tracing and the signer tracing.

6.1 Owner Tracing

Owner tracing is that the Central Bank CB can recover U_i's identity from the coin $coin$. $coin$ includes the commitment $c_{pk_{U_i}}$ to U_i's public key. $c_{pk_{U_i}}$ is obtained by using the CB's commitment key. To trace the owner of the coin, B_i sends $coin$ to CB. CB extracts the owner's public key pk_{U_i} by using the extraction key ek_{cb}.

6.2 Coin Tracing

Coin tracing is that the Center Bank CB can trace the coin from the record of the withdrawal protocol which is kept in the bank. When U_i withdraws $coin$ from B_i, B_i obtains $\{c_S, c_{pk_{U_i}}\}$ from U_i. In order to achieve the coin tracing, B_i sends c_S to CB. CB extracts the serial number S from c_S by using the extraction key ek_{cb}. Then CB sends the serial number S to B_i. When U_i spends the coin $coin' = \{S, L, R, c'_{pk_{U_i}}, TA, \pi_{coin'}\}$ to M, B_i can trace the coin from the record of the withdrawal protocol by using S which is extracted from c_S.

6.3 Signer Tracing

Signer tracing is that the Center Bank CB can recover the signer B_i's identity. U_i obtains the group blind signature $NGBS(S, c_{pk_{U_i}})$. Using the Open algorithm in Section 3.1, CB extracts $(\sigma_S, \sigma_{pk_{U_i}}, cert_i, bgpk_i)$ from $NGBS(S, c_{pk_{U_i}})$ in $coin$. Therefore, we know which bank signs the coin.

7 Security Analysis

This section gives the security analysis of the scheme. The scheme fulfills all the security requirements given in Section 3.3. An adversary is defined as $\mathcal{A}$. A challenger is defined as $\mathcal{C}$. A series of games is given between $\mathcal{A}$ and $\mathcal{C}$ to prove the security properties.

Theorem 1. The scheme provides the bank anonymity under the following assumptions: the SXDH assumption and the zero-knowledge of the Groth-Sahai proofs.

$Proof$. The advantage that $\mathcal{A}$ breaks the bank anonymity is $Adv^{BAnon}_{\Pi,\mathcal{A}}(\lambda)$, where Π is our multiple-bank e-cash system. In the following, a series of games is given to prove that $Adv^{BAnon}_{\Pi,\mathcal{A}}(\lambda)$ is negligible.

Game 0. This is the real scheme where the real oracle $Join, BWith$ are executed at each joining and withdrawal query.

Game 1. As Game 0 except that in ParamGen the public key *mbpk* of the system is replaced by the perfectly hiding one. Under SXDH assumption, this change is negligible.

Game 2. As Game 1 except that in oracle $Join$ the signature of the bank's public key issuing the group signature is replaced by the simulation signature which is generated using the perfectly hiding public key. The certificate is zero-knowledge proofs. We know the Groth-Sahai is zero-knowledge, this Game and Game 1 are indistinguishable.

Game 3. As Game 2 except that in oracle $BWith$ the group signature is replaced by the simulation group signature which is generated using the perfectly hiding public key. The group signature is also zero-knowledge proofs. Thus, this Game and Game 2 are indistinguishable.

By the above series of games, we know if $\mathcal{A}$ can distinguish which bank signs the coin he can break the SXDH assumption and the zero-knowledge of the Groth-Sahai proofs.

Theorem 2. This scheme provides the user anonymity under the following assumptions: the SXDH assumption, the zero-knowledge of the Groth-Sahai proofs and the pseudo-randomness of the BCKL's sVRF.

Proof. The advantage that $\mathcal{A}$ breaks the user anonymity is $Adv_{\Pi,\mathcal{A}}^{UAnon}(\lambda)$, where Π is our multiple-bank e-cash system. In the following, a series of games is given to prove that $Adv_{\Pi,\mathcal{A}}^{UAnon}(\lambda)$ is negligible.

Game 0. This is the real scheme where the real oracle $Issue, Spd$ are executed at each joining and spending query.

Game 1. As Game 0 except that in ParamGen the public key *mbpk* of the system is replaced by the perfectly hiding one. Under SXDH assumption, this change is negligible.

