Linkable Spontaneous Anonymous Group 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

Abstract. We present a linkable spontaneously anonymous group (LSAG) signature scheme (alternatively known as linkable ring signature scheme) satisfying the following three properties. (1) Anonymity, or signer indistinguishability. (2) Linkability: That two signatures by the same signer can be linked. (3) Spontaneity: No group secret, therefore no group manager or group secret sharing setup. We reduce the security of our scheme to well-known problems under the random oracle model. Using the scheme, we construct a new efficient one-round e-voting system which does not have a registration phase. We also present a new efficient reduction of famous rewind simulation lemma which only relies on elementary probability theory. Threshold extensions of our scheme are also presented.

30th April, 2004.

1 Introduction

We present a 1-out-of-n group signature scheme which satisfy three properties: (1) Anonymity, or signer-indistinguishability. (2) Linkability: That two signatures by the same signer can be linked. (3) Spontaneity: No group secret, and thus no group manager or secret-sharing setup stage.

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. They are typically achieved by generating a group secret and then share it out using centralized methods (with group manager) or distributed methods (with a all-n-member secret sharing setup stage) [13, 14, 8, 9].

* The work described in this paper was fully supported by a grant from CityU (Project No. 7200005).

In Cramer, et al. [15] and Rivest, et al. [29] a new paradigm for achieving 1-out-of-n group signature is presented. Any single user/signer can conscript the public keys of $n - 1$ other users to form a group of n members. Then a signature can be generated by that single signer which can be publicly verified to be signed by one of the n group members. But the group formation and the signature generation are both *spontaneous*, meaning that no participation or even knowledge of the other $n - 1$ users are needed. The 1-out-of-n signature generated this way is also anonymous (signer indistinguishable). Furthermore the anonymity is unconditional (information-theoretic) and exculpable (signer anonymous even after subpoenaing all n secret keys and all communications transcripts). Rivest, et al. [29] formalized this kind of signature to be 'Ring Signature' because their construction of the signature forms a ring structure. Some other works in the literature also call this kind of signature (with the above properties) 'Ring Signature' although some of them may not have a ring structure for their construction. In alternative terminology, we call this kind of signature 'Spontaneous Anonymous Group (SAG) Signature' as they fulfill Spontaneity, Anonymity and Group properties regardless of the construction structure.

This paradigm of SAG signature schemes have found many applications where maximum or near maximum privacy protection is needed such as whistle blowing. It has also found applications in group signatures for *ad hoc groups* where group secret setup and maintenance are too expensive due to frequent membership joins and drops. This kind of structure raises new challenges for security issues as the instance of ad hoc groups are not dependent on any particular network infrastructure. For example, a group of users who spontaneously decide to communicate some sensitive data which do not involve any trusted third party for participation while privacy need to be preserved at the same time. SAG signatures are perfectly suited to such situation due to its spontaneity property. Additional works on this topic includes [1, 6, 7, 33, 32, 24].

In this paper, we present linkable SAG (LSAG) signatures. Linkability means two signatures by the same actual signer can be identified as such, but the signer remains anonymous.

There are several applications of the new LSAG signatures. (1) Linked whistle-blowing. SAG signature can be used to leak secret information [29]. However, some media or journalists may not believe what the secret leaker tells and may think that he is telling lies. They may only believe two or three different sources with the same piece of information. In this case, SAG signature cannot be used as one cannot distinguish whether two different SAG signatures are generated by the same signer or not. Instead, LSAG should be used to allow people to verify that two given signatures are in fact generated by two distinct signers. (2) A new efficient e-voting system can be built upon LSAG signatures. This new e-voting system has efficiency advantage in eliminating one of three typical phases in e-voting systems. Typical e-voting systems have three major phases: Registration, Voting, Vote Opening and Tallying Phases. Our e-voting eliminates the Registration phase, and thus achieve great efficiency and user friendliness.

The tradeoff is in increased bandwidth requirements. We are going to discuss this in the later section of this paper.

1.1 Contributions

Our contributions consists of (1) the first linkable SAG (ring) signature. (2) application of this new paradigm. (3) a new efficient and accessible reduction of the forking lemma.

We add the property of linkability to ring signatures by presenting the first linkable ring signature scheme. In alternative terminology, our scheme is the first LSAG (Linkable Spontaneous Anonymous Group) signature scheme. Unlike the original ring signature which is exculpable, our scheme is culpable. 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. In general, culpability implies claimability but not vice versa.

