Proof-of-Knowledge of Representation of Committed Value and Its Applications*

Man Ho Au, Willy Susilo and Yi Mu

Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
{aau,wsusilo,ymu}@uow.edu.au

Abstract. We present a zero-knowledge argument system of representation of a committed value. Specifically, for commitments $C = \mathsf{Commit}_1(y)$, $D = \mathsf{Commit}_2(\boldsymbol{x})$, of value y and a tuple $\boldsymbol{x} = (x_1, \ldots, x_L)$, respectively, our argument system allows one to demonstrate the knowledge of $(\boldsymbol{x}, y)$ such that $\boldsymbol{x}$ is a representation of y to bases $h_1, \ldots, h_L$. That is, $y = h_1^{x_1} \cdots h_L^{x_L}$. Our argument system is zero-knowledge and hence, it does not reveal anything such as $\boldsymbol{x}$ or y. We note that applications of our argument system are enormous. In particular, we show how round-optimal cryptography systems, where privacy is of a great concern, can be achieved. We select three interesting applications with the aim to demonstrate the significance our argument system. First, we present a concrete instantiation of two-move concurrently-secure blind signature without interactive assumptions. Second, we present the first compact e-cash with concurrently-secure withdrawal protocol. Finally, we construct two-move traceable signature with concurrently-secure join. On the side note, we present a framing attack against the original traceable signature scheme within the original model.

1 Introduction

The notion of zero-knowledge proof protocol was put forth by Goldwasser, Micali and Rackoff in [33]. In a zero-knowledge proof protocol, a prover convinces a verifier that a statement is true, while the verifier learns nothing except the validity of the assertion. A proof-of-knowledge [6] is a protocol such that the verifier is convinced that the prover knows a certain quantity w satisfying some kinds of relation R with respect to a commonly known string x. That is, the prover convinces the verifier that he knows some w such that $(w, x) \in R$. If it can be done in such a way that the verifier learns nothing besides the validity of the statement, this protocol is called a zero-knowledge proof-of-knowledge (ZKPoK) protocol. Various efficient ZKPoK protocols about knowledge of discrete logarithms and their relations have been proposed in the literature. For instance, knowledge of discrete logarithm [45], polynomial relations of discrete logarithms [14, 26], inequality of discrete logarithms [17], range of discrete logarithms [12] and double discrete logarithm [18].

ZKPoK protocols have been used extensively as building blocks of many cryptosystems. In this paper, we present a ZKPoK protocol for the knowledge of representation of a committed value. We demonstrate that our protocol can be used to construct round-optimal cryptosystems, including blind signatures, traceable signatures and compact e-cash.

1.1 Related Work

ZKPoK of Double-Discrete Logarithm Our protocol generalizes the ZKPoK protocol of double discrete logarithm ,introduced by Stadler [46], when it is used to construct a verifiable secret sharing scheme. Roughly speaking, a double discrete logarithm of an element y to base g and h is an element

* This paper is the full version of the paper to appear in ACISP 2010 under the same title.

x such that $y = g^{h^x}$. Stadler introduces a ZKPoK protocol to demonstrate the knowledge of such x with respect to y. This protocol was employed in the construction of group signatures [18, 2] and a divisible e-cash scheme [19]. Looking ahead, our zero-knowledge protocol further extends Stadler's protocol in which it allows the prover to demonstrate the knowledge of a set of values $(x_1, \ldots, x_L, r)$ such that $y = g^{h_1^{x_1} \cdots h_L^{x_L}} g_0^r$. We would like to stress that there is a *subtle difference* between Stadler's protocol and ours when $L = 1$. Specifically, with the introduction of the variable r, no information about x is leaked to the verifier. This turns out to be very useful when the prover wishes to demonstrate the same x, without being linked, to different verifiers.

Blind signatures Introduced by Chaum [22], blind signature schemes allow a user to obtain interactively a signature on message m from a signer in such a way that the signer learns nothing about m (*blindness*) while at the same time, the user cannot output more signatures than the ones produced from the interaction with the signer (*unforgeability*). The formal definition of blind signatures was first proposed in [44], with the requirement that any user executing the protocol ℓ times with the signer cannot output $\ell + 1$ valid signatures on $\ell + 1$ distinct messages. One important feature of security offered by any blind signature construction is whether the execution of the signing protocol can be performed concurrently, that is, in an arbitrarily-interleaved manner. As pointed out in [30], a notable exception to the problems of constructing schemes secure against interleaving executions are those with an optimal two-move signing protocol, of which the problem of concurrency is solved immediately.

Table 1 summarizes existing schemes that are secure under concurrent execution. Note that [35], [30] and [34] provide generic construction only. [30] relies on generic NIZK while [34] utilizes ZAP. On the other hand, as pointed out in [34], [35] makes use of generic concurrently-secure 2-party computation and constructing such a protocol without random oracle or trusted setup is currently an open problem. Lindell's result [39] states that it is impossible to construct concurrently-secure blind signatures in the plain model if simulation-based definitions are used. Hazay *et al.* [34] overcome this limitation by employing a game-based definition. A construction achieving all properties is proposed in [31] recently.

Schemes	Round-Optimal?	W/o RO?	Non-Interactive Assumption?	Instantiation?
[34]	$\times$	✓	✓	?
[35]	$\times$	✓	✓	$\times$
[30]	✓	✓	✓	?
[7]	✓	$\times$	$\times$	✓
[9]	✓	$\times$	$\times$	✓
[41]	$\times$	✓	✓	✓
[31]	✓	✓	✓	✓
Our Scheme	✓	$\times$	✓	✓

Table 1. Summary of Existing Blind Signatures Secure under Concurrent Signature Generation

Traceable Signatures Introduced by Chaum and van Heyst [23], group signatures allow a group member to sign anonymously on behalf of the group. Whenever required, the identity of the signature's originator can be revealed only by the designated party. Traceable signatures, introduced in [36], are group signatures with added functionality in which a designated party could output some tracing information on a certain user that allows the bearer to trace *all* signatures generated by that user. Subsequently, another traceable signature is propose in [24]. We discover a flaw in the security proof of [36] and are able to develop a concrete attack against their scheme under their model. Table 2 summarizes existing traceable signatures. Note that *none* of the existing schemes is secure when the join protocol is executed concurrently. In contrast, group signature scheme with concurrent join has been proposed in [38] and can also be constructed based on group encryption [21].

Schemes	Round-Optimal?	W/o RO?	Support Concurrent-Join?	Secure?
[36]	$\times$	$\times$	$\times$	$\times$
[24]	$\times$	$\times$	$\times$	$\checkmark$
Our Scheme	$\checkmark$	$\times$	$\checkmark$	$\checkmark$

Table 2. Summary of Existing Traceable Signatures

Compact E-Cash Invented by Chaum [22], electronic cash (E-Cash) is the digital counterpart of paper cash. In an e-cash scheme, a user withdraws an electronic coin from the bank and the user can spend it to any merchant, who will deposit the coin back to the bank. Compact e-cash, introduced in [15], aims at improving bandwidth efficiency. In compact e-cash, users can withdraw efficiently a wallet containing K coins. These coins, however, must be spent one by one. Other constructions of compact e-cash include [4, 3, 20]. Table 3 summarizes existing compact e-cash. Note that *none* of the existing schemes is secure when the withdrawal protocol is executed concurrently.

Schemes	Round-Optimal?	W/o RO?	Support Concurrent-Withdrawal?
[15]	$\times$	$\times$	$\times$
[4]	$\times$	$\times$	$\times$
[3]	$\times$	$\times$	$\times$
[20]	$\times$	$\times$	$\times$
Our Scheme	$\checkmark$	$\times$	$\checkmark$

Table 3. Summary of Existing Compact E-Cash Systems

1.2 Overview of Our Approach

As discussed in [38], the most efficient and conceptually simple joining procedure for a group signature is for the user to choose a one way function f and compute $x = f(x')$ for some user secret x'. Next, the user sends x to the group manager (GM) and obtains a signature σ on x. A group signature from the user will then consist of a probabilistic encryption of x into ψ under the GM's public key, and a signature-of-knowledge of (1) the correctness of ψ as an encryption of some value x, (2) knowledge of x', a pre-image of x, and (3) knowledge of σ which is a valid signature on x. This approach is suggested by Camenisch and Stadler [18], and is given the name "single-message and signature-response paradigm" in [38]. Nonetheless, it turns out that a concrete instantiation of this approach is not as simple as it looks, since it is hard to choose a suitable signature scheme and function f so that efficient and secure proof is possible.

It turns out that our argument system together with the Boneh-Boyen signature [10] fits in perfectly with the above paradigm. In our construction, f is chosen to be a perfectly hiding malleable commitment scheme which allows the commitment of a block of values. This expands the flexibility of the paradigm and allows the construction of traceable signatures, compact e-cash as well as blind signature. Taking traceable signature as an example, a user first computes a commitment $f(x)$ of a secret value x. Due to the malleability of the commitment scheme, the group manager changes it to a commitment of a block of values $f(x,t)$ and issues a signature σ on this commitment. To generate a traceable signature, the user computes a probabilistic encryption of $f(x,t)$ into ψ[1], a random base $\tilde{g} = g^r$ and a tracing tag $T = \tilde{g}^t$. Next, the user generates a signature-of-knowledge of (1) the correctness of ψ, $\tilde{g}$ and T with respect to x and t, (2) knowledge of x, t, a pre-image of $f(x,t)$, and (3) knowledge of σ which is a valid signature on $f(x,t)$. To trace the user, the GM simply outputs t and everyone can test whether the tracing tag T and the random base $\tilde{g}$ associated with each group signature satisfies $T = \tilde{g}^t$.

1.3 Organization of The Paper

The rest of this paper is organized as follows. In Section 2, we review preliminaries that will be used throughout this paper. We then present our argument system, its security and efficiency analysis in Section 3. Then, we apply our argument system in constructing blind signatures, traceable signatures and compact e-cash. Those constructions are presented in Section 4, 5 and 6, respectively. Finally, we conclude the paper in Section 7.