Game 2. As Game 1 except that in oracle $Issue$ the certificate is replaced by the simulation certificate which is generated using the perfectly hiding public key. The certificate is zero-knowledge proofs. We know the Groth-Sahai proofs is zero-knowledge, this Game and Game 1 are indistinguishable.

Game 3. As Game 2 except that in oracle Spd the NIZK proof is replaced by the simulation proof which is generated using the perfectly hiding public key. We know the Groth-Sahai proofs is zero-knowledge, this Game and Game 1 are indistinguishable.

Game 4. As Game 3 except that in oracle Spd the security tag is replaced by the random value. The security tag is generated by using the BCKL's sVRF. If $\mathcal{A}$ can distinguish between this Game and Game 3 we can break the pseudo-randomness of the BCKL's sVRF.

Game 5. As Game 4 except that in oracle Spd the commitment of user's public key is replaced by the random value. The commitment of user's public key is generated by the re-randomness of the

Groth-Sahai proof. If $\mathcal{A}$ can distinguish between this Game and Game 4 we can break the re-randomness of the Groth-Sahai proofs.

By the above series of games, we know if $\mathcal{A}$ can distinguish which user spends the coin he can break the SXDH assumption, pseudo-randomness of the BCKL's sVRF and the zero-knowledge and re-randomness of the Groth-Sahai proofs.

Theorem 3. This scheme provides unforgeability under the following assumptions: the soundness of the Groth-Sahai proof and the unforgeability of the new group blind signature.

Proof. The advantage that $\mathcal{A}$ breaks the user anonymity is $Adv_{\Pi,\mathcal{A}}^{unfor}(\lambda)$. A deposited coin can be parsed as $coin = (S, L, R, c_{pk_{U_i}}, TA, NGBS(S, c_{pk_{U_i}}), \pi_Y, \pi_T)$. We consider the following multiple games.

Game 0. This is the real scheme.

Game 1. As Game 0 except that in oracle $UDepo$ $\mathcal{A}$ supplies a coin that the serial number is the same as that in $coin$, but the user's public key is different. If the event occurs, one of user's public keys does not correspond to the opening of $c_{pk_{U_i}}, c'_{pk_{U_i}}$ in which case we found a forgery for one of the two new group signature or if we broke the soundness of the Groth-Sahai proof.

Game 2. As Game 1 except that in oracle $UDepo$ $\mathcal{A}$ supplies a coin that the serial number is not obtained by the $BWith$ query. If the event occurs, we break the unforgeability of the new group blind signature. This is because a serial number is correspond to a group signature. The group signature is not obtained by the $BWith$ query. Thus, we break the unforgeability of the new group blind signature.

It is known $\mathcal{A}$ can win if he breaks the soundness of the Groth-Sahai proof and the unforgeability of the new group blind signature.

Theorem 4. This scheme provides identification of double-spender under the following assumptions: the pseudo-randomness of the BCKL's sVRF, the soundness of the Groth-Sahai proof, the unforgeability of the new group blind signature and the collision-resistant of the hash function.

Proof. The advantage that $\mathcal{A}$ breaks the user anonymity is $Adv_{\Pi,\mathcal{A}}^{ident}(\lambda)$. A successful adversary $\mathcal{A}$ in the identification game outputs two coins $(coin_1, coin_2)$ that verify and have the same serial number S but different merchant.

Game 0. This is the real scheme.

Game 1. As Game 0 except that in $BWith$ one of user's public keys does not correspond to the opening of $c_{pk_{U_i}}, c'_{pk_{U_i}}$ in which case we found a forgery for one of the two new group signature or if we broke the soundness of the Groth-Sahai proof.

Game 2. As Game 1 except that in security tag the s_m is different. If the event occurs, we breaks the the pseudo-randomness of the BCKL's sVRF or zero-knowledge of the Groth-Sahai proof.

Game 3. As Game 2 except that the two coin $coin_1$ and $coin_2$ are such that $(pk_{M_1}||Date_1) \neq (pk_{M_2}||Date_2)$ but result in the same hash value $\mathcal{H}(pk_{M_1}||Date_1) = \mathcal{H}(pk_{M_2}||Date_2)$. If the hash function is collision-resistant, this game is indistinguishable from Game 2.

