Linkable and Culpable Ring Signatures

Joseph K. Liu[1], Victor K. Wei[1], and Duncan S. Wong[2]

[1] Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
{ksliu9,kwwei}@ie.cuhk.edu.hk
[2] Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
duncan@cityu.edu.hk

Abstract. We introduce a new variant for ring signature that supports *linkability* and *culpability*. A ring signature is a 1-out-of-n signer-ambiguous group signature. One of the most distinguishing features of a ring signature among other group signature schemes is *spontaneity* — the lack of any group manager, TTP (Trusted Third Party), and there is also no collaboration between any two group members. By linkability, a public verifier is able to tell if two ring signatures are created by the same signer. This feature is useful for applications in which each signer is allowed to generate one signature in each event, for example, detecting double-voting in an e-voting system. By culpability, it can be shown that a ring signature is indeed created by a particular signer after his private key is revealed. In the paper, we also propose a realization for DL-type public key pairs over the same domain. Our scheme yields an efficient one-round e-voting scheme achieving anonymity, fairness, spontaneity, and other properties. In addition, we show how to extend it to a linkable and culpable t-out-n threshold ring signature scheme. On security analysis, we establish the ROS (Rewind-on-Success) lemma. Base on this lemma, our tape rewinding simulation can be nested without reducing the success rate.

Keywords: ring signature, linkable ring signature, spontaneous group signature

1 Introduction

Rivest et al. [25] presented the notion of ring signature in 2001. A $(1, n)$-ring signature convinces a public verifier that a message is signed by one of n possible signers without revealing the identity of the actual signer. It is different from a group signature [13]. The group signature has a group manager who can always revoke the identity of the actual signer. A ring signature does not have a group manager and no one can revoke the identity of the actual signer. In addition, it does not require any coordination among the n possible signers during the

signature generation. The formation of the signer group is *spontaneous*. No secret sharing or pseudonym systems that require membership collaboration are needed to implement. Any signer can choose any set of possible signers that includes himself, and sign any message by using his private key and the others' public keys without their consent.

Unlinkability and inculpability are some implicit features of all the current ring signature schemes [25, 1, 6, 29, 28]. Unlinkability means that a public verifier (that is, someone who has all the public keys of the n possible signers) cannot tell if two ring signatures are actually generated by the same signer, even the signatures include the same set of public keys. Inculpability allows the actual signer of a ring signature to claim that the signature was *not* signed by him, even after his private key has been revealed. All the current ring signature schemes have these two properties. However in some cases, they may not be as desirable as one may think.

Let us consider the following scenario. Suppose there is an organization which wants to conduct an anonymous and voluntary questionnaire among its members. It is required that only the legitimate members can submit the questionnaire, and at the same time, each member cannot submit more than one questionnaire. In addition, the organization may want to check the authorship of a particular questionnaire some time later when this becomes necessary. For example, when a user leaves and joins a competitor of the organization. The user has to surrender all properties belonging to the organization before he leaves including the private key. Conventional ring signature can be used to solve part of the problem. It can ensure those who submitted the questionnaire are members of the organization and at the same time, maintain user anonymity. However, without any additional mechanisms, it cannot prevent a member from submitting more than one questionnaire as it is unlinkable. It is also unable to allow the organization to find out who has submitted a particular questionnaire even the organization has collected all the private keys of its members as it is inculpable.

Using other cryptographic protocols such as e-voting systems [18, 14, 15], blind signature schemes [10, 7], or pseudonym systems [12, 16, 21] seem to be able to solve the problem. Yet they require *all* the users to participate in some preparation or setup stage even they do not want to take part in the subsequent protocols. In other words, the *voluntariness* may not be flawlessly realized.

Furthermore, inculpability may become an even more problematic feature if we consider a usual legal practice when an investigator backed by law to subpoena a private key and determine the responsibility of having generated a signature by someone, say a drug dealer.

1.1 Contributions

In this paper, we introduce a new variant of ring signature. It provides *linkability* and *culpability* while preserving all other features of current ring signature such as anonymity and spontaneity or uncoordinated generation of signatures.

By linkability, a public verifier is able to tell if two $(1, n)$-ring signatures, which include the same set of n public keys, are actually generated by the same

signer. The two ring signatures may correspond to two different messages but the two associating sets of n public keys must be identical. Since the scheme will still maintain anonymity, the verifier will not be able to identify who the actual signer is.

By *culpability*, it can be shown that a ring signature is indeed created by a particular signer after his private key is revealed. A culpable ring signature provides an option to determine the authorship of the signature from a private key corresponding to one of the public keys associated to it. We would like to emphasize that it is the signers' interest to safeguard their private keys and therefore revealing the identity of the actual signer is at the signers' own discretion, if it is not coerced. We stress the distinction between this new feature and the feature of *claimability*. Claimability allows a signer to come forth on his own volition and claim responsibility by providing proof of having generated a given signature. This feature is easy to achieve in any of the current ring signature schemes by embedding some *secret* proof of knowledge [25]. However, culpability is not known to any of the current ring signature schemes. In general, culpability implies claimability but not vice versa.

Besides introducing the two new features, linkability and culpability, other contributions are in the following. First, we propose a ring signature scheme for DL-type (discrete logarithm) public keys over the same domain to realize linkability and culpability. Second, we provide a t-out-n threshold extension to the basic scheme. In the scheme, a public verifier is able to find out the number of signers who participate in generating two signatures on the same set of n possible signers, that is, linkability. Culpability is also maintained in the same sense described above. Third, we describe an efficient one-round e-voting system which is based on linkable ring signature. The e-voting system has a simple preparation stage which does not involve any of the voters. In addition, double voting can readily be detected due to the linkability feature of the underlying ring signature scheme.

Fourth on security analysis, we show the ROS (Rewind-on-Success) lemma. Base on this lemma, our tape rewind simulation can be nested without reducing the success rate. In current (indiscriminate) tape rewind simulation approaches which are based on heavy-row lemma or forking lemma, the success rate decreases exponentially when the number of nested layers increases.

The rest of the paper is organized as follows. Some related work is reviewed in Sec. 2. It is followed by the description of the security model in Sec. 3. Our linkable and culpable ring signature scheme is described in Sec. 4 and the security analysis is given in Sec. 5. In Sec. 6, we construct an e-voting system using linkable ring signature schemes. The threshold extension is then described in Sec. 7 and the paper is concluded in Sec. 8.

2 Related Work

(Ring Signatures) Since the notion of ring signature was first formalized by Rivest et al. in [25], other ring signature schemes have been proposed [1, 6, 29,

28]. In [6, 28], (t, n)-threshold ring signature schemes have also been proposed. All these schemes are unlinkable and claimable but inculpable. In Sec. 1, we explain that linkability could be a very useful feature which can help detect double-voting in an e-voting system. In addition, culpability implies claimability but not vice versa. Hence one of our objectives in this paper is to propose a ring scheme scheme which is linkable, culpable and therefore claimable as well. At the same time, anonymity is to be maintained.

(E-voting Schemes) The first e-voting scheme was proposed by Chaum in [9]. Since then, there were many different e-voting models proposed. In general, an e-voting system consists of a group of voters, a Registry to specify the group of eligible voters, a Voting Center for collecting votes and a Tally to count votes. A voting event can be divided into three phases : Registration Phase, Voting Phase and Vote-Open Phase. In the Registration phase, the Registry may need to interact with voters to define the group of eligible voters. In the Voting phase, eligible voters send their votes to the Voting Center. In the Vote-Open phase, the Tally counts the votes and publishes the result.

As one of the applications of linkable ring signatures, we propose a new e-voting system which has the following properties. There is no voter involved in the Registration phase. The Voting phase is one-round. The Tally is just a public bulletin so that everyone can do the counting. In Appendix A, we provide a more detailed description on the common security definitions of an e-voting scheme and the classification of current e-voting schemes.

3 Security Model

In this section, we define a $(1, n)$-ring signature scheme and specify the security requirements of a linkable and culpable $(1, n)$-ring signature scheme. Let $k \in \mathbb{N}$ be a security parameter.

Definition 1 (A Ring Signature Scheme). *A $(1, n)$-ring signature scheme is a triple $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ such that*

- $(\hat{s}, P) \leftarrow \mathcal{G}(1^k)$ *is a probabilistic polynomial time algorithm (PPT) which accepts as input a security parameter k, produces a private key $\hat{s}$ and a public key P.*
- $\sigma \leftarrow \mathcal{S}(1^k, \hat{s}, L, m)$ *is a PPT which accepts as inputs a private key $\hat{s}$, a set of n public keys L including the one that corresponds to $\hat{s}$ and a message $m \in \{0,1\}^*$, produces a signature σ.*
- $1/0 \leftarrow \mathcal{V}(1^k, L, m, \sigma)$ *is a polynomial-time algorithm which accepts as inputs a set of public keys L, a message $m \in \{0,1\}^*$ and a signature σ, returns 1 or 0 for* accept *or* reject*, respectively. We require that $\mathcal{V}(1^k, L, m, \mathcal{S}(1^k, \hat{s}, L, m)) = 1$ for any $m \in \{0,1\}^*$, $(\hat{s}, P) \leftarrow \mathcal{G}(1^k)$ and any L that includes P.*

For simplicity, we usually omit the input of security parameter when using $\mathcal{S}$ and $\mathcal{V}$ in the rest of the paper.