2 Preliminaries

2.1 Notations

We employ the following notation throughout this paper. Let $\mathbb{G}_1$ be a cyclic group of prime order p. Let $\mathbb{G}_\mathrm{q} \subset \mathbb{Z}_p^*$ be a cyclic group of prime order q. This can be generated by setting p to be a prime of the form $p = \gamma\mathrm{q} + 1$ for some integer γ and set $\mathbb{G}_\mathrm{q}$ to be the group generated by an element of order q in $\mathbb{Z}_p^*$.

Let $g, g_0, g_1, g_2 \in_R \mathbb{G}_1$ be random elements of $\mathbb{G}_1$ and $\mathrm{h}, \mathrm{h}_0, \mathrm{h}_1, \ldots, \mathrm{h}_L \in_R \mathbb{G}_\mathrm{q}$ be random elements of $\mathbb{G}_\mathrm{q}$ (with the requirement that none of them being the identity element of their respective group). Since $\mathbb{G}_1$ and $\mathbb{G}_\mathrm{q}$ are of prime order, those elements are generators of their respective groups.

We say that a function $\mathsf{negl}(\lambda)$ is a negligible function [5], if for all polynomials $f(\lambda)$, for all sufficiently large λ, $\mathsf{negl}(\lambda) < 1/f(\lambda)$.

2.2 Bilinear Map

A pairing is a bilinear mapping from a pair of group elements to a group element. Specifically, let $\mathbb{G}_T$ be cyclic group of prime order p. A function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be a pairing if it satisfies the following properties:

- (Bilinearity.) $\hat{e}(u^x, v^y) = \hat{e}(u,v)^{xy}$ for all $u, v \in \mathbb{G}_1$ and $x, y \in \mathbb{Z}_p$.
- (Non-Degeneracy.) $\hat{e}(g,g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity element in $\mathbb{G}_T$.
- (Efficient Computability.) $\hat{e}(u,v)$ is efficiently computable for all $u, v \in \mathbb{G}_1$.
- (Unique Representation.) All elements in $\mathbb{G}_1$, $\mathbb{G}_T$ have unique binary representation.

Looking ahead, while we are assuming $\mathbb{G}_1$ is equipped with a bilinear map, it is not necessary for our zero-knowledge proof of knowledge of representation of committed value. Its presence is mainly for the many applications associated with our protocol.

[1] In fact, this is for revealing signer's identity and encryption of either $f(x)$, x or σ also serves the purpose.

2.3 Number-Theoretic Assumptions

We present below the number-theoretic problems related to the schemes presented in this paper. The respective assumptions state that no PPT algorithm has non-negligible advantage in security parameter in solving the corresponding problems. Let $\mathbb{G} = \langle g \rangle = \langle g_1 \rangle = \cdots = \langle g_k \rangle$ be a cyclic group.

- The *Discrete Logarithm Problem* (DLP) in $\mathbb{G}$ is to output x such that $Y = g^x$ on input $Y \in \mathbb{G}$.
- The *Representation Problem* (RP) [13] in $\mathbb{G}$ is to compute a k-tuple $(x_1, \ldots, x_k)$ such that $Y = g_1^{x_1} \cdots g_k^{x_k}$ on input Y. RP is as hard as DLP if the relative discrete logarithm of any of the g_i's are not known.
- The *Decisional Diffie-Hellman Problem* (DDHP) $\in \mathbb{G}$ is to decide if $z = xy$ on input a tuple (g^x, g^y, g^z).
- The *Decisional Linear Diffie-Hellman Problem* (DLDH problem) [11] in $\mathbb{G}$ is to decide if $z = x + y$ on input a tuple (g_1^x, g_2^y, g_3^z). The DLDH problem is strictly harder than the DDH problem.
- The *q-Strong Diffie-Hellman Problem* (q-SDH problem) [10] in $\mathbb{G}$ is to compute a pair (A, e) such that $A^{x+e} = g$ on input $(g^x, g^{x^2}, \ldots, g^{x^q})$.
- The *y-Decisional Diffie-Hellman Inversion Problem* (y-DDHI problem) [28, 15] in $\mathbb{G}$ is to decide if $z = 1/x$ on input $(g^x, g^{x^2}, \ldots, g^{x^y}, g^z)$.

2.4 Cryptographic Tools

Commitment Schemes A commitment scheme is a protocol between two parties, namely, committer Alice and receiver Bob. It consists of two stages: the *Commit* stage and the *Reveal* stage. In the *Commit* stage, Alice receives a value x as input, which is revealed to Bob at the *Reveal* stage. Informally speaking, a commitment scheme is secure if at the end of the *Commit* stage, Bob cannot learn anything about the committed value (a.k.a. hiding) while at the *Reveal* stage, Alice can only reveal one value, that is x (a.k.a. binding). Formally, we review the security notion from [32].

Definition 1. *A commitment scheme* $(\textsf{Gen}, \textsf{Commit})$[2] *is secure if holding the following two properties:*

1. *(Perfect Hiding.) For all algorithm $\mathcal{A}$ (even computationally unbounded one), we require that*

$$Pr\begin{bmatrix} \textsf{param} \leftarrow \textsf{Gen}(1^\lambda); (x_0, x_1) \leftarrow \mathcal{A}(\textsf{param}); & \\ b \in_R \{0,1\}; r \in_R \{0,1\}^\lambda; & : b' = b \\ C = \textsf{Commit}(\textsf{param}, x_b; r); b' \leftarrow \mathcal{A}(C); & \end{bmatrix} \leq \frac{1}{2} + \textsf{negl}(\lambda).$$

2. *(Binding.) No PPT adversary $\mathcal{A}$ can open a commitment in two different ways. Specifically,*

$$Pr\begin{bmatrix} \textsf{param} \leftarrow \textsf{Gen}(1^\lambda); (x_0, x_1, r_0, r_1) \leftarrow \mathcal{A}(\textsf{param}) : \\ x_0 \neq x_1 \wedge \\ \textsf{Commit}(\textsf{param}, x_0; r_0) = \textsf{Commit}(\textsf{param}, x_1; r_1) \end{bmatrix} = \textsf{negl}(\lambda).$$

In this paper, we restrict ourselves to a well-known non-interactive commitment scheme, the Pedersen Commitment [42], which is reviewed very briefly here. On input a value $x \in \mathbb{Z}_p$, the committer randomly chooses $r \in \mathbb{Z}_p$, computes and outputs commitment $C = g_0^x g^r$ as the commitment of value x. To reveal commitment C, the committer outputs (x, r). Everyone can test if $C = g_0^x g^r$. Sometimes (x, r) is referred to as an opening of the commitment C.

[2] With $\textsf{Gen}$ being the parameter generation function.

Recall that Pedersen Commitment is perfect hiding and computationally binding provided that the g_0 and g are randomly and independently generated and that relative discrete logarithm of g_0 to base g is unknown. One can easily extend the scheme to allow commitment of a block of values, say, $\boldsymbol{x} = (x_0, x_1, \ldots, x_k)$ by setting the commitment $C = g_0^{x_0} g_1^{x_1} \cdots g_k^{x_k} g^r$ with additional random generators $g_1, \ldots, g_k$ of $\mathbb{G}_1$.

Boneh-Boyen Short Signature Boneh and Boyen introduced a short signature scheme in [10], which, is used extensively in the applications of our argument system. Hereafter, we shall refer to this scheme as BB-signature.

KeyGen. Let $\alpha, \beta \in_R \mathbb{Z}_p^*$ and $u = g^\alpha$ and $v = g^\beta$. The secret key sk is (α, β) while the public key pk is $(\hat{e}, \mathbb{G}_1, \mathbb{G}_T, p, g, u, v)$.

Sign. Given message $m \in \mathbb{Z}_p^*$, pick a random $e \in_R \mathbb{Z}_p$ and compute $A = g^{\frac{1}{\alpha+m+\beta e}}$. The term $\alpha + m + \beta e$ is computed modulo p. In case it is zero, choose another e. The signature σ on m is (A, e).

Verify. Given a message m and signature $\sigma = (A, r)$, verify that

$$\hat{e}(A, ug^m v^e) = \hat{e}(g, g)$$

If the equality holds, output valid. Otherwise, output invalid.

***Σ*-Protocol** We restrict ourselves to a special class of ZKPoK protocol called Σ-protocol which is defined below. Informally speaking, Σ-protocols only guarantee zero-knowledgeness when the verifier is honest. We are interested in Σ-protocol since they can be transformed to 4-move perfect zero-knowledge ZKPoK protocol [25]. They can also be transformed to 3-move concurrent zero-knowledge protocol in the auxiliary string model using trapdoor commitment schemes [27].

Definition 2. *A Σ-protocol for a binary relation $\mathcal{R}$ is a 3-round ZKPoK protocol between two parties, namely, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. For every input $(w, x) \in \mathcal{R}$ to $\mathcal{P}$ and x to $\mathcal{V}$, the first round of the protocol consists of $\mathcal{P}$ sending a commitment t to $\mathcal{V}$. $\mathcal{V}$ then replies with a challenge c in the second round and $\mathcal{P}$ concludes by sending a response z in the last round. At the end of the protocol, $\mathcal{V}$ outputs accept or reject. We say a protocol transcript (t, c, z) is valid if the output of an honest verifier $\mathcal{V}$ is accept. A Σ-protocol has to satisfy the following two properties:*

- *(Special Soundness.) A cheating prover can at most answer one of the many possible challenges. Specifically, there exists an efficient algorithm KE, called knowledge extractor, that on input x, a pair of valid transcripts (t, c, z) and (t, c', z') with $c \neq c'$, outputs w such that $(w, x) \in \mathcal{R}$.*
- *(Special Honest-Verifier Zero-Knowledgeness(HVZK).) There exists an efficient algorithm KS, called zero-knowledge simulator, that on input x and a challenge c, outputs a pair (t, z) such that (t, c, z) is a valid transcript having the same distribution as a real protocol transcript resulted from the interaction between $\mathcal{P}$ with input $(w, x) \in \mathcal{R}$ and an honest $\mathcal{V}$.*

Signature of Knowledge Any Σ-protocol can be turned into non-interactive form, called signature of knowledge [18], by setting the challenge to the hash value of the commitment together with the message to be signed [29]. Pointcheval and Stern [43] showed that any signature scheme obtained this way is secure in the random oracle model [8].

