This is the full version of the extended abstract which will appear in: The 9th Australasian Conference on Information Security and Privacy (ACISP 2004), Springer-Verlag, 2004. LNCS.

Linkable and Anonymous Signature for Ad Hoc Groups

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

March 15, 2004

Abstract. We present a 1-out-of-n group signature scheme that satisfies three major properties. (1) Anonymity, or signer-indistinguishability. (2) Linkability: That two signatures by the same signer can be linked. (3) Spontaneity: No group manager or TTP, no setup stage such as in secret sharing. The scheme has many applications where maximum privacy is protected, copycatting is prevented, and impromptu linkup is prerogative. For example: whistle blowing, e-voting, anonymous membership authentication, and digital rights management. We show under the random oracle model that our scheme satisfies all the three properties. Additionally, we present an e-voting system based on our scheme. In the system, there is no involvement of voters in the registration phase and the voting phase is only one-round. We also present a new proof of the core lemma in rewind simulation. This is the literature's third such proofs, after the forking lemma [30] and the heavy-row lemma [28]. Our proof is the most accessible of the three, relying on only the moment inequality from elementary probability theory. And our proof has the best simulation efficiency of the three. Threshold extensions of our scheme are also proposed.

1 Introduction

The paper introduces a 1-out-of-n group signature scheme which has some interesting properties and many useful applications. The three major properties are: (1) Anonymity, or signer-indistinguishability. (2) Linkability: That two signatures by the same signer can be linked. (3) Spontaneity: No group manager or TTP, no setup stage such as in secret sharing.

A 1-out-of-n group signature scheme allows any member of a group of n signers to generate a signature such that any public verifier can determine if the signature is generated by a group member. By anonymity, the signature hides the identity of its author in a group so that no one can tell who the actual signer is among the n group members. By linkability, the scheme allows one to determine whether two signatures of the group are generated by the same signer.

Early 1-out-of-n anonymous group signature schemes [14, 15, 9, 10] often require a powerful group manager or a TTP (Trusted Third Party) which has the power to revoke the identity of the actual signer. In addition, the group is pre-defined with a group public key and there is a setup stage which requires collaboration of all the group members. Spontaneity is to remove any external party and setup stage from a 1-out-of-n group signature scheme. A spontaneously formed group has no manager, no group public key and no setup stage at all. By assuming that everyone in a system has an independently-generated public key associated with and the key is already published, a signer can impromptually and arbitrarily conscript $n-1$ diversion members into an ad hoc group without their awareness, and generate a group signature without their participation. The signature can be verified based on the n published public keys of the group members, and it essentially establishes that the actual signer is a member of the group.

The concept of spontaneity is similar to the formation of an ad hoc group. In this paper, we present a linkable, spontaneous, and anonymous group (LSAG) signature scheme for ad hoc groups.

The LSAG signature scheme has many applications where maximum or near maximum privacy is required and impromptu linkup is prerogative. One scenario is on protecting the anonymity of the so-called "Deepthroat" who helped divulge the Watergate scandal that led to the resignation of the US President Richard Nixon some thirty years ago. Deepthroat wants to leak a sequence of secrets to the press from time to time without giving out his identity. At the same time, the press also want to ensure that Deepthroat is really working for the White House and the sequence of secrets is really given out by the same person. This is the scenario where both anonymity and linkability are required. Other applications of LSAG signature schemes include detecting double voting while maintaining anonymity of the voters in an e-voting system; providing anonymous membership authentication with restriction on simultaneous log-ins; and implementing digital rights management without compromising users' privacy.

1.1 Contributions

In this paper, we present the first 1-out-of-n group signature scheme which simultaneously achieves linkability, spontaneity, and anonymity. All these properties of our scheme are proven secure under the random oracle model. Our scheme has many applications, especially where maximum or near maximum privacy protection, copycat prevention, and impromptu linkup are desired or required. An e-voting system based on our linkable, spontaneous and anonymous group signature scheme is proposed. In the system, there is no involvement of voters in the registration phase and the voting phase is only one-round. The Tally is just a public bulletin so that everyone can do the counting.

Additionally, we present a new proof of the core lemma in rewind simulation. This is the literature's third such proofs, after the forking lemma [30] and the heavy-row lemma [28]. Our proof is the most accessible of the three, relying on only the moment inequality from elementary probability theory. And our proof has the best simulation efficiency of the three.