Remark on Spontaneity: The signature generation algorithm $\mathcal{S}$ requires inputs of only one private key $\hat{s}$ and n public keys including the one corresponding to $\hat{s}$ from the public when generating a signature. This implies that an actual signer does not need any TTP, group manager or cooperation among group membership, in pre-processing or in computing the scheme itself.

The security of a linkable and culpable ring signature scheme consists of four requirements. They are *Unforgeability*, *Signer Ambiguity*, *Linkability* and *Culpability*.

Definition 2 (Existential Unforgeability against Adaptive Chosen Message and Chosen Public-key Attacks). *Let $\mathcal{SO}'(L', \pi, m)$ be a signing oracle that accepts as inputs any set of public keys L' in which each public key is generated by $\mathcal{G}$ with corresponding security parameter, any index of the actual signer π, $1 \leq \pi \leq |L'|$, and any message $m \in \{0,1\}^*$, produces a signature σ such that $\mathcal{V}(L', m, \sigma) = 1$. Let $(\hat{s}_i, P_i) \leftarrow \mathcal{G}(1^{k_i})$ for each i, $1 \leq i \leq n$, where $k_i \in \mathbb{N}$. Let $k = \min(k_1, \cdots, k_n)$ and $L = \{P_1, \cdots, P_n\}$. A $(1,n)$-ring signature scheme is unforgeable if, for any PPT $\mathcal{A}$ with signing oracle $\mathcal{SO}'$, any L, and any sufficiently large k,*

$$\Pr[\mathcal{A}(1^k, L) \rightarrow (m, \sigma) : \mathcal{V}(L, m, \sigma) = 1] \leq \epsilon(k)$$

where ϵ is some negligible function. Restriction is that (L, m, σ) should not be found in the set of all oracle queries and replies between $\mathcal{A}$ and $\mathcal{SO}'$. The probability is taken over all the possible inputs of $\mathcal{A}$, oracle queries and coin flips of $\mathcal{A}$.

A real-valued function ϵ is negligible if for every $c > 0$ there exists a $k_c > 0$ such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

Definition 3 (Signer Ambiguity). *Let $\mathcal{SO}''(L', m)$ be a signing oracle that accepts as inputs any set of public keys L' in which each public key is generated by $\mathcal{G}$ with corresponding security parameter, and any message $m \in \{0,1\}^*$, produces a signature σ such that $\mathcal{V}(L', m, \sigma) = 1$. Let $L = \{P_1, \cdots, P_n\}$ where each key is generated as $(\hat{s}_i, P_i) \leftarrow \mathcal{G}(1^{k_i})$ where $k_i \in \mathbb{N}$. Let $k = \min(k_1, \cdots, k_n)$. Let $\mathrm{Sym}(n)$ be the symmetric group of order n. It consists of all permutations on n objects. A ring signature scheme is signer ambiguous if for any PPT $\mathcal{A}$ with signing oracle $\mathcal{SO}''$, any L, any message $m \in \{0,1\}^*$, any $\pi \in \{1, \cdots, n\}$, any signature $\sigma \leftarrow \mathcal{S}(\hat{s}_\pi, L, m)$, any $t < n-1$, and all sufficiently large k,*

$$\begin{aligned}\Pr[\mathcal{A}(1^k, L, m, \sigma) \rightarrow \phi,\ \mathcal{A}(1^k, L, m, \sigma, \hat{s}_{\phi(1)}, \cdots, \hat{s}_{\phi(t)}) \rightarrow \hat{\pi} :\\ \phi \in \mathrm{Sym}(n),\ \pi = \hat{\pi},\ \hat{s}_\pi \neq \hat{s}_{\phi(i)}, \forall 1 \leq i \leq t] \leq \frac{1}{n-t} + \epsilon(k)\end{aligned}$$

where ϵ is some negligible function. Restriction is that $(L, m, \sigma) \notin \{(L'_i, m_i, \sigma_i)\}$ where $\{(L'_i, m_i, \sigma_i)\}$ is the set of oracle queries and replies between $\mathcal{A}$ and $\mathcal{SO}''$. The probability is taken over all the possible inputs, all possible values of π and coin flips of $\mathcal{A}$.

Note that the signing oracles in Definition 2 and 3 are different. In Definition 3, the attacker cannot actively choose an actual signer for generating a signature when using the signing oracle. This is different from the one in Definition 2. If the same signing oracle as the one in Definition 2 were used, linkability feature defined below will help find out the identity of all linked signatures once the identity of the actual signer of at least one of these linked signatures is revealed. Two signatures are *linked* if they are from the same signer. Formally,

Definition 4 (Linkability). *Let $L = \{P_1, \cdots, P_n\}$ where each key is generated as $(\hat{s}_i, P_i) \leftarrow \mathcal{G}(1^{k_i})$. Let $k = \min(k_1, \cdots, k_n)$. A ring signature scheme is linkable if for any L, any $\pi_1, \pi_2 \in \{1, \cdots, n\}$, any messages $m_1, m_2 \in \{0,1\}^*$ and any $\sigma_1 \leftarrow \mathcal{S}(\hat{s}_{\pi_1}, L, m_1)$, $\sigma_2 \leftarrow \mathcal{S}(\hat{s}_{\pi_2}, L, m_2)$, there exists a PPT $\mathcal{F}_1$ such that for all sufficiently large k, it always outputs $1/0$ with*

$$\Pr[\mathcal{F}_1(1^k, L, m_1, m_2, \sigma_1, \sigma_2) = 0 : \pi_1 = \pi_2] \leq \epsilon(k)$$

and

$$\Pr[\mathcal{F}_1(1^k, L, m_1, m_2, \sigma_1, \sigma_2) = 1 : \pi_1 \neq \pi_2] \leq \epsilon(k)$$

where ϵ is some negligible function. The probability is taken over all the possible inputs, all possible values of π_1 and π_2, and coin flips of $\mathcal{F}_1$.

The algorithm $\mathcal{F}_1$ outputs 1 if it thinks the two signatures are linked, that is, are signed by the same ring member.

Linkability defined above requires that the set of public keys L is fixed while there is no additional requirement on the messages of the signatures. A general form of linkability is discussed in Sec. 4.4. When describing our basic scheme in Sec. 4, we follow the definition above for simplicity.

A $(1, n)$-ring signature scheme is *culpable* if an external entity, given the private key of a group member, can ascertain whether that member is the signer. Formal definition is below. But first a technical definition. Let $f_0(L, \hat{s}) = i$, $1 \leq i \leq n$, where $L = P_1 || \cdots || P_n$ and $(\hat{s}, P_i)$ is a key pair. Let $f_0(L, \hat{s}) = 0$ if no such key pair exists.

Definition 5 (Culpability). *Let $L = \{P_1, \cdots, P_n\}$ where each key is generated as $(\hat{s}_i, P_i) \leftarrow \mathcal{G}(1^{k_i})$ and $k = \min(k_1, \cdots, k_n)$. A ring signature scheme is culpable if for any L, any message $m \in \{0,1\}^*$, any signature $\sigma = \mathcal{S}(\hat{s}, L, m)$ where $f_0(L, \hat{s}) = \pi$ for some π, $1 \leq \pi \leq n$, and any j, $1 \leq j \leq n$, there exists a PPT $\mathcal{F}_2$ such that for all sufficiently large k, it outputs $1/0$ with*

$$\Pr[\mathcal{F}_2(1^k, L, m, \sigma, \hat{s}, j) = 0 : j = f_0(L, \hat{s})] \leq \epsilon(k)$$

and

$$\Pr[\mathcal{F}_2(1^k, L, m, \sigma, \hat{s}, j) = 1 : j \neq f_0(L, \hat{s})] \leq \epsilon(k)$$

where ϵ is some negligible function. The probability is taken over all possible inputs, all possible values of f_0 and coin flips of $\mathcal{F}_2$.

The algorithm $\mathcal{F}_2$ outputs 1 if it thinks member j is the signer (culprit). The signing algorithm $\mathcal{S}(\hat{s}, L, m)$ can compute π such that $\hat{s} = s_\pi$ from L and $\hat{s}$.

For any valid culpable ring signature, by revealing the actual signer's private key, the public can be convinced that the signature is generated by the signer. On the other side, if the private key of a non-participating user specified in L is revealed, the public can also be convinced that the signature is not generated by that user.