3 A Zero-Knowledge Proof-of-Knowledge Protocol for RCV

We present the main result of this paper, namely, a zero-knowledge proof-of-knowledge protocol of **R**epresentation of **C**ommitted **V**alue, RCV. Specifically, let $C = g_0^x g_1^r \in \mathbb{G}_1$ be a commitment of x with randomness r. Let $\mathrm{D} = \mathrm{h}_1^{m_1} \cdots \mathrm{h}_L^{m_L} \mathrm{h}^s \in \mathbb{G}_\mathrm{q}$ be the commitment of x's representation (to bases $\mathrm{h}_1, \ldots, \mathrm{h}_L$, denoted as $\boldsymbol{m}$) with randomness $s \in_R \mathbb{Z}_\mathrm{q}$. We construct a ZKPoK protocol of $(x, \boldsymbol{m})$, denoted as $\mathrm{PK}_{\mathsf{RCV}}$. Technically speaking, our protocol is an *argument* system rather than a *proof* system in the sense that soundness in our system only holds against a PPT cheating prover. This is sufficient for all our purposes when adversaries in the applications of our $\mathrm{PK}_{\mathsf{RCV}}$ are modeled as PPT algorithms. $\mathrm{PK}_{\mathsf{RCV}}$ for C, D can be abstracted as follows.

$$\mathrm{PK}_{\mathsf{RCV}}\Big\{ (x, r, s, m_1, \ldots, m_L) : \\ C = g_0^x g^r \;\wedge\; \mathrm{D} = \mathrm{h}_1^{m_1} \cdots \mathrm{h}_L^{m_L} \mathrm{h}^s \;\wedge\; x = \mathrm{h}_1^{m_1} \cdots \mathrm{h}_L^{m_L} \Big\}$$

The construction of $\mathrm{PK}_{\mathsf{RCV}}$ consists of two parts. Note that while we describe them separately, they can be executed in parallel in its actual implementation.

3.1 The Actual Protocol

We construct a Σ-Protocol of $\mathrm{PK}_{\mathsf{RCV}}$. Let λ_k be a security parameter. In practice, we suggest λ_k should be at least 80. The first part of $\mathrm{PK}_{\mathsf{RCV}}$ is a zero-knowledge proof-of-knowledge of representation of an element, and we adapt the protocol from [40].

(Commitment.) The prover randomly generates $\rho_x, \rho_r \in_R \mathbb{Z}_p$, computes and sends $T = g_0^{\rho_x} g^{\rho_r}$ to the verifier.
(Challenge.) The verifier returns a random challenge $c \in_R \{0,1\}^{\lambda_k}$.
(Response.) The prover, treating c as an element in $\mathbb{Z}_p$[3], computes $z_x = \rho_x - cx \in \mathbb{Z}_p$, $z_r = \rho_r - cr \in \mathbb{Z}_p$ and returns (z_x, z_r) to the verifier.
(Verify.) Verifier accepts if and only if $T = C^c g_0^{z_x} g^{z_r}$.

The second part is more involved and can be thought of as the extension of the ZKPoK of double-discrete logarithm in combination with ZKPoK of equality of discrete logarithm.

(Commitment.) For $i = 1$ to λ_k, the prover randomly generates $\rho_{m_1,i}$, …, $\rho_{m_L,i}$, $\rho_{s,i} \in_R \mathbb{Z}_\mathrm{q}$ and $\rho_{r,i} \in_R \mathbb{Z}_p$. Then the prover computes $T_{1,i} = g_0^{\mathrm{h}_1^{\rho_{m_1,i}} \cdots \mathrm{h}_L^{\rho_{m_L,i}}} g^{\rho_{r,i}} \in \mathbb{G}_1$ and $T_{2,i} = \mathrm{h}_1^{\rho_{m_1,i}} \cdots \mathrm{h}_L^{\rho_{m_L,i}} h^{\rho_{s,i}} \in \mathbb{G}_\mathrm{q}$. After that, the prover sends $(T_{1,i}, T_{2,i})_{i=1}^{\lambda_k}$ to the verifier.
(Challenge.) The verifier returns a random challenge $c \in_R \{0,1\}^{\lambda_k}$.
(Response.) Denote $c[i]$ as the i-th bit of c. That is, $c[i] \in \{0,1\}$. For $i = 1$ to λ_k, the prover computes $z_{m_1,i} = \rho_{m_1,i} - c[i]m_1 \in \mathbb{Z}_\mathrm{q}$, …, $z_{m_L,i} = \rho_{m_L,i} - c[i]m_L \in \mathbb{Z}_\mathrm{q}$, $z_{s,i} = \rho_{s,i} - c[i]s \in \mathbb{Z}_\mathrm{q}$ and $z_{r,i} = \rho_{r,i} - c[i]\mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}} r \in \mathbb{Z}_p$. The prover sends $\big(z_{m_1,i}, \ldots, z_{m_L,i}, z_{s,i}, z_{r,i}\big)_{i=1}^{\lambda_k}$ to the verifier.
(Verify.) The verifier accepts if the following equations hold for $i = 1$ to λ_k.

$$T_{2,i} \stackrel{?}{=} \mathrm{D}^{c[i]} \mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}} h^{z_{s,i}}$$
$$T_{1,i} \stackrel{?}{=} g_0^{\mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}}} g^{z_{r,i}} \text{ if } c[i] = 0$$
$$T_{1,i} \stackrel{?}{=} C^{\mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}}} g^{z_{r,i}} \text{ if } c[i] = 1$$

[3] Consequently, the bit-length of p should be longer than λ_k.

The two parts should be executed in parallel using the same challenge. Regarding the security of $\mathrm{PK}_{\mathsf{RCV}}$, we have the following theorem whose proof can be found in Appendix A.

Theorem 1. $\mathrm{PK}_{\mathsf{RCV}}$ *is a Σ-Protocol.*

3.2 Efficiency Analysis of $\mathrm{PK}_{\mathsf{RCV}}$

Table 4 summarizes the time and space complexities of $\mathrm{PK}_{\mathsf{RCV}}$. We breakdown the time complexity of the protocol into the number of multi-exponentiations (multi-EXPs)[4] in various groups. Note that with pre-processing, prover's online computation is minimal and does not involve any exponentiations. As for the bandwidth requirement, the non-interactive version is more space-efficient since the prover does not need to include the commitment using the technique of [1].

In practice, we can take $\lambda_k = 80$ and p (resp. q) to be a 1024-bit (resp. 160-bit) prime. Thus, $\mathbb{Z}_p$, $\mathbb{Z}_\mathrm{q}$ and $\mathbb{G}_1$ will take 1024, 160 and roughly 1024 bit, respectively. The non-interactive form (of which our applications employ) takes up around $(12 + 1.5L)$kB. Looking ahead, L is 1, 3 and 3 in our construction of blind signature, traceable signature and compact e-cash, respectively. The most dominant operation in our applications is the Multi-EXPs in group $\mathbb{G}_1$ since we are using the elliptic curve group equipped with pairing. As a preliminary analysis, we find out that one multi-EXP in $\mathbb{G}_1$ takes about 25ms. The timing is obtained on a Dell GX620 with an Intel Pentium 4 3.0 GHz CPU and 2GB RAM running Windows XP Professional SP2 as the host. We used Sun xVM VirtualBox 2.0.0 to emulate a guest machine of 1GB RAM running Ubuntu 7.04. Our implementation is written in C and relies on the Pairing-Based Cryptography (PBC) library (version 0.4.18). $\mathbb{G}_1$ is taken to be an elliptic curve group equipped with type A1 pairing and the prime p is 1048 bits. In a nutshell, the verifier takes around 2 seconds in verifying the proof $\mathrm{PK}_{\mathsf{RCV}}$.

Time Complexities			
	Prover w/o Preproc.	Prover w/ Preproc.	Verifier
$\mathbb{G}_1$ multi-EXP	$\lambda_k + 1$	0	$\lambda_k + 1$
$\mathbb{G}_\mathrm{q}$ multi-EXP	$\lambda_k(\lceil L/3 \rceil + 1) + 1$	0	$\lambda_k(\lceil L/3 \rceil + 2)$

Bandwidth Requirement		
	Interactive Form	Non-Interactive Form
$\mathbb{G}_1$	$2\lambda_k + 1$	0
$\mathbb{Z}_p$	$\lambda_k + 2$	$\lambda_k + 2$
$\mathbb{Z}_\mathrm{q}$	$\lambda_k(L+1)$	$\lambda_k(L+1)$

Table 4. Time and Space Complexities of $\mathrm{PK}_{\mathsf{RCV}}$.

4 Application to Round-Optimal Concurrently-Secure Blind Signature without Interactive Assumptions

4.1 Syntax

We review the definition of blind signature from Hazay *et al.* [34].

Definition 3. *A blind signature scheme is a tuple of PPT algorithms* BGen*,* BVer *and an interactive protocol* BSign *between a user and a signer such that:*

[4] A multi-EXP computes the product of exponentiations faster than performing the exponentiations separately. Normally, a multi-based exponentiation takes only 10% more time compared with a single-based exponentiation. We assume that one multi-EXP operation multiplies up to 3 exponentiations.