Besides the linkable, spontaneous and anonymous 1-out-of-n group signature scheme, we describe how to convert it into a suite of t-out-of-n threshold extensions.

(Organization) 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 of a LSAG signature scheme in Sec. 3. Our LSAG signature scheme is then described in Sec. 4 and the security analysis is given in Sec. 5. In Sec. 6, we construct an e-voting system using our group signature schemes. The paper is concluded in Sec. 7.

2 Related Work

(Ring Signatures) Since the notion of ring signature was first formalized by Rivest, et al. in [31] and similar idea appeared in the literature earlier in [16], many other ring signature schemes have been proposed [1, 7, 8, 35, 34, 25]. A ring signature is an anonymous spontaneous group signature which allows a signer to impromptually and spontaneously form a group of n members by conscripting $n-1$ diversion signers and generate a signature such that anyone can tell if the signature is generated by a group member without identifying who the actual signer is. All of these schemes are unlinkable. In Sec. 1, we explain that linkability could be a very useful feature which can help detect double-voting in an e-voting system, leak a sequence of secrets, prevent copycatting, and implementing efficient anonymous membership management. Also, all the schemes are exculpable. Exculpability 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. This feature may become problematic 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.

In our linkable, spontaneous and anonymous group (LSAG) signature scheme, culpability is an additional feature which provides an option to determine the authorship of the signature from a private key corresponding to one of the public keys associated to it. Hence by culpability, it can be shown that an anonymous group signature is indeed created by a particular signer after his private key is revealed. We 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 [31]. However, culpability is not known to any of the current ring signature schemes. In general, culpability implies claimability but not vice versa.

(E-voting Schemes) The first e-voting scheme was proposed by Chaum in [12]. 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 our LSAG signature scheme, we propose a new e-voting system which has the following properties. There is no involvement of voters in the registration phase and the voting phase is only one-round. The Tally is just a public bulletin so that everyone can do the counting. In Appendix B, 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 The Security Model

A (1-out-of-n) linkable spontaneous anonymous group (LSAG) signature scheme is a triple of algorithms ($\mathcal{G}$, $\mathcal{S}$, $\mathcal{V}$).

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

$$\mathcal{V}(1^k, L, m, \mathcal{S}(1^k, \hat{s}, L, m)) = 1.$$

We omit the denotation of the security parameter as an input of the algorithms in the rest of the paper.

3.1 Two Models of Adaptive Chosen Message Attack for Unforgeability of LSAG Signature Schemes

A secure LSAG signature scheme should be able to thwart signature forgery under certain security models. Under the model of adaptive chosen message attack, existential unforgeability [22] means that given the public keys of all group members but not any of the corresponding private keys, an adversary, who can even adaptively obtain valid signatures for any messages that he wishes, cannot forge a signature for any message m.

We address one interesting subtlety of this model for LSAG signature schemes. This subtlety for signature schemes at large has already been in many cryptographers' minds for quite a while. However, it has not been clearly explored in the setting of ring or LSAG signature schemes before.

In the security model of Rivest, et al. [31], the adversary who targets to forge a signature of message m subjects to the condition that m has *never* been presented to a signing oracle[3]. Hence given a ring or LSAG message-signature pair (m', σ') with respect to a public-key set L, if another pair (m, σ) with respect to L is forged such that $m \neq m'$, then this is considered as a successful forgery in the model. However given a signature σ of m, if the adversary forges another signature σ', $\sigma' \neq \sigma$, of the same message m, then this is not considered as a successful forgery in the model. In particular, it is not considered as a successful forgery even if the corresponding public-key sets of σ and σ' are different. Hence a ring or LSAG signature scheme has been proven to be existential unforgeable under this security model may still allow anyone to add or remove public keys associated to a given signature. This case is not considered in the security model. Comparing to another model described below, we refer to this model as the model of *restricted* adaptive chosen message attack.

Another model first specified by Abe, et al. [1] allows the adversary who targets to forge a signature of message m to query the signing oracle even with m. The only restriction is that the forged message-signature pair of the adversary should not appear in the transcript of the signing oracle. This model is stronger than the restricted one in the sense that existential unforgeability in this model does not allow the list of public-keys corresponding to an SAG signature of m to be altered.

