Generic Construction of Linkable Ring Signature

Xueli Wang [*] Yu Chen [†] Xuecheng Ma [‡]

Abstract

We propose a generic construction of linkable ring signature from any compatible ring signature scheme and one-time signature scheme. Our construction has both theoretical and practical interest. In theory, our construction gives the first generic transformation from ring signature to linkable ring signature, which brings at least two main benefits: first, the transformation achieves the lowest bound of the complexity that constructing linkable ring signature schemes. Second, ours preserves the anonymity of underlying ring signature schemes. In practice, our transformation incurs a very small overhead in size and running time. By instantiating our construction using the ring signature scheme [ESS$^+$18] and the one-time signature scheme [DKL$^+$18], we obtain a lattice-based linkable ring signature scheme whose signature size is logarithmic in the number of ring members. This scheme is practical, especially the signature size is very short: for 2^{30} ring members and security level of 100-bit, our signature size is only 4MB.

In addition, we give a new proof approach in proving the linkability, which might be of independent interest towards the proof in the random oracle model.

Keywords: ring signature, linkable ring signature, generic construction, lattice-based

1 Introduction

Ring signature (RS) was first proposed by Rivest et al. [RST01], which allows a signer to sign a message on behalf of a self-formed group. RS can provide not only unforgeability but also anonymity. Unforgeability requires an adversary cannot forge a signature on behalf of a ring which he does not know any secret key of ring members. Anonymity requires signatures do not leak any information about the identity of the signer, which can be categorized into two types: computational (against probabilistic polynomial adversary) and unconditional (against unbounded adversary).

As an extension of RS, Liu et al. [LWW04] first proposed the concept of linkable ring signature (LRS). LRS requires three properties: anonymity, linkability and nonslanderability. Anonymity is the same as that of RS. Linkability requires that if a signer signs twice, then a public procedure can link the two signatures to the same signer. Nonslanderability requires a user should not be entrapped that he has signed twice. Due to the security of LRS, it is widely used in many privacy-preserving scenarios which require accountable anonymity. For instance, LRS can be applied in e-voting system [TW05] to ensure that the voters can vote anonymously and will not repeat their votes. In a more popular setting, cryptocurrency, LRS plays a crucial

[*]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences & School of Cyber Security, University of Chinese Academy of Sciences. Email: wangxueli@iie.ac.cn

[†]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences & School of Cyber Security, University of Chinese Academy of Sciences. Email: yuchen.prc@gmail.com

[‡]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences & School of Cyber Security, University of Chinese Academy of Sciences. Email: maxuecheng@iie.ac.cn

role in providing anonymity of spenders while resisting the double-spending attack, and hence LRS has received much attention with the rise of Monero [Noe15] and other cryptocurrencies based on CryptoNote protocol [vS13].

The richer functionality of LRS makes it suited for a wide range of privacy-preserving applications, but also renders it relatively difficult to realize. Up to now, there are only a handful LRS schemes compared to numerous standard RS schemes. In light of the state of affairs described above, we are motivated to consider the generic construction of LRS, in particularly, whether LRS can be built from RS in a black-box manner. From a theoretical point of view, one is interested in the weakest assumptions needed for LRS. From a practical point of view, it is highly desirable to obtain general methods for constructing LRS rather than designing from scratch each time.

1.1 Our Contributions

In this paper, we give an affirmative answer to the above questions. The contribution of this paper is threefold:

- We give a generic construction of LRS from compatible RS and one-time signature (OTS). The construction achieves a lower bound of the complexity that constructing LRS scheme since RS is an arguably weaker primitive compared to chameleon hash plus function[1] (CH+) which is used as the underlying primitive by a recent generic construction of LRS [LAZ18]. In particular, the requirement for the underlying RS schemes is mild: the space of public keys $\hat{PK}$ has some group structure $(\hat{PK}, \odot)$ (e.g. bit-strings with $\oplus$) and the distribution of public keys generated by the key generation algorithm should be close to the uniform distribution over $\hat{PK}$, which can be satisfied by almost all the known RS schemes [AOS02, GK15, BDR15, BK10, LLNW16, ESS+18]. Moreover, our transformation preserves the same anonymity of the underlying RS schemes, which can simplify the task of constructing unconditionally anonymous LRS schemes. The reason is that most of RS schemes [RST01, DKNS04, CYH05, CWLY06, BK10, GK15, BDR15, LLNW16] provide unconditional anonymity but constructing unconditionally anonymous LRS schemes is stated as an open problem in [LWW04] and settled only by [LASZ14, TSS+18] recently. Finally, our transformation gives LRS schemes with small overhead in size and running time as compared to the underlying RS schemes.

- We develop a new proof approach to reduce the linkability of LRS to the unforgeability of RS. The new proof approach might be of independent interest towards the proof in the random oracle model.

- By instantiating our generic transformation based on the RS scheme in [ESS+18] and the OTS scheme Dilithium[2] [DKL+18] , we obtain a lattice-based LRS scheme whose signature size is logarithmic in the number of ring members. The signature size of our scheme is very short even for a large ring, for 2^{30} ring members and security level of 100-bit, our signature size is only 4MB comparing to 166MB[3] in the prior shortest lattice-based LRS scheme [YAL+17]. In addition, the experimental results demonstrate the concrete scheme is practical.

[1] CH+ is a similar concept of chameleon hash function but there exists public parameters and only the hash value is needed when computing the new randomness

[2] Dilithium is a signature scheme, we use it as a OTS scheme

[3] The signature size is from [LLNW16], the RS scheme in [LLNW16] is the major component of [YAL+17] and they have the same asymptotic size.

1.2 Technique Overview

To describe our construction, it is instructive to recall the generic construction of LRS [LAZ18], which is called Raptor. In [LAZ18], they introduced the concept of CH+ and gave a generic construction of LRS based on CH+ and OTS. In the key generation procedure, the signer generates a hash key hk and its trapdoor td, and a pair of public key and secret key (ovk, osk) of OTS. Then, the user computes the public key pk by masking hk with the hash value $H(ovk)$, and sets the secret key $sk = (td, ovk, osk)$. In the signing procedure, the signer first reconstructs a new set of hash keys $\{hk'_i = pk_i \oplus H(ovk_s)\}_{i\in[N]}$, where N is the number of ring members and s is the index of the signer, then runs the signing algorithm of RS on the set of public keys $\{hk'_i\}_{i\in[N]}$ to get signature $\hat{\sigma}$. Finally, the signer runs the signing algorithm of OTS to sign the message $(\hat{\sigma}, \{hk'_i\}_{i\in[N]}, ovk_s)$ with the secret key osk_s and get the signature $\tilde{\sigma}$. The signature σ is set as $(\hat{\sigma}, \tilde{\sigma}, ovk)$. In the security proof, the anonymity and linkability are reduced to the associated properties of CH+ and the nonslanderability is reduced to the unforgeability of OTS. However, there is a gap in the proof of linkability. The linkability is based on the collision-resistance of CH+, but the proof fails to embed the challenge hk_c of collision-resistance into the output of the linkability game. See Appendix A for details on these issues.

Inspired by the idea in [LAZ18], we give a generic construction of LRS from RS directly, rather than from CH+. The security proof of our scheme is not trivial, especially it is difficult to reduce the linkability of LRS to the unforgeability of RS. The reason is that in the security definition of unforgeability, a valid signature forgery must be generated on the ring which the adversary does not have the related secret keys, but this condition is hard to achieve by the forgery contained in the output of linkability game for our construction. We resolve it by developing a new proof approach.

Construction Sketch. In the key generation procedure, the signer firstly generates the public/secret key pair $(\hat{pk}, \hat{sk})$ and (ovk, osk) of LRS and OTS respectively. Then, he computes the public key $pk = \hat{pk} \odot H(ovk)$ and sets the secret key $sk = (\hat{sk}, osk, ovk)$. In the signing procedure, if the signer signs a message m on the ring $T = \{pk_i\}_{i\in[N]}$, he firstly reconstruct a new ring $\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}$, where $\hat{pk}'_i$ is equal to pk_i combined with the same value $H(ovk_s)$. It is easy to see that $\hat{pk}_i = \hat{pk}'_i$ only when $i = s$ and hence the signer knows the related secret key $\hat{sk}_s$ of $\hat{pk}'_s$. Then, he runs RS.Sign with $\hat{sk}_s$ and $\hat{T}'$ to get the signature $\hat{\sigma}$. Finally, the signer runs OTS.Sign on the message $(\hat{\sigma}, T, ovk_s)$ by the secret key osk_s, then he gets the signature $\tilde{\sigma}$ and sets $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$, where ovk_s acts as the linkability tag.

