Retrofitting Fairness on the Original RSA-Based E-cash

Shouhuai Xu[1] and Moti Yung[2]

[1] Dept. of Information and Computer Science, University of California at Irvine
shxu@ics.uci.edu
[2] Dept. of Computer Science, Columbia University
moti@cs.columbia.edu

Abstract. The notion of *fair* e-cash schemes was suggested and implemented in the last decade. It balances anonymity with the capability of tracing users and transactions in cases of crime or misbehavior. The issue was raised both, in the banking community and in the cryptographic literature. A number of systems were designed with an off-line fairness, where the tracing authorities get involved only when tracing is needed. However, none of them is based on the original RSA e-cash. Thus, an obvious question is whether it is possible to construct an *efficient* fair e-cash scheme by retrofitting the fairness mechanism on the original RSA-based scheme. The question is interesting from, both, a practical perspective (since investment has been put in developing software and hardware that implement the original scheme), and as a pure research issue (since retrofitting existing protocols with new mechanisms is, at times, harder than designing solutions from scratch). In this paper, we answer this question in the affirmative by presenting an efficient fair off-line e-cash scheme based on the original RSA-based one.

Keywords. E-cash, Fairness, Conditional Anonymity, RSA.

1 Introduction

In his seminal paper [C82], David Chaum introduced the notion of e-cash and presented the first e-cash scheme based on RSA ([RSA78]) blind signature. This scheme prevents the users from double-spending e-coins by on-line checking their freshness against a list of e-coins that have been spent. The overhead of this on-line checking process had inspired the notion and first implementation of off-line e-cash scheme [CFN88], which is also based on RSA blind signature but ensures that the identity of a double-spender will be exposed (after the fact). The above schemes preserve information-theoretic anonymity for users who do not double-spend any e-coin; this leads to potential abuses as was initially discussed in [vSN92]. In the last decade, this vulnerability has inspired numerous *fair* (or, conditionally anonymous) off-line e-cash schemes, but none of them (see, for example, [CMS96,FTY96,JY96]) is based on the original RSA e-cash; instead,

R.N. Wright (Ed.): FC 2003, LNCS 2742, pp. 51–68, 2003.

they all use discrete logarithm based cryptosystems. Thus, it has been open whether it is possible to construct an *efficient* fair off-line e-cash scheme that retains the original RSA-based scheme. In this paper, we answer this question in the affirmative.

1.1 Our Contributions

We show how to incorporate *fairness* into the original RSA-based e-cash by presenting an *efficient* fair off-line RSA e-cash scheme. In the resulting scheme, fairness is implemented by deploying a set of servers that maintain a threshold cryptosystem such that (1) the servers stay off-line when nothing goes wrong (i.e., they are invoked only when there is a need to revoke anonymity), and (2) the servers are only assumed to preserve user anonymity. Our scheme preserves the system architecture of its underlying scheme, namely the Chaum-Fiat-Naor one [CFN88], which has been developed by companies and has been deployed experimentally in the real world. These facts make retrofitting fairness while retaining the system architecture important from a business perspective. Our scheme is flexible in the sense that, as a special case, it can retain the original Chaum-Fiat-Naor coin structure as is (for software compatibility reasons), while it also allows other desirable coin structures.

Beyond achieving the above solution, there is a crucial technical question related to our solution that we believe deserves special mention: In order to implement fairness, what properties should the cryptosystem possess (e.g., is it enough to deploy a standard semantically-secure cryptosystem like ElGamal)? Although it may seem intuitively sufficient to deploy the ElGamal cryptosystem, we actually show that it is not so in the context of fair e-cash where the adversary has additional capability in the possible fault scenario (see Section 3.3 for details).

Remark 1. The e-cash scheme due to Juels [J99] inherits the e-coin structure of [C82]. However, this scheme involves an *on-line* Trusted Third Party (TTP) in the withdrawal sessions; the use of such an on-line TTP follows [BGK95]. Moreover, this scheme assumes a trust model that is strictly stronger than its counterpart in our scheme: there the TTP, in addition to being liable for preserving anonymity, is also assumed not to frame a user or steal a user's money; in our scheme, the TTP (namely the revocation servers) is only liable for preserving anonymity (i.e., the revocation servers have no capability to frame users or steal users' money).

Remark 2. The distribution of the revocation capability among a set of servers is done in order to implement better anonymity, as was shown in [MP98].

Organization:

In Section 2 we present the model and some basic goals of e-cash schemes. In Section 3 we introduce the basic ideas underlying our approach and construction. In Section 4 we present the cryptographic tools that will be used to implement fairness. We present our fair off-line RSA e-cash scheme in Section 5, and analyze its properties in Section 6. We discuss some deployment issues and extensions in

Section 7 and conclude in Section 8. Due to space limitation, we have eliminated in the current version certain details which will be given in the full version of this paper.

2 The Model and Goals

The Participants. We consider the following entities: a bank that issues e-coins (which possess a structure we call the "coin structure"), a set of n revocation servers $\mathcal{P}_1, \cdots, \mathcal{P}_n$ that maintain a threshold decryption capability for revoking anonymity, a set of users that withdraw and spend e-coins, and a set of merchants that accept e-coins. All of the entities are modeled as probabilistic polynomial-time interactive Turing machines.

The Communication Channels. The communication channels, except those channels between the servers in the revocation process, are asynchronous. Regarding the communication channels between the servers in the revocation process, we assume that $\mathcal{P}_1, \cdots, \mathcal{P}_n$ are connected by a complete network of private (i.e., untappable) point-to-point channels, and that they have access to a dedicated broadcast channel. Furthermore, we assume a *fully synchronous* communication model such that messages of a given round in the protocol are sent by all parties simultaneously and are delivered to their recipients. However, all the results in this paper apply to the more realistic *partially synchronous* communication model, in which messages sent on either a point-to-point or the broadcast channel are received by their recipients within some fixed time bound. Note that when we deploy the system in the real world, these assumptions can be substituted with appropriate cryptographic protocols that enforce "rounds of communication" using commitment schemes before decommitment of actual values.

The Adversary. We consider a probabilistic polynomial-time adversary that may initiate various protocols. The adversary is t-threshold meaning that it is able to corrupt at most t revocation servers. The adversary is malicious in that it may cause the corrupt parties to arbitrarily divert from the specified protocol. We assume that the adversary is static, which means that it chooses the parties it will corrupt at the initialization of the system. (We defer to Section 7 a detailed discussion on the issue of a subtle "suicide" attack against the anonymity of the honest users; in this attack an element in the coin structure of a honest user is embedded into a coin structure of a dishonest user who will commit a crime in order to compromise the anonymity of the honest user. Such an attack requires knowledge of coin structure of other users and potentially involves a bank which does not keep secret the coin database.)

2.1 The Goals

We focus on the following basic goals of e-cash (presented informally):

1. Unforgeability: After initiating polynomially many withdrawal sessions, the adversary is still unable to output an e-coin that is different from all the e-coins obtained in the withdrawal sessions. In other words, no adversary can succeed in conducting a "one more coin forgery."
2. Revocability: The revocation servers can collaboratively expose the e-coin issued in a withdrawal session and associate a given e-coin with the corresponding withdrawal session.
3. Anonymity: An adversary succeeds in breaking anonymity if it can associate an e-coin with the corresponding withdrawal session initiated by a honest user, or associate two e-coins with the same honest user (although the user's identity may be unknown). We require the probability of an adversary successfully breaking anonymity to be negligible.

3 The Basic Ideas

In this section, we first recall Chaum-Fiat-Naor's off-line RSA e-cash scheme [CFN88]; this is necessary for understanding our approach. Then, we present the basic ideas underlying our approach.

3.1 The Chaum-Fiat-Naor RSA E-cash Scheme

This scheme consists of four protocols: The Initialization, The Withdrawal, The Payment, and The Deposit.

The Initialization Protocol. Let f and g be two-argument collision-free functions, where f behaves like a random oracle and g has the property that fixing the first argument gives a one-to-one or c-to-1 map from the second argument *onto* the range.