Our scheme is proven existentially unforgeable against adaptive chosen-plaintext, adaptive chosen-public-key attackers provided DL is hard, under the random oracle model (ROM). Signer anonymity (resp. linkability) is reduced to DDH (resp. DL) under ROM.

We present an application of the new LSAG signature paradigm: an e-voting system without registration phase.

Additionally, we present a new efficient reduction of the forking lemma in rewind simulation. The reduction efficiency is four times better than that of heavy-row lemma [26] per rewind. Our reduction proof is the most accessible in the literature, relying only on the moment inequality from elementary probability theory.

We also present threshold versions of our LSAG signature.

(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 LSAG signature schemes. The paper is concluded in Sec. 7.

2 Related Work

(SAG Signatures) The Spontaneous Anonymous Group (SAG) Signature was presented by Cramer, et al. [15] and Rivest, et al. [29]. Later on, a separable and ID-based version were given by Abe, et al. [1] and Zhang, et al. [33] respectively. Threshold version have been proposed in [6, 7, 32, 24] as well. All of these schemes are unlinkable.

(E-voting Schemes) The first e-voting scheme was proposed by Chaum in [11]. 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.

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 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 polynomial time 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 polynomial time algorithm which takes as inputs security parameter k, private key $\hat{s}$, a list 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 a polynomial time algorithm which accepts as inputs security parameter k, a list 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 Existential Unforgeability against Adaptive Chosen-Plaintext, adaptive Chosen-Public-Key Attackers

A secure LSAG signature scheme should be able to thwart signature forgery under certain security models. Under the model of adaptive chosen-plaintext attack, existential unforgeability [21] 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.

In this paper, we adopt the stronger model of adaptive chosen-plaintext 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.

Definition 1 (Existential unforgeability against adaptive chosen plaintext, adaptive chosen-public-key attackers). *Let $\mathcal{SO}(L', m')$ be a signing oracle that accepts as inputs any public key list $L' = \{PK_1, \cdots, PK_{n'}\}$ and any message m', produces a signature σ' such that $\mathcal{V}(L', m', \sigma') = 1$. An LSAG signature scheme is* **existentially unforgeable (against adaptive chosen-plaintext and adaptive chosen public-key attackers)** *if, for any PPT (probabilistic polynomial-time) algorithm $\mathcal{A}$ with signing oracle $\mathcal{SO}$ such that $(L, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{SO}}(L)$, for a list L of n public keys chosen by $\mathcal{A}$, its output satisfies $\mathcal{V}(L, m, \sigma) = 1$ only with negligible probability. Note that (L, m, σ) should not correspond to any query-response pair to the signing oracle.*

3.2 Signer Ambiguity

Signer ambiguity means that it is infeasible to identify which private key was actually used in generating a given LSAG signature. Formally,

Definition 2 (Signer Ambiguity). *An LSAG signature scheme is signer ambiguous if for any PPT algorithm E, on inputs of any message m, any list L of n public keys, any set of t private keys $\mathcal{D}_t = \{\hat{s}_1, \cdots, \hat{s}_t\} \subset L$, and any valid signature σ on L and m generated by user π, we have*

$$\Pr[E(m, L, \mathcal{D}_t, \sigma) \to \pi] \begin{cases} \in (\frac{1}{n-t} - \frac{1}{Q(k)}\ ,\ \frac{1}{n-t} + \frac{1}{Q(k)}), \\ \quad \text{if } \hat{s}_\pi \notin \mathcal{D}_t \text{ and } 0 \le t < n-1 \\ > 1 - \frac{1}{Q(k)},\ \text{o.w.} \end{cases}$$

for any polynomial $Q(k)$.

Remark: Note that this implies culpability. That is, the actual signer is able to prove to others that he is the actual signer by revealing his signing key.

The definition of signer ambiguity for LSAG signature scheme differs from that of other SAG signature schemes such as [29, 1, 6, 32, 24] in two aspects: (1) The former signer-ambiguity comes with culpability, and the later is signer-ambiguous with exculpability because the latter achieves the probability of finding out the actual signer to $1/n$, independent of t and whether the private key of the actual signer is included or not; (2) The signer-ambiguity is now reducible to a hard problem (to DDHP as we shall see soon) for the former while the latter case reduces to information-theoretic security.