4 A Linkable and Culpable Ring Signature Scheme

Let $G = \langle g \rangle$ be a group of prime order q such that the underlying discrete logarithm problem is intractable. Let $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_2 : \{0,1\}^* \to G$ be some statistically independent cryptographic hash functions. We require that for any $\alpha \in \{0,1\}^*$, the discrete logarithm of $H_2(\alpha)$ to the base g in G is intractable. For $i = 1, \cdots, n$, each user i has a distinct public key y_i and a private key x_i such that $y_i = g^{x_i}$. Let $L = \{y_1, \cdots, y_n\}$ be the set of the n public keys. Sometimes, we may pass in the set L for hashing and we implicitly assume that certain appropriate encoding method is applied.

For some message $m \in \{0,1\}^*$, a user (signer) π uses his private key x_π and generate a linkable and culpable $(1, n)$-ring signature with respect to L as follows.

4.1 Signature Generation

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute
$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute
$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$
4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

4.2 Signature Verification

A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a set of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z_i' = g^{s_i} y_i^{c_i}$, $z_i'' = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

4.3 Linkability and Culpability

For a fixed set of public keys L, given two signatures associating with L, namely $\sigma'_L(m') = (c'_1, s'_1, \cdots, s'_n, \tilde{y}')$ and $\sigma''_L(m'') = (c''_1, s''_1, \cdots, s''_n, \tilde{y}'')$, where m' and m'' are some messages, a public verifier after verifying the signatures to be valid, checks if $\tilde{y}' = \tilde{y}''$. If the congruence holds, the verifier concludes that the signatures are created by the same signer. Otherwise, the verifier concludes that the signatures are generated by two different signers.

For a valid signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on some message m and some set of public keys L, an investigator subpoenas a private key x_i from user i. If x_i is the private key of some $y_i \in L$ (that is, $y_i = g^{x_i}$) and $\tilde{y} = H_2(L)^{x_i}$, then the investigator conducts that the authorship of the signature belongs to user i.

4.4 Generalization

We denote the signature with respect to L by σ_L. That it, linkability is only meaningful in the context of having two ring signatures associate with the same set of parameters, which is L in the description above. In general, a linkable ring signature can be specified with respect to *any* other parameters, not limited to L. For example, in Sec. 6, we specify the set of parameters to be L and a unique e-voting event identifier ID. In this case, we compute $h = H_2(L, ID)$. The corresponding signature is denoted by $\sigma_{L,ID}$. We give more examples in Sec. 7.1 when describing some variations of our linkable threshold ring signature scheme.

5 Security Analysis

In this section, we analyze the security of our proposed scheme with the assumption that all the hash functions are distinct and behave like random oracles [3]. According to the following theorem, which is shown in Appendix B, our basic scheme is existentially unforegable against adaptive chosen message and chosen public-key attacks.

Theorem 1 (Existential Unforgeable). *Let $\mathcal{SO}$ be a signing oracle which performs signature generation and returns valid signatures according to our basic scheme described in Sec. 4. Let H_1 and H_2 be two distinct and ideal hash functions [3]. Let L be a set of n distinct public keys defined in Sec. 4. Let $\hat{S}$ be a set of private keys unrelated to L. Suppose there exists a PPT $\mathcal{A}$ which makes at most q_H queries to H_1 and H_2 combined and at most q_S queries to $\mathcal{SO}$ such that*

$$\Pr[\mathcal{A}(1^k, \hat{S}, L) \rightarrow (m, \sigma) : \mathcal{V}(L, m, \sigma) = 1] > \frac{1}{Q_1(k)}$$

for some polynomial Q_1 where $\mathcal{V}$ is the signature verification algorithm according to Sec. 4.2. Then, there exists a PPT which can solve the Discrete Logarithm Problem (DLP) with probability at least $(\frac{1}{n(q_H+nq_S)Q_1(k)})^2$.

Based on the proof and security analysis in Appendix C, we have the following theorem.

Theorem 2 (Signer Ambiguous). *Let L be a set of n public keys defined in Sec. 4. Suppose there exists a PPT $\mathcal{A}$ such that for any $m \in \{0,1\}^*$, $\pi \in \{1,\cdots,n\}$, and t $(t < n)$ private keys $\hat{s}_{i_1},\cdots,\hat{s}_{i_t}$ where $i_j \neq \pi$, $1 \leq j \leq t$,*

$$\Pr[\mathcal{A}(1^k, m, L, \hat{s}_{i_1},\cdots,\hat{s}_{i_t}, \sigma) \rightarrow \pi : \sigma \leftarrow \mathcal{S}(\hat{s}_\pi, L, m)] > \frac{1}{n} + \frac{1}{Q_2(k)}$$

for some polynomial Q_2 and the signature generation algorithm $\mathcal{S}$ according to Sec. 4.2. Then another PPT can be constructed which solves the Decisional Diffie-Hellman Problem (DDHP) with probability at least $\frac{1}{2} + \frac{1}{4Q_2(k)}$.

On linkability and culpability, we obtain the following theorems with proofs given in Appendix D and E.

Theorem 3 (Linkable). *Let $L = \{y_1,\cdots,y_n\}$ be a set of public keys where each key is defined according to Sec. 4. Let x_i, $1 \leq i \leq n$, be the corresponding private keys. Suppose there is a PPT $\mathcal{A}$ such that for any L, $\pi \in \{1,\cdots,n\}$ and k,*

$$\Pr[\mathcal{A}(1^k, L, x_\pi) \rightarrow (m', m'', \sigma'_L, \sigma''_L) : \mathcal{V}(L, m', \sigma'_L) = 1, \mathcal{V}(L, m'', \sigma''_L) = 1, \tilde{y}' \neq \tilde{y}''] > \frac{1}{Q_3(k)}$$

for some polynomial Q_3 where $\tilde{y}'$ and $\tilde{y}''$ are the last components of σ'_L and σ''_L, respectively, and $\mathcal{V}$ is the signature verification algorithm according to Sec. 4.2. Then another PPT can be constructed which computes the discrete logarithm of at least one public key in L other than y_π with non-negligible probability.

Theorem 4 (Culpable). *Let $L = \{y_1,\cdots,y_n\}$ be a set of public keys where each key is defined according to Sec. 4. Let x_i, $1 \leq i \leq n$, be the corresponding private keys. Let $\sigma_L = (c_1, s_1,\cdots,s_n,\tilde{y})$ where $c_1, s_i \in \mathbb{Z}_q$, $1 \leq i \leq n$ and $\tilde{y} \in G$. Let $m \in \{0,1\}^*$ be a message. For any L and (m, σ_L) such that $\mathcal{V}(L, m, \sigma) = 1$, $\tilde{y} = H_2(L)^{x_\pi}$ for some $\pi \in \{1,\cdots,n\}$ where $\mathcal{V}$ is the signature verification algorithm described in Sec. 4.2.*

6 An E-voting System

An e-voting system consists of a Registry (R), a Voting Center (VC), a Tally (T) and a group of users who are identified by their distinct public keys. In each voting event, there are three phases: Registration phase, Voting phase and Vote-Open phase. Let k be a system-wide security parameter and $h : \{0,1\}^* \rightarrow \{0,1\}^k$ be a cryptographic hash function. We assume that each voter is associated with a certified public key of some standard signature scheme such as DSA [22]. In the following, we describe each phase of a voting event one by one.

Registration Phase.

1. The Registry R randomly picks a number $ID \in_R \{0,1\}^k$ and publishes it as the unique identifier of this voting event.
2. R then specifies a group of eligible voters and announces the group to the public by publishing L which is the set of public keys of all the eligible voters. Let n be the number of eligible voters.

Voting Phase. The Voting Center C is a public bulletin. We assume that once something is sent to C and 'published' on the bulletin, the information becomes public and cannot be altered further. In addition, C is assumed to know nothing about sender's identity when receiving something. This assumption is similar to the requirement of having an anonymous channel from voters to C.

For $i = 1, \cdots, n$,

1. voter i casts a vote $Vote_i$;
2. randomly picks $b_i \in_R \{0,1\}^k$ and computes $C_i = h(Vote_i, b_i)$ as a *commitment*;
3. generates a linkable ring signature $\sigma_{L,ID}(C_i)$ using the signing algorithm described in Sec. 4.1;
4. and sends $(C_i, \sigma_{L,ID}(C_i))$ to C.

Each signature is publicly verified using the signature verification algorithm described in Sec. 4.2. For each pair of signatures, linkability is determined according to Sec. 4.3. If any pair of signatures are linked, then both signatures are marked to be void in the bulletin. They would not be counted in the Vote-Open phase.

Vote-Open Phase. The Tally T is another public bulletin with the same assumptions as C. The Vote-Open phase begins only after the Voting phase is completed. Hence no one can send any more commitment with signature to C when the Vote-Open phase starts.