Proof Sketch. We will omit the proofs of anonymity and nonslanderability and just sketch the new proof approach here. As described above, the linkability of our construction is reduced to the unforgeability of underlying RS schemes. For the sake of contradiction, suppose that there exists an adversary $\mathcal{A}$ that break linkability of our LRS scheme. Then, we construct an adversary $\mathcal{B}$ that break unforgeability of underlying RS scheme by using $\mathcal{A}$. If $\mathcal{A}$ succeeds, that is $\mathcal{A}$ outputs $N + 1$ unlinked valid signatures for the same ring whose size is N, then at least one of the signatures, denoted as σ^*, contains the linkability tag which is not used in the key generation procedure. We set $\hat{\sigma}^*$ contained in σ^* as the output of $\mathcal{B}$. The core problem that we face in reduction is how to simulate the public key for $\mathcal{A}$ to make $\hat{\sigma}^*$ is generated on the ring $\hat{T}'$ which $\mathcal{B}$ does not know the secret keys. At a high level, we resolve this problem by fixing every $\hat{pk}' \in \hat{T}'$ for $\mathcal{B}$ in advance instead of making it generated by $\mathcal{A}$. More specifically, we assume $\mathcal{A}$ and $\mathcal{B}$ have access to the joining oracle $\mathcal{O}_{\text{join}}$ and $\hat{\mathcal{O}}_{\text{join}}$ respectively, where $\mathcal{O}_{\text{join}}$ and $\hat{\mathcal{O}}_{\text{join}}$ output public keys of LRS and RS at random. For every query to $\mathcal{O}_{\text{join}}$ made by $\mathcal{A}$, $\mathcal{B}$ should query $\hat{\mathcal{O}}_{\text{join}}$ twice to get two public keys $\hat{pk}, \hat{pk}''$. $\hat{pk}$ is used to simulate the response of $\mathcal{O}_{\text{join}}$, and $\hat{pk}''$ is used to fix the elements in $\hat{T}'$. By the programmability of H, we generate pk in two different ways using $\hat{pk}, \hat{pk}''$ respectively:

$$pk = \hat{pk} \odot h = \hat{pk}'' \odot h'$$

where h' is chosen randomly and programmed as the output of the Ith H-query, h is computed by the above equation and programmed as the output of the H-query whose input is related ovk. If the input of the Ith H-query is ovk_s, then the forgery of RS contained in the output of $\mathcal{A}$ is generated on the public keys output by $\hat{\mathcal{O}}_{\text{join}}$ which $\mathcal{B}$ does not know the secret keys. The real execution of $\mathcal{O}_{\text{join}}$ is depicted in Fig.1 and the simulation of $\mathcal{O}_{\text{join}}$ is depicted in Fig.2.

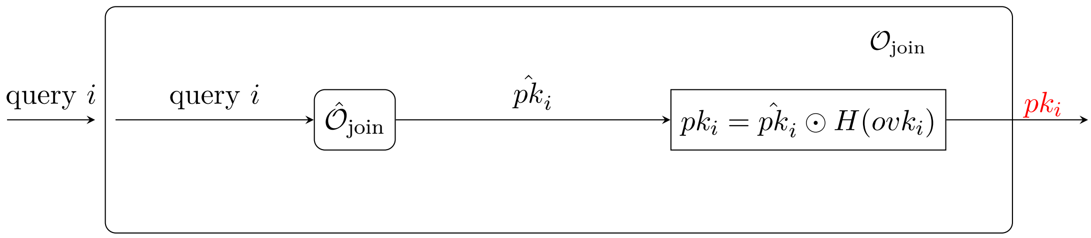

Figure 1: real $\mathcal{O}_{\text{join}}$

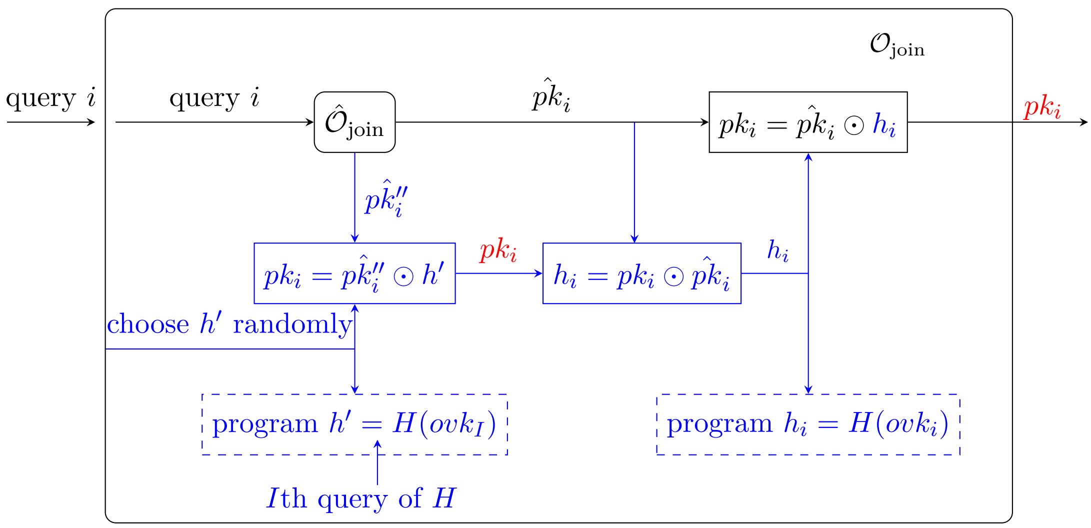

Figure 2: Simulation of $\mathcal{O}_{\text{join}}$

1.3 Related Work

Ring Signature. Abe et al. [AOS02] showed how to construct a RS scheme from a three-move sigma protocol based signature scheme and presented the first RS scheme under the discrete-logarithm assumption whose public keys are group elements. Groth and Kohlweiss [GK15] proposed a RS scheme whose signature size grows logarithmically in the number of ring members from a sigma protocol for a homomorphic commitment. They instantiated their scheme with Pedersen commitment and set the public keys as the commitments to 0. Bose et al. [BDR15] gave a generic technique to convert a compatible signature scheme to a RS scheme whose signature size is independent of the number of ring members and instantiated it from Full Boneh-Boyen signature. Brakerski and Kalai [BK10] proposed the first lattice-based RS scheme from ring trapdoor functions whose public keys are matrices over a group. Libert et al.

[LLNW16] proposed the first lattice-based RS scheme with logarithmic size in the number of ring members which is from zero-knowledge arguments for lattice-based accumulators and the public keys of their scheme are binary strings. We show that all of the above RS schemes satisfy the requirements of our transformation, and hence they can be extended to the LRS schemes directly by using our generic construction.

Linkable Ring Signature. Besides [LAZ18], there are two works tried to build a generic framework of LRS. Tsang and Wei [TW05] extended the generic RS constructions in [DKNS04] to their linkable version, but their schemes are under a weak security model which does not consider the nonslanderability. Franklin and Zhang [FZ12] proposed a general and unified framework for constructing unique ring signature which captures the spirit of LRS, but the linkability of their work is restricted to the same message, which is not suited for cryptocurrency. Except that, there are a series of outstanding results on constructing concrete linkable ring signature schemes. Yuen et al. [YLA+13] proposed a LRS scheme whose signature size is square root of the number of ring members. Sun et al. [SALY17] presented an accumulator-based LRS scheme whose signature size is independent of the number of ring members. Yang et al. [YAL+17] presented a construction of weak-PRF from LWR and designed a LRS scheme based on lattice by combining with an accumulator scheme in [LLNW16] and the supporting ZKAoKs. Zhang et al. [ZZTA17] proposed an anonymous post-quantum cryptocash which contains an ideal lattice-based LRS scheme. Baum et al. [BLO18] proposed a LRS scheme based on module lattice, which is mainly constructed from a lattice-based collision-resistant hash function. At the same time, Torres et al. [TSS+18] proposed a post-quantum one-time LRS scheme, which generalized a practical lattice-based digital signature BLISS [DDLL13] to LRS and successfully applied to the privacy protection protocol lattice ringCT v1.0.

Note. The issues we discovered about Raptor exist in the previous eprint version, available at https://eprint.iacr.org/2018/857 (version: 20180921:135633). We have communicated with the authors of Raptor, they confirmed our findings and the issues have been discovered independently by them as well. They shared with us their revised version which does not have the same flaws.

2 Preliminary

2.1 Notations

Throughout this paper, we use $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ to denote the set of natural numbers, integers and real numbers respectively. For $N \in \mathbb{N}$, we define $[N]$ as shorthand for the set $\{1, ..., N\}$. We use lower-case bold letters and upper-case bold letters to denote vectors and matrices (e.g. $\mathbf{x}$ and $\mathbf{A}$). We denote the Euclidean norm of a vector $\mathbf{x} = (x_0, ..., x_{n-1})$ and a polynomial $f(x) = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$ in variable X as $||\mathbf{x}|| = \sqrt{\sum_{i=0}^{n-1} x_i^2}$ and $||f|| = \sqrt{\sum_{i=0}^{n=1} a_i^2}$. For a vector $\mathbf{f} = (f_0, \cdots, f_{n-1})$ of polynomials, $||\mathbf{f}|| = \sqrt{\sum_{i=0}^{n-1} ||f_i||^2}$. The infinity norm of f is $||f||_\infty = \max_i |a_i|$. Let q be an odd prime integer and assume $q \equiv 5 \bmod 8$. We define the rings $R = \mathbb{Z}[X]/\langle X^d + 1\rangle$ and $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$, where $d > 1$ is a power of 2. If S is a set then $s \leftarrow S$ denotes the operation of uniformly sampling an element s from S at random. We use the same notation to sample s from a distribution S. If S is an algorithm, the same notation is used to denote that the algorithm outputs s. We denote a negligible function by $\mathsf{negl}(\lambda)$, which is a function $g(\lambda) = \mathcal{O}(\lambda^c)$ for some constant c. We denote the set of integers

$\{a, a+1, \cdots, b-1, b\}$ by $[a, b]$. We use $D_{v,\sigma}$ to denote the discrete normal distribution centered at v with standard deviation σ. We write D_σ as shorthand for $v = 0$.