3.3 Linkability

Two LSAG signatures with the same public key list L are *linked* if they are generated using the same private key. Formally,

Definition 3 (Linkability). *Let $L = \{P_1, \cdots, P_n\}$ be a list of n public keys. A LSAG signature scheme is linkable if there exists a PPT algorithm $\mathcal{F}_1$ which outputs $1/0$ with*

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

and

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

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 for sufficiently large k.

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.

Remark: Note that linkability defined above requires that the list of public keys L is fixed while there is no additional requirement on the messages of the signatures. Any two LSAG signatures with different L cannot be linked.

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. 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 list of n public keys.

4.1 Signature Generation

Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

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

Remarks: Other methods of generating h can be used, provided it is computable by the public verifier and that it does not damage security. We shall see examples in some of our applications.

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 list 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 list 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 list 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 Threshold Extension

It is trivial to extend our 1-out-of-n LSAG signature to t-out-of-n threshold LSAG signature by using the linkable property. Each signer simply generates his own 1-out-of-n LSAG signature and concatenate together to form the t-out-of-n threshold LSAG signature. By linkability, verifier can ensure the signature is generated by t distinct signers.

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].

5.1 Security of Our LSAG Signature Scheme

We give security theorems of our LSAG signature.

Theorem 1 (Existential Unforgeability). *Our LSAG signature scheme is existentially unforgeable against adaptive chosen-plaintext and adaptive chosen public-key attackers provided the DLP (Discrete Logarithm Problem) is hard, under the random oracle model.*

Corollary 1. *Assume there exists a PPT algorithm $\mathcal{A}$ which makes at most q_H queries to random oracles H_1 and H_2 combined and at most q_S queries to signing oracle $\mathcal{SO}$ defined in Def. 1 such that*

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

for some polynomial Q_1 and sufficiently large k. Then, there exists a PPT algorithm which can solve the Discrete Logarithm Problem (DLP) with probability at least $(\frac{1}{n(q_H + nq_S)Q_1(k)})^2$ and expected running time no more than twice that of $\mathcal{A}$.

Theorem 2 (Signer Ambiguity). *Our LSAG signature is signer ambiguous provided DDHP (Decisinal Diffie-Hellman Problem) is hard, in the random oracle model.*

Theorem 3 (Linkability). *Our LSAG signature is linkable provided DLP is hard, in the random oracle model.*

Proofs of the Theorems are in Appendices B, C, and D.

5.2 The ROS (Rewind-on-Success Lemma)

In the proof of the above theorems, we have used a new proof of the core lemma in rewind simulation. This is the literature's third such proofs, after the forking lemma [28] and the heavy-row lemma [26]. We call it the ROS (Rewind-on-Success) Lemma. 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. Details of the ROS Lemma will be addressed in Appendix E.

6 A New E-voting System without Registration Phase

We present a new e-voting scheme without a registration phase. The scheme uses LSAG signatures.

Most e-voting schemes that appeared after the paper by [20] consists of three phases: Registration phase, Voting phase and Vote-Open phase. In the Registration phase, voters obtain untraceable blank ballots from one or more Registry. The untraceability is often achieved by using blind signature techniques. In the Voting phase, each voter casts its filled ballot, in encrypted/blinded form, to one or more Voting Centers. Some e-voting schemes require that all cast ballots (in blinded or encrypted form) be published so each voter can ensure his vote is included. Then one or more independent tallies opens the votes using deblinding parameters sent in by satisfied voters and compute the voting results.

Using LSAG signatures, we can construct a new e-voting system which contains only two phases: the Voting phase and the Vote Opening phase. There is no Registration phase. It is believed that the eliminating of one of three phases will result in large efficiency improvement.

Infrastructure assumptions: We have in mind a voter population where each voter has a published public key. There is a trustworthy list of all voters' public keys that can be downloaded from a reliable repository. For example, the list of all citizens' public keys published by the national government. Or the list of all voting members' public keys published by a society or a board. For the time being, we assume all voters have discrete log key pairs with the same discrete log parameters p, q, and g. The list of all voters' public keys is denoted L.

To initiate a voting event: To initiate a voting event, some messages are generated by reliable means. There are many centralized or distributed methods to do so. Then the messages are published. For simplicity, we consider a yes/no referendum. In this case, two messages M_{yes} and M_{no} are published.

Voting Phase: Each yes-voter send in a LSAG signature on message M_{yes} and public key list L. Each no-voter does same on message M_{no}.