It is known $\mathcal{A}$ can win if he breaks the soundness of the Groth-Sahai proof, the unforgeability of the new group blind signature and the collision-resistant of the hash function.

Theorem 5. This scheme provides exculpability under the following assumptions: the pseudo-randomness of the BCKL's sVRF, the soundness of the Groth-Sahai proof, the unforgeability of the new group blind signature and the collision-resistant of the hash function.

Proof. The advantage that $\mathcal{A}$ breaks the user anonymity is $Adv_{\Pi,\mathcal{A}}^{excul}(\lambda)$. A successful adversary $\mathcal{A}$ acting the bank in the exculpability game outputs two coins $(coin_1, coin_2)$ that verify and have the same serial number S but different merchant.

Game 0. This is the real scheme.

Game 1. As Game 0 except that one certificate of two coins is not issued by the group manager. If the event occurs, we break the unforgeability of the automorphic blind signature.

Game 2. As Game 1 except that in two coins the group signature is different. If the event occurs, we break the unforgeability of the new group blind signature.

Game 3. As Game 2 except that in two coins the security tag is different. If the event occurs, we break the pseudo-randomness of the BCKL's sVRF.

It is known $\mathcal{A}$ can win if he breaks the pseudo-randomness of the BCKL's sVRF and the unforgeability of the new group blind signature and the automorphic blind signature.

8 Conclusion

In this paper, we present a fair multiple-bank e-cash which is proved secure in the standard model. We propose a new group blind signature by extending the Ghadafi's group blind signature. Then we achieve the dual privacy requirement (the users anonymity and the bank anonymity) by using the new group blind signature. To hide the identity of the user, we re-randomize the commitment and corresponding proof to the user's public key by using the re-randomness of the Groth-Sahai proofs system. To ensure the security of the security tag, we use the pseudo-randomness of BCKL's sVRF. Finally, we prove the security properties in the standard model.

References

1. Zhang J. X., Li Z. J., Guo H.: Multiple-bank E-cash without Random Oracles. In CSS'13,2013.
2. Canard, S., and Gouget A.: Divisible e-cash systems can be truly anonymous. In EUROCRYPT'07, pages 482-497. Springer, 2007.
3. Belenkiy, M., Chase, M., Kohlweiss, M. and Lysyanskaya, A.: Compact e-cash and simulatable VRFs revisited. In PAIRING'09, volume 5671 of LNCS, pages 114-131, 2009.
4. Blazy, O., Canard, S., Fuchsbauer, G., Gouget, A., Sibert, H. and Traore, J.: Achieving optimal anonymity in transferable e-cash with a judge. In AFRICACRYPT'11, volume 6737 of LNCS, pages 206-223, 2011.
5. Zhang J. X., Li Z. J., Guo H. Anonymous transferable conditional E-cash. In SECURECOMM'12, Padua, Italy, 3-5 September, 2012.
6. Izabachene, M. and Libert, B.: Divisible e-cash in the standard model. In Pairing'12, volume 7708 of LNCS, pages 314-332, 2012.
7. Lysyanskaya, A., Z.Ramzan. Group blind signature: a scalable solution to electronic cash. In Financial Cryptography'98. Anguilla, British West Indies. pp. 184-197.
8. Jeong I. R. and Lee D. H. Anonymity control in multi-bank E-cash system. In INDOCRYPT'00, volume 1977 of LNCS, pages 104-116, 2000.
9. Wang S., Chen Z. and Wang X. A new certificateless electronic cash scheme with multiple banks based on group signatures, In Proc. of 2008 International Symposium on Electronic Commerce and Security, pp. 362-366, 2008.
10. Chen M., Fan C., Juang W. and Yeh Y. An efficient electronic cash scheme with multiple banks using group signature. In International Journal of Innovative Computing, Information and Control. (7)8, 2012.
11. Canetti, R., Goldreich, O. and Halevi, S.: The random oracle methodology, revisited. In 30th AcM STOC, pages 209-218. ACM Press, 1998.
12. Bellare, M., Boldyreva, A. and Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In EUROCRYPT'04, volume 3027 of LNCS, pages 171-188. Springer, 2004.
13. Groth, J. and Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In Eurocrypt'08, volume 4965 of LNCS, pages 415-432. Springer, 2008.
14. Ghadafi E. Formalizing group blind signatures and practical constructions without random oracles. In ACISP 2013, LNCS 7959, pp. 330-346, 2013.
15. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K. and Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In CRYPTO'10, volume 6223 of LNCS, pages 209-236. Springer, 2010.
16. Fuchsbauer, G.: Commuting signatures and verifiable encryption. In CRYPTO 2011, volume 6632 of LNCS, pages 224–245.
17. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signa- tures. In Cryptology ePrint Archive, Report 2009/320, http://eprint.iacr.org/2009/320.pdf.