The RST scheme of Rivest, et al. has been shown to be existential unforgeable in the model of *restricted* adaptive chosen message attack under the ideal cipher model [31]. It is also believed to be secure in the stronger model of adaptive chosen message attack without modifying the scheme. In fact, we find that most of the existing ring signature schemes are secure in both models. This observation leads us to find examples of which it can make distinction between these two models. Luckily, we find that the ring signature scheme of Boneh, et al. [6] based on bilinear maps is proven secure against *restricted* adaptive chosen message attack but not against the stronger attack. In Appendix A, we show that their scheme is forgeable under the stronger model by describing how to let anyone add an arbitrary number of public keys to given signatures.

In this paper, we adopt the stronger model of adaptive chosen message attack for defining the existential unforgeability of a LSAG signature scheme. The following definition is similar to that of [1] which also captures the adaptive chosen public-key attack.

[3] The signing oracle receives a message m' and a set L' of public keys and returns a signature σ' such that $1 \leftarrow \mathcal{V}(L', m', \sigma')$.

Definition 1 (Existential Unforgeability against Adaptive Chosen Message Attack). *Let $\mathcal{L} = \{P_1, \cdots, P_n\}$ be a set of n public keys. Each public key in $\mathcal{L}$ is generated by $\mathcal{G}$ with security parameter k. Let $\mathcal{SO}(L', m')$ be a signing oracle that accepts as inputs any $L' \subseteq \mathcal{L}$ and any message m', produces a signature σ' such that $\mathcal{V}(L', m', \sigma') = 1$. A LSAG signature scheme is unforgeable if, for any PPT (probabilistic polynomial-time algorithm) $\mathcal{A}$ with signing oracle $\mathcal{SO}$ such that $(L, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{SO}}(1^k, \mathcal{L})$, its output satisfies $\mathcal{V}(L, m, \sigma) = 1$ only with negligible probability in k. Restriction is that $L \subseteq \mathcal{L}$ and (L, m, σ) should not be found in the set of all oracle queries and replies between $\mathcal{A}$ and $\mathcal{SO}$.*

Some additional features can be introduced into this model as well. For example, the model can be changed to allow the queries of the signing oracle to specify which private key corresponding to the set of public keys should be used in generating the signatures. This model captures the scenario that the adversaries are given not only some signatures but also the identities of the actual signers. We call this model an adaptive chosen insider attack. However this model cannot be used in defining the signer anonymity below. We give more details in the following.

3.2 Signer Anonymity

By signer anonymity, we require that given a LSAG signature with respect to a group of keys, it is infeasible to identify which private key is used to generate the signature. In this paper, we focus on providing near maximum privacy in such a way that the chance of guessing correctly which key of a set of n keys generated by $\mathcal{G}$ with security parameter k is used to generate a given signature is negligibly greater than $1/n$ provided that the signing key is chosen at random and the adversary only knows the public keys (but not the private keys). If t private keys of the set are also known by the adversary, where $t < n$, but the signing private key is not one of them, then the success probability of the adversary should still be negligibly greater than $1/(n-t)$.

The notion of signer anonymity above is different from those of [31, 1] which require that revealing private keys does not reduce the level of signer anonymity. However, we specify the weaker form for supporting the property of culpability. On the other hand, the original notion of [31, 1] should be used if exculpability is required.

On the security model, our LSAG signature scheme described later in this paper can be shown to be signer anonymous under the same chosen message attack model described in Def. 1. The additional feature in the signer anonymity model is that the adversary can adaptively choose up to t group members to corrupt by asking their private keys. We require that the best way of figuring out the actual signer's identity of a LSAG signature σ of message m with respect to a set L of n public keys is to take a wild guess. The condition is that the private key of the actual signer is still a secret even though a few other private keys (up to t) of the n keys specified by L may be chosen by the adversary after (L, m, σ) is given.

Note that the adaptive chosen insider attack model described in Sec. 3.1 cannot be applied to LSAG signer anonymity. This can be seen more clearly when considering the property of linkability which will be formalized shortly. Suppose the signing oracle for the adversary of signer anonymity also receives an index for the actual signer from the query and the oracle has to return a signature generated by the actual signer. By linkability defined below, the query 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.