2.2 Module-SIS Problem and Commitment Scheme

We recall the definition of Module-SIS problem and commitment scheme in [ESS+18].

Definition 2.1 (Module-SIS$_{n,m,q,\theta}$)**.** Let $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$. Given $\mathbf{A} \leftarrow R_q^{n\times m}$, find $\mathbf{x} \in R_q^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ and $0 < ||\mathbf{x}|| \leq \theta$.

Module-SIS problem is a bridge between SIS problem and Ring-SIS problem, that is if $m = n\log q$ and $R_q = \mathbb{Z}_q$, then the above problem is SIS problem; if $n = m = 1$, then the above problem is R-SIS problem.

Definition 2.2. Let $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$, $S_{\mathbf{r}}(\epsilon_{\mathbf{r}}) = \{\mathbf{r} \in R_q^m : ||\mathbf{r}||_\infty \leq \epsilon_{\mathbf{r}}\}$ be the randomness domain with χ as the probability distribution of $\mathbf{r}$ on $S_{\mathbf{r}}(\epsilon_{\mathbf{r}})$ for a positive real number $\epsilon_{\mathbf{r}}$, and $S_M(\epsilon_M) = \{\mathbf{m} \in R_q^v : ||\mathbf{m}||_\infty \leq \epsilon_M\}$ be the message domain for a positive real number ϵ_M for $m, v \in \mathbb{Z}^+$. The commitment of a message vector $\mathbf{m} = (m_1, ..., m_v) \in S_M(\epsilon_M)$ using a randomness $\mathbf{r} \in S_{\mathbf{r}}$ is given as

$$\mathrm{Com}_{ck}(\mathbf{m}; \mathbf{r}) = \mathrm{Com}_{ck}(m_1, \cdots, m_v; \mathbf{r}) = \mathbf{G} \cdot \begin{pmatrix} \mathbf{r} \\ m_1 \\ \vdots \\ m_v \end{pmatrix} \in R_q^n$$

where $ck = \mathbf{G} \leftarrow R_q^{n\times(m+v)}$ and it is used as the commitment key.

The Lemma 2 in [ESS+18] shows the above commitment scheme is statistically hiding if $q > 8\epsilon_{\mathbf{r}^2}$ and χ is greater than $n\log q/m + 2\lambda/(md)$, and computationally strong binding if Module-SIS$_{n,m+v,q,\theta}$ problem for $\theta = 2\sqrt{\epsilon_{\mathbf{r}}^2 md + \epsilon_M^2 vd}$ is hard.

2.3 Rejection Sampling

Lemma 2.1 ([Lyu12])**.** *Let V be a subset of $\mathbb{Z}^d$ where all the elements have norms less $\mathcal{T}$, and h be a probability distribution over V. Define the following algorithms:*

$\mathcal{A}$: $\boldsymbol{v} \leftarrow h$; $\mathbf{z} \leftarrow D_{v,\sigma}^d$; output $(\mathbf{z}, \mathbf{v})$ with probability $\min\{\frac{D_\sigma^d(\mathbf{z})}{\mathcal{M}D_{\mathbf{v},\sigma}^d(\mathbf{z})}, 1\}$

$\mathcal{F}$: $\boldsymbol{v} \leftarrow h$; $\mathbf{z} \leftarrow D_\sigma^d$; output $(\mathbf{z}, \mathbf{v})$ with probability $\frac{1}{\mathcal{M}}$,

where $\sigma = 12\mathcal{T}$ and $\mathcal{M} = e^{1+\frac{1}{288}}$. Then the output of algorithm $\mathcal{A}$ is within statistical distance $2^{-100}/\mathcal{M}$ of the output of $\mathcal{F}$. Moreover, the probability that $\mathcal{A}$ outputs something is more than $\frac{1-2^{-100}}{\mathcal{M}}$.

2.4 Ring Signature

A RS scheme consists of four algorithms (Setup, KeyGen, Sign, Vrfy):

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, outputs public parameter pp. We assume pp is an implicit input to all the algorithms listed below.
- $\mathsf{KeyGen}(pp)$: On input the public parameter pp, outputs secret key sk and public key pk.

- $\mathsf{Sign}(sk, m, T)$: On input the secret key sk, a signing message m and a set of public keys T, outputs a signature σ.
- $\mathsf{Vrfy}(T, m, \sigma)$: On input the set of public keys T, signing message m and signature σ, outputs *accept*/*reject*.

Correctness. For any security parameter λ, any $\{pk_i, sk_i\}_{i\in[N]}$ output by KeyGen, any $s \in [N]$, and any message m, we have $\mathsf{Vrfy}(T, m, \mathsf{Sign}(sk_s, m, T)) = accept$ where $T = \{pk_i\}_{i\in[N]}$.

Before introducing the security definitions of RS, we first assume there are three oracles as follows:

- Joining Oracle $pk \leftarrow \mathcal{O}_{\text{join}}(\perp)$: $\mathcal{O}_{\text{join}}$ generates a new user and returns the public key pk of the new user.
- Corruption Oracle $sk \leftarrow \mathcal{O}_{\text{corrupt}}(pk)$: On input a public key pk which is a output of $\mathcal{O}_{\text{join}}$, returns the corresponding secret key sk.
- Signing Oracle $\sigma \leftarrow \mathcal{O}_{\text{sign}}(T, m, pk_s)$: On input a set of public keys T, message m and the public key of the signer $pk_s \in T$, returns a valid signature σ on m and T.

There are two security properties of RS: *anonymity* and *unforgeability*.

Anonymity. The anonymity of a RS scheme requires that an adversary can not tell the real signer from all the ring members. The anonymity can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$:

1. Setup: The challenger $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter pp to $\mathcal{A}$.
2. Query: The adversary $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$.
3. Challenge: $\mathcal{A}$ picks a set of public keys $T = \{pk_i\}_{i\in[N]}$ and a message m. $\mathcal{A}$ sends (T, m) to $\mathcal{CH}$. $\mathcal{CH}$ picks $s \in [N]$ and runs $\sigma \leftarrow \mathsf{Sign}(sk_s, m, T)$. $\mathcal{CH}$ sends σ to $\mathcal{A}$.
4. Output: $\mathcal{A}$ outputs a guess $s^* \in [N]$.

$\mathcal{A}$ wins if $s^* = s$. The advantage of $\mathcal{A}$ is defined by $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}} = |\Pr[s^* = s] - \frac{1}{N}|$.

Definition 2.3. A RS scheme is said to be anonymous (resp.unconditionally anonymous) if for any PPT adversary (unbounded adversary) $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}}$ is negligible in λ.

Unforgeability. The unforgeability of a RS scheme captures the intuition that an outside adversary cannot forge a signature for a ring, whose formal definition is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$.

1. Setup: The challenger $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter pp to $\mathcal{A}$.
2. Query: The adversary $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}, \mathcal{O}_{\text{corrupt}}$ and $\mathcal{O}_{\text{sign}}$.
3. Output: $\mathcal{A}$ outputs a forgery (m^*, σ^*, T^*).

$\mathcal{A}$ wins if

- $\mathsf{Vrfy}(m^*, \sigma^*, T^*) = accept$;

- all of the public keys in T^* are query outputs of $\mathcal{O}_{\text{join}}$;
- no public key in T^* has been input to $\mathcal{O}_{\text{corrupt}}$; and
- (m^*, T^*) has not been queried to $\mathcal{O}_{\text{sign}}$.

The advantage of $\mathcal{A}$, denoted as $\mathsf{Adv}_{\mathcal{A}}^{\text{forge}}$, is defined by the probability that $\mathcal{A}$ wins in the above game.

Definition 2.4. A RS scheme is said to be unforgeable if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{forge}}$ is negligible in λ.

2.5 Linkable Ring Signature

A LRS scheme consists of five algorithms (Setup, KeyGen, Sign, Vrfy, Link):

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, outputs public parameter *pp*. We assume *pp* is an implicit input to all the algorithms listed below.
- $\mathsf{KeyGen}(pp)$: On input the public parameter *pp*, outputs secret key *sk* and public key *pk*.
- $\mathsf{Sign}(sk, m, T)$: On input the secret key *sk*, a signing message m and a set of public keys T, outputs a signature σ.
- $\mathsf{Vrfy}(T, m, \sigma)$: On input the set of public keys T, the signing message m and the signature σ, outputs *accept/reject*.
- $\mathsf{Link}(m_1, m_2, \sigma_1, \sigma_2, T_1, T_2)$: On input two sets of public keys T_1, T_2, two signing messages m_1, m_2 and their signatures σ_1, σ_2, outputs *linked/unlinked*.

Correctness. For any security parameter 1^λ, any $\{pk_i, sk_i\}_{i\in[N]}$ output by KeyGen, any $s \in [N]$, and any message m, we have $\mathsf{Vrfy}(T, m, \mathsf{Sign}(sk_s, m, T)) = \mathit{accept}$ where $T = \{pk_i\}_{i\in[N]}$.

There are three security properties of LRS: *anonymity, linkability* and *nonslanderability.* According to [ASY06], the *unforgeability* of LRS can be implied by the linkability and the nonslanderability defined below.