1. The bank initially publishes an RSA signature verification key $(3, N)$, where the modulus N is the multiplication of two prime numbers P and Q that are chosen according to the main security parameter κ.
2. The bank sets a secondary parameter l.
3. A user Alice opens a bank account numbered u and the bank keeps a counter v associated with it.

The Withdrawal Protocol. Let $||$ denote string concatenation. This protocol has the following steps.

1. Alice chooses a_i, c_i, d_i, and r_i, $1 \leq i \leq l$, independently and uniformly at random from Z_N^*.
2. Alice sends to the bank l blinded candidates $B_i = r_i^3 \cdot f(x_i, y_i) \bmod N$ for $1 \leq i \leq l$, where $x_i = g(a_i, c_i)$ and $y_i = g(a_i \oplus (u||(v+i)), d_i)$.
3. The bank chooses a random subset of $l/2$ blinded candidate indices $R = \{i_j\}_{1 \leq i_j \leq l, 1 \leq j \leq l/2}$ and transmits it to Alice.
4. Alice presents the a_i, c_i, d_i, and r_i for all $i \in R$, and the bank checks their semantical correctness (i.e., that their structure follows the protocol specification). To simplify notations, let $R = \{l/2+1, l/2+2, \cdots, l\}$.

5. The bank gives Alice $\prod_{1 \le i \le l/2} B_i^{1/3} \bmod N$ and charges her account one dollar. The bank also increments Alice's counter v by l.
6. Alice can easily extract the e-coin $C = \prod_{1 \le i \le l/2} f(x_i, y_i)^{1/3} \bmod N$. Alice re-indexes the candidates in C according to the f values: $f(x_1, y_1) < f(x_2, y_2) < \cdots < f(x_{l/2}, y_{l/2})$. Alice also increases her copy of the counter v by l.

The Payment Protocol. The payment protocol has the following steps.

1. Alice (anonymously) sends $C = \prod_{1 \le i \le l/2} f(x_i, y_i)^{1/3} \bmod N$ to Bob.
2. Bob chooses a random binary string $z_1, z_2, \cdots, z_{l/2}$.
3. For all $1 \le i \le l/2$: if $z_i = 1$, Alice sends Bob a_i, c_i, and y_i; otherwise, Alice sends Bob x_i, $a_i \oplus (u||(v+i))$, and d_i.
4. Bob verifies the semantical correctness of the responses.

The Deposit Protocol. The deposit protocol has the following steps.

1. Bob sends an e-coin C and Alice's responses in a payment to the bank.
2. The bank verifies their semantical correctness and credits Bob' account. The bank stores C, the binary string $z_1, z_2, \cdots, z_{l/2}$ and the values a_i, c_i, and y_i (for $z_i = 1$) and x_i, $a_i \oplus (u||(v+i))$, and d_i (for $z_i = 0$).

3.2 Key Observations Underlying Our Approach

An intuitive solution to incorporating *fairness* into the Chaum-Fiat-Naor scheme is to deploy a set of servers for revoking anonymity such that, ideally, the servers stay completely off-line (i.e., they are involved only when there is a need to revoke anonymity). The key observations are:

1. We can force a user to encrypt a coin representation using a public key whose corresponding private key is shared among the servers.
2. The revocation capability can be completely independent of the e-cash issuing capability, thus we can modularly integrate different cryptosystems (for different purposes) into a single scheme.

For concreteness, in the sequel we consider the case of DLOG-based cryptosystem for the servers (though other systems are possible). More specifically, we let a user encrypt $H_1(m) \in \mathcal{G}$, corresponding to which $H(m) \in Z_N^*$ appears in an e-coin, using a DLOG-based cryptosystem over a cyclic sub-group $\mathcal{G}$ of Z_p, where both $H(\cdot)$ and $H_1(\cdot)$ are appropriate functions. An intriguing question is: Can we employ the ElGamal cryptosystem [E85] for this purpose? Next we present a detailed analysis that shows that ElGamal is insufficient.

3.3 Why ElGamal Cryptosystem Is Insufficient for Our Purpose?

ElGamal cryptosystem, which has been proven to be semantically secure [TY98], may not be enough to facilitate the task of revoking anonymity. For the sake of proving anonymity, we need to present a simulator that is able to emulate a real-world system while embedding a Decision Diffie-Hellman (DDH) challenge

tuple. The key observation underlying the infeasibility is that the adversary is allowed to invoke the revocation process, which means: the simulator must somehow be able to decrypt ciphertexts generated by the adversary; otherwise, the adversary is able to distinguish a simulated system from the real-world system. (The rationale that we have to allow the adversary to invoke the revocation process can be justified by, for instance, the following scenario: if the adversary commits some suspicious activities then the revocation process will be invoked in the real world; the adversary misbehaves anyway, or simply for the purpose of distinguishing a simulation from the real-world system.) A simple solution to this problem is to deploy some cryptosystem that is simulatable while allowing the simulator to hold the private key.

One may suggest that we can seek to utilize the random oracle and the cut-and-choose technique (both of them are deployed anyway) so that the simulator not knowing the private key can get through. However, this intuition is incorrect. First, even if the simulator knows all the plaintexts that are output by a random oracle (namely $H_1(\cdot)$ in our setting), the adversary can arbitrarily plant a single plaintext as an element in an e-coin structure. In order to have a better understanding about this issue, let us consider the following simplified scenario: (1) the adversary asks the oracle at two points 0 and 1, which means that the simulator knows $H_1(0)$ and $H_1(1)$; (2) the adversary presents an ElGamal encryption of $H_1(b)$ where $b \in_R \{0,1\}$; (3) in the revocation process the simulator not knowing the private key needs to output b. This is exactly the game used in defining security of the cryptosystem, which means that if the simulator does not know the corresponding private key but is able to output the correct plaintext, then we can easily make use of the simulator to break the DDH assumption. It is easy to see that the adversary can always play such a game in the e-cash scheme, because it can always pass the cut-and-choose test with probability 0.5. Since the adversary can dynamically "open" values and play with them in the game itself, we have to cope with this more dynamic nature of adversarial behavior. To this end, the bare ElGamal cryptosystem seems insufficient.

4 Cryptographic Tools for Revoking Anonymity

Now, we review some DLOG-based cryptographic tools that will be used as subroutines in the rest of this paper. The reader familiar with these tools may skip the technical description.

The Setting. Let κ' be a security parameter of DLOG-based cryptosystems, corresponding to which two prime numbers p and q are chosen so that $q|(p-1)$. Suppose $g \in Z_p$ is of order q, which means that g specifies a unique cyclic subgroup $\mathcal{G}$ in which the DDH assumption is assumed to hold; namely, it is infeasible to distinguish a random tuple (g, h, u, v) of four independent elements in $\mathcal{G}$ from a random tuple satisfying $log_g u = log_h v$.

A Semantically-Secure Encryption Scheme. We adopt the semantically-secure encryption scheme appeared in [JL00], which is similar to the standard ElGamal scheme except that it uses two generators (to allow the more

dynamic adversary). Let $\langle p, q, g, h, y = g^{x_1} h^{x_2} \bmod p \rangle$ be a public key and $\langle p, q, g, h, x_1, x_2 \rangle$ be the corresponding private key, where $(x_1, x_2) \in_R Z_q^2$, g and h are two random elements in $\mathcal{G}$ such that nobody knows $log_g h$. To encrypt a message $M \in \mathcal{G}$, one chooses $k \in_R Z_q$ and computes the ciphertext $(g^k \bmod p, h^k \bmod p, y^k M \bmod p)$. To decrypt a ciphertext (α, β, γ), one computes $M = \gamma / \alpha^{x_1} \beta^{x_2} \bmod p$.

Proof of the Same Logarithm. The following protocol is a zero-knowledge proof that $log_g u = log_h v$. Let $DLOG(g, u) = DLOG(h, v)$ denote such a proof. Suppose $x = log_g u = log_h v \bmod q$.

1. The prover chooses $k \in_R Z_q$, computes $A = g^k \bmod p$ and $B = h^k \bmod p$, and sends (A, B) to the verifier.
2. The verifier chooses a challenge $c \in_R Z_q$, and sends it to the prover.
3. The prover sends the verifier $d = c \cdot x + k \bmod q$.
4. The verifier accepts if $g^d = u^c A \bmod p$ and $h^d = v^c B \bmod p$.