3.3 Linkability

Two LSAG signatures are *linked* with respect to the same set L of public keys if they are generated using the same private key. Formally,

Definition 2 (Linkability). *Let $L = \{P_1, \cdots, P_n\}$ be a set of n public keys. Each public key in L is generated by $\mathcal{G}$ with security parameter k. A LSAG signature scheme is linkable if there exists a PPT $\mathcal{F}_1$ which 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

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

for all sufficiently large k, any $\pi_1, \pi_2 \in \{1, \cdots, n\}$, any messages m_1, m_2 and any $\sigma_1 \leftarrow \mathcal{S}(\hat{s}_{\pi_1}, L, m_1)$, $\sigma_2 \leftarrow \mathcal{S}(\hat{s}_{\pi_2}, L, m_2)$. ϵ is some negligible function.

The algorithm $\mathcal{F}_1$ outputs 1 if it thinks the two signatures are linked, that is, are signed by the same group member. Otherwise, it outputs 0.

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.

3.4 Culpability

A LSAG signature scheme is *culpable* if given a message-signature pair and the private key of a group member, anyone can determine if that group member is the actual 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 3 (Culpability). *Let $L = \{P_1, \cdots, P_n\}$ be a set of n public keys. Each key is generated by $\mathcal{G}$ with security parameter k. A LSAG signature scheme is culpable if there exists a PPT $\mathcal{F}_2$ which 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

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

for all sufficiently large k, any message m, 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, $1 \leq j \leq n$. ϵ is some negligible function.

The algorithm $\mathcal{F}_2$ outputs 1 if it thinks member j is the signer (culprit). For any valid culpable LSAG 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. This requirement of culpability induces the chance of guessing correctly the identity of the actual signer in the definition of signer anonymity (Sec. 3.2).

4 A LSAG 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 LSAG 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 $c_1 \overset{?}{=} H_1(L, \tilde{y}, m, z'_n, z''_n)$. 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 LSAG signatures associate with the same set of parameters, which is L in the description above. In general, a LSAG 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. F.1 when describing a suite of threshold extensions of our LSAG 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]. First, we present a new proof of the core lemma in rewind simulation. This is the literature's third such proofs, after the forking lemma [30] and the heavy-row lemma [28]. Our proof is the most accessible of the three, relying on only the moment inequality from elementary probability theory. And our proof has the best simulation efficiency of the three.

5.1 ROS (Rewind-on-Success Lemma)

In a typical rewind simulation [20, 30, 28], a reduction master $\mathcal{M}$ invokes an adversarial algorithm $\mathcal{A}$ to obtain a certain output. The simulation proceeding, including the coin flip sequences, are recorded on a simulation transcript tape $\mathcal{T}$. $\mathcal{M}$ rewinds $\mathcal{T}$ to a certain header position H, and redo the simulation from then onward to obtain another 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 [30], or the heavy-row lemma [28], 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 an adaptively-chosen header H and re-simulates $\mathcal{A}$ to obtain $\mathcal{T}'$. Finally, $\mathcal{M}$ processes $\mathcal{T}$ and $\mathcal{T}'$ to obtain answers.

In the unforgeability proof below, we only use a very simple adaptive mechanism: *rewind on success*. Suppose there are q_H queries of H_1 and H_2 altogether in one simulation. $\mathcal{M}$ rewinds 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 rewind 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-simulates $\mathcal{A}$ to obtain transcript $\mathcal{T}'$. If* $\Pr[\mathcal{T} \text{ succeeds}] = \epsilon$, *then* $\Pr[\mathcal{T}' \text{ succeeds}] = \epsilon$.

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

$$\begin{aligned}\epsilon_H &= \sum_{\mathcal{T}:H \text{ prefixes } \mathcal{T}, \mathcal{T} \text{ succeeds}} \Pr[\mathcal{T}] \\ &= \Pr[\mathcal{T} \text{ succeeds} | H \text{ prefixes } \mathcal{T}] \\ &= \Pr[\mathcal{T}' \text{ succeeds} | H \text{ prefixes } \mathcal{T}']\end{aligned}$$

Then

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

$$= \sum_H \frac{\Pr[H]\epsilon_H}{\sum_{H'} \Pr[H']\epsilon_{H'}} \epsilon_H$$
$$= \frac{\langle \epsilon_H^2 \rangle}{\langle \epsilon_H \rangle} \geq \langle \epsilon_H \rangle = \epsilon$$

□

(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 $\Pr[\mathcal{T}'$ succeeds$|H$ 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.