– *BGen: On input security parameter 1^λ, this algorithm outputs a key pair (pk, sk).*
– *BSign: Signer, with private input sk interacts with a user having input pk and a message m in the protocol. At the end of the execution, user obtains a signature σ_m on the message m, assuming neither party abort.*
– *BVer: On input pk, m, σ_m, outputs valid or invalid.*

As usual, correctness requires that for all (pk, sk) output by BGen(1^λ), and for all σ_m which is the output of the user upon successful completion of the protocol run of BSign with appropriate inputs $((pk, m)$ and sk for user and signer respectively) to both parties, BVer with input pk, m, σ_m outputs valid.

Definition 4. *Blind signature scheme (BGen, BSign, BVer) is unforgeable if the winning probability for any PPT adversary $\mathcal{A}$ in the following game is negligible:*

– *BGen outputs (pk, sk) and pk is given to $\mathcal{A}$.*
– *$\mathcal{A}$ interact concurrently with ℓ signer clones with input sk in BSign protocol.*
– *$\mathcal{A}$ outputs $\ell + 1$ signatures σ_i on $\ell + 1$ distinct messages m_i.*

$\mathcal{A}$ wins the game if all m_i are distinct and BVer$(pk, m_i, \sigma_i) = 1$ for all $i = 1$ to $\ell + 1$.

Definition 5. *Blind signature scheme (BGen, BSign, BVer) satisfies blindness if the advantage for any PPT adversary $\mathcal{A}$ in the following game is negligible:*

– *$\mathcal{A}$ outputs an arbitrary public key pk and two equal-length messages m_0, m_1.*
– *A random bit $b \in_R \{0, 1\}$ is chosen, and $\mathcal{A}$ interacts concurrently with two user clones, say U_0 and U_1, with input (pk, m_b) and (pk, m_{1-b}) respectively. Upon completion of both protocols, define σ_0 and σ_1 as follows:*
  - *If either of the U_0 or U_1 aborts, set $(\sigma_0, \sigma_1) = (\perp, \perp)$.*
  - *Otherwise, define σ_i be the output of U_i for $i = 0$ and 1.*

  (σ_0, σ_1) are given to $\mathcal{A}$.
– *$\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$.*

$\mathcal{A}$ wins the game if all $b' = b$. The advantage of $\mathcal{A}$ is defined as $|Pr[b' = b] - 1/2|$.

4.2 Construction

BGen. Let $\alpha, \beta \in_R \mathbb{Z}_p$ and $u = g^\alpha$ and $v = g^\beta$. Let $H : \{0, 1\}^* \to \mathbb{Z}_{\mathrm{q}}$ be a collision-resistant hash function. The signer's secret key sk is (α, β) while its public key pk is $(\mathbb{G}_1, \mathbb{G}_T, \hat{e}, \mathbb{G}_{\mathrm{q}}, p, \mathrm{q}, g, u, v, \mathrm{h}, \mathrm{h}_0, \mathrm{h}_1, H)$.
BSign. On input message $m \in \mathbb{Z}_{\mathrm{q}}$, the user computes $\mathrm{x} = \mathrm{h}_0^m \mathrm{h}^s$ for some randomly generated $s \in_R \mathbb{Z}_{\mathrm{q}}$. The user sends x to the signer. The signer selects $e \in_R \mathbb{Z}_p$ and computes $A = g^{\frac{1}{\alpha + \mathrm{x} + \beta e}}$. The signer returns (A, e) to the user.

The user computes Π_m as an non-interactive zero-knowledge proof-of-knowledge of a BB signature (A, e) on a hidden value x, and that x is a commitment of m and output Π_m as the signature of m.

Specifically, denote $\mathrm{y} = \mathrm{h}^s$. The user computes $\mathfrak{A}_1 = A g_2^{r_1}$, $\mathfrak{A}_2 = g_1^{r_1} g_2^{r_2}$, $\mathfrak{A}_3 = g_1^{\mathrm{y}} g_2^{r_3}$ for some randomly generated $r_1, r_2, r_3 \in_R \mathbb{Z}_p$ and $\mathrm{A}_4 = \mathrm{h}_0^s \mathrm{h}^t$ for some randomly generated $t \in_R \mathbb{Z}_{\mathrm{q}}$. Parse $M = \mathfrak{A}_1 || \mathfrak{A}_2 || \mathfrak{A}_3 || \mathrm{A}_4$. The user computes the following non-interactive zero-knowledge proof-of-knowledge Π_m comprising two parts, namely, SPK_1 and SPK_2. SPK_1 can be computed using standard techniques, while SPK_2 is computed using our newly constructed $\mathrm{PK}_{\mathsf{RCV}}$. Finally, parse Π_m as $(\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathrm{A}_4, \mathrm{SPK}_1, \mathrm{SPK}_2)$.

$$\Pi_m : \begin{cases} \mathrm{SPK}_1\Big\{(r_1, r_2, r_3, \mathrm{y}, e, \beta_1, \beta_2, \beta_3, \beta_4) : \\ \qquad \mathfrak{A}_2 = g_1^{r_1} g_2^{r_2} \ \wedge\ 1 = \mathfrak{A}_2^{-e} g_1^{\beta_1} g_2^{\beta_2} \ \wedge \\ \qquad 1 = \mathfrak{A}_2^{-\mathrm{y}} g_1^{\beta_3} g_2^{\beta_4} \ \wedge\ \mathfrak{A}_3 = g_1^{\mathrm{y}} g_2^{r_3} \ \wedge\ \frac{\hat{e}(\mathfrak{A}_1, u)}{\hat{e}(g,g)} = \\ \qquad \hat{e}(g_2, u)^{r_1} \hat{e}(\mathfrak{A}_1, v)^{-e} \hat{e}(g_2, v)^{\beta_1} \hat{e}(\mathfrak{A}_1, g^{\mathrm{h}_0^m})^{-\mathrm{y}} \hat{e}(g_2, g^{\mathrm{h}_0^m})^{\beta_3}\Big\}(M) \\ \mathrm{SPK}_2\Big\{ (r_3, \mathrm{y}, s, t) : \mathfrak{A}_3 = g_1^{\mathrm{y}} g_2^{r_3} \ \wedge \mathrm{A}_4 = \mathrm{h}_0^s \mathrm{h}^t \ \wedge\ \mathrm{y} = \mathrm{h}_0^s\Big\}(M) \end{cases}$$

BVer. On input message m and its signature Π_m, parse Π_m as $(\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathrm{A}_4, \mathrm{SPK}_1, \mathrm{SPK}_2)$ and verify that SPK_1 and SPK_2 are valid.

Regarding the security of our construction, we have the following theorems whose proofs can be found in Appendix B.

Theorem 2. *Our blind signature is unforgeable under the q-SDH assumption in $\mathbb{G}_1$ and DL assumption in $\mathbb{G}_\mathrm{q}$ in the random oracle model.*

Theorem 3. *Our blind signature satisfies blindness unconditionally in the random oracle model.*

5 Application to Traceable Signatures with Concurrent Join

We describe the construction of our traceable signatures. Since traceable signatures are group signatures with added functionalities, it is easy to modify our scheme into a 'regular' group signature. An attack to the traceable signature due to [36] is given in Appendix C.

5.1 Syntax

We review briefly the definition of traceable signature from Choi *et al.* [24] which is an adaptation of the definition of traceable identification from Kiayias *et al.* [36]. Note that Traceable identifications can be turned into traceable signatures using the Fiat-Shamir Heuristics [29].

Definition 6. *A traceable signature scheme is a tuple of nine PPT algorithms / protocols (GGen, Join, GSign, GVer, Open, Trace, Claim, $\mathsf{ClaimVer}$) between three entities, namely group manager (GM), users and tracing agents:*

- GGen: *On input security parameter 1^λ, this algorithm outputs a key pair (pk, sk) for the group manager.*
- Join: *This is a protocol between a user and GM. Upon successful completion of the protocol, user U_i obtains a membership certificate cert_i. The GM stores the whole protocol transcript Jtrans_i.*
- GSign: *User U_i with membership certificate cert_i signs a message m and produces a group signature σ_m.*
- GVer: *On input pk, m, σ_m, outputs valid or $\mathsf{invalid}$.*
- Open: *On input m, σ_m, the group manager outputs the identity of the signer.*
- Reveal: *On input Jtrans_i, the group manager outputs tracing information tr_i, which is the tracing trapdoor that allows party to identity signatures generated by user U_i.*
- Trace: *On input a signature σ and a tracing information tr_i, output $0/1$ indicating the signature is generated by user U_i or not.*

- *Claim: On input a signature σ and a membership certificate cert_i, user U_i produces a proof τ to prove that he is the originator of the signature.*
- *ClaimVer" On input a signature σ, a proof τ, output $0/1$ indicating the signature is generated by claimer or not.*

Security Requirements. We informally review the security notion of a traceable signature. Due to page limitation, please refer to [36, 24] for formal definition. A traceable signature should be secure against three types of attack.

(Misidentification.) The adversary is allowed to observe the operation of the system while users are engaged with GM during the joining protocol. It is also allowed to obtain a signature from existing users on any messages of its choice. They are also allowed to introduce users into the system. The adversary's goal is to produce a valid signature on new message that is not open to users controlled by the adversary.

(Anonymity.) The adversary is allowed to observe the operation of the system while users are engaged with GM during the joining protocol. It is also allowed to obtain signature from existing users on any messages of its choice. They are also allowed to introduce users into the system. Finally, the adversary chooses a message and two target users he does not control, and then receives a signature of the message he returned from one of these two target users. The adversary's goal is to guess which of the two target users produced the signature.

(Framing.) The adversary plays the role of a malicious GM. It is considered successful with the following scenarios. Firstly, the adversary may construct a signature that opens to an honest user. Secondly, it may construct a signature, output some tracing information and that when traced, this maliciously-constructed signature will be traced to be from an honest user. Thirdly, it may claim a signature that was generated by an honest user as its own.