1. For $i = 1, \cdots, n$, voter i sends $Vote_i$ and b_i to T.
2. It is publicly verified if $C_i \stackrel{?}{=} h(Vote_i, b_i)$ for $1 \leq i \leq n$.

A dishonest T may refuse to post a vote in practice. This can be identified easily by having the corresponding voter send the vote anonymously to several "independent parties" (e.g. some solicitors) who are assumed not collaborating with each other. There may no be such concern in the Voting phase. As the value of each vote and the identity of its sender are both not known to C, C does not have any motivation to alter or reject posting any incoming commitments.

6.1 Additional Features

("Vote-and-Go" Feature) In the e-voting system above, a voter sends a commitment and a linkable ring signature in the Voting phase and reveals the vote in the Vote-Open phase. A "Vote-and-Go" e-voting system does not require the voter to be involved in the Vote-Open phase. This feature may improve efficiency and reduce the complexity of the entire system. Our system above can be modified to a "Vote-and-Go" e-voting system easily by requiring each voter to encrypt their vote using a probabilistic encryption function under a public key of a trusted third party called an Arbitrator (A). The public key is assumed to be unique for each voting event and is published during the Registration phase. In practice, the role of A can also be played by R. In the Vote-Open phase, A publishes the corresponding private key for decrypting the votes and conducting the ballot count by the public.

(Receipt-freeness) A receipt-free e-voting system prevents a voter from claiming the authorship of a particular vote. Since all the ring signature schemes are claimable, the e-voting systems above are not receipt-free. Our systems can be modified to support receipt-freeness by using a tamper-resistant randomizer [20]. A built-in randomizer is responsible for generating all the random numbers for probabilistic functions carried out in the device. Users are only allowed to enter their vote choices and their devices do the rest without further intervention. This is a practical model and we omit the details due to the page limitation.

6.2 Comparison with Previous Schemes

In Sec. 2, previous e-voting schemes are classified into two types. There are several advantages of this scheme over previous ones. Compare with Type 1 schemes, anonymity is maintained without trusting the Tally. The system does not leak any information on who has voted and who has not. In addition, voters among different voting events are unlinkable. The scheme is more scalable and performs almost the same no matter the votes are "yes/no" type, 1-out-of-n type or even t-out-of-n type. Compare with Type 2 schemes, the Registration phase is simplified and there is no interaction between voters and the Registry. Detecting double voting can be done by employing the linkability property of the underlying linkable ring signature scheme.

7 Threshold Extension

A (t, n)-threshold ring signature allows a public verifier to tell if the signature is generated by t distinct signers out of n possible signers without getting any information on which t signers are. In the following, we describe how to extend the basic scheme described in Sec. 4 and construct a linkable and culpable (t, n)-threshold ring signature scheme.

We use the same notations as we denoted in Sec. 4. For a message $m \in \{0, 1\}^*$ and a set of n public keys L, a linkable and culpable (t, n)-threshold ring signature is generated as follows.

1. For $i = 1, \cdots, t$, user i constructs a signature $\sigma^i_{L,t}(m)$ with respect to L and t using the signing algorithm described in Sec. 4.1. That is, we compute $h = H_2(L, t)$.
2. The (t, n)-threshold ring signature, denoted by $\sigma^{(t,n)}_{L,t}(m)$, is

$$(\sigma^1_{L,t}(m),\ \cdots,\ \sigma^t_{L,t}(m))$$

The signature verification is done in the obvious way. First, each of the t $(1, n)$-ring signature is verified using the verification algorithm given in Sec. 4.2. Second, the algorithm for detecting linkability described in Sec. 4.3 is carried out. If all the signature verifications are passed successfully and no two signatures are generated by the same signer, the verifier concludes that the signature is valid. Otherwise, the verifier rejects the signature.

7.1 Variations

Dropping t from the subscript of the signature notation $\sigma^{(t,n)}_{L,t}(m)$ above means that linkability is now in the context of the public key set L only. This allows one to find out how many signers have participated in the generation of two threshold ring signatures even the two signatures are having different values of t, provided that L is the same for both signatures.

One technical remark is that the value of t must be committed in the signature. The reason can be easily seen by noting that the value of t of such a (t, n)-threshold ring signature can later be reduced arbitrarily by removing some $(1, n)$-ring signatures from it. This issue can easily be fixed by requiring the 'message' to be (m, t) instead of m alone. Hence we denote this variant of (t, n)-threshold ring signature by $\sigma^{(t,n)}_{L}(m, t)$.

Continue working on this variant and replace L by a nonce denoted by r which is distinct for each signature, then we obtain an *unlinkable* but culpable (t, n)-threshold ring signature scheme. We denote this variant by $\sigma^{(t,n)}_{r}(m, t)$. Similar idea can also be applied to our basic scheme for generating unlinkable but culpable $(1, n)$-ring signatures. Details are omitted due to the page limitation.

8 Conclusions

In this paper, we introduce a new variant of ring signature. It provides *linkability* and *culpability*, while preserving all other features such as anonymity and spontaneity. By linkability, a public verifier is able to tell if two signatures are created by the same signer. By culpability, an investigator can subpoena a private key corresponding to one of the n public keys of a signature and determine if the signature is created using the private key.

We propose a realization of a linkable and culpable ring signature scheme for DL-type public keys over the same domain and extend it to a (t, n)-threshold ring signature scheme. On security analysis, we establish the ROS (Rewind-on-Success) lemma. It allows rewind simulation to be nested without reducing

the success rate. This is potentially more powerful than current (indiscriminate) rewind simulation in proof scenarios where multiple layers of rewinding are nested.

A suite of e-voting systems based on the linkable ring signature scheme have been presented. Our basic e-voting system is one-round (that is, vote-and-go) and can detect double-voting while maintaining voter anonymity efficiently. The Registration phase is also greatly simplified. Voters do not need to collaborate with others nor interact with the Registry before being enrolled as eligible voters.

In general, a linkable ring signature scheme can be used as a primitive mechanism to detect duplicated signature generations in a system. In an e-voting system, linkable ring signature can be used to detect double-voting while maintaining voter anonymity. We believe that linkable ring signature can also be used to construct a more efficient e-cash system as it may provide a simple way to detect double-spending, which usually entails the major complexity in an e-cash systems.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *Proc. ASIACRYPT 2002*, pages 415–432. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501.
2. O. Baudron, P. A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *Proc. the 20th ACM Symposium on Principles of Distributed Computing*, pages 274–283. ACM Press, 2001.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
4. J. Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Department of Computer Science, New Haven, Yale University, September 1987.
5. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th ACM Symp. on Theory of Computing (STOC)*, pages 544–553. ACM, 1994.
6. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proc. CRYPTO 2002*, pages 465–480. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2442.
7. J. Camenisch, J. Piveteau, and M. Stadler. Blind signatures based on the discrete logarithm problem. In *Proc. EUROCRYPT 94*, pages 428–432. Springer-Verlag, 1995. Lecture Notes in Computer Science No. 950.
8. R. Chan, J. Wong, and A. Chan. Anonymous electronic voting system with non-transferable voting passes. In *SEC 2000*, volume 175 of *IFIP Conference Proceedings*, pages 321–330. Kluwer, 2000.
9. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
10. D. Chaum. Blind signatures for untraceable payments. In *Proc. CRYPTO 82*, pages 199–203. Plenum Press, 1982.
11. D. Chaum. Elections with unconditionally secret ballots and disruption equivalent to breaking rsa. In *Proc. EUROCRYPT 88*, pages 177–182. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 330.

12. D. Chaum. Security without identification: Transaciton systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 2001.
13. D. Chaum and E. Van Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 547.
14. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multiauthority secret ballot elections with linear work. In *Proc. EUROCRYPT 96*, pages 72–83. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1070.
15. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election schemes. In *Proc. EUROCRYPT 97*, pages 103–118. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1233.
16. I. Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In *Proc. CRYPTO 88*, pages 328–335. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 403.
17. Y. Desmedt and Y. Frankel. Threshold cryptosystem. In *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1989. Lecture Notes in Computer Science No. 435.
18. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale election. In *AUSCRYPT 91*, pages 244–260. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 718.
19. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. EUROCRYPT 2000*, pages 539–556. Springer-Verlag, 2000. Lecture Notes in Computer Science No. 1807.
20. B. Lee and K. Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Proc. ICISC 2002*, pages 389–406. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2587.
21. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1758.
22. National Bureau of Standards FIPS Publication 186. *Digital Signature Standard*, 1994.
23. M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto. An improvement on a practical secret voting scheme. In *Proc. Information Security 1999*, pages 225–234. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1729.
24. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Proc. EUROCRYPT 93*, pages 248–259. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 765.
25. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2248.
26. K. Sako and J. Kilian. Secure voting using partial compatible homomorphisms. In *Proc. CRYPTO 94*, pages 411–424. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 839.
27. A. Shamir. How to share a secret. In *Communications of the ACM*, volume 22(2), pages 612–613. ACM Press, 1979.
28. D. Wong, K. Fung, J. Liu, and V. Wei. On the RS-code construction of ring signature schemes and a threshold setting of RST. In *5th Intl. Conference on Information and Communication Security (ICICS 2003)*, pages 34–46. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2836.
29. F. Zhang and K. Kim. ID-Based blind signature and ring signature from pairings. In *Proc. ASIACRYPT 2002*, pages 533–547. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501.