Anonymity. The anonymity of LRS is the same as that of RS. The security can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$:

1. Setup: The challenger $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter *pp* to $\mathcal{A}$.
2. Query: The adversary $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$.
3. Challenge: $\mathcal{A}$ picks a set of public keys $T = \{pk_i\}_{i\in[N]}$ and a message m. $\mathcal{A}$ sends (T, m) to $\mathcal{CH}$. $\mathcal{CH}$ picks $s \in [N]$ and runs $\sigma \leftarrow \mathsf{Sign}(sk_s, m, T)$. $\mathcal{CH}$ sends σ to $\mathcal{A}$.
4. Output: $\mathcal{A}$ outputs a guess $s^* \in [N]$.

$\mathcal{A}$ wins if $s^* = s$. The advantage of $\mathcal{A}$ is defined by $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}} = |\Pr[s^* = s] - \frac{1}{N}|$.

Definition 2.5. A LRS scheme is said to be anonymous (resp.unconditionally anonymous) if for any PPT adversary (unbounded adversary) $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}}$ is negligible in λ.

Linkability. The linkability of LRS is used to go against the dishonest signers. The intuition of linkability is that a signer cannot generate two valid unlinked signatures. It can be translated into that the users in a ring with size N cannot produce $N+1$ valid signatures and any two of them are unlinkable. Linkability can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$:

1. Setup: The challenger $\mathcal{CH}$ runs Setup and gives $\mathcal{A}$ public parameter pp.
2. Query: The adversary $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$, $\mathcal{O}_{\text{corrupt}}$, $\mathcal{O}_{\text{sign}}$.
3. Output: $\mathcal{A}$ outputs $N+1$ messages/signature pairs $\{T, m_i, \sigma_i\}_{i\in[N+1]}$, where T is a set of public keys with size N.

$\mathcal{A}$ wins if

- all public keys in T are query outputs of $\mathcal{O}_{\text{join}}$;
- $\mathsf{Vrfy}(m_i, \sigma_i, T) = \textit{accept}$ for all $i \in [N+1]$;
- $\mathsf{Link}(m_i, m_j, \sigma_i, \sigma_j) = \textit{unlinked}$ for all $i, j \in [N+1]$ and $i \neq j$.

The advantage of $\mathcal{A}$, denoted as $\mathsf{Adv}_{\mathcal{A}}^{\text{link}}$, is defined by the probability that $\mathcal{A}$ wins in the above game.

Definition 2.6. A LRS scheme is said to be linkable if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{link}}$ is negligible in λ.

Nonslanderability. The nonslanderability of LRS means no adversary can entrap a user to be considered to have signed twice even he corrupted all the users except the target user. Nonsladerabiliy can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$:

1. Setup: The challenger $\mathcal{CH}$ runs Setup and gives $\mathcal{A}$ public parameter pp.
2. Query: The adversary $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$, $\mathcal{O}_{\text{corrupt}}$, $\mathcal{O}_{\text{sign}}$.
3. Challenge: $\mathcal{A}$ gives $\mathcal{CH}$ a set of public keys T, a message m and a public key $pk_s \in T$. $\mathcal{CH}$ runs $\mathsf{Sign}(sk_s, m, T)$ and returns the corresponding signature σ to $\mathcal{A}$.
4. Output: $\mathcal{A}$ outputs a set of public keys T^*, a message m^* and a signature σ^*.

$\mathcal{A}$ wins if

- $\mathsf{Vrfy}(m^*, \sigma^*, T^*) = \textit{accept}$;
- pk_s is not queried by $\mathcal{A}$ to $\mathcal{O}_{\text{corrupt}}$ and as an insider to $\mathcal{O}_{\text{sign}}$;
- all public keys in T and T^* are query outputs of $\mathcal{O}_{\text{join}}$; and
- $\mathsf{Link}(m, m^*, \sigma, \sigma^*) = \textit{linked}$.

The advantage of $\mathcal{A}$, denoted as $\mathsf{Adv}_{\mathcal{A}}^{\text{slander}}$, is defined by the probability that $\mathcal{A}$ wins in the above game.

Definition 2.7 (Nonslanderability). A LRS scheme is said to be nonslanderable if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{slander}}$ is negligible in λ.

3 Generic Construction of Linkable Ring Signature

3.1 Construction

The generic construction is based on two primitives: (1) a ring signature scheme RS=(Setup,KeyGen, Sign,Vrfy); (2) a one-time signature scheme OTS=(KeyGen, Sign,Vrfy).

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, this algorithm runs $\hat{pp} \leftarrow \text{RS}.\mathsf{Setup}(1^\lambda)$. It also chooses a hash function $H : OVK \rightarrow \hat{PK}$ modeled as random oracle, where OVK and $\hat{PK}$ are public key spaces of OTS and RS respectively. Finally, it outputs public parameter $pp = \hat{pp}$. We assume pp is an implicit input to all the algorithms listed below.

- $\mathsf{KeyGen}(pp)$: On input the public parameter pp, runs $(\hat{pk}, \hat{sk}) \leftarrow \text{RS}.\mathsf{KeyGen}(\hat{pp})$. Then, the algorithm runs $(ovk, osk) \leftarrow \text{OTS}.\mathsf{KeyGen}$. It returns public key $pk = \hat{pk} \odot H(ovk)$ and secret key $sk = (\hat{sk}, osk, ovk)$.

- $\mathsf{Sign}(sk_s, m, T)$: On input the secret key sk_s, a signing message m and a set of public keys $T = \{pk_i\}_{i\in[N]}$, computes $\hat{pk}'_i = pk_i \odot H(ovk_s)$ for $i \in [N]$ and sets $\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}$. Next, the algorithm runs

$$\hat{\sigma} \leftarrow \text{RS}.\mathsf{Sign}(\hat{sk}_s, m, \hat{T}'),$$
$$\tilde{\sigma} \leftarrow \text{OTS}.\mathsf{Sign}(osk_s, \hat{\sigma}, T, ovk_s).$$

  Finally, it returns the signature $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$.

- $\mathsf{Vrfy}(T, m, \sigma)$: On input the set of public keys T, a signing message m and the signature σ, this algorithm first parses σ as $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk)$ and computes $\hat{pk}'_i = pk_i \odot H(ovk)$ for $i \in [N]$. Next, it runs $\text{RS}.\mathsf{Vrfy}(\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}, m, \hat{\sigma})$ and $\text{OTS}.\mathsf{Vrfy}(ovk, (\hat{\sigma}, T, ovk), \tilde{\sigma})$. Finally, it outputs *accept* if RS.Vrfy returns *accept* and OTS.Vrfy returns *accept*; otherwise outputs *reject*.

- $\mathsf{Link}(m_1, m_2, \sigma_1, \sigma_2, T_1, T_2)$: On input two sets of public keys T_1, T_2, two signing messages m_1, m_2 and their signatures σ_1, σ_2, runs $\mathsf{Vrfy}(m_1, \sigma_1, T_1)$ and $\mathsf{Vrfy}(m_2, \sigma_2, T_2)$. If $\mathsf{Vrfy}(m_1, \sigma_1, T_1) = reject$ or $\mathsf{Vrfy}(m_2, \sigma_2, T_2) = reject$, it aborts. Otherwise, the algorithm parses σ_1 and σ_2 as $\sigma_1 = (\hat{\sigma}_1, \tilde{\sigma}_1, ovk_1)$ and $\sigma_2 = (\hat{\sigma}_2, \tilde{\sigma}_2, ovk_2)$, and compares ovk_1 and ovk_2. If $ovk_1 = ovk_2$, then outputs *linked*; otherwise outputs *unlinked*.

3.2 Security

Theorem 3.1. *Our LRS scheme is anonymous (resp. unconditionally anonymous) if the underlying RS scheme is anonymous (resp. unconditionally anonymous).*

Proof. If there exists an adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_{\text{join}}$ can break the anonymity of LRS, then we can construct an adversary $\mathcal{B}$ with oracles access to $\hat{\mathcal{O}}_{\text{join}}, \hat{\mathcal{O}}_{\text{sign}}$ to break the anonymity of RS with the same advantage, where $\hat{\mathcal{O}}_{\text{join}}$ and $\hat{\mathcal{O}}_{\text{sign}}$ are oracles in security games of RS.

Given a signature $\hat{\sigma}$ on the set of public keys T and a message m chosen by $\mathcal{B}$, $\mathcal{B}$ interacts with $\mathcal{A}$ with the aim to guess the signer s.

1. Setup: Given the public parameter $\hat{pp}$, $\mathcal{B}$ selects a hash function $H : OVK \rightarrow \hat{PK}$, where H is modeled as random oracle and OVK and $\hat{PK}$ are public key spaces of OTS and RS respectively. $\mathcal{B}$ then sends $pp = \hat{pp}$ to $\mathcal{A}$.

2. Oracle Simulation: $\mathcal{A}$ is allowed to access the joining oracle $\mathcal{O}_{\text{join}}$: $\mathcal{B}$ runs $(ovk, osk) \leftarrow$OTS. KeyGen at first. Upon receiving a joining query, $\mathcal{B}$ queries $\hat{\mathcal{O}}_{\text{join}}$ to obtain a public key $\hat{pk}$ of RS, computes $pk = \hat{pk} \odot H(ovk)$. $\mathcal{B}$ then sends pk to $\mathcal{A}$.