The Fiat-Shamir transformation [FS86] can make this protocol non-interactive.

Proof of the Same Representation. The following protocol allows a prover to prove $REP(g, h; y) = REP(\alpha, \beta; \gamma)$, where $REP(a, b; c)$ denotes the representation of c with respect to the pair of bases (a, b) [CP92]. Suppose $y = g^{x_1} h^{x_2} \bmod p$ and $\gamma = \alpha^{x_1} \beta^{x_2} \bmod p$.

1. The prover chooses $k_1, k_2 \in_R Z_q$, and computes $A = g^{k_1} h^{k_2} \bmod p$ and $B = \alpha^{k_1} \beta^{k_2} \bmod p$. The prover sends (A, B) to the verifier.
2. The verifier chooses $c \in_R Z_q$, and sends it to the prover.
3. The prover sends the verifier $d_1 = c \cdot x_1 + k_1 \bmod q$ and $d_2 = c \cdot x_2 + k_2 \bmod q$.
4. The verifier accepts if $g^{d_1} h^{d_2} = y^c A \bmod p$ and $\alpha^{d_1} \beta^{d_2} = \gamma^c B \bmod p$.

The Fiat-Shamir transformation [FS86] can make this protocol non-interactive.

Feldman-VSS(t). This protocol allows a dealer to share $s \in_R Z_q$ among a set of players $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ via a t-degree polynomial [F87]. As a consequence, $\mathcal{P}_i$ holds a share s_i such that $s \overset{(t+1,n)}{\longleftrightarrow} (s_1, \cdots, s_n)$.

1. The dealer chooses a random t-degree polynomial $f(z) = a_0 + a_1 z + \cdots + a_t z^t$ over Z_q such that $s = a_0$. It broadcasts $A_l = g^{a_l} \bmod p$ for $0 \leq l \leq t$, computes and secretly sends $s_j = f(j)$ to player $\mathcal{P}_j$ for $1 \leq j \leq n$.
2. $\mathcal{P}_j$ $(1 \leq j \leq n)$ verifies if $g^{s_j} = \prod_{l=0}^{t} (A_l)^{j^l} \bmod p$. We call this equation "Feldman verification equation". If the verification fails, $\mathcal{P}_j$ broadcasts a *complaint* against the dealer.
3. The dealer receiving a complaint from player $\mathcal{P}_j$ broadcasts s_j that satisfies the Feldman verification equation.
4. The dealer is *disqualified* if either there are more than t complaints in Step 2, or its answer to a complaint in Step 3 does not satisfy the Feldman verification equation.

Pedersen-VSS(t). This protocol allows a dealer to share a pair of secrets $(s, s') \in_R Z_q^2$ among a set of players $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ via two t-degree polynomials [P91]. This protocol also uses a pair of bases (g, h) so that $log_g h$ is unknown. As a consequence, $\mathcal{P}_i$ holds a pair of shares (s_i, s'_i) such that $s \overset{(t+1,n)}{\longleftrightarrow} (s_1, \cdots, s_n)$ and $s' \overset{(t+1,n)}{\longleftrightarrow} (s'_1, \cdots, s'_n)$.

1. The dealer chooses two random polynomials $f(z) = a_0 + a_1 z + \cdots + a_t z^t$ and $f'(z) = b_0 + b_1 z + \cdots + b_t z^t$ over Z_q. Let $s \stackrel{def}{=} a_0$ and $s' \stackrel{def}{=} b_0$. It broadcasts $C_l = g^{a_l} h^{b_l} \bmod p$ for $0 \le l \le t$, computes and secretly sends $s_j = f(j)$, $s'_j = f'(j)$ to $\mathcal{P}_j$ for $1 \le j \le n$.
2. $\mathcal{P}_j$ $(1 \le j \le n)$ verifies if $g^{s_j} h^{s'_j} = \prod_{l=0}^{t} (C_l)^{j^l} \bmod p$. We call this equation "Pedersen verification equation". If the verification fails, $\mathcal{P}_j$ broadcasts a *complaint* against the dealer.
3. The dealer receiving a complaint from $\mathcal{P}_j$ broadcasts the values s_j, s'_j that satisfy the Pedersen verification equation.
4. The dealer is *disqualified* if either there are more than t complaints in Step 2, or its answer to a complaint in Step 3 does not satisfy the Pedersen verification equation.

Joint-Pedersen-RVSS(t). This protocol allows a set of players $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ to jointly generate a pair of secrets $(a, b) \in_R Z_q^2$ via a pair of t-degree polynomials. As a consequence, $\mathcal{P}_i$ holds a pair of shares (a_i, b_i) of (a, b) such that $a \stackrel{(t+1,n)}{\longleftrightarrow} (a_1, \cdots, a_n)$ and $b \stackrel{(t+1,n)}{\longleftrightarrow} (b_1, \cdots, b_n)$.

1. $\mathcal{P}_i$ $(1 \le i \le n)$, as a dealer, performs an instance of Pedersen-VSS(t) to share a pair of secrets $(a_{i0}, b_{i0}) \in_R Z_q^2$ such that $a_{i0} \stackrel{(t+1,n)}{\longleftrightarrow} (s_{i1}, \cdots, s_{in})$ and $b_{i0} \stackrel{(t+1,n)}{\longleftrightarrow} (s'_{i1}, \cdots, s'_{in})$.
2. $\mathcal{P}_i$ $(1 \le i \le n)$ builds the set of non-disqualified players $QUAL$. This is a unique global name depending on the broadcast information available to all of the honest players.
3. $\mathcal{P}_i$ $(1 \le i \le n)$ holds (a_i, b_i) so that $a_i = \sum_{j \in QUAL} s_{ji} \bmod q$ is its share of $a = \sum_{i \in QUAL} a_{i0} \bmod q$, and $b_i = \sum_{j \in QUAL} s'_{ji} \bmod q$ is its share of $b = \sum_{i \in QUAL} b_{i0} \bmod q$.

Rand-Gen(t). This protocol allows a set of n servers to collaboratively generate $g^a \bmod p$ such that $a \in_R Z_q$ and $a \stackrel{(t+1,n)}{\longleftrightarrow} (a_1, \cdots, a_n)$ [GJKR99].

1. The severs $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ execute Joint-Pedersen-RVSS(t).
2. $\mathcal{P}_i$ $(i \in QUAL)$ exposes $y_i = g^{a_i} \bmod p$ as follows.
   a) $\mathcal{P}_i$ broadcasts $A_{il} = g^{a_{il}} \bmod p$ for $0 \le l \le t$.
   b) $\mathcal{P}_j$ $(1 \le j \le n)$ verifies if $g^{s_{ij}} = \prod_{l=0}^{t} (A_{il})^{j^l} \bmod p$. If the verification fails for some index i, $\mathcal{P}_j$ broadcasts a complaint against $\mathcal{P}_i$ by broadcasting s_{ij} and s'_{ij} that satisfy the Pedersen verification equation but not the Feldman verification equation.
   c) If there is a valid complaint against $\mathcal{P}_i$ (i.e., the broadcast s_{ij} and s'_{ij} satisfy the Pedersen verification equation but not the Feldman verification equation), the servers reconstruct and publish a_{i0}, $f_i(z)$, A_{il} for $0 \le l \le t$. Each server in $QUAL$ sets $y_i = A_{i0} = g^{a_{i0}} \bmod p$ and $y = \prod_{i \in QUAL} y_i \bmod p$.
   d) As a result, $\mathcal{P}_j$ holds its share $a_j = \sum_{i \in QUAL} s_{ij} \bmod q$ of a. Note that $g^{a_j} = \prod_{i \in QUAL} g^{s_{ij}} \bmod p$ for $1 \le j \le n$ are publicly known.