A Background on E-Voting Schemes

In this section, we provide an overview of the security requirements of an e-voting scheme and specify the two major types of previously proposed e-voting schemes.

A.1 Security Requirements

According to a popular definition of a secure e-voting scheme given by Fujioka et al. in [18], there are seven requirements needed to fulfill: *completeness*, *soundness*, *privacy*, *unreusability* (detecting double voting), *eligibility*, *fairness* and *verifiability*. Completeness requires that all valid votes should be counted correctly. Soundness requires that all invalid votes should not be counted. Privacy means that all votes should be kept secret until all votes have been collected and are ready to count. Unreusability prevents any voter to vote twice or more. Eligibility prevents unauthorized entities to vote. Fairness requires that nothing can affect the result. Verifiability ensures that the voting result is publicly verifiable.

Recent researches suggest some additional requirements. One is receipt-free voting systems [5, 19]. Such a system prevents a voter from claiming the authorship of a particular vote. Another requirement is non-transferability. In most of the e-voting systems, the voting right can be transferred because the authentication document is irrelevant to the voter. A non-transferable e-voting system ensures the transfer of the voting right is equivalent to the transfer of all the secret information owned by the voter. This requirement is considered in [8].

A.2 Classification: The Two Types

Previous e-voting systems can mainly be classified into two types [18]. In the first type, each voter sends the ballot to a trusted third party, the Voting Center, in an encrypted form. In the second type, each voter sends the ballot to the Voting Center through an anonymous channel [9, 24].

Type 1 In the Voting phase, voters send their votes with their signatures to a public bulletin which acts as the Voting Center. Votes are encrypted with the public key of a trusted third party, say the Tally, using homomorphic encryption schemes. The homomorphic encryption scheme allows the Tally to sum up the votes and obtain the final result without decrypting each individual vote.
There are several advantages. First, double voting is prevented since voters sign their encrypted votes which are publicly verifiable in the bulletin. Second, no interaction with the voters is required for the Registry to define the group of eligible voters. Hence the registration phase is simplified. Third, the Tally outputs the vote result without decrypting each individual vote. Hence the vote of each voters is protected from being known by others.

However, the system leaks information on who has voted and who has not. In addition, the Tally needs to be trusted for protecting privacy. In order to reduce the risk, secret sharing techniques [27] or threshold cryptosystems [17] are suggested to use with this type of e-voting systems. Another drawback is that although homomorphic encryption protocols work efficiently for "yes/no" type ballot, it becomes inefficient when it applies to 1-out-of-n type. It is even less practical when there is a large election scale or a t-out-of-n type of votes are conducted for t being close to $n/2$.
Examples of this type of e-voting systems are [4, 26, 14, 15, 19, 2].

Type 2 In Voting phase, a voter sends a vote to the Voting Center through an anonymous channel which can be established easily in practice. The anonymous channel protects the identity of the voter. It is more practical for large scale election since the communication and computation overheads are reasonable even if the number of voters is large [18].
However, as the channel is anonymous, special mechanisms are needed to prevent or detect double voting and to check the eligibility of a voter. Hence interaction with voters is necessary in the Registration phase to have the Registry dispatch some token or voting pass to each eligible voter. Blind signature is usually used. Examples of this type are [11, 18, 23].

B Theorem 1 (Unforgeability) Proof Sketch

Parameters p, q, g are fixed throughout this paper, and omitted from notations. Assume PPT adversary $\mathcal{A}$ can forge (1,n)-ring signature with non-negligible probability, i.e.

$$\Pr[\mathcal{A}(1^k, L, H_1, H_2, \mathcal{S}) \to (m, \sigma) : \mathcal{V}(L, m, \sigma, H_1, H_2)] > \frac{1}{Q_1(k)}$$

for some polynomial Q_1. Note $\mathcal{S}$ is a signing oracle which performs signature generation according to our (1,n)-ring signature scheme and returns a valid signature, upon $\mathcal{A}$'s query, other than the one $\mathcal{A}$ eventually produces. Assume $\mathcal{A}$ makes no more than q_H queries to the random oracles H_1 and H_2 combined, and it makes no more than q_S queries to the signing oracle $\mathcal{S}$, where q_H and q_S are no more than polynomially growing. The independent random oracles H_1 and H_2 produces random outcomes, except to maintain consistencies among duplicated queries. Note that $\mathcal{S}$ also makes queries to H_1 and H_2, and consistencies between its queries and $\mathcal{A}$'s queries are maintained. We construct a PPT simulator (i.e. reduction master) $\mathcal{M}$ which calls (i.e. enslaves) $\mathcal{A}$ and solves DLP of at least one of the public keys in L with non-negligible probability. Denote $L = E_1||\cdots||E_n$, and denote the forged signature by

$$\sigma = (c_1, s_1, \cdots, s_n, y_0)$$

where it satisfies the Verification including the following n equations:

$$\begin{aligned} c_{i+1} &= H_1(L, y_0, m, g^{s_i} y_i^{c_i}, h^{s_i} y_0^{c_i}), \text{ for } 1 \le i \le n-1; \\ c_1 &= H_1(L, y_0, m, g^{s_n} y_n^{c_n}, h^{s_n} y_0^{c_n}) \end{aligned}$$

The master $\mathcal{M}$ will invoke $\mathcal{A}$ with constructed inputs, receive and process outputs from $\mathcal{A}$, and invoke $\mathcal{A}$ multiply depending on $\mathcal{A}$'s outputs from previous invocations. In the random oracle model, $\mathcal{M}$ flips the coins for the random oracles H_1 and H_2 and record queries to the oracles.

Consider each invocation of $\mathcal{A}$ to be recorded on a Turing simulation-run transcript tape. Some transcripts produce successful ring signature forgeries. Others do not.

Let $\mathbf{E}$ denote the event that each of the n queries corresponding to the n Verification queries have been included in the q_H queries $\mathcal{A}$ made to the random oracles, or in the queries made by the signing oracle on behalf of $\mathcal{A}$ in its q_S signing queries. In the event $\bar{\mathbf{E}}$, $\mathcal{M}$ needs to flip additional coins in order to Verify $\mathcal{A}$'s signature forgery. Then the probability of c_1 satisfying the (final) Verification equation is less than $1/(q - q_H - nq_S)$ because $\mathcal{A}$ can only guess the outcomes of queries used in Verification that he has not made. Therefore

$$\begin{aligned}\frac{1}{Q_1(k)} &< \Pr[\mathbf{E}]\Pr[\mathcal{A} \text{ forges}|\mathbf{E}] + \Pr[\bar{\mathbf{E}}]\Pr[\mathcal{A} \text{ forges}|\bar{\mathbf{E}}] \\ &\leq \Pr[\mathbf{E}]\Pr[\mathcal{A} \text{ forges}|\mathbf{E}] + 1 \cdot (\frac{1}{q - q_H - nq_S})\end{aligned}$$

and

$$\Pr[\mathbf{E} \text{ and } \mathcal{A} \text{ forges}] \geq \frac{1}{Q_1(k)} - (\frac{1}{q - q_H - nq_S})$$

The probability of $\mathcal{A}$ returning a valid signature and having already queried the random oracles with n queries used in Verification is essentially $1/Q_1(k)$.

Therefore, in each $\mathcal{A}$ transcript which produced a valid signature, there exists n queries to H_1, denoted by $X_{i_1}, \cdots, X_{i_n}$, $1 \leq i_1 < \cdots < i_n$, such that they match the n queries made in Verification. This happens with each transcript that $\mathcal{A}$ successfully produces a valid signature, with negligible exceptions. Queries to random oracles H_1 and H_2 made by the signing oracle $\mathcal{S}$ when it responds to $\mathcal{A}$'s requests to sign have a negligible effect.

In a successful signature forgery σ by $\mathcal{A}$, consider the set of all queries made by $\mathcal{A}$ that were used (including duplicate queries) in Verification. Let $X_{i_1}, \cdots, X_{i_n}$ denote the first appearance of each of the queries used in Verification, $i_1 < \cdots < i_n$. Let π be such that

$$X_{i_n} = H_1(L, y_0, m, g^{s_{\pi-1}} y_{\pi-1}^{c_{\pi-1}}, h^{s_{\pi-1}} y_0^{c_{\pi-1}})$$

in Verification. We call π the *gap* of σ.