18. Ghadafi E., Smart N.P. and Warinschi B.: Groth-Sahai proofs revisited. In PKC'10, volume 6056 of LNCS, pages 177-192. Springer, 2010.

19. Canard, S., Delerablee, Cecile, Gouget, A., Hufschmitt, E., Laguillaumie, F., Sibert, H., traore, J. and Vergnaud, D. Fair e-Cash: be compact, spend faster. In ISC 2009, volume 5735 of LNCS, Pisa, Italy, 7-9 September, pages 294-309.

Appendix A

In this section, We define SXDH-based commitments, randomization of the SXDH-based commitments, SXDH-based Proof, randomization of the SXDH-based Proof and Groth-Sahai Proof of Committing to Constants as follows.

- *SXDH-based commitments.*
  - *Setup.* On input of the public parameter $pp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \widehat{e}, G, H)$, choose $\alpha_1, \alpha_2, t_1, t_2 \in \mathbb{Z}_p$, then output is $crs = (\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2)$, where $\mathbf{U}_1 = (U_{1,1}, U_{1,2}) = (G, G^{\alpha_1}), \mathbf{U}_2 = (U_{2,1}, U_{2,2}) = (G^{t_1}, G^{\alpha_1 t_1}), \mathbf{V}_1 = (V_{1,1}, V_{1,2}) = (H, H^{\alpha_1})$ and $\mathbf{V}_2 = (V_{2,1}, V_{2,2}) = (H^{t_1}, H^{\alpha_1 t_1})$.
  - *Commit.* Define the commitment to a group element $X \in \mathbb{G}_1$ as

    $$C_X = Com(crs, X, \mathbf{r} = (r_1, r_2)) = (U_{1,1}^{r_1} \cdot U_{2,1}^{r_2}, X \cdot U_{1,2}^{r_1} \cdot U_{2,2}^{r_2}),$$

    where $r_1, r_2 \in \mathbb{Z}_p$. So the commitment to $Y \in \mathbb{G}_2$ is

    $$C_Y = Com(crs, Y, \mathbf{s} = (s_1, s_2)).$$

    where $s_1, s_2 \in \mathbb{Z}_p$.
- *Randomization to commitment.* Define the randomization to the commitment C_X as

  $$RdCom(crs, C_X, \mathbf{r}') = C_X \odot Com(crs, 1, \mathbf{r}') = (C_{X,1} \cdot U_{1,1}^{r'_1} \cdot U_{2,1}^{r'_2}, C_{X,2} \cdot U_{1,2}^{r'_1} \cdot U_{2,2}^{r'_2}),$$

  where $\mathbf{r}' = (r'_1, r'_2), r'_1, r'_2 \in \mathbb{Z}_p$, $\odot$ denotes component-wise multiplication.
- *SXDH-based proof.*

  Groth-Sahai proofs assert that some group elements satisfy the class of pairing-product equations ε mentioned above.

  Following the definition in [16], we also define the prove that $X_i \in \mathbb{G}_1, Y_i \in \mathbb{G}_2$ satisfy ε as $Prove(crs, (X_i, \mathbf{r}_i)_{i=1}^m, (Y_j, \mathbf{s}_j)_{j=1}^n, \varepsilon; Z)$, where $\mathbf{r}_i, \mathbf{s}_j \in \mathbb{Z}_p^2$, and $Z \in \mathbb{Z}_p^{2\times 2}$ is the internal randomness.
- *Randomization to proof.* It is similar to the randomization to the commitment. We random the proof by replacing the internal randomness of the commitment. Therefore, the randomization to proof is defined as

