Adding Linkability to Ring Signatures with One-Time Signatures

Xueli Wang^{1,2,3}, Yu Chen^{1,2,3,4*}, and Xuecheng Ma^{1,3}

^{1} State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
^{2} State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
^{3} School of Cyber Security, University of Chinese Academy of Sciences, Beijing 101408, China
wangxueli@iie.ac.cn, maxuecheng@iie.ac.cn
^{4} Ant Financial, Hangzhou 310012, China
cycosmic@gmail.com

Abstract. We propose a generic construction that adds linkability to any ring signature scheme with one-time signature scheme. Our construction has both theoretical and practical interest. In theory, the construction gives a formal and cleaner description for constructing linkable ring signature from ring signature directly. In practice, the transformation incurs a tiny overhead in size and running time. By instantiating our construction using the ring signature scheme [13] and the one-time signature scheme [12], 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 100 bit security, our signature size is only 4 MB.
In addition, when proving the linkability we develop a new proof technique in the random oracle model, which might be of independent interest.

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

1 Introduction

Ring signature (RS) was first proposed by Rivest et al. [22], which allows a signer to sign a message on behalf of a self-formed group. RS provides 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: anonymity against probabilistic polynomial adversary and anonymity against unbounded adversary.

* Corresponding author.

As an extension of RS, Liu et al. [18] 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 [26] to ensure that the voters can vote anonymously and will not repeat their votes. In a more popular setting, cryptocurrency, LRS plays a crucial role in providing anonymity of spenders while defeating the double-spending attack, and hence LRS has received much attention with the rise of Monero [21] and other cryptocurrencies based on CryptoNote protocol [23].

The richer functionality of LRS makes it suited for a wide range of privacy-preserving applications, but also renders it relatively complicated to realize. The only known generic construction of LRS is proposed by Franklin and Zhang [14], but the linkability of their work is restricted to the same message, which may not be suited for cryptocurrency. In addition, there are some existing works that proposed a RS scheme firstly and then extended it to the linkable version, such as [26,19,3], but these transformations are not generic. In light of the state of affairs described above, we are motivated to consider the generic construction of LRS, in particular, whether LRS can be built from RS. 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 question. The contribution of this paper is threefold:

– We give a generic construction that adds linkability to any RS scheme with one-time signature (OTS) scheme. 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 (CH$^+$) which is used as the underlying primitive by a recent generic construction of LRS [19]. 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. additive group with $+$) and the distribution of public keys generated by the key generation algorithm should be statistically close to the uniform distribution over $\hat{PK}$, which are naturally satisfied by most of RS schemes [1,15,5,6,16,13]. Moreover, almost all the known RS schemes [22,10,9,8,6,15,5,16] provide the anonymity against unbounded adversaries but LRS schemes with the anonymity against unbounded adversaries are only given in [18,17,25] recently. Our transformation preserves the same anonymity of the underlying RS schemes and hence it can be used to enrich LRS schemes with the anonymity against unbounded adversaries. Finally, our transformation introduces a small overhead on the size and running time compared to the underlying RS schemes.

– We develop a new proof approach to reduce the linkability of LRS to the unforgeability of RS, which can bridge the gap between all public keys must be generated honestly in security definition of RS and generated by adversaries in reduction. Although this approach may do not help in proving many other existing LRS schemes, it gives inspiration for designing other LRS schemes from RS schemes with other primitives. In addition, we believe the new proof approach might be of independent interest in the random oracle model.
– By instantiating our generic transformation based on the RS scheme in [13] and the OTS scheme Dilithium[5] [12] , we obtain a lattice-based LRS scheme whose signature size is logarithmic in the number of ring members. Compared with the underlying primitive CH$^+$ of [19], which only can be used to construct LRS scheme with linear signature size, RS schemes can be instantiated with logarithmic signature size and transformed to LRS schemes with the same size by our construction. Hence, the signature size of our scheme is very short even for a large ring, for 2^{30} ring members and 100 bit security, our signature size is only 4 MB comparing to 166 MB[6] in the prior shortest lattice-based LRS scheme [27]. In addition, the experimental results demonstrate the concrete scheme is practical.