We call a successful forgery σ by $\mathcal{A}$ a (ℓ, π)-forgery if $i_1 = \ell$. I.e. the first appearance of all Verification-related queries is the ℓ-th query and the gap equals π. Queries made by $\mathcal{S}$ to random oracles on behalf of $\mathcal{A}$ are counted. There exist ℓ and π, $1 \leq \ell \leq q_H$, $1 \leq \pi \leq n$, such that the probability $\mathcal{A}$ produces (ℓ, π)-forgery is no less than $1/(n(q_H + nQ_S)Q_1(k))$.

In the following, $\mathcal{M}$ will do a rewind-simulation for each value of ℓ and π.

In the rewind-simulation for a given (ℓ, π), $\mathcal{M}$ first invokes $\mathcal{A}$ to obtain its output and its Turing transcript $\mathcal{T}$. $\mathcal{M}$ computes the output and the transcript to determine whether they form a successful (ℓ, π)-forgery. If not, abort. Otherwise continue. This can be done in at most polynomial time because $\mathcal{M}$ records queries made by $\mathcal{A}$ to the random oracles. The transcript $\mathcal{T}$ is rewound to the ℓ-th query and given to $\mathcal{A}$ for a rewind-simulation to generate transcript $\mathcal{T}'$. New coin flips independent of those in $\mathcal{T}$ are made for all queries subsequent to the ℓ-th query while maintaining consistencies with the prior queries. $\mathcal{T}$ and $\mathcal{T}'$ use the same code in $\mathcal{A}$. The ℓ-th query, common to $\mathcal{T}$ and $\mathcal{T}'$, is denoted

$$H_1(L, m, y_0, g^u, h^v).$$

$\mathcal{M}$ knows g^u and h^v but not u or v at the time of the rewind. After $\mathcal{A}$ returns the output from the rewind simulation, $\mathcal{M}$ proceeds to compute the DL of y_π.

Let H_ℓ denote the common prefix of $\mathcal{T}$ and $\mathcal{T}'$ whose length is exactly up to the ℓ-th query. Let

$$\begin{aligned}
\epsilon_{\ell,\pi}(H_\ell) &= \sum_{\mathcal{T}:H_\ell \text{ prefixes } \mathcal{T}, \mathcal{T} \ (\ell,\pi)\text{-forgers}} \Pr[\mathcal{T}]/\Pr[H_\ell] \\
&= \Pr[\mathcal{T}(\ell,\pi)\text{-forges}|H_\ell \text{ prefixes } \mathcal{T}] \\
&= \Pr[\mathcal{T}'(\ell,\pi)\text{-forges}|H_\ell \text{ prefixes } \mathcal{T}']
\end{aligned}$$

Note that

$$\begin{aligned}
&\Pr[H_\ell] = \sum_{H_\ell \text{ prefixes } \mathcal{T}} \Pr[\mathcal{T}] \\
&\sum_{\ell,\pi} \sum_{H} \epsilon_{\ell,\pi}(H) \geq 1/Q_1(k) \\
&\Pr[H_\ell \text{ prefixes } \mathcal{T}'] = \frac{\epsilon_{\ell,\pi}(H_\ell)\Pr[H_\ell]}{\sum_H \epsilon_{\ell,\pi}(H)\Pr[H]}
\end{aligned}$$

Then

$$\begin{aligned}
&\Pr[\mathcal{T}' \ (\ell,\pi)\text{-forges}] \\
&= \sum_{H_\ell} \Pr[H_\ell \text{ prefixes } \mathcal{T}']\Pr[\mathcal{T}' \ (\ell,\pi)\text{-forges}|H_\ell \text{ prefixes } \mathcal{T}'] \\
&= \sum_{H_\ell} \left(\frac{\epsilon_{\ell,\pi}(H_\ell)\Pr[H_\ell]}{\sum_H \epsilon_{\ell,\pi}(H)\Pr[H]}\right) \cdot \epsilon_{\ell,\pi}(H_\ell) \\
&= \frac{\langle \epsilon_{\ell,\pi}(H_\ell)^2 \rangle}{\langle \epsilon_{\ell,\pi}(H_\ell) \rangle} \\
&\geq \langle \epsilon_{\ell,\pi}(H_\ell) \rangle
\end{aligned}$$

The last inequality is well-known in probability theory.

Remark: The above proves the technical lemma critical to the proof concerning ROS (Rewind-on-Success) lemma. It depends on the well-known moment

inequality $\langle \chi^2 \rangle \geq \langle \chi \rangle^2$ from probability theory. Alternatively, the heavy-row lemma or the forking lemma can be used to prove a similar technical lemma for our rewind simulation, albeit with inferior constant. The main difference is that our rewinding is *adaptive* – it rewinds only on successful transcripts $\mathcal{T}$. In the traditional forking lemma or heavy-row lemma, $\mathcal{T}$ is indiscriminately rewound whether it is a success or not. More details about our "ROS lemma" are in Appendix F.

The tape $\mathcal{T}$ and a rewind-simulation tape $\mathcal{T}'$ produce two (ℓ, π)-forgery signatures with

$$\begin{aligned}
g^u &= g^{s_\pi} y_\pi^{c_\pi} = g^{s_\pi + x_\pi c_\pi} \bmod p_\pi, \text{ from } \mathcal{T} \\
h^v &= h^{s_\pi} y_0^{c_\pi} = h^{s_\pi + r_\pi c_\pi} \bmod p_0, \text{ from } \mathcal{T} \\
g^u &= g^{s'_\pi} y_\pi^{c'_\pi} = g^{s'_\pi + x_\pi c'_\pi} \bmod p_\pi, \text{ from } \mathcal{T}' \\
h^v &= h^{s'_\pi} y_0^{c'_\pi} = h^{s'_\pi + r_\pi c'_\pi} \bmod p_0, \text{ from } \mathcal{T}'
\end{aligned}$$

where $y_0 = h^{r_\pi} \bmod q_0$. Solve to obtain

$$x_\pi = \frac{s'_n - s_n}{c_n - c'_n} \bmod q_\pi \text{ and } r_\pi = \frac{s'_n - s_n}{c_n - c'_n} \bmod p_0 \tag{1}$$

The reduction master $\mathcal{M}$ solves for x_π based on recorded and computed c_π, s_π, $\bar{c}_\pi$, $\bar{s}_\pi$ as shown above. There exists (ℓ, π) such that

$$\langle \epsilon_{\ell,\pi}(H_\ell) \rangle \geq \frac{1}{n(q_H + nq_S)} \frac{1}{Q_1(k)}$$

Therefore $\mathcal{M}$ achieves a solution in at least one of its runs for all possible values of (ℓ, π), $1 \leq \ell \leq q_H + nq_S$, $1 \leq \pi \leq n$, with the above probability. The complexity of $\mathcal{M}$ is no more than $n(q_H + nq_S)$ times that of $\mathcal{A}$. The probability of success of $\mathcal{M}$ is at least $(\frac{1}{n(q_H+nq_S)Q_1(k)})^2$, still non-negligible. Desired contradiction. Theorem 1 is proved.

Remark: The following improved simulator $\mathcal{M}$ can achieve complexity no more than twice that of $\mathcal{A}$, and success probability at least $1/(n(q_H+nq_S)Q_1^2(k))$. $\mathcal{M}$ invokes $\mathcal{A}$ to obtain a transcript $\mathcal{T}$. If $\mathcal{M}$ confirms $\mathcal{T}$ is a successful (ℓ, π)-forgery then rewind to ℓ-th query. Otherwise abort. an (ℓ, π)-forgery with probability $\langle \epsilon_{\ell,\pi} \rangle$. The probability of $\mathcal{M}$ succeeding both in the first invocation and the rewind simulation is

$$\begin{aligned}
&\sum_{\ell,\pi} \Pr[\mathcal{T}' \ (\ell,\pi)\text{-forges}, \mathcal{T} \ (\ell,\pi)\text{-forges}] \\
&= \sum_{\ell,\pi} \Pr[\mathcal{T}' \ (\ell,\pi)\text{-forges} | \mathcal{T} \ (\ell,\pi)\text{-forges}] \Pr[\mathcal{T} \ (\ell,\pi)\text{-forges}] \\
&= \sum_{\ell,\pi} \sum_{H_\ell} \Pr[\mathcal{T}' \ (\ell,\pi)\text{-forges} | H_\ell \text{ prefixes } \mathcal{T}', \mathcal{T} \ (\ell,\pi)\text{-forges}]
\end{aligned}$$

$$\cdot \Pr[\mathcal{T} \ (\ell,\pi)\text{-forges}|H_\ell \text{ prefixes } \mathcal{T}] \ \Pr[H_\ell]$$
$$= \sum_{\ell,\pi} \sum_{H_\ell} \epsilon_{\ell,\pi}(H_\ell)\epsilon_{\ell,\pi}(H_\ell)\Pr[H_\ell]$$
$$= \sum_{\ell,\pi} \langle \epsilon_{\ell,\pi}^2 \rangle \geq \frac{1}{n(q_H + nq_S)} (\sum_{\ell,\pi} \langle \epsilon_{\ell,\pi} \rangle)^2 \geq \frac{1}{n(q_H + nq_S)} \frac{1}{Q_1^2(k)}$$

C Proof Sketch of Theorem 2 (Signer-Ambiguity)

For simplicity, we prove for the case $t = 0$ only and drop the signing oracle $\mathcal{S}$ from the algorithm specification. Other cases are similar and it is trivial to argue that the signing oracle does not help break the signer ambiguity. The parameters p, q, g are fixed throughout this paper, and omitted from notations.