EXP-INTERPOLATE. Given a set of values $(v_1, \cdots, v_n)$ where $n \geq 2t+1$, if at most t of them are *null* and the remaining are of the form $g^{a_i} \bmod p$ where the a_i's lie on some t-degree polynomial $F(\cdot)$ over Z_q, then we can compute $g^{F(0)} = \prod_{i \in \Gamma}(v_i)^{\lambda_{i,\Gamma}} = \prod_{i \in \Gamma}(g^{a_i})^{\lambda_{i,\Gamma}}$, where Γ is a $(t+1)$-subset of the correct v_i's and the $\lambda_{i,\Gamma}$'s are the corresponding Lagrange interpolation coefficients. Let $v = \text{EXP-INTERPOLATE}(v_1, \cdots, v_n)$.

DOUBLE-EXP-INTERPOLATE. Suppose that $(u, v) \in \mathcal{G}^2$, and that (a_i, b_i) for $1 \leq i \leq n$ are the shares output in an instance of JOINT-PEDERSEN-RVSS(t) with polynomials $F(\cdot)$, $F'(\cdot)$ and bases g, h. Given a set of values $(\varphi_1, \cdots, \varphi_n)$ where $n \geq 2t+1$, if at most t of them are *null* and the remaining are of the form $u^{a_i} v^{b_i} \bmod p$, then we can compute

$$\begin{aligned}\varphi &= \prod_{i \in \Gamma}(\varphi_i)^{\lambda_{i,\Gamma}} = \prod_{i \in \Gamma}(u^{a_i} v^{b_i})^{\lambda_{i,\Gamma}} = \prod_{i \in \Gamma}(u^{a_i})^{\lambda_{i,\Gamma}}(v^{b_i})^{\lambda_{i,\Gamma}} \\ &= (\prod_{i \in \Gamma}(u^{a_i})^{\lambda_{i,\Gamma}})(\prod_{i \in \Gamma}(v^{b_i})^{\lambda_{i,\Gamma}}) = u^{F(0)} v^{F'(0)} \bmod p,\end{aligned}$$

where Γ is a $(t+1)$-subset of the correct φ_i's, and the $\lambda_{i,\Gamma}$'s are the Lagrange interpolation coefficients. Let $\varphi = \text{DOUBLE-EXP-INTERPOLATE}(\varphi_1, \cdots, \varphi_n)$.

5 A Fair Offline RSA E-cash Scheme

We assume both $H : \{0,1\}^* \to Z_N^*$ and $H_1 : \{0,1\}^* \to \mathcal{G}$ behave like random oracles [BR93], where N is an RSA modulus and $\mathcal{G}$ is a group over which a DLOG-based public key cryptosystem is defined (see details below). The scheme consists of five protocols: THE INITIALIZATION, THE WITHDRAWAL, THE PAYMENT, THE DEPOSIT, and THE REVOCATION.

5.1 The Initialization Protocol

This protocol has the following steps.

1. Given a security parameter κ, the bank generates a pair of RSA public and private keys $(\langle e, N\rangle, \langle d, N\rangle)$ (e.g., $e = 3$) such that $ed = 1 \bmod \phi(N)$. Let l be the secondary security parameter.
2. Given a security parameter κ', the n revocation servers $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ generate a pair of public and private keys $(\langle p, q, g, h, y\rangle, \langle p, q, g, h, x_1, x_2\rangle)$ where $y = g^{x_1} h^{x_2} \bmod p$. This may include the following steps.
   a) GENERATING (p, q, g). This is a standard process.
   b) GENERATING h. The revocation servers run a distributed coin-flipping protocol to generate $r \in_R Z_p^*$. Then, it is sufficient to set $h = r^{(p-1)/q} \bmod p$. This is so because if q^2 does not divide $p-1$ then h is a random element in the group generated by g.

c) GENERATING y. The revocation servers run JOINT-PEDERSEN-RVSS(t) to share a random pair of $(a, b) \in_R Z_q^2$. Let $y = g^{x_1}h^{x_2} \stackrel{def}{=} g^a h^b = \prod_{i \in QUAL} g^{a_{i0}} h^{b_{i0}} \bmod p$, which is publicly known. Note that $x_1 = a = \sum_{i \in QUAL} a_{i0} \bmod q$ and $x_2 = b = \sum_{i \in QUAL} b_{i0} \bmod q$, that $\mathcal{P}_i$ $(1 \leq i \leq n)$ holds its shares $[x_1]_i = a_i = \sum_{j \in QUAL} s_{ji} \bmod q$ and $[x_2]_i = b_i = \sum_{j \in QUAL} s'_{ji} \bmod q$, and that $g^{[x_1]_i} h^{[x_2]_i} = \prod_{j \in QUAL} g^{s_{ji}} h^{s'_{ji}} \bmod p$ for $1 \leq i \leq n$ are publicly known.

5.2 The Withdrawal Protocol

The withdrawal protocol (between Alice and the bank) goes as follows.

1. Alice chooses $r_i \in_R Z_N^*$, m_i, and $k_i \in_R Z_q$, computes blinded candidates $B_i = r_i^e \cdot H(m_i) \bmod N$ and encryptions
$$(\alpha_i = g^{k_i} \bmod p, \beta_i = h^{k_i} \bmod p, \gamma_i = H_1(m_i) \cdot y^{k_i} \bmod p)$$
for $1 \leq i \leq l$. Then, Alice sends $\{(B_i, \alpha_i, \beta_i, \gamma_i)\}_{1 \leq i \leq l}$ to the bank. Here we assume that the m_i's are signature verification keys without pinning down any concrete signature scheme; as having said before, a special instantiation is an one-time signature scheme as in the Chaum-Fiat-Naor scheme.
2. The bank chooses a random subset of $l/2$ indices $R = \{i_j\}_{1 \leq i_j \leq l, 1 \leq j \leq l/2}$, and sends R to Alice.
3. Alice sends $\{(r_i, m_i, k_i)\}_{i \in R}$ to the bank.
4. The bank checks the semantical correctness of the responses. To simplify notations, let $R = \{l/2+1, \cdots, l\}$. Now, the bank gives Alice $\prod_{i=1}^{l/2} B_i^{1/e} \bmod N$ and charges her account (for instance) one dollar.
5. Alice obtains the e-coin
$$C = (m_1, \cdots, m_{l/2}; \prod_{i=1}^{l/2} H(m_i)^{1/e} \bmod N).$$
Alice re-indexes the m's in C to be lexicographic on their representation: $H(m_1) < H(m_2) < \cdots < H(m_{l/2})$.

5.3 The Payment Protocol

Suppose there is an anonymous channel between a user, Alice, and a merchant, Bob. The payment protocol has the following steps.

1. Alice sends Bob an e-coin $C = (m_1, \cdots, m_{l/2}; \prod_{i=1}^{l/2} H(m_i)^{1/e} \bmod N)$, and signatures that can be verified using the verification keys $(m_1, \cdots, m_{l/2})$.
2. Bob verifies the semantical correctness of the e-coin and the signatures.

5.4 The Deposit Protocol

The deposit protocol (between Bob and the bank) goes as follows.

1. Bob sends the bank an e-coin C and the corresponding signatures that can be verified using the verification keys $(m_1, \cdots, m_{l/2})$.
2. The bank verifies the semantical correctness of the payment. If the e-coin C has not been deposited before, then it credits Bob's account; otherwise, the bank invokes THE OWNER REVOCATION PROTOCOL below.

5.5 The Revocation Protocols

We consider two types of revocations: The Coin Revocation Protocol and The Owner Revocation Protocol.

The Coin Revocation Protocol. This protocol enables the servers to compute the e-coin issued in a given withdrawal session. Given a withdrawal session with ciphertexts

$$(\alpha_i = g^{k_i} \bmod p, \beta_i = h^{k_i} \bmod p, \gamma_i = H_1(m_i) \cdot y^{k_i} \bmod p)$$

for $1 \le i \le l/2$, the servers only need to collaboratively decrypt the ciphertexts and publish the corresponding plaintexts: $H_1(m_1), \cdots, H_1(m_{l/2})$. The distributed decryption operation goes as follows.