   The only difference between this simulation and the real game is that in this simulation, every pk is generated by the same ovk. $\mathcal{A}$ cannot distinguish this simulation from the real game since the distribution of $\hat{pk}$ is close to the uniform distribution over $\hat{PK}$, which means the pk generated in two games are both close to the uniform distribution over $\hat{PK}$.

3. Challenge: Received $(T = \{pk_i\}_{i\in[N]}, m)$ from $\mathcal{A}$, $\mathcal{B}$ computes $\hat{pk}'_i = pk_i \odot H(ovk)$ for all $i \in [N]$ and sets $\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}$. $\mathcal{B}$ then sends $(\hat{T}', m)$ to the challenger $\mathcal{CH}$ and received a signature $\hat{\sigma}$ (signed by $\hat{pk}_s \in \hat{T}$ which is chosen by $\mathcal{CH}$). $\mathcal{B}$ runs $\tilde{\sigma} \leftarrow$OTS.$\mathsf{Sign}(osk, \hat{\sigma}, T, ovk)$ and sends $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk)$ to $\mathcal{A}$.

4. Output: $\mathcal{A}$ outputs the index s^*.

Finally, $\mathcal{B}$ forwards s^* to the challenger $\mathcal{CH}$. $\mathcal{A}$ essentially guesses which index is used to generate $\hat{\sigma}$ since $\tilde{\sigma}, ovk$ are identical no matter which index $\mathcal{CH}$ has chosen. If $\mathcal{A}$ succeeds, $\mathcal{B}$ also succeeds due to $\mathcal{B}$ is also aim to guess which index is used to generate $\hat{\sigma}$. We have $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}} = \mathsf{Adv}_{\mathcal{B}}^{\text{anon}}$. □

Theorem 3.2. *Our LRS scheme is linkable in the random oracle model if the underlying RS is unforgeable .*

Proof. We proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ succeeds in Game i.

Game 0. This is the standard linkability game for LRS. The challenger $\mathcal{CH}$ interacts with $\mathcal{A}$ as below:

1. Setup: $\mathcal{CH}$ runs $\hat{pp} \leftarrow$RS.$\mathsf{Setup}(1^\lambda)$, selects a hash function $H : OVK \to \hat{PK}$, where H is modeled as random oracle and OVK and $\hat{PK}$ are public key spaces of OTS and RS respectively. $\mathcal{CH}$ then sends $pp = \hat{pp}$ to $\mathcal{A}$.

2. Oracle Simulation: $\mathcal{A}$ is allowed to access the following four oracles:

   Random Oracle H: To make our proof explicit, we separate the queries of H as two categories: querying directly and querying in $\mathcal{O}_{\text{join}}$ and $\mathcal{O}_{\text{sign}}$. $\mathcal{CH}$ initializes an empty set RO. Upon receiving a random oracle query i, if it has been queried, $\mathcal{CH}$ returns corresponding output in RO; else, $\mathcal{CH}$ picks $h_i \leftarrow \hat{PK}$ at random, sends h_i to $\mathcal{A}$ and stores the pair of (i, h_i) in RO.

   Joining Oracle $\mathcal{O}_{\text{join}}$: $\mathcal{CH}$ initializes an empty set JO. Upon receiving a joining query, $\mathcal{CH}$ runs $(\hat{pk}, \hat{sk}) \leftarrow$RS.$\mathsf{KeyGen}$ and $(ovk, osk) \leftarrow$OTS.KeyGen, computes $pk = \hat{pk} \odot H(ovk)$, sets $sk = (\hat{sk}, osk, ovk)$. $\mathcal{CH}$ then sends pk to $\mathcal{A}$ and stores pk in JO.

   Corruption Oracle $\mathcal{O}_{\text{corrupt}}$: Upon receiving a corruption query pk, $\mathcal{CH}$ sends corresponding sk to $\mathcal{A}$ if $pk \in JO$; else, $\mathcal{CH}$ return $\perp$.

   Signing Oracle $\mathcal{O}_{\text{sign}}$: Upon receiving a signing query $(T = \{pk_i\}_{i\in[N]}, m, pk_s \in T)$, $\mathcal{CH}$ computes $\hat{pk}'_i = pk_i \odot H(ovk_s)$ for $i \in [N]$, sets $\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}$, runs $\hat{\sigma} \leftarrow$RS.$\mathsf{Sign}(\hat{sk}_s, m, \hat{T}')$ and $\tilde{\sigma} \leftarrow$OTS.$\mathsf{Sign}(osk_s, \hat{\sigma}, T, ovk_s)$. $\mathcal{CH}$ then sends $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$ to $\mathcal{A}$.

3. Outputs: $\mathcal{A}$ outputs $N + 1$ message/signature pairs $\{m_i, \sigma_i\}_{i\in[N+1]}$ on the same set of public keys $T = \{pk_i\}_{i\in[N]}$. $\mathcal{A}$ wins if

- all public keys in T are query outputs of $\mathcal{O}_{\text{join}}$;
- $\mathsf{Vrfy}(m_i, \sigma_i, T) = \textit{accept}$ for all $i \in [N+1]$;
- $\mathsf{Link}(m_i, m_j, \sigma_i, \sigma_j)=\textit{unlinked}$ for all $i, j \in [N+1]$ and $i \neq j$.

According to the definition, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{link}} = \Pr[S_0]$$

Game 1. Same as Game 0 except that in $\mathcal{O}_{\mathsf{join}}$ of Oracle Simulation stage, $\mathcal{CH}$ additionally choose $h' \leftarrow \hat{PK}$ at first before receiving queries. This change is purely conceptual and thus we have

$$\Pr[S_1] = \Pr[S_0]$$

Game 2. Same as Game 1 except that in H of Oracle Simulation stage, $\mathcal{CH}$ chooses a index $I \leftarrow [1, ..., q_h]$, where q_h is the maximum number of times $\mathcal{A}$ directly queries H, then $\mathcal{CH}$ programs the output of the Ith query as h'.

By the programmability of H and h' is chosen uniformly and independently, we have

$$\Pr[S_2] = \Pr[S_1]$$

Game 3. Same as Game 2 except that in $\mathcal{O}_{\mathsf{join}}$ of Oracle Simulation stage, $\mathcal{CH}$ additionally runs $(\hat{pk}'', \hat{sk}'') \leftarrow$ RS.KeyGen and computes h such that $\hat{pk} \odot h = \hat{pk}'' \odot h'$ upon receiving a joining query. $\mathcal{CH}$ then sends $pk = \hat{pk} \odot h$ to $\mathcal{A}$. Due to the distribution of $\hat{pk}$ is close to the uniform distribution over $\hat{PK}$, hence

$$\Pr[S_3] = \Pr[S_2]$$

Game 4: Same as Game 3 except that in H of Oracle Simulation stage, $\mathcal{CH}$ programs the output of the query on ovk (querying in $\mathcal{O}_{\mathsf{join}}$) as corresponding h. By the programmability of H, we have

$$\Pr[S_4] = \Pr[S_3]$$

Lemma 3.1. *If the RS is unforgeable, then the probability that any adversary wins in Game 4 is negligible in λ.*

If $\mathcal{A}$ wins in Game 4, then we can construct an adversary $\mathcal{B}$ with oracles access to $\hat{\mathcal{O}}_{\text{join}}$, $\hat{\mathcal{O}}_{\text{corrupt}}$ and $\hat{\mathcal{O}}_{\text{sign}}$ to break the unforgeability of RS with the advantage $q_h \cdot \mathsf{Adv}_{\mathcal{A}}^{\text{forge}}$, implying $\Pr[S_4]$ must be negligible, where $\hat{\mathcal{O}}_{\text{join}}, \hat{\mathcal{O}}_{\text{corrupt}}$ and $\hat{\mathcal{O}}_{\text{sign}}$ are oracles in security games of RS.

$\mathcal{B}$ interacts with $\mathcal{A}$ in Game 4 with the aim to output $(m^*, \sigma^*, \hat{T}^*)$ satisfying the conditions in Definition 4.

1. Setup: Given the public parameter $\hat{pp}$, $\mathcal{B}$ selects a hash function $H : OVK \to \hat{PK}$, where H is modeled as random oracle and OVK and $\hat{PK}$ are public key spaces of OTS and RS respectively. $\mathcal{B}$ then sends $pp = \hat{pp}$ to $\mathcal{A}$.

2. Oracle Simulation:

   Random Oracle H: To make our proof explicit, we separate the queries of H as two categories: querying directly and querying in $\mathcal{O}_{\text{join}}$ and $\mathcal{O}_{\text{sign}}$. $\mathcal{B}$ initializes an empty set RO, chooses a index $I \leftarrow [1, ..., q_h]$, where q_h is the maximum number of times $\mathcal{A}$ directly queries H. $\mathcal{A}$ then programs the output of the Ith query as h' and stores them in RO. On receiving a random oracle query i, if it has been queried, $\mathcal{A}$ returns corresponding output in RO; otherwise, $\mathcal{B}$ programs the output as corresponding h if it is the query on ovk (querying in $\mathcal{O}_{\mathsf{join}}$), else $\mathcal{B}$ picks $h_i \leftarrow \hat{PK}$, sends h_i to $\mathcal{A}$ and stores the pair (i, h_i) in RO.