$$RdProve(crs, (X_i, \mathbf{r}_i + \mathbf{r}'_i)_{i=1}^m, (Y_j, \mathbf{s}_j + \mathbf{s}'_j)_{j=1}^n, \{\varepsilon_1 \wedge \ldots \wedge \varepsilon_j\}),$$

where $\mathbf{r}'_i$, $\mathbf{s}'_j \in \mathbb{Z}_p^2$, $\mathbf{r}'_i$ and $\mathbf{s}'_j$ are the randomness of the commitments $C_{X_i} = Com(crs, X_i, \mathbf{r}'_i)$ and $C_{Y_j} = Com(crs, Y_j, \mathbf{s}'_j)$.

Appendix B

In this section, we give the simple descriptions of Abe et al.'s Automorphic Blind Signature on one message and two messages.

Automorphic Blind Signature on one Message. Note that *algorithm*(a) represents the input of the parameters of the algorithm is a, and *protocol*$[A \leftrightarrow B]$ represents A and B interacts in the protocol.

- *Setup*(1^λ). Output the public parameters of bilinear groups $bgpp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \hat{e}, G, H)$, where λ is the security parameter. Choose random elements $G, F, K, T \in \mathbb{G}_1, H \in \mathbb{G}_2$ and output the public parameters $abspp = (bgpp, F, K, T)$. On input $bgpp$ run the setup algorithms for Groth-Sahai proofs and return the reference string crs and the extraction key ek. The message space is $\mathcal{DH} = \{(G^m, H^m) | m \in \mathbb{Z}_p\} = \{M_1, M_2\}$.
- $Key-Generation$. Choose $s \leftarrow \mathbb{Z}_p$ and set $S_1 = G^s, S_2 = H^s$. The public key is $pk = (S_1, S_2)$. The signing key is $sk = s$.
- $The-First-Round$ $[U \rightarrow Signer]$. Choose $q \leftarrow \mathbb{Z}_p$ and compute $Q_1 = G^q$,$Q_2 = H^q$ and $U = T^q \cdot M_1$. U generates the proof $\Psi \leftarrow GSPOK(crs, \{M_1, M_2, Q_1, Q_2\}, \{\hat{e}(M_1, H) = \hat{e}(G, M_2) \wedge \hat{e}(Q_1, H) = \hat{e}(G, Q_2) \wedge \hat{e}(T, Q_2) \cdot \hat{e}(M_1, H) = \hat{e}(U, H)\})$. Finally, U sends (U, Ψ) to signer.
- $The-Second-Round$ $[Signer \rightarrow U]$. If Ψ is valid the signer chooses $c, r \leftarrow \mathbb{Z}_p$ and computes $A = (K \cdot T^r \cdot U)^{\frac{1}{s+c}}, C_1 = F^c, C_2 = H^c, R_1 = G^r, R_2 = H^r$. Finally, the signer sends $\sigma = (A, C_1, C_2, R_1, R_2)$ to U.
- *Obtain* $[U]$. If $\hat{e}(R_1, H) = \hat{e}(G, R_2), \hat{e}(F, C_2) = \hat{e}(C_1, H)$ and $\hat{e}(A, S_2 \cdot C_2) = \hat{e}(K \cdot M_1, H) \cdot \hat{e}(T, R_2)$, U computes $R'_i = R_i \cdot Q_i$ for $i = 1, 2$. U outputs $\Omega \leftarrow GSPOK(crs, \{A, C_1, C_2, R'_1, R'_2\}, \{\hat{e}(R'_1, H) = \hat{e}(G, R'_2) \wedge \hat{e}(C_1, H) = \hat{e}(F, C_2) \wedge \hat{e}(A, S_2 \cdot C_2) \cdot \hat{e}(T^{-1}, R_2) = \hat{e}(K \cdot M_1, H)\})$. Finally, U obtains the signature $\Sigma = \Omega$.
- $Verify(pk, M_1, M_2, \Sigma, crs)$. Anyone can verify the correctness of the blind signature. The output is 1 or 0.

Automorphic Blind Signature on Two Messages. In order to sign two messages, Fuchsbauer gives the following method in [16] to finish a generic transformation from the automorphic blind signature on single message to one singing two messages at once. The messages signed are (V, W) and (M, N)

which are the Diffie-Hellman pair. We generate two key pairs (vk, sk) and (vk^*, sk^*). The first one is used for signing vk^*. The second one is used for signing the messages.