1. For $1 \le i \le l/2$, server $\mathcal{P}_a$ $(1 \le a \le n)$ publishes $\delta_{i,a} = \alpha_i^{[x_1]_a} \beta_i^{[x_2]_a} \bmod p$. To ensure robustness, $\mathcal{P}_a$ also proves
$$REP(g, h; g^{[x_1]_a} h^{[x_2]_a}) = REP(\alpha_i, \beta_i; \delta_{i,a})$$
using a non-interactive or interactive proof (in the latter case the verifier could be a honest one played by the servers that collaboratively choose a random challenge).
2. For $1 \le i \le l/2$, the servers compute $H_1(m_i) = \gamma_i / \delta_i \bmod p$, where
$$\delta_i = \text{Double-EXP-Interpolate}(\delta_{i,1}, \cdots, \delta_{i,n}).$$

Once the plaintexts $H_1(m_1), \cdots, H_1(m_{l/2})$ are known, an e-coin with

$$C = (m'_1, \cdots, m'_{l/2}; \textstyle\prod_{i=1}^{l/2} H(m'_i)^{1/e} \bmod N)$$

matches the withdrawal session if $H_1(m_i) = H_1(m'_j)$ for some $1 \le i, j \le l/2$.

The Owner Revocation Protocol. This protocol enables the servers to compute the owner of a given e-coin. For this purpose, we let the servers compare a given e-coin, $C = (m'_1, \cdots, m'_{l/2}; \prod_{i=1}^{l/2} H(m'_i)^{1/e} \bmod N)$, with *each* candidate withdrawal session of publicly known $\{(\alpha_i, \beta_i, \gamma_i)\}_{1 \le i \le l/2}$. We say "the e-coin C matches the withdrawal session" if there exist $1 \le i, j \le l/2$ such that $REP(g, h; y) = REP(\alpha_i, \beta_i; \gamma_i / H_1(m'_j))$.

An intuitive solution to the problem of deciding whether a given e-coin matches a given withdrawal session is to let the servers conduct a distributed proof $REP(g, h; y) = REP(\alpha_i, \beta_i; \gamma_i / H_1(m'_j))$. However, this would leak the plaintext information. To avoid this information leakage, it is natural to inject randomness into the revocation process. For example, the servers could compute $\theta = g^z \bmod p$, $\delta = \gamma_i / H_1(m'_j) \bmod p$, and $\Delta = [(\alpha_i)^z]^{x_1} [(\beta_i)^z]^{x_2} \bmod p$ for some $z \in_R Z_q$, and conduct a distributed protocol attempting to prove $DLOG(g, \theta) = DLOG(\delta, [(\alpha_i)^z]^{x_1} [(\beta_i)^z]^{x_2})$. While this is reasonable, we find an even better way to facilitate the task, namely letting the servers compute and compare $\delta^z \stackrel{?}{=} \Delta$. Note that $\delta^z = \Delta$ (therefore, the revocation process outputs Yes) if and only if $REP(g, h; y) = REP(\alpha_i, \beta_i; \gamma_i / H_1(m'_j))$ except for a negligible probability.

Specifically, the following protocol loops for $1 \le i \le l/2$ and $1 \le j \le l/2$, and may halt when it outputs the first Yes (there are various alternatives depending on the system management policy that is beyond the scope of this paper). Note that the protocol, for the sake of clarifying the presentation, is not round-optimal, but it is easy to see that rounds 3-8 can be merged into 2 rounds.

1. Define $\delta = \gamma_i / H_1(m'_j) \bmod p$.
2. The servers execute RANDOM-GEN(t) to generate $\theta = g^z \bmod p$ so that $z \overset{(t+1,n)}{\longleftrightarrow} (z_1, \cdots, z_n)$, where $\theta_a = g^{z_a} \bmod p$, $1 \le a \le n$, are publicly known.
3. Server $\mathcal{P}_a$ $(1 \le a \le n)$ broadcasts $\sigma_a = \delta^{z_a} \bmod p$. To ensure robustness, $\mathcal{P}_a$ proves in zero-knowledge $DLOG(g, \theta_a) = DLOG(\delta, \sigma_a)$.
4. The servers compute $\sigma = \text{EXP-INTERPOLATE}(\sigma_1, \cdots, \sigma_n)$.
5. Server $\mathcal{P}_a$ $(1 \le a \le n)$ broadcasts $\mu_a = \alpha_i^{z_a} \bmod p$. To ensure robustness, $\mathcal{P}_a$ proves in zero-knowledge $DLOG(g, \theta_a) = DLOG(\alpha_i, \mu_a)$.
6. The servers compute $\mu = \text{EXP-INTERPOLATE}(\mu_1, \cdots, \mu_n)$.
7. Server $\mathcal{P}_a$ $(1 \le a \le n)$ broadcasts $\nu_a = \beta_i^{z_a} \bmod p$. To ensure robustness, $\mathcal{P}_a$ proves in zero-knowledge $DLOG(g, \theta_a) = DLOG(\beta_i, \nu_a)$.
8. The servers compute $\nu = \text{EXP-INTERPOLATE}(\nu_1, \cdots, \nu_n)$.
9. Server $\mathcal{P}_a$ $(1 \le a \le n)$ broadcasts $\Delta_a = \mu^{[x_1]_a} \nu^{[x_2]_a} \bmod p$. To ensure robustness, $\mathcal{P}_a$ proves in zero-knowledge $REP(g, h; \psi_a) = REP(\mu, \nu; \Delta_a)$, where $\psi_a = g^{[x_1]_a} h^{[x_2]_a} \bmod p$ is publicly known.
10. The servers compute $\Delta = \text{DOUBLE-EXP-INTERPOLATE}(\Delta_1, \cdots, \Delta_n)$.
11. If $\Delta = \sigma$, then the protocol outputs YES meaning that $H_1(m'_j)$ is the plaintext corresponding to the ciphertext $(\alpha_i, \beta_i, \gamma_i)$; otherwise, the protocol outputs NO meaning that $H_1(m'_j)$ is not the plaintext corresponding to the ciphertext $(\alpha_i, \beta_i, \gamma_i)$.

6 Properties of the Fair Offline RSA E-cash Scheme

Let cfn denote the original Chaum-Fiat-Naor scheme [CFN88], and folc denote the above fair off-line scheme. As we said before, we focus on unforgeability, revocability, and anonymity.

6.1 Unforgeability

We reduce the unforgeability of cfn to the unforgeability of folc. Due to space limitation, we leave formal analysis to the full version of this paper.

6.2 Revocability

Revocability of folc is ensured by the cut-and-choose technique which assures that one element in the coin structure is decryptable by the authorities. Specifically, as in cfn, the probability that a dishonest user passes the cut-and-choose verification without presenting any appropriately ciphertext is negligible.

6.3 Anonymity

We consider two types of anonymity. First, no adversary can link a withdrawal session initiated by a honest user to the corresponding e-coin. To prove this (in Theorem 1), we need to ensure that THE COIN REVOCATION PROTOCOL does not leak any significant information about the private key except the plaintext

(this is proved in Lemma 1), and that THE OWNER REVOCATION PROTOCOL leaks no information about the private key and no information about the corresponding plaintext in the case that $(\alpha_i, \beta_i, \gamma_i)$ is no encryption of $H_1(m'_j)$ (this is proved in Lemma 2). Second, no adversary can link two e-coins withdrawn by a honest user (i.e., *unlinkability*) even if the user is not identified (this is proved in Theorem 2).

Before we prove the theorems, let us recall that the adversary is static and t-threshold (namely, it corrupts at most t revocation servers). Without loss of generality, we assume that $\mathcal{P}_1, \cdots, \mathcal{P}_{t'}$ $(t' \leq t)$ were corrupted at initialization, and thus their internal states are known to the adversary.

Lemma 1. *Given a ciphertext (α, β, γ) that appears in a withdrawal session.* THE COIN REVOCATION PROTOCOL *does not leak any information about (x_1, x_2) but the corresponding plaintext $H_1(m) = \gamma/\alpha^{x_1}\beta^{x_2} \bmod p$.*

Proof. (sketch) We construct a polynomial-time algorithm $\mathcal{S}$ to simulate THE COIN REVOCATION PROTOCOL. Suppose $\mathcal{S}$ is given $H_1(m)$ and the shares of (x_1, x_2) held by the corrupt servers: $([x_1]_1, [x_2]_1), \cdots, ([x_1]_{t'}, [x_2]_{t'})$. Recall that $g^{[x_1]_b} h^{[x_2]_b} \bmod p$ for $1 \leq b \leq n$ are publicly known. Let $\delta = \gamma/H_1(m) \bmod p$.