Vote Opening Phase: It is a simple matter to count the yes-votes and the no-votes.

If anyone votes twice (or more), whether both times in the same column or otherwise, his votes will be linked and appropriate actions taken.

6.1 Discussions

The above represents the bare essence of an anonymous e-voting scheme based on LSAG signatures. The most striking distinction between this scheme and popular schemes in the literature is the lack of the Registration phase. Many details are left out. Here we discuss some of them.

(Anonymous Channel and Bulletin Board) In order to ensure voter anonymity, the send-in of LSAG signatures should be via anonymous channels. Furthermore, We assume that once the LSAG signature is sent in to a voting center, it will be published to a public bulletin and that public information cannot be altered further.

(Multiple Voting Events) The above scheme works for one voting. If there are future voting events, we will need to alter the value of $h = H_2(L)$ for each event. This can be accomplished by several methods. For example, using a different H_2 each event, or change to $h = H_2(L, eventlabel)$.

(Vote-and-Go) In comparison to most e-voting schemes in the literature, our scheme has the *vote-and-go* feature. That is, the voters are not involved in the Vote-Opening Phase.

(Receipt-freeness) A receipt-free e-voting system prevents a voter from claiming the authorship of a particular vote. This property deters vote-buying. The voter cannot provide evidence of compliance to vote buyers. 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 [23]. 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.

7 Conclusions

In this paper, we present the first LSAG 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, and impromptu linkup are desired or required.

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 [28] and the heavy-row lemma [26]. 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. LNCS Vol. 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. LNCS Vol. 2442.
7. 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.
8. J. Camenisch. Efficient and generalized group signatures. In *Proc. EUROCRYPT 97*, pages 465–479. Springer-Verlag, 1997. LNCS Vol. 1233.
9. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proc. CRYPTO 97*, pages 410–424. Springer-Verlag, 1997. LNCS Vol. 1294.
10. 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.
11. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

12. D. Chaum. Elections with unconditionally secret ballots and disruption equivalent to breaking rsa. In *Proc. EUROCRYPT 88*, pages 177–182. Springer-Verlag, 1988. LNCS Vol. 330.
13. D. Chaum and E. Van Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. LNCS Vol. 547.
14. L. Chen and T. Pedersen. New group signature schemes. In *Proc. EUROCRYPT 94*, pages 171–181. Springer-Verlag, 1994. LNCS Vol. 950.
15. 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. LNCS Vol. 839.
16. 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. LNCS Vol. 1070.
17. 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. LNCS Vol. 1233.
18. Y. Desmedt and Y. Frankel. Threshold cryptosystem. In *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1989. LNCS Vol. 435.
19. 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. LNCS Vol. 263.
20. 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. LNCS Vol. 718.
21. 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.
22. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. EUROCRYPT 2000*, pages 539–556. Springer-Verlag, 2000. LNCS Vol. 1807.
23. 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. LNCS Vol. 2587.
24. J. K. Liu, V. K. Wei, and D. S. Wong. A separable threshold ring signature scheme. In *ICISC 2003*, pages 12–26. Springer-Verlag, 2004. LNCS Vol. 2971.
25. 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. LNCS Vol. 1729.
26. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *Proc. CRYPTO 98*, pages 354–369. Springer-Verlag, 1998. LNCS Vol. 1462.
27. 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. LNCS Vol. 765.
28. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. EUROCRYPT 96*, pages 387–398. Springer-Verlag, 1996. LNCS Vol. 1070.
29. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. LNCS Vol. 2248.
30. K. Sako and J. Kilian. Secure voting using partial compatible homomorphisms. In *Proc. CRYPTO 94*, pages 411–424. Springer-Verlag, 1994. LNCS Vol. 839.
31. A. Shamir. How to share a secret. In *Communications of the ACM*, volume 22(2), pages 612–613. ACM Press, 1979.

32. 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. LNCS Vol. 2836.
33. F. Zhang and K. Kim. ID-Based blind signature and ring signature from pairings. In *Proc. ASIACRYPT 2002*, pages 533–547. Springer-Verlag, 2002. LNCS Vol. 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 [20], 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, 22]. 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 [10].

A.2 Classification: The Two Types

Previous e-voting systems can mainly be classified into two types [20]. 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 [11, 27].

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 [31] or threshold cryptosystems [18] 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, 30, 16, 17, 22, 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 [20].