- *Sign*. The signature is $\sigma_{(V,W),(M,N)} = (vk^*, \sigma_0 = ABSign(sk, vk^*), \sigma_1 = ABSign(sk^*, (M, N)),$ $\sigma_2 = ABSign(sk^*, (V, W) \odot (M, N)), \sigma_3 = ABSign(sk^*, (V, W)^3 \odot (M, N)))$, where $\odot$ denotes applying the group operation componentwise.
- *Verify*. In order to verify the signature, we verify $ABSVerify(vk, vk^*, \sigma_0), ABSVerify(vk^*, (M, N), \sigma_1),$ $ABSVerify(vk^*, (V, W) \odot (M, N), \sigma_2)$ and $ABSVerify(vk^*, (V, W)^3 \odot (M, N), \sigma_3)$.

Appendix C

In this section, we give the description of Ghadafi's group blind signature. We define by Si_i the ith signer. We define by GM the group manager. We define by U_i the ith user. In the following, we give the simple description. Note that $algorithm(a)$ represents the input of the parameters of the algorithm is a, and $protocol[A \leftrightarrow B]$ represents A and B interacts in the protocol.

- *Group–Key–Generation*(1^λ). Output the public parameters of bilinear groups $bgpp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \hat{e}, G, H)$, where λ is the security parameter. Choose random elements $G, F, K, T \in \mathbb{G}_1, H \in \mathbb{G}_2$ and output $abspp = (bgpp, F, K, T)$. On input $bgpp$ run the setup algorithms for Groth-Sahai proofs and return two reference strings crs_1, crs_2 and the corresponding extraction keys ek_1, ek_2. The two reference strings and the extraction keys are used for the first round and the second round of the automorphic blind signature scheme which is used for issuing the group blind signature. The second extraction key ek_2 is given to the opener for recovering the malicious bank. On input $bgpp$ and output the key pair of group manager (sk_{GM}, pk_{GM}). The public key of group blind signature is $gpk = (abspp, crs_1, crs_2, pk_{GM})$.
- *Signer − Key − Generation* $[Si_i]$. On input $bgpp$ run the setup algorithms for automorphic blind signature scheme and return two key pairs (sk_i, pk_i) and (ssk_i, spk_i). The first one is used for issuing the group blind signature. The second one is used for signing the public key $bgpk_i$ in the joining protocol.
- *Join − Protocol* $[Si_i \leftrightarrow GM]$. Si_i chooses $s \leftarrow \mathbb{Z}_p$. The secret key is $sk_i = s$. The public key is $pk_i = (S_1 = G^s, S_2 = H^s)$. Si_i gives the signature $sig_i \leftarrow ABSSign(sk_i, spk_i)$ for spk_i using sk_i. Finally, Si_i sends sig_i, pk_i to GM. If S_1 and S_2 are well-formed, pk_i is signed by GM, and sig_i is valid, GM gives the signature $cert_i \leftarrow ABSSign(sk_{GM}, pk_i)$ which is used for B_i's certificate. The registration information is set to $reg_i = (pk_i, sig_i)$. If $cert_i$ is valid, B_i's group signing key is $(sk_i, pk_i, cert_i)$.