1. $\mathcal{S}$ simulates the process that $\mathcal{P}_b$ $(1 \leq b \leq n)$ publishes $\delta_b = \alpha^{[x_1]_b}\beta^{[x_2]_b} \bmod p$, which is intended to guarantee $\delta =$ DOUBLE-EXP-INTERPOLATE$(\delta_1, \cdots, \delta_n)$. For this purpose $\mathcal{S}$ executes as follows.
   a) $\mathcal{S}$ chooses $a_0 \in_R Z_q$, computes $\delta' = \delta/\alpha^{a_0} \bmod p$ which can be understood as $\beta^{a'_0} \bmod p$ for some unknown a'_0. Note that $\delta = \alpha^{a_0}\beta^{a'_0} \bmod p$.
   b) Note that $A_0 \stackrel{def}{=} \alpha^{a_0}$, $f(1) \stackrel{def}{=} [x_1]_1, \cdots, f(t') \stackrel{def}{=} [x_1]_{t'}$, $f(t'+1) \stackrel{def}{=} [x_1]^*_{t'+1} \in_R Z_q, \cdots, f(t) \stackrel{def}{=} [x_1]^*_t \in_R Z_q$ uniquely determine a t-degree polynomial $f(\eta) = a_0 + a_1\eta + \cdots + a_t\eta^t \bmod q$. So, $\mathcal{S}$ computes $A_v = \alpha^{a_v} = (A_0)^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\alpha^{[x_1]_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\alpha^{[x_1]^*_b})^{\lambda_{vb}}$ for $1 \leq v \leq t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\alpha^{[x_1]^*_b} \stackrel{def}{=} \alpha^{f(b)} = \prod_{v=0}^{t} (A_v)^{b^v}$ for $t+1 \leq b \leq n$.
   c) Note that $A'_0 \stackrel{def}{=} \sigma' = \beta^{a'_0}$ where a'_0 is unknown to $\mathcal{S}$, $f'(1) \stackrel{def}{=} [x_2]_1, \cdots, f'(t') \stackrel{def}{=} [x_2]_{t'}$, $f'(t'+1) \stackrel{def}{=} [x_2]^*_{t'+1} \in_R Z_q, \cdots, f'(t) \stackrel{def}{=} [x_2]^*_t \in_R Z_q$ uniquely determine a t-degree polynomial $f'(\eta) = a'_0 + a'_1\eta + \cdots + a'_t\eta^t \bmod q$. So, $\mathcal{S}$ computes $A'_v = \beta^{a'_v} = (A'_0)^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\beta^{[x_2]_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\beta^{[x_2]^*_b})^{\lambda_{vb}}$ for $1 \leq v \leq t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\beta^{[x_2]^*_b} \stackrel{def}{=} \beta^{f'(b)} = \prod_{v=0}^{t} (A'_v)^{b^v}$ for $t+1 \leq b \leq n$.
   d) For $1 \leq b \leq t'$, $\mathcal{S}$ executes at the adversary's will (e.g., the adversary may ask $\mathcal{S}$ to deviate from the protocol); for $t'+1 \leq b \leq n$, $\mathcal{S}$ publishes $\delta_b = \alpha^{[x_1]^*_b}\beta^{[x_2]^*_b} \bmod p$ and proves $REP(g, h; g^{[x_1]_b} h^{[x_2]_b}) = REP(\alpha, \beta; \delta_b)$ using the corresponding simulator.
2. It is guaranteed that $\delta =$ DOUBLE-EXP-INTERPOLATE$(\delta_1, \cdots, \delta_n)$ and thus $H_1(m) = \gamma/\delta \bmod p$.

Lemma 2. *Given $H_1(m'_j)$ and a ciphertext $(\alpha_i, \beta_i, \gamma_i)$,* The Owner Revocation Protocol *does not leak any information about the secret (x_1, x_2) except the bit whether $(\alpha_i, \beta_i, \gamma_i)$ is an encryption of $H_1(m'_j)$. Furthermore, in the case the process outputs* YES*, $H_1(m_i)$ is of course publicly known; in the case the process outputs* NO*, no information about $H_1(m_i) = \gamma_i/(\alpha_i)^{x_1}(\beta_i)^{x_2}$ is leaked.*

Proof. (sketch) We construct a polynomial-time algorithm $\mathcal{S}$ to simulate The Owner Revocation Protocol. Specifically, suppose an instance of The Owner Revocation Protocol outputs $\theta = g^z$, $\delta = \gamma_i/H_1(m'_j)$, $\sigma = \delta^z$, $\mu = (\alpha_i)^z$, $\nu = (\beta_i)^z$, and $\Delta = \mu^{x_1}\nu^{x_2}$, where either $\Delta = \sigma$ (i.e., the instance outputs Yes) or $\Delta \neq \sigma$ (i.e., the instance outputs No). Let $H_1(m_i) = \gamma_i/(\alpha_i)^{x_1}(\beta_i)^{x_2}$. Note that in the case $\Delta \neq \sigma$, the leakage of information about z may result in the leakage of information about the plaintext $H_1(m_i)$ because $\sigma/\Delta = [H_1(m_i)/H_1(m'_j)]^z$. Note also that $([x_1]_1, [x_2]_1)$, $\cdots$, $([x_1]_{t'}, [x_2]_{t'})$ are given to the simulator $\mathcal{S}$. $\mathcal{S}$ executes as follows.

1. $\mathcal{S}$ computes $\delta = \gamma_i/H_1(m'_j) \bmod p$, which is the same as the one given to $\mathcal{S}$.
2. $\mathcal{S}$ emulates the execution of Random-GEN(t) to generate $\theta = g^z \bmod p$, which is given as an input. This is done by calling the simulator in Appendix A. As a consequence, it is guaranteed that $z \overset{(t+1,n)}{\longleftrightarrow} (z_1, \cdots, z_n)$, where $\theta_b = g^{z_b} \bmod p$ for $1 \leq b \leq n$ are publicly known.
3. To simulate the generation of $\sigma = \delta^z$, $\mathcal{S}$ executes as follows.
   a) Note that $A_0^* \overset{def}{=} \sigma = \delta^{a_0^*}$ where $a_0^* \overset{def}{=} z$, $f^*(1) \overset{def}{=} z_1$, $\cdots$, $f^*(t') \overset{def}{=} z_{t'}$, $f^*(t'+1) \overset{def}{=} z^*_{t'+1} \in_R Z_q$, $\cdots$, $f^*(t) \overset{def}{=} z_t^* \in_R Z_q$ uniquely determine a t-degree polynomial $f^*(\eta) = a_0^* + a_1^*\eta + \cdots + a_t^*\eta^t \bmod q$. So, $\mathcal{S}$ computes $A_v^* = \delta^{a_v^*} = (A_0^*)^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\delta^{z_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\delta^{z_b^*})^{\lambda_{vb}}$ for $1 \leq v \leq t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\delta^{z_b^*} \overset{def}{=} \delta^{f^*(b)} = \prod_{v=0}^{t} (A_v^*)^{b^v}$ for $t+1 \leq b \leq n$.
   b) For $1 \leq b \leq t'$, $\mathcal{S}$ emulates $\mathcal{P}_b$ at the adversary's will; for $t'+1 \leq b \leq n$, $\mathcal{S}$ emulates $\mathcal{P}_b$ by broadcasting $\sigma_b = \delta^{z_b^*} \bmod p$ and proving $DLOG(g, \theta_b) = DLOG(\delta, \sigma_b)$ using the corresponding simulator.
4. It is guaranteed that $\sigma = \delta^z = \text{EXP-Interpolate}(\sigma_1, \cdots, \sigma_n)$.
5. To simulate the generation of $\mu = \alpha_i^z$, $\mathcal{S}$ executes as follows.
   a) Note that $A_0^* \overset{def}{=} \mu = \alpha_i^{a_0^*}$ where $a_0^* \overset{def}{=} z$, $f^*(1) \overset{def}{=} z_1$, $\cdots$, $f^*(t') \overset{def}{=} z_{t'}$, $f^*(t'+1) \overset{def}{=} z^*_{t'+1} \in_R Z_q$, $\cdots$, $f^*(t) \overset{def}{=} z_t^* \in_R Z_q$ uniquely determine a t-degree polynomial $f^*(\eta) = a_0^* + a_1^*\eta + \cdots + a_t^*\eta^t \bmod q$. So, $\mathcal{S}$ computes $A_v^* = \alpha_i^{a_v^*} = (A_0^*)^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\alpha_i^{z_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\alpha_i^{z_b^*})^{\lambda_{vb}}$ for $1 \leq v \leq t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\alpha_i^{z_b^*} \overset{def}{=} \alpha_i^{f^*(b)} = \prod_{v=0}^{t} (A_v^*)^{b^v}$ for $t+1 \leq b \leq n$.
   b) For $1 \leq b \leq t'$, $\mathcal{S}$ emulates $\mathcal{P}_b$ at the adversary's will; for $t'+1 \leq b \leq n$, $\mathcal{S}$ emulates $\mathcal{P}_b$ by broadcasting $\mu_b = \alpha_i^{z_b^*} \bmod p$ and proving $DLOG(g, \theta_b) = DLOG(\alpha_i, \mu_b)$ using the corresponding simulator.
6. It is guaranteed that $\mu = \alpha_i^z = \text{EXP-Interpolate}(\mu_1, \cdots, \mu_n)$.
7. To simulate the generation of $\nu = \beta_i^z$, $\mathcal{S}$ executes as follows.