Joining Oracle $\mathcal{O}_{\text{join}}$: $\mathcal{B}$ initializes an empty set JO and chooses $h' \leftarrow \hat{PK}$. Upon receiving a joining query from $\mathcal{A}$, $\mathcal{B}$ queries $\hat{\mathcal{O}}_{\text{join}}$ twice to get two public keys $\hat{pk}, \hat{pk}''$, computes h such that $\hat{pk} \odot h = \hat{pk}'' \odot h'$, runs $(ovk, osk) \leftarrow$ OTS.KeyGen. $\mathcal{B}$ then sends $pk = \hat{pk} \odot h$ to $\mathcal{A}$ and stores pk in JO.

Corruption Oracle $\mathcal{O}_{\text{corrupt}}$: Upon receiving a corruption query pk, $\mathcal{B}$ queries the oracle $\hat{\mathcal{O}}_{\text{corrupt}}$ on input $\hat{pk}$ to obtain $\hat{sk}$ if $pk \in JO$; else $\mathcal{B}$ returns $\perp$. $\mathcal{B}$ then sends $sk = (\hat{sk}, ovk, osk)$ to $\mathcal{A}$.

Signing Oracle $\mathcal{O}_{\text{sign}}$: Upon receiving a signing query $(T = \{pk_i\}_{i\in[N]}, m, pk_s \in T)$, $\mathcal{B}$ queries the oracle $\hat{\mathcal{O}}_{\text{sign}}$ on input $(\hat{T}' = \{\hat{pk}'_i = pk_i \odot h_i\}_{i\in[N]}, m, \hat{pk}_s \in \hat{T}')$ to get a signature $\hat{\sigma}$, runs $\tilde{\sigma} \leftarrow$OTS.$\mathsf{Sign}(osk_s, \hat{\sigma}, T, ovk_s)$. $\mathcal{B}$ then sends $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$ to $\mathcal{A}$.

3. Output: $\mathcal{A}$ outputs $N+1$ message/signature pairs $\{m_i, \sigma_i\}_{i\in[N+1]}$ on the same set of public keys $T = \{pk_i\}_{i\in[N]}$ and wins in Game 4.

Upon receiving $\{m_i, \sigma_i, T = \{pk_j\}_{j\in[N]}\}_{i\in[N+1]}$, $\mathcal{B}$ parses every signature σ_i as $\sigma_i = (\hat{\sigma}_i, \tilde{\sigma}_i, ovk_i)$. Since $\mathcal{A}$ outputs $N+1$ unlinked signatures on N public keys, so there exits at least one of ovk_i in σ_i which is not produced by $\mathcal{O}_{\text{join}}$. We assume it is ovk^*. Hence, the probability of $\hat{pk}_j^* = pk_j \odot H(ovk^*)$ has been input to $\hat{\mathcal{O}}_{\mathsf{corrupt}}$ and $\hat{\mathcal{O}}_{\text{sign}}$ is negligible for all $j \in [N]$. Furthermore, if ovk^* is the Ith query of H, which happens with probability at least $\frac{1}{q_h}$, then $\{\hat{pk}_j^*\}_{j\in[N]}$ are all query outputs of $\hat{\mathcal{O}}_{\mathsf{join}}$. Hence, $\mathcal{B}$ can outputs a successful forgery $(m^*, \hat{\sigma}^*, \hat{T}^* = \{\hat{pk}_j^*\}_{j\in[N]})$ if $H(ovk^*) = h'$; else it returns $\perp$.

It is straightforward to verify that $\mathcal{B}$'s simulation for Game 4 is perfect, we can conclude

$$\Pr[S_4] = q_h \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{forge}}$$

Putting all the above together, the theorem immediately follows. □

Theorem 3.3. *Our LRS is nonslanderable in the random oracle model if the underlying one-time signature is unforgeable .*

Proof. If there exists an adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_{\text{join}}, \mathcal{O}_{\text{corrupt}}$ and $\mathcal{O}_{\text{sign}}$ can break the nonslanderability of LRS, then we can construct an adversary $\mathcal{B}$ allowed to query the signature once for any message of his choosing to break the unforgeability of OTS with the same advantage, implying that $\mathsf{Adv}_{\mathcal{B}}^{\text{slander}}$ must be negligible.

Given a public key ovk^* of OTS, $\mathcal{B}$ interacts with $\mathcal{A}$ with the aim to forge (m^*, σ^*) such that OTS.$\mathsf{Vrfy}(ovk^*, m^*, \sigma^*) = accept$.

1. Setup: $\mathcal{B}$ runs $\hat{pp} \leftarrow$ RS.Setup and chooses a hash function $H : OVK \rightarrow \hat{PK}$, where H is modeled as random oracle and OVK and $\hat{PK}$ are public key spaces of OTS and RS respectively. $\mathcal{B}$ then sends $pp = \hat{pp}$ to $\mathcal{A}$.

2. Oracle Simulation:

Joining Oracle $\mathcal{O}_{\text{join}}$: $\mathcal{B}$ initializes an empty set JO. Upon receiving a joining query, $\mathcal{B}$ samples $pk \leftarrow \hat{PK}$. $\mathcal{B}$ then sends it to $\mathcal{A}$ and stores it in JO.

This change cannot be distinguished from the real game since $pk = \hat{pk} \odot H(ovk)$ in real game and H is the random oracle.

Corruption Oracle $\mathcal{O}_{\text{corrupt}}$: $\mathcal{B}$ initializes an empty set CO. Upon receiving a corruption query pk, if $pk \in JO$, $\mathcal{B}$ runs $(ovk, osk) \leftarrow$ OTS.KeyGen and $(\hat{pk}, \hat{sk}) \leftarrow$ RS.KeyGen, computes $h = pk \odot \hat{pk}$, sets $sk = (\hat{sk}, ovk, osk)$. $\mathcal{B}$ then sends sk to $\mathcal{A}$ and stores (pk, sk) in CO.

This change is purely conceptual and thus $\mathcal{A}$ cannot distinguish this change from the real game.

Signing Oracle $\mathcal{O}_{\text{sign}}$: Upon receiving $(m, T = \{pk_i\}_{i\in[N]}, pk_s)$, $\mathcal{B}$ runs $(ovk_s,\ osk_s) \leftarrow$ OTS.KeyGen and $(\hat{pk}_s, \hat{sk}_s) \leftarrow$ RS.KeyGen, computes $h = pk_s \odot \hat{pk}_s$ and $\hat{pk}_i = pk_i \odot h$ for other $i \in [N]$. $\mathcal{B}$ then runs $\hat{\sigma} \leftarrow$ RS.$\mathsf{Sign}(\hat{sk}_s, m, \hat{T} = \{\hat{pk}_i\}_{i\in[N]})$ and $\tilde{\sigma} \leftarrow$ OTS.$\mathsf{Sign}(osk_s, \hat{\sigma}, T, ovk_s)$. Finally, $\mathcal{B}$ sends $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$ to $\mathcal{A}$.

This change is purely conceptual and thus $\mathcal{A}$ cannot distinguish this change from the real game.

Random Oracle H: $\mathcal{B}$ initializes an empty set RO. Upon receiving a random oracle query, if it has been queried, $\mathcal{B}$ returns corresponding output in RO; otherwise, $\mathcal{B}$ programs the output as corresponding h if it is the query in $\mathcal{O}_{\mathsf{join}}$ and $\mathcal{O}_{\mathsf{sign}}$, else $\mathcal{B}$ picks $h \leftarrow \hat{PK}$ at random, sends h to $\mathcal{A}$ and stores it in RO.

By the programmability of H we conclude that A cannot distinguish this change from the real game.

3. Challenge: Upon receiving $(T = \{pk_i\}_{i\in[N]}, m, pk \in T)$ from $\mathcal{A}$, $\mathcal{B}$ runs $(\hat{pk}, \hat{sk}) \leftarrow$ RS.KeyGen, programs $H(ovk^*) = pk \odot \hat{pk}$, computes $\hat{pk}_i = pk_i \odot H(ovk^*), i \in [1, ..., N]$ and runs $\hat{\sigma} \leftarrow$ RS.$\mathsf{Sign}(\hat{sk}, m, \hat{T} = \{\hat{pk}_i\}_{i\in[N]})$. $\mathcal{B}$ then queries the one-time signature $\tilde{\sigma}$ on input the message $(\hat{\sigma}, T, ovk^*)$. Finally, $\mathcal{B}$ sends $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk^*)$ to $\mathcal{A}$.

   $\mathcal{B}$ embeds ovk^* into the challenge which is sent to $\mathcal{A}$ by programming ovk^* as the OTS public key to generate the challenge.

4. Output: $\mathcal{A}$ outputs a set of public keys T^*, a message m^* and a signature σ^* and wins if $\mathsf{Link}(\sigma^*, \sigma, m^*, m, T^*, T) = linked$.