- $The - First - Round$ $[U_i \rightarrow Si_i]$. Choose $q \leftarrow \mathbb{Z}_p$ and compute $Q_1 = G^q$,$Q_2 = H^q$ and $U = T^q \cdot M_1$. U_i generates the proof $\Psi \leftarrow GSPOK(crs_1, \{M_1, M_2, Q_1, Q_2\}, \{\hat{e}(M_1, H) = \hat{e}(G, M_2) \wedge \hat{e}(Q_1, H) = \hat{e}(G, Q_2) \wedge \hat{e}(T, Q_2) \cdot \hat{e}(M_1, H) = \hat{e}(U, H)\})$. Finally, U_i sends (U, Ψ) to B_i.
- $The - Second - Round$ $[Si_i \rightarrow U_i]$. If (crs_1, Ψ) is valid the signer chooses $c, r \leftarrow \mathbb{Z}_p$ and computes $A = (K \cdot T^r \cdot U)^{\frac{1}{s+c}}, C_1 = F^c, C_2 = H^c, R_1 = G^r, R_2 = H^r$. B_i sets $\sigma = (A, C_1, C_2, R_1, R_2)$. In order to hide the parts which identify Si_i's identity, Si_i commits to $cert_i, pk_i, A, C_1$ and C_2. Then Si_i gives the proof $\Omega \leftarrow GSPOK(crs_2, \{cert_i, S_1, S_2, A, C_1, C_2\}, \{ABSSign(gpk, (S_1, S_2), cert_i) = 1 \wedge \hat{e}(S_1, H) = \hat{e}(G, S_2) \wedge \hat{e}(C_1, H) = \hat{e}(F, C_2) \wedge \hat{e}(A, S_2 \cdot C_2) \cdot \hat{e}(T^{-1}, R_2) = \hat{e}(K \cdot U, H)\})$. Finally, Si_i sends (R_1, R_2, Ω) to U_i.
- $Obtain$ $[U_i]$. If (crs_2, Ω) is valid and $\hat{e}(R_1, H) = \hat{e}(G, R_2)$, U_i computes $R'_i = R_i \cdot Q_i$ for $i = 1, 2$. U_i re-randomizes Ω as Ω'. Then U_i gives the proof $\Omega' \leftarrow GSPOK(crs_2, \{cert_i, S_1, S_2, A, C_1, C_2, R'_1, R'_2\}, \{ABSSign(gpk, (S_1, S_2), cert_i) = 1 \wedge \hat{e}(S_1, H) = \hat{e}(G, S_2) \wedge \hat{e}(C_1, H) = \hat{e}(F, C_2) \wedge \hat{e}(A, S_2 \cdot C_2) \cdot \hat{e}(T^{-1}, R_2) = \hat{e}(K \cdot M_1, H) \wedge \hat{e}(R'_1, H) = \hat{e}(G, R'_2)\})$. Finally, U_i obtains the signature $\Sigma = \Omega'$.
- $Verify(gpk, \Sigma)$. Anyone can verify the correctness of the blind signature Σ. The output is 1 or 0.
- $Open(crs_2, ek_2, \Sigma)$. The opener extracts the public key pk_i, the signature σ and the membership certificate $cert_i$ from the proof Σ.
- $Judge(gpk, spk_i, M_1, M_2, \Sigma, \tau = (i, A, C_1, C_2, R'_1, R'_2, cert_i, pk_i, sig_i))$. If $i > 0$, $\hat{e}(A, S_2 \cdot C_2) \cdot \hat{e}(T^{-1}, R_2) = \hat{e}(K \cdot M_1, H)$ and $\hat{e}(R'_1, H) = \hat{e}(G, R'_2)$, $ABSVerify(spk_i, pk_i, sig_i) = 1$, $\hat{e}(C_1, H) = \hat{e}(F, C_2)$ and $ABSVerify(pk_{GM}, pk_i, cert_i) = 1$, the opener's claim is correct.

Appendix D

In this section, we give the description of BCKL's sVRF.

- $Setup(1^k)$. Output parameters $params_{VRF} = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, G, H), params_{GS})$, where $params_{GS}$ is the parameters of the corresponding Groth-Sahai NIZK proof system.
- $Keygen(params_{VRF})$. Pick a random seed $s \leftarrow \mathbb{Z}_p$ and random opening information $open_s$, and output secret key $sk = (s, open_s)$ and public key $pk = Com(h^s, open_s)$.
- $Eval(params_{VRF}, sk, x)$. Compute $y = g^{1/(s+x)}$.
- $Prove(params_{VRF}, sk, x)$. Compute $y = g^{1/(s+x)}$ and $C_y = Com(y, open_y)$ from random opening $open_y$. Next create the following two proofs: π_1 is a composable NIZK proof that C_y is a commitment to y; π_2 is a GS composable witness indistinguishable proof that C_y is a commitment to Y and pk is a commitment to h^s.
- $Verify(params_{GS}, pk, x, y, C_y, \pi_1, \pi_2)$. Use the Groth-Sahai verification to verify π_1 and π_2 with respect to C_y, x, pk, y.