5.2 Construction

GGen. Let $\alpha, \beta \in_R \mathbb{Z}_p^*$ and $u = g^\alpha$ and $v = g^\beta$. $H : \{0,1\}^* \to \mathbb{Z}_\mathrm{q}$ be a collision-resistant hash function. Further, randomly generate $\gamma_1, \gamma_2 \in_R \mathbb{Z}_p$, $w_3 \in_R \mathbb{G}_1$ and compute $w_1 = w_3^{\frac{1}{\gamma_1}}$ and $w_2 = w_3^{\frac{1}{\gamma_2}}$. GM's secret key sk is $(\alpha, \beta, \gamma_1, \gamma_2)$ while its public key pk is $(\hat{e}$, $\mathbb{G}_1$, $\mathbb{G}_T$, $\mathbb{G}_\mathrm{q}$, p, q, g, u, v, w_1, w_2, w_3, h, h_0, $\mathrm{h}_1, \ldots, \mathrm{h}_4$, $H)$.

Join. A user U_i randomly selects $s, x \in_R \mathbb{Z}_\mathrm{q}$ and sends $\mathrm{C}' = \mathrm{h}_0^s \mathrm{h}_1^x \in \mathbb{G}_\mathrm{q}$ to GM. GM computes $t = H(\mathrm{C}') \in \mathbb{Z}_\mathrm{q}$. It then computes $\mathrm{C} = \mathrm{C}'\mathrm{h}_2^t$ and selects $e \in_R \mathbb{Z}_p$. Next, it computes $A = g^{\frac{1}{\alpha + \mathrm{C} + \beta e}}$. The GM returns (A, e, t) to the user. User checks if $\hat{e}(A, uv^e g^{\mathrm{h}_0^s \mathrm{h}_1^x \mathrm{h}_2^t}) = \hat{e}(g, g)$ and $t = H(\mathrm{C}')$. He then stores (A, e, s, t, x) as his membership certificate cert_i. GM records t as the tracing information tr_i for this user. GM also stores the whole communication transcript.

GSign. Let the user membership certificate be (A, e, s, t, x). The user computes $S = \mathrm{h}_3^k$, $U = \mathrm{h}_3^{k'}$ for some randomly generated $k, k', k'' \in_R \mathbb{Z}_\mathrm{q}$ and $T_1 = S^t$, $T_2 = S^{k''}$, $T_3 = \mathrm{h}_0^s \mathrm{h}_1^x T_1^{k''}$, $V = U^x$. Denote $\mathrm{y} = \mathrm{h}_0^s \mathrm{h}_1^x \mathrm{h}_2^t$. The user then randomly generates $r_1, r_2, r_3 \in_R \mathbb{Z}_p$, computes $\mathfrak{A}_1 = A w_3^{r_1 + r_2}$, $\mathfrak{A}_2 = w_1^{r_1}$, $\mathfrak{A}_3 = w_2^{r_2}$, $\mathfrak{A}_4 = g_1^{\mathrm{y}} g_2^{r_3}$ and $\mathrm{A}_5 = \mathrm{h}^r \mathrm{h}_0^s \mathrm{h}_1^x \mathrm{h}_2^t$ for some randomly generated $r \in_R \mathbb{Z}_\mathrm{q}$. To generate a traceable signature for message m, parse $M = m||S||U||T_1||T_2||T_3||V||\mathfrak{A}_1||\mathfrak{A}_2||\mathfrak{A}_3||\mathfrak{A}_4||\mathrm{A}_5$.

The user computes the following non-interactive zero-knowledge proof-of-knowledge Π_{grp} comprising two parts, namely, SPK_3 and SPK_4. SPK_3 can be computed using standard techniques, while SPK_4 is computed using $\mathrm{PK}_{\mathsf{RCV}}$. Finally, parse Π_{grp} as $(\mathfrak{A}_1$, $\mathfrak{A}_2$, $\mathfrak{A}_3$, $\mathfrak{A}_4$, A_5, SPK_3, $\mathrm{SPK}_4)$ and the signature σ_m as $(\Pi_{\mathsf{grp}}, S, T_1, T_2, T_3, U, V)$.

$$
\mathit{\Pi}_{\mathsf{grp}} : \begin{cases}
\mathrm{SPK}_3\Big\{(r_1, r_2, r_3, \mathrm{y}, e, \beta_1, \beta_2, \beta_3, \beta_4, r, s, t, x, k, k', k'') : \\
\qquad \mathfrak{A}_2 = w_1^{r_1} \ \wedge\ 1 = \mathfrak{A}_2^{-e} w_1^{\beta_1} \ \wedge\ 1 = \mathfrak{A}_2^{-\mathrm{y}} w_1^{\beta_2} \wedge \\
\qquad \mathfrak{A}_3 = w_2^{r_2} \ \wedge\ 1 = \mathfrak{A}_3^{-e} w_1^{\beta_3} \ \wedge\ 1 = \mathfrak{A}_3^{-\mathrm{y}} w_1^{\beta_4} \wedge \\
\qquad \mathfrak{A}_4 = g_1^{\mathrm{y}} g_2^{r_3} \ \wedge\ \mathrm{A}_5 = \mathrm{h}^r \mathrm{h}_0^s \mathrm{h}_1^x \mathrm{h}_2^t \wedge \\
\qquad S = \mathrm{h}_3^k \ \wedge\ T_1 = S^t \ \wedge\ T_2 = S^{k''} \ \wedge\ T_3 = \mathrm{h}_0^s \mathrm{h}_1^x T_1^{k''} \wedge \\
\qquad U = \mathrm{h}_3^{k'} \ \wedge\ V = U^x \ \wedge\ \frac{\hat{e}(\mathfrak{A}_1, u)}{\hat{e}(g, g)} = \\
\qquad \hat{e}(w_3, u)^{r_1 + r_2} \hat{e}(\mathfrak{A}_1, v)^{-e} \hat{e}(w_2, v)^{\beta_1 + \beta_3} \hat{e}(\mathfrak{A}_1, g)^{-\mathrm{y}} \hat{e}(w_3, g)^{\beta_2 + \beta_4}\Big\}(M) \\
\mathrm{SPK}_4\Big\{ (r_3, \mathrm{y}, r, s, t, x) : \\
\qquad \mathfrak{A}_4 = g_1^{\mathrm{y}} g_2^{r_3} \ \wedge \mathrm{A}_5 = \mathrm{h}^r \mathrm{h}_0^s \mathrm{h}_1^x \mathrm{h}_2^t \ \wedge\ \mathrm{y} = \mathrm{h}_0^s \mathrm{h}_1^x \mathrm{h}_2^t\Big\}(M)
\end{cases}
$$

Basically, $\mathfrak{A}_1$, $\mathfrak{A}_2$ and $\mathfrak{A}_3$ is the linear encryption of A (part of the membership certificate), T_1, T_2, T_3 is the El-Gamal encryption of $\mathrm{h}_0^s \mathrm{h}_1^x$ (under the public key S^t), while the rest of the proof is to assure the verifier that the encryptions are properly done and that values U, V, S are correctly formed with respective to values s, t, x, r.

Open. On input a signature σ_m, GM computes $A := \frac{\mathfrak{A}_1}{\mathfrak{A}_2^{\gamma_1} \mathfrak{A}_3^{\gamma_2}}$. From A, GM looks up its list of join transcripts and identify the underlying user.

Reveal. To allow tracing of user U_i, the GM outputs tracing information tr_i.

Trace. Given a valid signature $\sigma_m = (\mathit{\Pi}_{\mathsf{grp}}, S, T_1, T_2, T_3, U, V)$ and tracing information tr_i, everyone can test if the signature is from user U_i by testing $T_1 \stackrel{?}{=} S^{\mathsf{tr}_i}$ and $\mathsf{tr}_i \stackrel{?}{=} H(\frac{T_3}{T_2^{\mathsf{tr}_i}})$.

Claim. On input a message $\sigma_m = (\mathit{\Pi}_{\mathsf{grp}}, S, T_1, T_2, T_3, U, V)$, the originator can produce an non-interactive proof τ as

$$\tau : \mathrm{SPK}_\tau\{(x) : V = U^x\}(\sigma_m)$$

ClaimVer. Given a signature σ_m and τ, everyone can verify τ.

Regarding the security of traceable signature, we have the following theorem whose proof can be found in Appendix B.

Theorem 4. *Our traceable signature is secure under the q-SDH assumption, the DLDH assumption in $\mathbb{G}_1$ and DL assumption in $\mathbb{G}_{\mathrm{q}}$ in the random oracle model.*

6 Compact E-Cash With Concurrent Withdrawal

Our technique can also be applied to construct compact e-cash systems with concurrently-secure withdrawal protocol. A high-level description is given here. In compact e-cash, there are three entities, namely, the bank, users and merchants. To withdraw a wallet of K coins, user obtains a BB signature cert on commitment of values (s, t, x), in a similar manner as user obtains a membership certificate in our construction of traceable signatures. Note that the major difference being in this case, none of the values are known to the bank (with s being a random number jointly generated by the bank and user).

To spend a electronic coin to a merchant, user computes a serial number $S = \mathrm{h}_3^{\frac{1}{s+J+1}}$, a tracing tag $T = \mathrm{h}_0^s \mathrm{h}_1^t \mathrm{h}_2^x \mathrm{h}_3^{\frac{R}{t+J+1}}$, where J is the counter of the number of times the user has spent his wallet and R is a random challenge issued by the merchant. User sends the pair (S, T) to the merchant, along with a signature of knowledge $\mathit{\Pi}_{\$}$, stating that S and T are correctly formed. Specifically, the proof assures the merchant that (1)user is in possession of a valid BB signature from the bank on values (s, t, x); (2)counter $0 \leq J < K$; (3)S and T are correctly formed with respect to (s, t, x).