$\mathcal{B}$ parses the signature σ^* as $\sigma^* = (\hat{\sigma}^*, \tilde{\sigma}^*, ovk^*)$ and sends $((\hat{\sigma}^*, T^*, ovk^*), \tilde{\sigma}^*)$ to $\mathcal{CH}$. We heve OTS.$\mathsf{Vrfy}(ovk^*, (\hat{\sigma}^*, T^*, ovk^*), \tilde{\sigma}^*) = accept$ according to the definition of $\mathsf{Link}(\sigma^*, \sigma, m^*, m, T^*, T) = linked$. If $\mathcal{A}$ succeeds, $\mathcal{B}$ also succeeds due to the above. We have $\mathsf{Adv}_{\mathcal{A}}^{\text{slander}}$=$\mathsf{Adv}_{\mathcal{B}}^{\text{forge}}$. □

4 Instantiation

We show an instantiation of our construction by using the RS in [ESS+18] and the signature Dilithium [DKL+18].

- $\mathsf{Setup}(1^\lambda)$: On input 1^λ, select the commitment key $ck = \mathbf{G} \leftarrow R_q^{n\times(m\times k\beta)}$, two hash functions $H : \{0,1\}^* \to \mathcal{C}$ and $H' : \{0,1\}^* \to R_q^n$, where H, H' are modeled as random oracles and $\mathcal{C} = \{X^\omega : 0 \le \omega \le 2d - 1\}$ is the challenge space.
- $\mathsf{KeyGen}(pp)$: On input public parameter pp, select $r_i \leftarrow \{-M, ..., M\}^d$ for $i \in [m]$ and set $\mathbf{r} = (r_1, ..., r_m)$, compute $\mathbf{c} = \text{Com}_{ck}(\mathbf{0}; \mathbf{r})$, where $\mathbf{0}$ is the all-zero vector, set $\hat{pk} = \mathbf{c}$, $\hat{sk} = \mathbf{r}$, run $(ovk, osk) \leftarrow$ Dilithium.$\mathsf{KeyGen}(1^\lambda)$. Output $pk = \hat{pk} + H'(ovk)$, $sk = (\hat{sk}, osk, ovk)$.

- $\mathsf{Sign}(sk_s, m, T)$: On input the secret key sk_s, a signing message m and a set of public keys $T = \{pk_i\}_{i\in[N]}$

  1. Compute $\hat{pk_i'} = pk_i - H'(ovk_s)$ for each $i \in [N]$ and set $\hat{T}' = \{\hat{pk_i'}\}_{i\in[N]}$.
  2. Sample $a_{0,1}, ..., a_{k-1,\beta-1} \leftarrow D^d_{12\sqrt{k}}$, compute $a_{j,0} = -\sum_{i=1}^{\beta-1} a_{j,i}$ for $j = 0, ..., k-1$, select $r_{b,i}, r_{c,i} \leftarrow \{-M, ..., M\}^d$ for $i \in [m]$ and set $\mathbf{r}_b = (r_{b,1}, ..., r_{b,m}), \mathbf{r}_c = (r_{c,1}, ..., r_{c,m})$, sample $r_{a,i}, r_{d,i} \leftarrow D^d_{12M\sqrt{2md}}$ for $i \in [m]$ and set $\mathbf{r}_a = (r_{a,1}, ..., r_{a,m})$, $\mathbf{r}_d = (r_{d,1}, ..., r_{d,m})$, compute
  $$\begin{aligned} A &= \mathrm{Com}_{ck}(a_{0,0}, ..., a_{k-1,\beta-1}; \mathbf{r}_a) \\ B &= \mathrm{Com}_{ck}(\delta_{s_0,0}, ..., \delta_{s_{k-1},\beta-1}; \mathbf{r}_b) \\ C &= \mathrm{Com}_{ck}(\{a_{j,i}(1-2\delta_{j,i})\}_{j,i=0}^{k-1,\beta-1}; \mathbf{r}_c) \\ D &= \mathrm{Com}_{ck}(-a_{0,0}^2, ..., -a_{k-1,\beta-1}^2; \mathbf{r}_d), \end{aligned}$$
  where $\delta_{j,i}$ is Kronecker's delta, $\delta_{j,i} = 1$ if $j = i$ and $\delta_{j,i} = 0$ otherwise. Sample $\rho_{j,i} \leftarrow D^{md}_{12M\sqrt{3md/k}}$ for $i \in [m]$ and set $\rho_j = (\rho_{j,1}, ..., \rho_{j,m})$, compute $E_j = \sum_{i=0}^{N-1} p_{i,j}c_i + \mathrm{Com}(\mathbf{0}; \rho_j)$ for $j = 0, ..., k-1$, where $p_{i,j}$ is computed by $p_i(x) = \prod_{j=0}^{k-1}(x \cdot \delta_{s_j,i_j} + a_{j,i_j}) = \prod_{j=0}^{k-1} x \cdot \delta_{s_j,i_j} + \sum_{j=0}^{k-1} p_{i,j}x^j = \delta_{s,i}x^k + \sum_{j=0}^{k-1} p_{i,j}x^j, i \in [N]$. Compute $x = H'(ck, m, \hat{T}', A, B, C, D, \{E_j\}_{j=0}^{k-1})$, $f_{j,i} = x \cdot \delta_{s_j,i_j} + a_{j,i_j}, \forall j, \forall i \neq 0$, $\mathbf{z}_b = x \cdot \mathbf{r}_b + \mathbf{r}_a$, $\mathbf{z}_c = x \cdot \mathbf{r}_c + \mathbf{r}_d$, $\mathbf{z} = x^k \cdot \hat{sk}_s - \sum_{j=0}^{k-1} x^j \cdot \rho_j$. Set CMT$= (A, B, C, D, \{E_j\}_{j=0}^{j=k-1})$ and RSP $= (\{f_{j,i}\}_{j=0,i=1}^{k-1,\beta-1}, \mathbf{z}, \mathbf{z}_b, \mathbf{z}_c)$.
  3. Repeat step 2 L times in parallel and get $\{\mathrm{CMT}_l\}_{l\in[L]}$, $\mathbf{x} = \{x_l\}_{l\in[L]}$ and $\{\mathrm{RSP}_l\}_{l\in[L]}$. If $\mathrm{RSP}_l \neq \perp$ for all $l \in [L]$, set $\hat{\sigma} = (\{\mathrm{CMT}_l\}_{l\in[L]}, \mathbf{x}, \{\mathrm{RSP}_l\}_{l\in[L]})$. Otherwise, go to Step 2 (repeat at most $\frac{-\lambda}{\log(1-1/\mathcal{M}^2)}$).
  4. Run $\hat{\sigma} \leftarrow \mathrm{Dilithium}.\mathsf{Sign}(osk, (\hat{\sigma}, T, ovk_s))$.
  5. Output $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$.

- $\mathsf{Vrfy}(T, m, \sigma)$: On input the set of public keys $T = \{pk_i\}_{i\in[N]}$, a signing message m and the signature σ, parse σ as $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk)$ and compute $\hat{pk}_i' = pk_i - H(ovk)$ for each $i \in [N]$, then

  1. For every $(\mathrm{CMT}_l, x_l, \mathrm{RSP}_l)$, $l \in [L]$ Check whether
     - $f_{j,0} = x - \sum_{i=1}^{\beta-1} f_{j,i}$ for $j = 0, ..., k-1$
     - $xB + A = \mathrm{Com}_{ck}(f_{0,0}, ..., f_{k-1,\beta-1}; \mathbf{z}_b)$
     - $xC + D = \mathrm{Com}_{ck}(f_{0,0}(x - f_{0,0}), ..., f_{k-1,\beta-1}(x - f_{k-1,\beta-1}); \mathbf{z}_c)$
     - $||f_{j,i}|| \leq 60\sqrt{dk}, \forall j, \forall i \neq 0$
     - $||f_{j,0}|| \leq 60\sqrt{dk(\beta-1)}, \forall j$
     - $||\mathbf{z}||, ||\mathbf{z}_s||, ||\mathbf{z}_c|| \leq 24\sqrt{3}Mmd$
     - $\sum_{i=0}^{N-1}(\prod_{j=0}^{k-1} f_{j,i_j})c_i - \sum_{j=0}^{k-1} E_j x^j = \mathrm{Com}_{ck}(\mathbf{0}; \mathbf{z})$ for $i = (i_0, ..., i_{k-1})$

     if not, return *reject*.
  2. Run $accept/reject \leftarrow \mathrm{Dilithium}.\mathsf{Vrfy}(ovk, (\hat{\sigma}, T, \hat{pk}))$.
  3. If neither 1 and 2 return *reject*, return *accept*.

- $\mathsf{Link}(m_1, m_2, \sigma_1, \sigma_2, T_1, T_2)$: On input two sets of public keys T_1, T_2, two signing messages m_1, m_2 and their signatures σ_1, σ_2, run $\mathsf{Vrfy}(m_1, \sigma_1, T_1)$ and $\mathsf{Vrfy}(m_2, \sigma_2, T_2)$. Parse σ_1 and σ_2 as $\sigma_1 = (\hat{\sigma}_1, \tilde{\sigma}_1, ovk_1)$ and $\sigma_2 = (\hat{\sigma}_2, \tilde{\sigma}_2, ovk_2)$. Compare ovk_1 and ovk_2. Return *linked* if $\mathsf{Vrfy}(m_1, \sigma_1, T_1) = \mathsf{Vrfy}(m_2, \sigma_2, T_2) = accept$ and $ovk_1 = ovk_2$.

5 Comparison and Experimental Analysis

5.1 Comparison

We compare the size of public key and signature of existing lattice-based LRS in Table 1. Like [ESS+18], our scheme is able to adjust the base representations for user indices and results in different asymptotic growths of signature length.