□

5.2 Security of Our LSAG Signature Scheme

According to the following theorem, which is shown in Appendix C, our LSAG signature scheme is existentially unforegable against adaptive chosen message and chosen public-key attacks defined in Def. 1.

Theorem 1 (Existential Unforgeable). *Let $\mathcal{SO}$ be a signing oracle which returns valid signatures according to our LSAG 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. 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*

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

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 D, 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 $i_j \neq \pi$, $1 \le j \le t$,*

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

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, we obtain the following theorem with proof given in Appendix 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, $1 \le i \le n$, 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, $1 \le i \le n$, be the corresponding private keys. Let $\sigma_L = (c_1, s_1,\cdots,s_n,\tilde{y})$ where $c_1, s_i \in \mathbb{Z}_q$, $1 \le i \le n$ 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.*

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

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 [26]. 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 LSAG 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 $C_i \overset{?}{=} h(Vote_i, b_i)$ for $1 \le i \le n$.

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 LSAG 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 the LSAG signature scheme is claimable, the e-voting system above is not receipt-free. Our systems can be modified to support receipt-freeness by using a tamper-resistant randomizer [24]. 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 no involvement of the voters is needed. Detecting double voting can be done by employing the linkability property of the underlying LSAG signature scheme.

7 Conclusions

In this paper, we present the first LSAG signature scheme which simultaneously achieves linkability, spontaneity, and anonymity. Another feature called culpability is also introduced and realized. All these properties of our scheme are proven secure under the random oracle model. Our scheme has many applications, especially where maximum or near maximum privacy protection, copycat prevention, and impromptu linkup are desired or required. In Appendix F, we describe how to convert our LSAG scheme into a suite of t-out-of-n threshold extensions.

An e-voting system based on our LSAG signature scheme is proposed. In the system, there is no involvement of voters in the registration phase and the voting phase is only one-round (that is, vote-and-go). The Tally is just a public bulletin so that everyone can do the counting.

Additionally, we present a new proof of the core lemma in rewind simulation. This is the literature's third such proofs, after the forking lemma [30] and the heavy-row lemma [28]. Our proof is the most accessible of the three, relying on only the moment inequality from elementary probability theory. And our proof has the best simulation efficiency of the three.

There are many interesting problems that are to be solved. For example, it is interesting to design a LSAG signature scheme which still maintains unconditional anonymity. In addition, LSAG signature schemes may also be constructed based on other hard problems such as factorization. To obtain more scalable e-voting systems, much shorter and more efficient LSAG signature schemes are to be devised.

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. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. EUROCRYPT 2003*, pages 416–432. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2656.
7. 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.
8. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. full version. http://www.di.ens.fr/~bresson, 2002.
9. J. Camenisch. Efficient and generalized group signatures. In *Proc. EUROCRYPT 97*, pages 465–479. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1233.
10. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proc. CRYPTO 97*, pages 410–424. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.
11. 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.

12. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
13. 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.
14. 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.
15. L. Chen and T. Pedersen. New group signature schemes. In *Proc. EUROCRYPT 94*, pages 171–181. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 950.
16. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. CRYPTO 94*, pages 174–187. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 839.
17. 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.
18. 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.
19. Y. Desmedt and Y. Frankel. Threshold cryptosystem. In *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1989. Lecture Notes in Computer Science No. 435.
20. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. CRYPTO 86*, pages 186–199. Springer-Verlag, 1987. Lecture Notes in Computer Science No. 263.
21. 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.
22. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, April 1988.
23. 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.
24. 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.
25. J. K. Liu, V. K. Wei, and D. S. Wong. A separable threshold ring signature scheme. In *ICISC 2003*, pages 7–22. Springer-Verlag, 2004. To appear in Lecture Notes in Computer Science series.
26. National Bureau of Standards FIPS Publication 186. *Digital Signature Standard*, 1994.
27. 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.
28. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *Proc. CRYPTO 98*, pages 354–369. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1462.
29. 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.

30. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. EUROCRYPT 96*, pages 387–398. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1070.
31. 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.
32. 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.
33. A. Shamir. How to share a secret. In *Communications of the ACM*, volume 22(2), pages 612–613. ACM Press, 1979.
34. 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.
35. 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 Security of a Bilinear SAG Signature Scheme Under Different Security Models

In Sec. 5 of [6], Boneh et al. devised a bilinear SAG signature scheme. We first review their scheme and show that their scheme allows anyone to add new public keys to given signatures.