In the deposit protocol, merchant sends the coin $(\Pi_\$, S, T, R)$ to the bank. Since counter J runs from 0 to $K-1$, user can at most spend his wallet for K times. If the user uses the counter for a second time, the serial number S of the double-spent coins will be the same and will thus be identified. Next, the bank can compute a value $\mathrm{C} := (\frac{T^{R'}}{T'^{R}})^{1/(R'-R)}$, the commitment of (s, t, x) which allows the bank to identify the underlying double-spender.

6.1 Syntax and Construction

EBGen. *This is the key generation algorithm for the bank, which includes the wallet size K of the system.*
Let $\alpha, \beta \in_R \mathbb{Z}_p^*$ and $u = g^\alpha$ and $v = g^\beta$. Let K be the size of the wallet and $H : \{0,1\}^* \to \mathbb{Z}_\mathrm{q}$ be a collision-resistant hash function. The bank's secret key bsk is (α, β) while its public key bpk is ($\hat{e}$, $\mathbb{G}_1$, $\mathbb{G}_T$, $\mathbb{G}_\mathrm{q}$, p, q, g, u, v, h, h_0, h_1, …, h_4, K, H).
Withdrawal. *User withdraws a wallet of K coins from the bank in this protocol.*
A user U_i randomly selects $s', t, x \in_R \mathbb{Z}_\mathrm{q}$ and sends $\mathrm{C}' = \mathrm{h}_0^{s'}\mathrm{h}_1^t\mathrm{h}_2^x \in \mathbb{G}_\mathrm{q}$ to the bank. The bank computes randomly selects $s'' \in_R \mathbb{Z}_\mathrm{q}$, computes $\mathrm{C} = \mathrm{C}'\mathrm{h}_0^{s''}$ and selects $e \in_R \mathbb{Z}_p$. It then computes $A = g^{\frac{1}{\alpha+\mathrm{C}+\beta e}}$. The bank returns (A, e, s'') to the user. User computes $s = s' + s'' \bmod \mathbb{Z}_\mathrm{q}$, checks if $\hat{e}(A, uv^e g^{\mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x}) = \hat{e}(g, g)$. He then stores $(A,\ e,\ s,\ t,\ x,\ J)$, where J is a counter initialized to 0, as his wallet. The bank records C as the identifer for this user. The bank also stores the whole communication transcript.
Spend. *User spends a electronic coin to a merchant in this protocol.*
Let the user wallet be $(A\ ,e, s, t, x, J)$ such that $J < K$. The user engages with merchant with identity I and they first agree on the transaction information info. Both parties compute $R = H(\mathrm{info}, I)$ locally.

Next, the user computes $S = \mathrm{h}_3^{\frac{1}{s+J+1}}$ and $T = \mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x\mathrm{h}_3^{\frac{R}{t+J+1}}$. S is the unique serial number associated with the electronic coin. Denote $\mathrm{y} = \mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x$. The user randomly generates $r_1, r_2, r_3 \in_R \mathbb{Z}_p$, computes $\mathfrak{A}_1 = A g_3^{r_1}$, $\mathfrak{A}_2 = g_1^{r_1}g_2^{r_2}$, $\mathfrak{A}_3 = g_1^{\mathrm{y}}g_2^{r_3}$ and $\mathrm{A}_4 = \mathrm{h}^r\mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x$ for some randomly generated $r \in_R \mathbb{Z}_\mathrm{q}$. Parse $M = R||S||T||\mathfrak{A}_1||\mathfrak{A}_2||\mathfrak{A}_3||\mathrm{A}_4$.

The user computes the following non-interactive zero-knowledge proof-of-knowledge $\Pi_\$$ comprising two parts, namely, SPK_5 and SPK_6. SPK_5 can be computed using standard techniques, while SPK_6 is computed using $\mathrm{PK}_{\mathsf{RCV}}$. Finally, parse $\Pi_\$$ as $(\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathrm{A}_4, \mathrm{SPK}_5, \mathrm{SPK}_6)$ and the electronic coin $\sigma_\$$ is $(\Pi_\$, S, T, R)$. The user increases counter J by 1.

$$\Pi_\$: \begin{cases} \mathrm{SPK}_5\Big\{(r_1, r_2, r_3, \mathrm{y}, e, \beta_1, \beta_2, \beta_3, \beta_4) : \\ \quad \mathfrak{A}_2 = g_1^{r_1}g_2^{r_2} \ \wedge\ 1 = \mathfrak{A}_2^{-e}g_1^{\beta_1}g_2^{\beta_2} \ \wedge\ 1 = \mathfrak{A}_2^{-\mathrm{y}}g_1^{\beta_3}g_2^{\beta_4} \ \wedge \\ \quad \mathfrak{A}_3 = g_1^{\mathrm{y}}g_2^{r_3} \ \wedge\ \mathrm{A}_4 = \mathrm{h}^r\mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x \ \wedge\ \mathrm{A}_4 = \mathrm{A}_4^{-t}\mathrm{A}_4^{-J}\mathrm{h}^{\beta_5}\mathrm{h}_0^{\beta_6}\mathrm{h}_1^{\beta_7}\mathrm{h}_2^{\beta_8} \ \wedge \\ \quad \frac{S}{\mathrm{h}_3} = S^sS^J \ \wedge\ \frac{T}{\mathrm{h}_3^R} = T^{-t}T^{-J}\mathrm{h}_0^{\beta 6}\mathrm{h}_1^{\beta_7}\mathrm{h}_2^{\beta_8} \ \wedge\ 0 \leq J < K \ \wedge \\ \quad \frac{\hat{e}(\mathfrak{A}_1,u)}{\hat{e}(g,g)} = \hat{e}(g_2,u)^{r_1}\hat{e}(\mathfrak{A}_1,v)^{-e}\hat{e}(g_2,v)^{\beta_1}\hat{e}(\mathfrak{A}_1,g)^{-\mathrm{y}}\hat{e}(g_2,g)^{\beta_3}\Big\}(M) \\ \mathrm{SPK}_6\Big\{(r_3, \mathrm{y}, r, s, t, x) : \mathfrak{A}_3 = g_1^{\mathrm{y}}g_2^{r_3} \ \wedge \mathrm{A}_4 = \mathrm{h}^r\mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x \ \wedge\ \mathrm{y} = \mathrm{h}_0^s\mathrm{h}_1^t\mathrm{h}_2^x\Big\}(M) \end{cases}$$

The user sends $\sigma_\$$ to the merchant, who accepts the coin if $\Pi_\$$ is valid.
Deposit. *Merchant deposit a electronic coin to the bank in this protocol.*
To deposit, the merchant simply sends $\sigma_\$$, along with info to the bank. The bank verifies the transcript

exactly as the merchant did. In addition, the bank has to verify that I is indeed the identity of the merchant and $R = H(\text{info}, I)$ is fresh. This is to prevent colluding users and merchants from submitting a double spent coin (which have identical transcripts). It also prevents a malicious merchant from eavesdropping an honest transaction and depositing it (in that case, identity of the malicious merchant does not match with I). The bank stores $(\sigma_\$, R)$ to the database.

Revoke. *The bank employs this algorithm to reveal the identity of the double-spender.*
When a new coin $\sigma_\$ = (\Pi_\$, S, T, R)$ is received, the bank checks if S exists in the database. If yes, then it is a double-spent coin. The bank identifies the double-spender as follows. Let the entry in the database be $(\Pi_\$, S, T', R')$. The bank computes $\mathrm{C} := (\frac{T^{R'}}{T'^{R}})^{1/(R'-R)}$ and identity the user.

6.2 Security Requirements.

We informally review the security requirements of a compact e-cash. Please refer to [15] for the formal treatment of the subject.

(Balance.) It is required that no collusion of users and merchants together can deposit more than they withdraw without being identified. More precisely, we require that collusion of users and merchants, having run Withdrawal for n times, cannot deposit more than nk coins back to the bank without being identified.

(Anonymity.) It is required that no collusion of users, merchants and the bank can ever learn the spending habit of an honest user.

(Exculpability.) It is required that an honest user cannot be proven to have double-spent, even all other users, merchants and the bank collude.

Regarding the security of our compact E-Cash system, we have the following theorem. See Appendix B for a brief discussion.

Theorem 5. *Our compact e-cash scheme is secure under the q-SDH assumption in $\mathbb{G}_1$ and y-DDHI assumption in $\mathbb{G}_\mathrm{q}$ in the random oracle model.*

7 Conclusion

We constructed a new zero-knowledge argument system and illustrated its significance with applications to blind signatures, traceable signatures and compact e-cash systems. We believe this system is useful in other cryptographic applications.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO'00*, volume 1880 of *LNCS*, pages 255–270, 2000.
2. G. Ateniese, D. X. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Financial Cryptography*, pages 183–197, 2002.
3. M. H. Au, W. Susilo, and Y. Mu. Practical compact e-cash. In *ACISP'07*, volume 4586 of *LNCS*, pages 431–445, 2007.
4. M. H. Au, Q. Wu, W. Susilo, and Y. Mu. Compact e-cash from bounded accumulator. In *CTRSA'07*, volume 4377 of *LNCS*, 2007.
5. M. Bellare. A note on negligible functions. *J. Cryptology*, 15(4):271–284, 2002.
6. M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420, 1993.
7. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. In *Financial Cryptography'01*, volume 2339 of *LNCS*, pages 309–328, 2001.