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 [12, 20, 25].

B Proof of Theorem 1 (Unforgeability)

Proof. We prove by rewind simulation, together with the classification technique.

Parameters p, q, g are fixed throughout this paper, and omitted from notations. Let $\mathcal{L}$ be a list 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.

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

for some polynomial Q_1, and q_H and q_S 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.

The Signing Oracle: Given any message m, any sequence of public keys $L = (y_1, \cdots, y_n)$, the signing oracle $\mathcal{SO}$ generates a signature. $\mathcal{M}$ simulates $\mathcal{SO}$ to generate a signature without knowing any secret key but by back patching H as follows:

Assume without loss of generality, $H_2(L)$ has been queried before and $\mathcal{M}$ has simulated it by randomly picking r and returning $H_2(L) = g^r$. To simulate $\mathcal{SO}$, $\mathcal{M}$ randomly picks $\pi \in_R \{1, \cdots, n\}$, and randomly picks c_1, $\cdots$, c_n, s_1, $\cdots$, s_n. Compute $\tilde{y} = y_\pi^r$. Back patch to

$$H_1(L, \tilde{y}, m, g^{s_i} y_i^{c_i}, g^{r s_i} y_\pi^{r c_i}) = c_{i+1}$$

for $1 \le i \le n$ with the short-hand notation that subscript $n+1$ means subscipt 1.

Remark: The signature returned by $\mathcal{SO}$ looks just like one actually signed by signer π.

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

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

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

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

Then

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

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. Details of the ROS lemma will be addressed in Appendix E. 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 [26] or the forking lemma [28] 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

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

□

C Proof of Theorem 2 (Signer-Ambiguity)

Proof. We first prove the case that $0 \leq t < n - 1$ where $\hat{s}_\pi \notin \mathcal{D}_t$. The DL parameters p, q, g are fixed throughout the rest of this paper, unless explicitly stated otherwise.

Suppose there exists a PPT algorithm A, on inputs of a message m, a list L of n public keys, a set of t private keys $\mathcal{D}_t = \{\hat{s}_1, \cdots, \hat{s}_t\} \subset \mathcal{L}$, $0 \leq t < n - 1$, and a valid signature σ on L and m generated by user π, with probability

$$\Pr[A(m, L, \mathcal{D}_t, \sigma) \rightarrow \pi] > \frac{1}{n - t} + \frac{1}{Q(k)}$$

for some polynomial $Q(k)$ in the case of $\hat{s}_\pi \notin \mathcal{D}_t$, then we construct below, a PPT simulator $\mathcal{M}$ which can solve the Decisional Diffie-Hellman Problem (DDHP):

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

for some polynomial $Q_2(k)$. For simplicity, we prove for the case $t=0$. Other value of t where $0<t<n-1$ are similar and omitted.

We will prove by back patching to $H_2(L)=\beta$.

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 $1\le i\le n$, 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, $1\le j\le n$, 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

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

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, $1\le\pi\le$

n, we obtain

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

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.

Now we prove for the case of $t=n-1$ and $\hat{s}_\pi \notin \mathcal{D}_t$.

The last component of a valid signature is $\tilde{y}=h^{\hat{s}_\pi}$. After trying $h^{\hat{s}_i}$ for $\hat{s}_i \in \mathcal{D}_t$, $1\leq i\leq t$, one can conclude that user i, $1\leq i\leq t$, are not the actual signer. As the signature is unforgeable, it must be generated by one of the n users. Thus, the remaining user must be the actual signer.

For the case that $\hat{s}_\pi \in \mathcal{D}_t$, one can find out the fact that $\tilde{y}=h^{\hat{s}_\pi}$ easily and conclude that user π is the actual signer. □

D Proof of Theorem 3 (Linkability)

Proof. If PPT $\mathcal{A}$ can produce two unlinkable signatures with non-negligible probability ϵ, then there exists a simulator $\mathcal{M}$ which can compute the discrete log of two public keys among $y_1,\cdots,y_n$. Since $\mathcal{A}$ is in possession of at most one secret key, $\mathcal{M}$ will have solved a hard problem: the DLP.

The proof is by rewinding twice the simulation transcript at two suitable forks. 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 proved. □

E Details of the ROS (Rewind-on-Success) Lemma

In a typical rewind simulation [19, 28, 26], 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 [28], or the heavy-row lemma [26], 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

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

□

(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.

□