Table 1: Comparison of Lattice-Based Linkable Ring Signature

Scheme	Public Key Size	Signature Size	Assumption
[YAL+17]	$n\log p$	$2m(\log q)^2 \cdot \boxed{\log N}$	SIS/LWR
[ZZTA17]	$md\log q$	$m^2 d\log q \cdot \boxed{\log N}$	I(f)-SVP$_\gamma$
[BLO18]	$nd\log q$	$m\log(2\sigma\sqrt{d}) \cdot \boxed{N}$	M-SIS/M-LWE
[TSS+18]	$d\log q$	$m\log(\eta\sigma\sqrt{d}) \cdot \boxed{N}$	R-SIS
[LAZ18]	$d\log q$	$(256 + 2d\log q) \cdot \boxed{N}$	NTRU
Ours	$nd\log q$	$(nd\log q + \beta d\log\sqrt{144L\log_\beta N})L \cdot \boxed{\log_\beta N}$	M-SIS

1. Constant terms are omittd.
2. n and m denote the row and column of matrix on $\mathbb{Z}_q$ or R_q, d denotes the dimension of polynomials, β denotes the base representations, σ and L denote the standard deviation of discrete normal distribution and the number of repetitions in our scheme.

5.2 Experimental Analysis

In order to compare the size and running time of the LRS scheme and the underlying RS scheme, we implement the instantiation of our scheme and the RS scheme [ESS+18] based on the NTL library and the source code of Dilithium.

Parameter setting and Experimental results. We set the parameter in the part of RS as in Table 3 and adopt the very high version of Dilithium.

Table 2: Experimental Parameter

Parameters	M	n	q	m	l	d	L	k
Values	100	9	2^{60}	71	100	76	17	2

Table 3: Experimental Results

		Size(KB)		Time(ms)		
	Ring Size	Public Key	Signature Size	KeyGen	Sign	Vrfy
Ring Signature	2^6	5.13	1083	84.04	603.18	418.94
	2^8	5.13	1100	84.04	1195.96	904.09
	2^{10}	5.13	1135	84.04	2993.27	2524.47
	2^{12}	5.13	1205	84.04	10310.9	9268.26
Linkable Ring Signature	2^6	5.13	1088	84.89	604.94	421.47
	2^8	5.13	1105	84.89	1201.53	906.17
	2^{10}	5.13	1140	84.49	2995.47	2527.49
	2^{12}	5.13	1210	84.49	10313.2	9272.20

As depicted in Table 3, the experimental results show the performance of the LRS scheme is close to the performance of the underlying RS.

References

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, pages 415–432, 2002.

[ASY06] Man Ho Au, Willy Susilo, and Siu-Ming Yiu. Event-oriented k-times revocable-iff-linked group signatures. In *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, pages 223–234, 2006.

[BDR15] Priyanka Bose, Dipanjan Das, and Chandrasekaran Pandu Rangan. Constant size ring signature without random oracle. In *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*, pages 230–247, 2015.

[BK10] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive*, 2010:86, 2010.

[BLO18] Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. *IACR Cryptology ePrint Archive*, 2018:107, 2018.

[CWLY06] Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006*, pages 297–302, 2006.

[CYH05] Sherman S. M. Chow, Siu-Ming Yiu, and Lucas Chi Kwong Hui. Efficient identity based ring signature. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 499–512, 2005.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 40–56, 2013.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 609–626, 2004.

[ESS+18] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. *IACR Cryptology ePrint Archive*, 2018:773, 2018.

[FZ12] Matthew K. Franklin and Haibin Zhang. A framework for unique ring signatures. *IACR Cryptology ePrint Archive*, 2012:577, 2012.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 253–280, 2015.

[LASZ14] Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Trans. Knowl. Data Eng.*, 26(1):157–165, 2014.

[LAZ18] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. *IACR Cryptology ePrint Archive*, 2018:857, 2018.

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 1–31, 2016.

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, pages 325–335, 2004.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 738–755, 2012.

[Noe15] Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptology ePrint Archive*, 2015:1098, 2015.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 552–565, 2001.

[SALY17] Shifeng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, pages 456–474, 2017.

[TSS+18] Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0). In *Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings*, pages 558–576, 2018.

[TW05] Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *Information Security Practice and Experience, First International Conference, ISPEC 2005, Singapore, April 11-14, 2005, Proceedings*, pages 48–60, 2005.

[vS13] Nicolas van Saberhagen. Cryptonote v 2.0. https://cryptonote.org/whitepaper.pdf, 2013.

[YAL+17] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Lattice-based techniques for accountable anonymity: Composition of abstract stern's protocols and weak PRF with efficient protocols from LWR. *IACR Cryptology ePrint Archive*, 2017:781, 2017.

[YLA+13] Tsz Hon Yuen, Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Efficient linkable and/or threshold ring signature without random oracles. *Comput. J.*, 56(4):407–421, 2013.

[ZZTA17] Huang Zhang, Fangguo Zhang, Haibo Tian, and Man Ho Au. Anonymous post-quantum cryptocash. *IACR Cryptology ePrint Archive*, 2017:716, 2017.

A Comment on [LAZ18].

Lu et al. [LAZ18] adopted the definitions of anonymity, linkability and nonslanderability from [LASZ14]. Then, they gave a theorem which shows that the unforgeability is implied by linkability and nonslanderability. We first review the definition of linkability and the theorem as follows:

The linkability in [LAZ18] is defined in terms of the following game between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$:

1. Setup. $\mathcal{CH}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$.
2. Query. $\mathcal{A}$ is given access to $\mathcal{O}_{\text{join}}, \mathcal{O}_{\text{corrupt}}, \mathcal{O}_{\text{sign}}$ and may query the oracles in an adaptive manner.
3. Output. $\mathcal{A}$ outputs two sets $\{T_1, m_1, \sigma_1\}$ and $\{T_2, m_2, \sigma_2\}$, where T_1 and T_2 are two sets of public keys, m_1 and m_2 are messages, σ_1 and σ_2 are signatures.

$\mathcal{A}$ wins the game if

- all public keys in T_1 and T_2 are query outputs of $\mathcal{O}_{\text{join}}$;
- $\mathsf{Vrfy}(T_1, m_1, \sigma_1) = \textit{accept}$;
- $\mathsf{Vrfy}(T_2, m_2, \sigma_2) = \textit{accept}$;
- $\mathcal{A}$ queried $\mathcal{O}_{\text{corrupt}}$ less than two times; and
- $\mathsf{Link}(m_1, \sigma_1, m_2, \sigma_2) = \textit{unlinked}$.

The advantage of $\mathcal{A}$, denoted as $\mathsf{Adv}_{\mathcal{A}}^{\text{link}}$, is defined by the probability that $\mathcal{A}$ wins in the above game.

Definition A.1 ([LAZ18], Definition 11)**.** A LRS scheme is linkable if for any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{link}}$ is negligible in λ.

Theorem A.1 ([LAZ18], Theorem 2)**.** *If a LRS scheme is linkable and nonslanderable, it is also unforgeable.*

Issue 1. Theorem A.1 does not hold for the definition of linkability in [LAZ18]. The content of theorem A.1 was introduced in [ASY06] which towards the security definitions in [ASY06]. However, the definition of linkability in [LAZ18] is different from the definition in [ASY06]. In [LAZ18], the adversary $\mathcal{A}$ against unforgeability is allowed to make polynomial many $\mathcal{O}_{\text{corrupt}}$ queries in the unforgeability game, whereas the adversary $\mathcal{B}$ against linkability is restricted to make at most one $\mathcal{O}_{\text{corrupt}}$ query in the linkability game. This means $\mathcal{B}$ cannot simulate $\mathcal{O}_{\text{corrupt}}$ for $\mathcal{A}$ and thus $\mathcal{B}$ cannot run $\mathcal{A}$ to break the linkability.

Issue 2. There is a gap in the proof of linkability. They reduced the linkability of the LRS to the collision resistance of CH+ as follows: First, they embedded the collision resistance challenge hk_c into one of the public keys pk_I by computing $pk_I = hk_c \oplus H(ovk_I)$. Second, the adversary $\mathcal{A}$ outputs two signatures and they concluded that at least one of the output signatures should be generated from the secret key that $\mathcal{A}$ does not obtain because $\mathcal{A}$ is allowed to make at most one $\mathcal{O}_{\text{corrupt}}$ query. The signature is denoted as (m^*, σ^*, T^*), where $\sigma^* = \{(m_1^*, r_1^*), ..., (m_N^*, r_N^*), ovk^*, \tilde{\sigma}^*\}$. Finally, they assumed $pk_I \in T^*$ and used (m^*, σ^*, T^*) to find a collision of hk_c according to the General Forking Lemma.

However, the collision resistance challenge may not be embedded into the output signatures of $\mathcal{A}$. This means that hk_c is not used to generate the signature (m^*, σ^*, T^*) although $pk_I \in T^*$. The reason is that ovk^* may not equal to $ovk_{\mathcal{I}}$ and thus $hk_c \neq hk_i = pk_i \oplus H(ovk^*)$ for every $i \in [N]$. According to the signing algorithm of the LRS in [LAZ18], we can conclude that hk_c is independent of σ^* if $ovk^* \neq ovk_I$. Thus, the collision resistance of CH+ cannot be broken although $\mathcal{A}$ has broken the linkability of the LRS.