8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM-CCS'93*, pages 62–73, 1993.
9. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC'03*, volume 2567 of *LNCS*, pages 31–46, 2003.
10. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 56–73, 2004.
11. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, volume 3152 of *LNCS*, pages 41–55, 2004.
12. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 431–444, 2000.
13. S. Brands. An Efficient Off-Line Electronic Cash System Based on the Representation Problem. Technical report, CWI, 1993.
14. J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. *PhD Thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.*, 1998.
15. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 302–321, 2005.
16. J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN'02*, volume 2576 of *LNCS*, pages 268–289, 2002.
17. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO'03*, volume 2729 of *LNCS*, pages 126–144, 2003.
18. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424, 1997.
19. S. Canard and A. Gouget. Divisible e-cash systems can be truly anonymous. In *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 482–497, 2007.
20. S. Canard, A. Gouget, and E. Hufschmitt. Handy Compact E-cash System. *Proceedings of the 2nd Conference on Security in Network Architectures and Information Systems - SAR-SSI'07*, 2007.
21. J. Cathalo, B. Libert, and M. Yung. Group encryption: Non-interactive realization in the standard model. In *ASIACRYPT*, pages 179–196, 2009.
22. D. Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum, New York, 1983.
23. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265, 1991.
24. S. G. Choi, K. Park, and M. Yung. Short traceable signatures based on bilinear pairings. In *IWSEC'06*, volume 4266 of *LNCS*, pages 88–103, 2006.
25. R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC'00*, volume 1751 of *LNCS*, pages 354–373, 2000.
26. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187, 1994.
27. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 418–430, 2000.
28. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC'05*, volume 3386 of *LNCS*, pages 416 – 431, 2005.
29. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194, 1986.
30. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO'06*, volume 4117 of *LNCS*, pages 60–77, 2006.
31. G. Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009.
32. S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
33. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In *STOC 1985*, pages 291–304, 1985.
34. C. Hazay, J. Katz, C.-Y. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *TCC'07*, volume 4392 of *LNCS*, pages 323–341, 2007.
35. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164, 1997.
36. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 571–589, 2004.
37. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. Cryptology ePrint Archive, Report 2004/007, 2004. http://eprint.iacr.org/.

38. A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 198–214, 2005.
39. Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC 2003*, pages 683–692, 2003.
40. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO'92*, pages 31–53, 1992.
41. T. Okamoto. Efficient blind and partially blind signatures without random oracles. Cryptology ePrint Archive, Report 2006/102, 2006. http://eprint.iacr.org/.
42. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140, 1992.
43. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398, 1996.
44. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.
45. C.-P. Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptology*, 4(3):161–174, 1991.
46. M. Stadler. Cryptographic protocols for revocable privacy. *PhD Thesis, ETH Zürich, 1996. Diss. ETH No. 11651*, 1996.

A Proof of Theorem 1

We show that $\mathrm{PK}_{\mathsf{RCV}}$ is a Σ-protocol by constructing a knowledge extractor and transcript simulator.

Soundness of $\mathbf{PK}_{\mathsf{RCV}}$ We construct a knowledge extractor KE for $\mathrm{PK}_{\mathsf{RCV}}$. On input two transcripts $\big(T,\, (T_{1,i},\, T_{2,i})_{i=1}^{\lambda_k},\, c,\, z_x,\, z_r,\, (z_{m_1,i},\, \ldots,\, z_{m_L,i},\, z_{r,i},\, z_{s,i})_{i=1}^{\lambda_k}\big)$ and $\big(T,\, (T_{1,i},\, T_{2,i})_{i=1}^{\lambda_k},\, \hat{c},\, \hat{z}_x,\, \hat{z}_r,\, (\hat{z}_{m_1,i}, \ldots, \hat{z}_{m_L,i}, \hat{z}_{r,i}, \hat{z}_{s,i})_{i=1}^{\lambda_k}\big)$, KE is constructed as follows.

Since $T = C^c g_0^{z_x} g^{z_r}$ and $T = C^{\hat{c}} g_0^{\hat{z}_x} g^{\hat{z}_r}$, we have $C^{\hat{c}-c} = g_0^{z_x - \hat{z}_x} g^{z_r - \hat{z}_r}$. Denote δ_c as $c - \hat{c}$, $\delta_x = z_x - \hat{z}_x$ and $\delta_r = z_r - \hat{z}_r$. The simulator obtains a relation $C = g_0^{\tilde{x}} g^{\tilde{r}}$, where $\tilde{x} = -\delta_x/\delta_c$ and $\tilde{r} = -\delta_r/\delta_c$.

On the other hand, as $c \neq \hat{c}$, there exists a position i such that $c[i] \neq \hat{c}[i]$. Without the loss of generality, assume $c[i] = 0$ and $\hat{c}[i] = 1$. We have $T_{2,i} = \mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}} h^{z_{s,i}}$ and $T_{2,i} = \mathrm{D}\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}} \mathrm{h}^{\hat{z}_{s,i}}$. Thus, $\mathrm{D} = \mathrm{h}_1^{\delta_{m_1}} \cdots \mathrm{h}_L^{\delta_{m_L}} \mathrm{h}^{\delta_s}$, where $\delta_{m_1} = z_{m_1,i} - \hat{z}_{m_1,i}, \ldots\ \delta_{m_L} = z_{m_L,i} - \hat{z}_{m_L,i}$ and $\delta_s = z_{s,i} - \hat{z}_{s,i}$.

We also have $T_{1,i} = g_0^{\mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}}} g^{z_{r,i}}$ and $T_{1,i} = C^{\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}}} g^{\hat{z}_{r,i}}$. Substituting $C = g_0^{\tilde{x}} g^{\tilde{r}}$ into the equation, we have $g_0^{\mathrm{h}_1^{z_{m_1,i}} \cdots \mathrm{h}_L^{z_{m_L,i}}} g^{z_{r,i}} = g_0^{\tilde{x}\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}}} g^{\tilde{r}\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}} + \hat{z}_{r,i}}$. That is, $g_0^{\tilde{x}} g^{\tilde{r}\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}}} = g_0^{\mathrm{h}_1^{\delta_{m_1}} \cdots \mathrm{h}_L^{\delta_{m_L}}} g^{\delta_{r,i}}$, where $\delta_{r,i}$ is defined as $z_{r,i} - \hat{z}_{r,i}$. Under the discrete logarithm assumption in $\mathbb{G}_1$, $\tilde{x} = \mathrm{h}_1^{\delta_{m_1}} \cdots \mathrm{h}_L^{\delta_{m_L}}$ and $\tilde{r} = \delta_{r,i}/(\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}})$[5].

Thus, the extractor KE has successfully extracted a value $\tilde{x}$, whose representation is $\delta_{m_1}, \ldots, \delta_{m_L}$ such that C is a commitment of x with opening $(\tilde{x}, \tilde{r})$ and D is a commitment with opening $\big((\delta_{m_1}, \ldots, \delta_{m_L}), \delta_s\big)$. Thus, $\mathrm{PK}_{\mathsf{RCV}}$ is sound.

Honest Verifier Zero-Knowledgeness of $\mathbf{PK}_{\mathsf{RCV}}$ We construct a zero-knowledge simulator KS for $\mathrm{PK}_{\mathsf{RCV}}$ that, on input a random challenge c, outputs a transcript which is indistinguishable from the actual transcript of a real protocol run.

For a given commitments C and D and a random challenge $c \in_R \{0,1\}^{\lambda_k}$, the simulator randomly generates $z_x, z_r \in_R \mathbb{Z}_p$ and for $i = 1$ to λ_k, $z_{r_i} \in_R \mathbb{Z}_p$, $z_{s,i} \in_R \mathbb{Z}_{\mathrm{q}}$ and $z_{m_1,i} \in_R \mathbb{Z}_{\mathrm{q}}, \ldots, z_{m_L,i} \in_R \mathbb{Z}_{\mathrm{q}}$.

---

[5] Otherwise the discrete logarithm of g_0 to base g can be computed as $\frac{\delta_{r,i} - \tilde{r}\mathrm{h}_1^{\hat{z}_{m_1,i}} \cdots \mathrm{h}_L^{\hat{z}_{m_L,i}}}{\tilde{x} - \mathrm{h}_1^{\delta_{m_1}} \cdots \mathrm{h}_L^{\delta_{m_L}}}$

Next, it computes $T = C^c g_0^{z_x} g^{z_r}$, $T_{2,i} = \mathrm{D}^{c[i]} \mathrm{h}_1^{z_{m_{1,i}}} \cdots \mathrm{h}_L^{z_{m_{L,i}}} \mathrm{h}^{z_{s,i}}$ and $T_{1,i} = g_0^{\mathrm{h}_1^{z_{m_{1,i}}} \cdots \mathrm{h}_L^{z_{m_{L,i}}}} g^{z_{r,i}}$ if $c[i] = 0$ or $T_{1,i} = C^{\mathrm{h}_1^{z_{m_{1,i}}} \cdots \mathrm{h}_L^{z_{m_{L,i}}}} g^{z_{r,i}}$ if $c[i] = 1$. Finally, it outputs $\big(T, c, z_x, z_r\big)$ as a transcript of PK_1 and $\big(T_{1,i}, T_{2,i}, c[i], z_{m_{1,i}}, \ldots, z_{m_{L,i}}, z_{s,i}, z_{r,i}\big)_{i=1}^{\lambda_k}$ as a transcript of PK_2.

It is straightforward to show that the distribution of the simulated transcript is indistinguishable from a real transcript.

B Security Analysis

We prove the lemma below which is central in the proofs of Theorem 2, Theorem 4 and Theorem 5.