Assume $\mathcal{A}$ is a PPT adversary who can crack anonymity, i.e.

$$\Pr[\mathcal{A}(H_1, H_2, L, m, \sigma) = f_0(L, \hat{s}) : \text{ random } L, m; (\hat{s}, P) \leftarrow \mathcal{G}(1^k);$$
$$\sigma \leftarrow \mathcal{S}(\hat{s}, L, m); \mathcal{V}(L, m, \sigma) = 1] > \frac{1}{n} + \frac{1}{Q(k)}$$

for some polynomial Q. The we construct below, a PPT simulator $\mathcal{M}$ which can solve the Decisional Diffie-Hellman Problem (DDHP)

$$\Pr[\mathcal{M}(\alpha, \beta, \gamma) = b : \text{ random } \ell_0, \ell_1, \ell_2, \ell_0', \ell_1', \ell_2' \in_R \{1, \cdots, q-1\};$$
$$\alpha_0 = g^{\ell_0}, \beta_0 = g^{\ell_1}, \gamma_0 = g^{\ell_2}, \alpha_1 = g^{\ell_0'}, \beta_1 = g^{\ell_1' 1}, \gamma_1 = g^{\ell_0' \ell_1'}; b \leftarrow \{0,1\};$$
$$(\alpha, \beta, \gamma) = (\alpha_b, \beta_b, \gamma_b)] = \frac{1}{2} + \frac{1}{Q_2(k)}$$

for some polynomial Q_2.

In the simulation by $\mathcal{M}$, H_1 and H_2 are assumed to be random oracles. They generate random outputs, and do so independently. One additional assumption is that $H_2(L) = \beta$. Note that β is random and therefore H_2 remains random. There is another assumption about queries to H_1, in that it has a similar "fixed point".

To simulate, $\mathcal{M}$ computes a "candidate signature" σ' as follows before calling $\mathcal{A}$:

1. Randomly generate n, L, m. Generate $\pi \in_R \{1, \cdots, n\}$. Randomly generate $\ell \in_R \{1, \cdots, q-1\}$ and let $c'_\pi = g^\ell$. Set $y_\pi = \alpha$.
2. For $i = \pi', \cdots, n, 1, \cdots, \pi' - 1$, randomly generate s_i and compute
$$c_{i+1} = H_1(L, \gamma, m, g^{s_i} y_i^{c_i}, h^{s_i} \gamma^{c_i})$$
querying the oracle H_1 along the way.
3. Set the oracle outcome
$$H_1(L, \gamma, m, g^{s_{\pi-1}} y_{\pi-1}^{c_{\pi-1}}, h^{s_{\pi-1}} \gamma^{c_{\pi-1}}) = c_\pi$$

4. $\sigma' = (c_1, s_1, \cdots, s_n, \gamma)$

In the simulation, $\mathcal{M}$ calls $\mathcal{A}$ with H_1, H_2, L, m, and σ'. The oracles H_1 and H_2 will produce random outcomes upon $\mathcal{A}$'s queries, except $H_2(L) = \beta$ and $H_1(L, \gamma, m, g^{s_i} y_i^{c_i}, h^{s_i} \gamma^{c_i})$ are predetermined, for $1 \leq i \leq n$, to maintain consistencies on duplicated inputs with queries already made by $\mathcal{M}$. By the random oracle model, these pre-dispositions affect the randomness of H_1 and H_2 only negligibly.

The adversary $\mathcal{A}$ returns an integer j, $1 \leq j \leq n$, to $\mathcal{M}$. By convention, $\mathcal{A}$ returns 0 if it cannot identify a signer. The simulator $\mathcal{M}$ outputs 1 if $j = \pi$; outputs 0 if $j = 0$; and outputs 1/0 with equal probability otherwise. Then

$$\begin{aligned}
&\Pr[\mathcal{M}(\alpha, \beta, \gamma) = b | b = 1] \\
&= \Pr[\mathcal{M}(\alpha, \beta, \gamma) = b | b = 1, \mathcal{A}(H_1, H_2, L, m, \sigma') = \pi] \\
&\quad + \Pr[\mathcal{M}(\alpha, \beta, \gamma) = b | b = 1, \mathcal{A}(H_1, H_2, L, m, \sigma') \neq \pi, \neq 0] \\
&\geq 1 \cdot (\frac{1}{n} + \frac{1}{Q(k)}) + \frac{1}{2}(1 - \frac{1}{n} - \frac{1}{Q(k)}) \\
&\geq \frac{1}{2} + \frac{1}{2n} + \frac{1}{2Q(k)}
\end{aligned}$$

If b=0, then all signers are symmetric from $\mathcal{A}$'s perspectives, and $\mathcal{A}$ can do no better than random guessing. Averaging over $\mathcal{M}$'s random choice of π, $1 \leq \pi \leq n$, we obtain

$$\begin{aligned}
&\Pr[\mathcal{M}(\alpha, \beta, \gamma) = b | b = 0] \\
&= \Pr[\mathcal{M}(\alpha, \beta, \gamma) = b | b = 0, \mathcal{A}(H_1, H_2, L, m, \sigma') = \pi] \\
&\quad + \Pr[\mathcal{M}(\alpha, \beta, \gamma) = b | b = 0, \mathcal{A}(H_1, H_2, L, m, \sigma') \neq \pi] \\
&\geq 0 \cdot \frac{1}{n} + \frac{1}{2}(1 - \frac{1}{n})
\end{aligned}$$

Combining, we have $\Pr[\mathcal{M}(\alpha, \beta, \gamma) = b] \geq \frac{1}{2} + \frac{1}{4Q(k)}$. Therefore $\mathcal{M}$ solves DDHP with probability non-negligibly over 1/2. Desired contradiction. Signer-ambiguity is proved.

Remark on the randomness of H_1 and H_2: $H_2(L) = \beta$. But L and β are random, therefore so is H_2. The randomness is H_1 is more complicated. The set of all random oracles can be partitioned into q parts, according to the value of i in

$$i = H_1(L, \gamma, m, g^{s_{\pi-1}} y_{\pi-1}^{c_{\pi-1}}, h^{s_{\pi-1}} \gamma^{c_{\pi-1}}) - c_\pi$$

Averaging over all q parts, $\mathcal{A}$ has a non-negligibly probability above $1/n$ of producing results. However, there could be pedagogical examples where $\mathcal{A}$ is a PPT adversary over random H_1 but not the kind of oracle randomized over q partitions. Under the random oracle model, we assume the oracle H_1 constructed above by $\mathcal{M}$ behaves like random oracles, and PPT adversary $\mathcal{A}$ can compute results given H_1 in place of a true random oracle.

D Proof Sketch of Theorem 3 (Linkability)

If PPT $\mathcal{A}$, in possession of one of the asymmetric decryption keys among x_1, $\cdots$, x_n, can produce two unlinkable signatures with non-negligible probability ϵ, then PPT $\mathcal{M}$ can compute the discret log of two public keys among y_1, $\cdots$, y_n.

Proof Sketch: We follow notations in the Proof of Theorem 1. Consider that $\mathcal{A}$ produces a pair of signatures (σ, σ') that are (ℓ, π)-forgeries and (ℓ', π')-forgeries, respectively, with Turing transcript $\mathcal{T}$. Denote $\sigma = (c_0, s_1, \cdots, s_n, y_0)$ and $\sigma' = (c_0', s_1', \cdots, s_n', y_0')$. Let $y_0 = h^{r_\pi}$ and $y_0' = h^{r_\pi'}$ denote the *linkability tag* in the two signatures respectively, where r_π and r_π' are yet to be determined.