Let G_1, G_2 and G_T be three (multiplicative) cyclic groups of prime order p. Let g_1 and g_2 be the generators of G_1 and G_2, respectively. Let $\psi : G_2 \to G_1$ be a computable isomorphism with $\psi(g_2) = g_1$. Let e be a computable bilinear map $e : G_1 \times G_2 \to G_T$ with the following properties: (1) Bilinear: for all $u \in G_1$, $v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$; (2) Non-degenerate: $e(g_1, g_2) \neq 1$. These properties imply that for any $u_1, u_2 \in G_1$, $v \in G_2$, $e(u_1u_2, v) = e(u_1, v) \cdot e(u_2, v)$; and for any $u, v \in G_2$, $e(\psi(u), v) = e(\psi(v), u)$.

For each user i, $1 \leq i \leq n$, pick random $x \in_R \mathbb{Z}_p$ and compute $v = g_2^x$. The user's public key is $v \in G_2$. The user's secret key is $x \in \mathbb{Z}_p$. Suppose π, $1 \leq \pi \leq n$, be the index of the actual signer. Let $H : \{0,1\}^* \to G_1$ be a hash function. The signature generation for message $m \in \{0,1\}^*$ proceeds as follows.

1. For each user i, $1 \leq i \leq n$ and $i \neq \pi$, randomly generate $a_i \in_R \mathbb{Z}_p$ and compute $\sigma_i = g_1^{a_i}$.
2. Solve the following system to obtain σ_π;
$$H(m) = h = \sigma_\pi^{x_\pi} \psi(\prod_{i=1, i\neq\pi}^{n} v_i^{a_i})$$
3. The signature is $\sigma = (\sigma_1, \cdots, \sigma_n) \in G_1^n$.

To verify the signature, check if $e(h = H(m), g_2) = \prod_{i=1}^{n} e(\sigma_i, v_i)$.

Add a New Public Key to a Given Signature. Given a bilinear signature $\sigma = (\sigma_1, \cdots, \sigma_n)$, we can add a public key, v_{n+1}, by setting the following.

1. For $i = 1, \cdots, n-1$, set $\sigma'_i \leftarrow \sigma_i$.
2. Set $\sigma'_n \leftarrow \sigma_n \cdot \psi(v_{n+1}^r)$ where $r \in_R \mathbb{Z}_p$.
3. Set $\sigma'_{n+1} \leftarrow \psi(v_n^{-r})$.

The new signature is $\sigma' = (\sigma'_1, \cdots, \sigma'_{n+1})$. The technique can be generalized to let anyone add an arbitrary number of public keys to given signatures.

This scheme is provably secure under random oracle model that it is existential unforgeable against *restricted* chosen message attack. However it is no longer true under the stronger model of adaptive chosen message attack (See Sec. 3.1 for details).

B 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.

B.1 Security Requirements

According to a popular definition of a secure e-voting scheme given by Fujioka et al. in [21], 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, 23]. 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 [11].

B.2 Classification: The Two Types

Previous e-voting systems can mainly be classified into two types [21]. 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 [12, 29].

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 [33] or threshold cryptosystems [19] 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, 32, 17, 18, 23, 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 [21].

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 [13, 21, 27].

C Proof of Theorem 1 (Unforgeability)

Proof. Parameters p, q, g are fixed throughout this paper, and omitted from notations. Let $\mathcal{L}$ be a set of public keys of which each key is generated according to the description in Sec. 4. Assume PPT adversary $\mathcal{A}$, which makes at most q_H queries to H_1 and H_2 combined and at most q_S queries to $\mathcal{SO}$, can forge (1,n)-LSAG signature with non-negligible probability, i.e.