a) Note that $A_0^* \stackrel{def}{=} \nu = \beta_i^{a_0^*}$ where $a_0^* \stackrel{def}{=} z$, $f^*(1) \stackrel{def}{=} z_1, \cdots, f^*(t') \stackrel{def}{=} z_{t'}$, $f^*(t'+1) \stackrel{def}{=} z^*_{t'+1} \in_R Z_q, \cdots, f^*(t) \stackrel{def}{=} z_t^* \in_R Z_q$ uniquely determine a t-degree polynomial $f^*(\eta) = a_0^* + a_1^*\eta + \cdots + a_t^*\eta^t \bmod q$. So, $\mathcal{S}$ computes $A_v^* = \beta_i^{a_v^*} = (A_0^*)^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\beta_i^{z_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\beta_i^{z_b^*})^{\lambda_{vb}}$ for $1 \le v \le t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\beta_i^{z_b^*} \stackrel{def}{=} \beta_i^{f^*(b)} = \prod_{v=0}^{t} (A_v^*)^{b^v}$ for $t+1 \le b \le n$.

b) For $1 \le b \le t'$, $\mathcal{S}$ emulates $\mathcal{P}_b$ at the adversary's will; for $t'+1 \le b \le n$, $\mathcal{S}$ emulates $\mathcal{P}_b$ by broadcasting $\nu_b = \beta_i^{z_b^*} \bmod p$ and proving $DLOG(g, \theta_b) = DLOG(\beta_i, \nu_b)$ using the corresponding simulator.

8. It is guaranteed that $\nu = \beta_i^z = \text{EXP-INTERPOLATE}(\nu_1, \cdots, \nu_n)$.
9. In order to simulate the generation of $\Delta = \mu^{x_1}\nu^{x_2}$, $\mathcal{S}$ executes as follows.

a) $\mathcal{S}$ chooses $a_0 \in_R Z_q$, computes $\sigma' = \Delta/\mu^{a_0} \bmod p$ which can be understood as $\nu^{a_0'} \bmod p$ for some unknown a_0'. Note that $\Delta = \mu^{a_0}\nu^{a_0'} \bmod p$.

b) Note that $A_0 \stackrel{def}{=} \mu^{a_0}$, $f(1) \stackrel{def}{=} [x_1]_1, \cdots, f(t') \stackrel{def}{=} [x_1]_{t'}$, $f(t'+1) \stackrel{def}{=} [x_1]^*_{t'+1} \in_R Z_q, \cdots, f(t) \stackrel{def}{=} [x_1]_t^* \in_R Z_q$ uniquely determine a t-degree polynomial $f(\eta) = a_0 + a_1\eta + \cdots + a_t\eta^t \bmod q$. So, $\mathcal{S}$ computes $A_v = \mu^{a_v} = (A_0)^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\mu^{[x_1]_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\mu^{[x_1]_b^*})^{\lambda_{vb}}$ for $1 \le v \le t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\mu^{[x_1]_b^*} \stackrel{def}{=} \mu^{f(b)} = \prod_{v=0}^{t} (A_v)^{b^v}$ for $t+1 \le b \le n$.

c) Note that $A_0' \stackrel{def}{=} \sigma' = \nu^{a_0'}$ where a_0' is unknown to $\mathcal{S}$, $f'(1) \stackrel{def}{=} [x_2]_1, \cdots, f'(t') \stackrel{def}{=} [x_2]_{t'}$, $f'(t'+1) \stackrel{def}{=} [x_2]^*_{t'+1} \in_R Z_q, \cdots, f'(t) \stackrel{def}{=} [x_2]_t^* \in_R Z_q$ uniquely determine a t-degree polynomial $f'(\eta) = a_0' + a_1'\eta + \cdots + a_t'\eta^t \bmod q$. So, $\mathcal{S}$ computes $A_v' = \nu^{a_v'} = (A_0')^{\lambda_{v0}} \cdot \prod_{b=1}^{t'} (\nu^{[x_2]_b})^{\lambda_{vb}} \cdot \prod_{b=t'+1}^{t} (\nu^{[x_2]_b^*})^{\lambda_{vb}}$ for $1 \le v \le t$, where the λ_{vb}'s are the Lagrange interpolation coefficients. Finally, $\mathcal{S}$ computes $\nu^{[x_2]_b^*} \stackrel{def}{=} \nu^{f'(b)} = \prod_{v=0}^{t} (A_v')^{b^v}$ for $t+1 \le b \le n$.

d) For $1 \le b \le t'$, $\mathcal{S}$ executes according to the adversary's requirements (e.g., the adversary may ask $\mathcal{S}$ to deviate from the protocol); for $t'+1 \le b \le n$, $\mathcal{S}$ publishes $\Delta_b = \mu^{[x_1]_b^*}\nu^{[x_2]_b^*}$ and proves $REP(g, h; g^{[x_1]_b}h^{[x_2]_b}) = REP(\mu, \nu; \mu^{[x_1]_b^*}\nu^{[x_2]_b^*})$ using the corresponding simulator.

10. It is guaranteed that $\Delta = \text{DOUBLE-EXP-INTERPOLATE}(\Delta_1, \cdots, \Delta_n)$.
11. If $\Delta = \sigma$, then $\mathcal{S}$ outputs YES; otherwise, $\mathcal{S}$ outputs NO.

Theorem 1. *Suppose the DDH assumption holds. Then, no t-threshold adversary, who is given a ciphertext (α, β, γ) that appears in a withdrawal session initiated by a honest user, and $H_1(m)$, is able to correctly decide whether (α, β, γ) is an encryption of $H_1(m)$ with non-negligible advantage over a random guess.*

The proof will be given in the full version of this paper.

Theorem 2. *(unlinkability)* folc *is unlinkable meaning that no t-threshold adversary can link two e-coins to the same honest user (although the user's identity is unknown).*

Proof. (sketch) This is so because (1) all the m_i's chosen by the honest users are independently and uniformly distributed, and (2) Theorem 1 showed that folc, compared with cfn, does not leak any computational information about the secret (x_1, x_2) or the $H_1(m)$'s of the honest users'.