Suppose $\mathcal{A}$ always produces, with negligible exception, signature pairs with $\pi = \pi'$. By rewinding $\mathcal{T}$ to just before the ℓ-th query and re-simulate, $\mathcal{A}$ can produce with non-negligible probability another signature pair $(\tilde{\sigma}, \tilde{\sigma}')$, which are (ℓ, π)-forgeries and (ℓ'', π'')-forgeries respectively with transcript $\mathcal{T}'$. Denote $\tilde{\sigma} = (\tilde{c}_0, \tilde{s}_1, \cdots, \tilde{s}_n, \tilde{y}_0)$ and and $\tilde{\sigma}' = (\tilde{c}_0', \tilde{s}_1', \cdots, \tilde{s}_n', \tilde{y}_0')$. By a derivation similar to that which led to Equation (1), we find that

$$x_\pi = r_\pi = \frac{\tilde{s}_n - s_n}{c_n - \tilde{c}_n} \bmod q$$

where $y_0 = h^{r_\pi}$ and $y_\pi = g_\pi^x \bmod p$.

Then a second rewind simulation. By rewinding $\mathcal{T}$ to the ℓ'-th query and re-simulate, $\mathcal{M}$ obtains with non-negligible probability, another signature pair $(\hat{\sigma}, \hat{\sigma}')$ where the second signature is an (ℓ', π)-forgery. A similar argument shows

$$r_\pi' = x_\pi' \bmod q$$

where $y_0' = h^{r_\pi'}$ and $y_\pi = g^{x_\pi'} \bmod p$. Therefore, $x_\pi = x_\pi' = r_\pi = r_\pi' \bmod q$ and $y_0 = y_0'$. The two signatures σ and σ' are linked. But then $y_0 = y_0' = h^{x_\pi} \bmod p$ and the two signatures are linked.

Therefore, $\mathcal{A}$ can generate with non-negligible probability signature pairs with different gaps, i.e. $\pi \neq \pi'$. In particular, there exists (ℓ, π, ℓ', π') satisfying $1 \leq \pi < \pi' \leq n$, $1 \leq \ell, \ell' \leq q_H + nq_S$, such that $\mathcal{A}$ can generate, with non-negligible probability, signatures pairs that are (ℓ, π)-forgery and (ℓ', π')-forgery respectively. Then the rewind simulation technique can be used, twice, to show that $\mathcal{M}$ can enslave $\mathcal{A}$ to compute the discret log of y_π and $y_{\pi'}$.

In the above, $\mathcal{A}$ is assumed to query the random oracles no more than q_H times and the signing oracle no more than q_S times. Theorem 3 is secure against adaptive chosen-plaintext attackers by the above proof.

E Proof Sketch of Theorem 4 (Culpability)

Test if $h^{x_\pi} = y_0$. The proof is trivial and omitted.

F ROS (Rewind-on-Success Lemma)

In a typical rewind simulation, a reduction master $\mathcal{M}$ invokes an adversarial algorithm $\mathcal{A}$ to obtain a certain output. The computation proceedings, including the coin flip sequence, are recorded on a Turing tape $\mathcal{T}$. Then $\mathcal{M}$ rewinds $\mathcal{T}$ to a certain header position H, and redo the simulation from then onward to obtain another Turing transcript $\mathcal{T}'$. The two transcripts $\mathcal{T}$ and $\mathcal{T}'$ use the same code $\mathcal{A}$, have the same coin flips up to H, but have different coin flips after H. After both simulations are done, $\mathcal{M}$ processes $\mathcal{T}$ and $\mathcal{T}'$ to obtain answers.

Assume the probability of success of $\mathcal{A}$, which equals the probability of success of $\mathcal{T}$, is ϵ. The forking lemma, or the heavy-row lemma, can be used to show that the probability of success of $\mathcal{T}'$, which equals the probability of success of $\mathcal{A}$ with given transcript header H, is at least $\epsilon/4$. The complexity of $\mathcal{M}$ is essentially twice that of $\mathcal{A}$, and the probability of success of $\mathcal{M}$ is at least $\epsilon^2/4$.

In our proof, we used the technique based on *ROS (Rewind-on-Success) lemma*. $\mathcal{M}$ invokes $\mathcal{A}$ once to obtain transcript $\mathcal{T}$. Then $\mathcal{M}$ processes $\mathcal{T}$ to conditionally decide the next step. Then $\mathcal{M}$ rewinds $\mathcal{T}$ to a adaptively-chosen header H and re-simulate $\mathcal{A}$ to obtain $\mathcal{T}'$. Finally, $\mathcal{M}$ processes $\mathcal{T}$ and $\mathcal{T}'$ to obtain answers.

We only used a very simple adaptive mechanism: *rewind on success*. $\mathcal{M}$ rewind to the ℓ-th query if produces a valid (ℓ, π)-forgery. abort if $\mathcal{T}$ is a failure. Assume the probability of success of $\mathcal{T}$ is ϵ, then the probability of success of $\mathcal{T}'$, which equals the probability of success of $\mathcal{A}$ with a given header H which was selected because it was the header of a successful $\mathcal{T}$, is also ϵ. Then the complexity of $\mathcal{M}$ is essentially twice that of $\mathcal{A}$, and its probability of success is essentially ϵ^2. The proof of this probability bound depends on a well-known moment inequality in probability theory: $\langle X^2 \rangle \geq \langle X \rangle^2$.

The success rate bound in our adaptive rewind simulation is 4 times better than that of the (indiscriminate) rewind simulation. More importantly, the success rate of subsequent rewound simulations remain the same as the success rate of the first simulation. This fact makes adaptive rewind simulation potentially more powerful than (indiscriminate) rewind simulation in proof scenarios where multiple layers of rewinding are nested.

In the following, we review proofs of the forking lemma for the (indiscriminate) rewind simulation and the new *ROS (Rewind-on-Success) lemma* for our adaptive rewind simulation.

Lemma 1 (ROS (Rewind-on-Success) Lemma). *Let $\mathcal{M}$ invokes $\mathcal{A}$ to obtain transcript $\mathcal{T}$. If $\mathcal{T}$ is successful, then $\mathcal{M}$ rewinds $\mathcal{T}$ to a header H and re-simulate $\mathcal{A}$ to obtain transcript $\mathcal{T}'$. If $\Pr[\mathcal{T}$ succeeds$] = \epsilon$, then $\Pr[\mathcal{T}'$ succeeds$] = \epsilon$.*

Sketch of Proof: All probabilities are with respect to all coin flips. For each H is a suitable domain of prefixes, let

$$\epsilon_H = \sum_{\mathcal{T}:H \text{ prefixes } \mathcal{T}, \mathcal{T} \text{ succeeds}} \Pr[\mathcal{T}]$$

$$= \Pr[\mathcal{T} \text{ succeeds} | H \text{ prefixes } \mathcal{T}]$$
$$= \Pr[\mathcal{T}' \text{ succeeds} | H \text{ prefixes } \mathcal{T}']$$

Then

$$\begin{aligned}\Pr[\mathcal{T}' \text{ succeeds}] &= \sum_H \Pr[H \text{ prefixes } \mathcal{T}'] \Pr[\mathcal{T}' \text{ succeeds} | H \text{ prefixes } \mathcal{T}'] \\ &= \sum_H \frac{\Pr[H]\epsilon_H}{\sum_{H'} \Pr[H']\epsilon_{H'}} \epsilon_H \\ &= \frac{\langle \epsilon_H^2 \rangle}{\langle \epsilon_H \rangle} \geq \langle \epsilon_H \rangle = \epsilon\end{aligned}$$

Q.E.D.

(Indiscriminate) Forking Lemma: Use notations as above. There exists a set of prefixes $\mathbf{H}$, such that $\sum_{H \in \mathbf{H}} \epsilon_H \geq \epsilon/2$ and $\Pr[\mathcal{T}' \text{ succeeds} | H \text{ prefixes } \mathcal{T}'] \geq \epsilon/2$ for each $H \in \mathbf{H}$.

Sketch of Proof: Let $\mathbf{H} = \{H_1, H_2, H_3, \cdots\}$ denote the set of all possible prefixes arranged in a way such that

$$\epsilon_{H_1} \geq \epsilon_{H_2} \geq \epsilon_{H_3} \geq \cdots$$

Let i be the integer such that

$$\sum_{j<i} \Pr[H_j] < \epsilon/2 \leq \sum_{j \leq i} \Pr[H_j]$$

We assert that $\epsilon_{H_i} \geq \epsilon/2$ which proves the lemma. Suppose the opposite that $\epsilon_{H_i} < \epsilon/2$. Then

$$\begin{aligned}\sum_j \epsilon_{H_j} \Pr[H_j] &= \sum_{j<i} \epsilon_{H_j} \Pr[H_j] + \sum_{j \geq i} \epsilon_{H_j} \Pr[H_j] \\ &= \sum_{j<i} \Pr[H_j] + \epsilon_{H_i} \sum_{j \geq i} \Pr[H_j] \\ &< \epsilon/2 + \epsilon_{H_i} < \epsilon/2 + \epsilon/2\end{aligned}$$

But the left-hand-side of the above equation equals ϵ, a desired contradiction.