Lemma 1. *Define* $\mathsf{param} = (\hat{e}, \mathbb{G}_1, \mathbb{G}_T, \mathbb{G}_\mathrm{q}, p, \mathrm{q}, g, u = g^\alpha, v = g^\beta, \mathrm{h}_1, \ldots, \mathrm{h}_L)$*, and an oracle* O_{BB} *that on input* M*, output a tuple* $\sigma = (A, e)$ *such that* $\hat{e}(A, uv^e g^M) = \hat{e}(g, g)$*. Also define function* $F : \mathbb{Z}_\mathrm{q}^L \to \mathbb{G}_\mathrm{q}$ *as* $F : (m_1, \ldots, m_i) \mapsto \prod_{i=1}^L h_i^{m_i}$*. Under the q-SDH assumption in* $\mathbb{G}_1$ *and the DL assumption in* $\mathbb{G}_\mathrm{q}$*, no PPT algorithm* $\mathcal{A}$ *with input* param *and oracle* O_{BB} *can output* $\ell + 1$ *distinct tuples* $(A_i, e_i, \boldsymbol{m_i}) \in (\mathbb{G}_1, \mathbb{Z}_p, \mathbb{Z}_\mathrm{q}^L)$ *such that* $\hat{e}(A_i, uv^{e_i} g^{F(\boldsymbol{m_i})}) = \hat{e}(g, g)$ *for* $i = 1$ *to* $\ell + 1$*, making only* ℓ *adaptive and possibly interleaving query to* O_{BB}*.*

Proof. The proof is by reduction. Suppose there exists such a PPT algorithm $\mathcal{A}$. $\mathcal{A}$ wins with two possibilities. (1) For all i, $F(\boldsymbol{m_i})$ are distinct. (2) There exists distinct indexes i, j such that $\boldsymbol{m_i}) = \boldsymbol{m_j}$. (3) There exists distinct indexes i, j such that $F(\boldsymbol{m_i}) = F(\boldsymbol{m_j})$ and $\boldsymbol{m_i} \neq \boldsymbol{m_j}$

Case (1) and (2): Obverse that O_{BB} is an signing oracle of the BB-signature. Due to the distinct nature of the $\ell + 1$ tuples, $(A_i, e_i, \boldsymbol{m_i})$, either $\mathcal{A}$ is able to output $\ell + 1$ BB signatures on $\ell + 1$ distinct messages defined as $F(\boldsymbol{m_i})$ (case I) or that $\mathcal{A}$ is able to output at least two different BB signatures on the same message (case 2). Thus, a simulator can easily be constructed, having blackbox access with $\mathcal{A}$, to break the strong unforgeability of BB signature.

Case (3): The condition $F(\boldsymbol{m_i}) = F(\boldsymbol{m_j})$ such that $\boldsymbol{m_i} \neq \boldsymbol{m_j}$ implies that $\mathrm{h}_1^{m_{1,i}} \cdots \mathrm{h}_L^{m_{L,i}} = \mathrm{h}_1^{m_{1,j}} \cdots \mathrm{h}_L^{m_{L,j}}$. The simulator can easily setup the parameter h_i's in $\mathbb{G}_\mathrm{q}$ so as to solve the relative discrete logarithm amongst two of them.

No PPT algorithm $\mathcal{A}$ exists under the q-SDH assumption in $\mathbb{G}_1$ (strong unforgeability of BB signature) and the DL assumption in $\mathbb{G}_\mathrm{q}$. □

Proof of Theorem 2 is given below. Proof of Theorem 4 and Theorem 5 are similar and are thus omitted. As a side note, since the signature of knowledge is probabilistic, our construction of blind signature cannot be strongly unforgeable.

Proof (Sketch). Under Lemma 1, any PPT adversary $\mathcal{A}$ cannot generate $\ell + 1$ distinct tuple of $\big(A_i, e_i, (m_i, s_i)\big)$ with only ℓ interactions with the signing oracle. Thus, any PPT adversary $\mathcal{A}$ will have to produce a fake signature of knowledge Π_m for some message m. This, however, happen only with negligible probability due to the soundness of $\mathrm{PK}_{\mathsf{RCV}}$.

More specifically, if there exists an adversary $\mathcal{A}$ with non-negligible probability in winning the game in Definition 4, we can construct a PPT simulator $\mathcal{S}$ that invalidates Lemma 1, in the random oracle model.

Suppose $\mathcal{A}$ makes q_H query to the hash oracle H. $\mathcal{S}$ randomly chooses one of the hash queries, denoted as query $*$. At the point of hash query $*$, $\mathcal{S}$ makes a copy of (fork) adversary $\mathcal{A}$ (into $\mathcal{A}'$) and replies with a different hash value. Finally, $\mathcal{A}$ and $\mathcal{A}'$ both outputs $\ell + 1$ signatures σ_i on $\ell + 1$ distinct messages m_i. With probability $\ell + 1/q_H$, one of those outputs from $\mathcal{A}$ and $\mathcal{A}'$ will be based on hash query $*$. $\mathcal{S}$ can thus invoke the extractor KE of Π_m to obtain the underlying tuple $\big(A_i, e_i, (m_i, s_I)\big)$. With probability at least $1/\ell + 1$, this tuple can be used to invalidate Lemma 1.

Proof of Theorem 3 is given below. It should be noted that, in the random oracle model, we can always ensure the signer generates the generators h_i's and $g_i's$ honestly by setting them as the output of some hash functions on some publicly known seed. In fact, unless some of these generators is taken to be the identity element, the signer cannot break the blindness property of our construction even if it is computationally unbounded.

Proof (Sketch). Recall that a signature on message m consists of values ($\mathfrak{A}_1$, $\mathfrak{A}_2$, $\mathfrak{A}_3$, A_4, SPK_1 and SPK_2). $\mathfrak{A}_2$, $\mathfrak{A}_3$, A_4 are information theoretically secure commitments of values r_1, y and s respectively and thus leak no information to even computationally unbounded adversary. $\mathfrak{A}_1$ is the product of A (values known to the signer) and a random number g^{r_1} and it also leaks no information. Finally, SPK_1 and SPK_2 are just non-interactive zero-knowledge proof-of-knowledge (or more precisely, signature-of-knowledge). Under the random oracle model and the honest-verifier zero-knowledge property of Π_m, it also leaks no information. Thus, our blind signature possesses blindness. □

C A Framing Attack on KTY Traceable Signatures

In this section, we present a high level description of the traceable signatures from [36] (KTY) and a concrete attack within their security model.

Overview of the KTY Traceable Signature

GGen: The group manager chooses a signature scheme. The signature scheme in KTY is in fact a variant of the CL signature [16].

Join: User chooses a random number x' and obtains a CL signature (denoted as cert) from the GM on values x', x using the signature generation protocol of CL signature. In particular, x' is unknown to GM while x is known. The value x is stored as the tracing information tr of the user. User stores cert as his membership certificate.

GSign: To sign a message m, user with membership certificate cert on values x', x first computes:
1. a tuple (T_1, T_2, T_3), which is the El-Gamal encryption of part of cert.
2. a tuple (T_4, T_5) such that $T_5 = g^k$ and $T_4 = T_5^x$ for some random number k.
3. a tuple (T_6, T_7) such that $T_7 = g^{k'}$ and $T_6 = T_7^{x'}$ for some random number k'.

The traceable signature is a signature of knowledge σ_m such that $(T_1, \ldots, T_7)$ are correctly formed.

GVer: The verifier simply verifies the signature-of-knowledge σ_m.

Open: On input m, σ_m, the group manager outputs the identity of the signer by decrypting T_1, T_2, T_3 and obtains cert of the user.

Reveal: On input Jtrans_i, the group manager outputs tracing information $\mathsf{tr} = x$.

Trace: On input a signature σ_m and a tracing information tr, test whether $T_4 \stackrel{?}{=} T_5^x$.

Claim/ClaimVer: To claim a signature, the signer produces a non-interactive proof-of-knowledge of discrete logarithm of T_6 to base T_7 (which is x').

The Framing Attack The framing attack is considered successful if the attacker can generate a signature that traces to an honest user. Specifically, the adversary is considered successful if it can output a signature σ_m^* such that $\mathsf{Trace}(\mathsf{Reveal}(\mathsf{Jtrans}_i), \sigma_m^*) = 1$ and that user U_i is an honest user who has not generated σ_m^* himself. The attack is based on the fact that σ_m^* does not need to open to U_i, and the attacker knows the corresponding tracing information, that is, x, of an honest user. To frame an honest user, the adversary generates another membership certificate cert^* on values x^*, x and uses it to produce a signature σ_m^*. Obviously, this signature will trace to the honest user.

The attack is possible due to a flaw in the security proof [37] (full version of [36] , Section 9.3), in which it is stated that "Then if the adversary outputs an identification transcript that either opens to user j traces to the user j, it is clear that we can rewind the adversary and obtain a witness for that transcript that will reveal the logarithm of C base b, and thus solving the discrete-logarithm problem." The argument is true when the identification transcript opens to user j in which it helps solving the discrete logarithm of C to base b (which is x', the user secret). However the same argument is not applicable to the case of tracing because the tracing information x for user j is in fact known to the adversary. The adversary is not required to use the same x' with the honest user in producing the signature for framing to be successful.

The Proposed Fix It turns out that the same attack is not applicable to the pairing-based traceable signatures [24] (CPY). The reason is that the tracing information tr is of the form g^x and, although tr is known to GM, the value x is unknown and correctness of tr is implicitly checked in a signature of knowledge of x. The same idea, however, is not applicable to the original KTY scheme because the tracing mechanism in CPY requires the use of a bilinear map[6] which is not known to exists in the group of which KTY is built on. Thus, we propose another fix. That is, the tracing information tr is no longer randomly chosen. Instead, it is set to be $H(C_i)$, where $C_i = b_i^{x'}$ is known to GM during the join protocol in KTY, for some collision-resistant hash function H. The group signature will be modified so that the user will encrypt C_i under the public key g^{tr} (using El-Gamal Encryption), together with a proof-of-correctness, including the knowledge of C_i to base b_i. The corresponding Trace algorithm is also modified to include a test that $\mathsf{tr} \stackrel{?}{=} H(C_i)$ when tr is given. Indeed, this idea is employed in our construction of traceable signatures.

[6] Specifically, for each signature, user produces values T_4, T_5 such that the tracing agent test if $\hat{e}(\mathsf{tr}, T_4) \stackrel{?}{=} T_5$. The user also includes a proof-of-knowledge of discrete logarithm (that is, knowledge of x) of T_5 to base $\hat{e}(g, T_5)$ in the signature.