7 Discussions and Extensions

On the Efficiency of the Fair Off-Line RSA E-Cash Scheme. The penalty for fairness is the computational overhead at the user side, namely a user needs to compute $3l$ exponentiations for each e-coin (but all the exponentiations are pre-computable). The communication overhead at the user side is almost the same as in the Chaum-Fiat-Naor scheme. The time complexity of deciding whether an e-coin matches a withdrawal session is $O(l^2)$ comparison operations; this may be no real concern because that the servers deployed in threshold cryptosystems are typically powerful, and that the technique developed in [JM99] could be adopted to improve the revocation efficiency.

On Anonymity against "Suicide" Attacks. In the analysis of Section 6, we have taken into consideration the attack that the adversary attempts to distinguish a simulation from the real-world system by invoking the revocation process. Here we take a further step in investigating the possibility that the adversary attempts to break the anonymity of a honest user by conducting the following "suicide" attack. Suppose an adversary intercepts (from a withdrawal session initiated by a honest user) a ciphertext $(\alpha = g^k \bmod p, \beta = h^k \bmod p, \gamma = H_1(m) \cdot y^k \bmod p)$ that remains un-opened after the cut-and-choose verification. Furthermore, the adversary initiates a withdrawal session into which it embeds $(\alpha \cdot g^{k'} \bmod p, \beta \cdot h^{k'} \bmod p, \gamma \cdot y^{k'} \bmod p)$ as the i^{th} component ciphertext, where $i \in_R \{1, \cdots, l\}$. Note in this case the adversary passes the cut-and-choose verification with probability 0.5, which is enough. Then, the adversary commits some suspicious activities so that The Coin Revocation Protocol will be invoked and $H_1(m)$ will be publicly known. As a consequence, the adversary is able to associate the honest user's payment with her withdrawal. Note that this issue is typically ignored in e-cash protocols, although such a "suicide" attack against a honest user's anonymity is expensive because the adversary, whose account has been charged, can not spend that e-coin.

One may suggest that the above "suicide" attack can be blocked by making the communication channel in the withdrawal protocol confidential (e.g., using an appropriate cryptosystem). Clearly, a simple-minded encryption so that each candidate $(B_i, \alpha_i, \beta_i, \gamma_i)$ is individually encrypted using (for example) the bank's public key does not prevent an adversary from applying the cut-and-paste technique. Even more involved protocol (e.g., each withdrawal session could be protected by deploying a fresh session key generated using an authenticated key exchange protocol) does not completely solve the problem because the model, which reflects the well-known security principle called *separation-of-duty*, implies that the bank is not necessarily trusted in preserving the users' anonymity. Clearly, it does not help us to let the bank distribute the decryption capabili-

ties (corresponding to the fresh session keys) among a set of servers, since these servers are under the control of the bank.

In summary, the powerful but *expensive* "suicide" attack, which perhaps involves the bank that is not necessarily trusted in preserving the users' anonymity, is able (in some extreme cases as above) to compromise the anonymity of some honest users in the fair off-line RSA e-cash scheme, but anonymity of most honest users will be preserved.

8 Conclusion

We showed how to incorporate fairness into the Chaum-Fiat-Naor e-cash scheme while preserving their system architecture. The disadvantage of our scheme is the computational overhead at the user side; this may be solvable by deploying certain RSA-based (instead of DLOG-based) cryptosystem.

Acknowledgement. We thank the anonymous reviewers for helpful comments.

References

[BR93] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. ACM CCS'93.

[BNPS01] M. Bellare, C. Namprempre, D. Pointcheval, M. Semanko. The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. Financial Crypto'01.

[BGK95] E. Brickell, P. Gemmell, and D. Kravitz. Trustee-based Tracing Extentions to Anonymous Cash and the Making of Anonymous Change. SODA'95.

[C82] D. Chaum. Blind Signatures for Untraceable Payments. Crypto'82.

[CFN88] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. Crypto'88.

[CMS96] J. Camenisch, U. Maurer, and M. Stadler. Digital Payment Systems with Passive Anonymity-Revoking Trusrees. ESORICS'96.

[CP92] D. Chaum and T. Pedersen. Wallet Databases with Observers. Crypto'92.

[E85] T. El Gamal. A Public-Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm. IEEE Trans. IT, 31(4), 1985, pp 469–472.

[F87] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. FOCS'87.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Crypto'86.

[FTY96] Y. Frankel, Y. Tsiounis, and M. Yung. Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-Cash. Asiacrypt'96.

[FR95] M. Franklin and M. Reiter. Verifiable Signature Scharing. Eurocrypt'95.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. Eurocrypt'99.

[GMR88] S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-message Attacks. SIAM J. Computing, 17(2), 1988.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game-A Completeness Theorem for Protocol with Honest Majority. STOC'87.

[JL00] S. Jarecki and A. Lysyanskaya. Concurrent and Erasure-Free Models in Adaptively-Secure Threshold Cryptography. Eurocrypt'00.
[JM99] M. Jakobsson and J. Mueller. Improved Magic Ink Signatures Using Hints. Financial Crypto'99.
[JY96] M. Jakobsson and M. Yung. Revokable and Versatile Electronic Money. ACM CCS'96.
[J99] A. Juels. Trustee Tokens: Simple and Practical Tracing of Anonymous Digital Cash. Financial Crypto'99.
[MP98] D. M'Raihl and D. Pointcheval. Distributed Trustees and Revocability: A Framework for Internet Payment. Financial Crypto'98.
[P91] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. Crypto'91.
[PS00] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. J. of Cryptology, 13(3), 2000.
[R98] T. Rabin. A Simplified Approach to Threshold and Proactive RSA. Crypto'98.
[RSA78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. CACM, 21(2), 1978, pp 120–126.
[S00] V. Shoup. Practical Threshold Signatures. Eurocrypt'00.
[TY98] Y. Tsiounis and M. Yung. On the Security of ElGamal Based Encryption. PKC'98.
[vSN92] S. von Solms and D. Naccache. On Blind Signatures and Perfect Crimes. Computer and Security, 11, 1992, pp 581–583.

A The Simulator of RANDOM-GEN(t)

This algorithm appeared in [GJKR99]. Given $\theta = g^z \bmod p$, the simulator emulates the generation of θ in the presence of an adversary that corrupts no more than t servers. Denote by $\mathbb{B}$ the set of servers corrupted by the adversary, and by $\mathbb{G}$ the set of honest server (run by the simulator). Without loss of generality, let $\mathbb{B} = \{1, \cdots, t'\}$ and $\mathbb{G} = \{t'+1, \cdots, n\}$, where $t' \leq t$.

INPUT: $\theta = g^z \bmod p$ and parameters (p, q, g, h).

The simulation goes as follows.

1. The simulator executes JOINT-PEDERSEN-RVSS(t) on behalf of the servers in $\mathbb{G}$.
2. The simulator emulates the extraction of $\theta = g^z$.
   - Compute $A_{il} = g^{z_{il}} \bmod p$ for $i \in QUAL \backslash \{n\}$ and $0 \leq l \leq t$.
   - Set $A^*_{n0} = \theta / \prod_{i \in QUAL \backslash \{n\}} A_{i0} \bmod p$.
   - Assign $s^*_{nj} = s_{nj}$ for $1 \leq j \leq t$.
   - Compute $A^*_{nl} = (A^*_{n0})^{\lambda_{l0}} \cdot \prod_{i=1}^{t} (g^{s^*_{ni}})^{\lambda_{li}} \bmod p$ for $1 \leq l \leq t$, where the λ_{li}'s are the Lagrange interpolation coefficients.
     a) Broadcast $\{A_{il}\}_{0 \leq l \leq t}$ for $i \in \mathbb{G} \backslash \{n\}$, and $\{A^*_{nl}\}_{0 \leq l \leq t}$.
     b) Check the values $\{A_{il}\}_{i \in \mathbb{B}, 0 \leq l \leq t}$ using the FELDMAN verification equation. If the verification fails for some $i \in \mathbb{B}$, $j \in \mathbb{G}$, broadcast a complaint (s_{ij}, s'_{ij}).
     c) If necessary, reconstruct the polynomial f_i and publish a_{i0} and $\theta_i = g^{a_{i0}} \bmod p$, where $i \in \mathbb{B}$.