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

for some polynomial Q_1, and q_H, q_S and $|\mathcal{L}|$ being no more than polynomially growing with respect to the security parameter k. Note $\mathcal{SO}$ is a signing oracle which returns valid signatures, upon $\mathcal{A}$'s query, other than the one $\mathcal{A}$ eventually produces. The independent random oracles H_1 and H_2 produces random outcomes, except to maintain consistencies among duplicated queries. Note that $\mathcal{SO}$ 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. the reduction master) $\mathcal{M}$ which calls $\mathcal{A}$ and solves DLP of at least one of the public keys in $\mathcal{L}$ with non-negligible probability. Denote a subset of $\mathcal{L}$ by $L = \{y_1, \cdots, y_n\}$, and denote the forged signature with respect to L by

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

where it satisfies the Verification (Sec. 4.2) 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 may invoke $\mathcal{A}$ for multiple times 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 simulation transcript tape. Some transcripts produce successful signature forgeries. Others do not.

Let $\mathbf{E}$ be 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 at most $1/(q - q_H - nq_S)$ because $\mathcal{A}$ can only guess the outcomes of queries used in Verification that he has not made. Therefore

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

and

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

Hence the probability of $\mathcal{A}$ returning a forged signature and having already queried the random oracles for all the n queries used in Verification is essentially greater than $1/Q_1(k)$ as $\frac{1}{q - q_H - nq_S}$ is negligibly small.

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}, $1 \le i_1 < \cdots < i_n$, 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 $\mathcal{SO}$ 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, $1 \le \ell \le q_H$, $1 \le \pi \le n$, 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

$$\Pr[H_\ell] = \sum_{H_\ell \text{ prefixes } \mathcal{T}} \Pr[\mathcal{T}]$$

$$\sum_{\ell,\pi} \sum_H \epsilon_{\ell,\pi}(H) \ge 1/Q_1(k)$$

$$\Pr[H_\ell \text{ prefixes } \mathcal{T}'] = \frac{\epsilon_{\ell,\pi}(H_\ell)\Pr[H_\ell]}{\sum_H \epsilon_{\ell,\pi}(H)\Pr[H]}$$

Then

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

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 (Sec. 5.1). 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 [28] or the forking lemma [30] 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.

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

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

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

$$\begin{aligned}
&\sum_{\ell,\pi}\Pr[\mathcal{T}'\ (\ell,\pi)\text{-forges}, \mathcal{T}\ (\ell,\pi)\text{-forges}]\\
&\quad=\sum_{\ell,\pi}\Pr[\mathcal{T}'\ (\ell,\pi)\text{-forges}|\mathcal{T}\ (\ell,\pi)\text{-forges}]\Pr[\mathcal{T}\ (\ell,\pi)\text{-forges}]\\
&\quad=\sum_{\ell,\pi}\sum_{H_\ell}\Pr[\mathcal{T}'\ (\ell,\pi)\text{-forges}|H_\ell \text{ prefixes } \mathcal{T}', \mathcal{T}\ (\ell,\pi)\text{-forges}]\\
&\qquad\qquad\qquad\cdot\Pr[\mathcal{T}\ (\ell,\pi)\text{-forges}|H_\ell \text{ prefixes } \mathcal{T}]\ \Pr[H_\ell]\\
&\quad=\sum_{\ell,\pi}\sum_{H_\ell}\epsilon_{\ell,\pi}(H_\ell)\epsilon_{\ell,\pi}(H_\ell)\Pr[H_\ell]\\
&\quad=\sum_{\ell,\pi}\langle\epsilon_{\ell,\pi}^2\rangle \geq \frac{1}{n(q_H+nq_S)}\Big(\sum_{\ell,\pi}\langle\epsilon_{\ell,\pi}\rangle\Big)^2 \geq \frac{1}{n(q_H+nq_S)}\frac{1}{Q_1^2(k)}
\end{aligned}$$

□

D Proof of Theorem 2 (Signer-Ambiguity)

Proof. 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.

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

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

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

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. □

E Proof 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 discrete log of two public keys among y_1, $\cdots$, y_n.

Proof. 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 discrete 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. □

F Threshold Extensions

A (t, n)-threshold LSAG 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 (t, n)-threshold LSAG 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 LSAG 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 LSAG 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 $(1, n)$-LSAG 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.

F.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 LSAG 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 LSAG signature can later be reduced arbitrarily by removing some $(1,n)$-LSAG 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 LSAG signature by $\sigma_L^{(t,n)}(m,t)$.

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 LSAG signature scheme. We denote this variant by $\sigma_r^{(t,n)}(m,t)$. Similar idea can also be applied to our basic scheme for generating unlinkable but culpable $(1,n)$-LSAG signatures. The security analysis will be given in the full paper.