1.2 Technique Overview

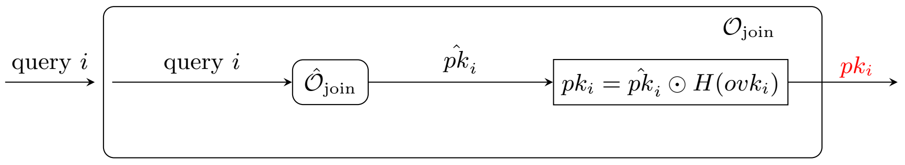
To describe our construction, it is instructive to recall the generic construction of LRS in [19], which is called Raptor. In [19], they introduced the concept of CH$^+$ and gave a generic construction of LRS based on CH$^+$ and OTS. In the key generation procedure, the signer generates a hash key hk and its trapdoor td, and a key pair (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 s firstly 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 the ring 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 gets the one-time signature $\tilde{\sigma}$. The final 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.

[5] Dilithium is a signature scheme, we use it as a OTS scheme.
[6] The signature size is from [16], the RS scheme in [16] is the major component of [27] and they have the same asymptotic size.

Inspired by the idea in [19], 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 with respect to a ring of which the adversary does not have any associated 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.

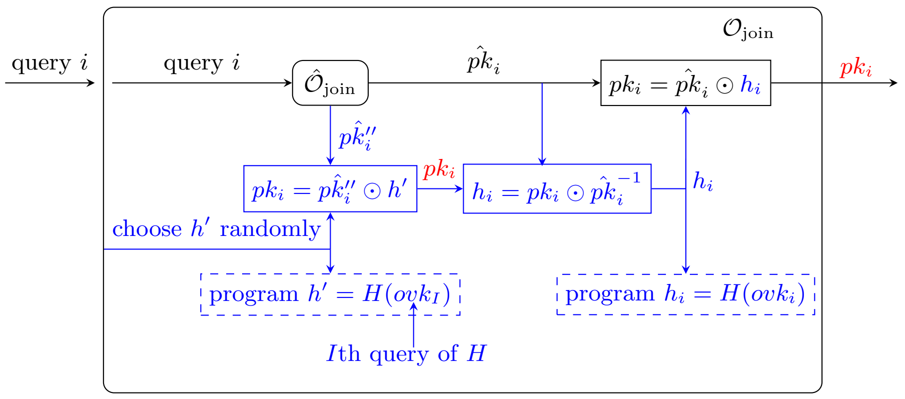
Construction Sketch. In the key generation procedure, the user firstly generates the key pair $(\hat{pk}, \hat{sk})$ and (ovk, osk) of LRS and OTS respectively. Then, he computes the public key $pk = \hat{pk} \odot H(ovk)$ and sets the secret key $sk = (\hat{sk}, osk, ovk)$. In the signing procedure, if the signer signs a message m on the ring $T = \{pk_i\}_{i\in[N]}$, he firstly reconstructs a new ring $\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}$, where $\hat{pk}'_i$ is equal to $pk_i \odot (H(ovk_s))^{-1}$, where $(H(ovk_s))^{-1}$ is the inverse element of $H(ovk_s)$ in the group $\hat{PK}$. It is easy to see that $\hat{pk}_i = \hat{pk}'_i$ only when $i = s$ and hence the signer knows the associated secret key $\hat{sk}_s$ of $\hat{pk}'_s$. Then, he runs RS.Sign on $\hat{T}'$ with $\hat{sk}_s$ to get the ring signature $\hat{\sigma}$. Finally, the signer runs OTS.Sign on the message $(\hat{\sigma}, T, ovk_s)$ with the secret key osk_s, then he gets the one-time signature $\tilde{\sigma}$ and sets the final signature $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk_s)$, 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. Suppose there exists an adversary $\mathcal{A}$ that breaks the linkability of our LRS scheme. Then, we construct an adversary $\mathcal{B}$ that breaks the unforgeability of underlying RS scheme by using $\mathcal{A}$. If $\mathcal{A}$ succeeds, i.e., $\mathcal{A}$ 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. $\hat{\sigma}^*$ contained in σ^* is set 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, $\mathcal{A}$ 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 associated 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.



query i
$\mathcal{O}_{\text{join}}$
query i
$\hat{\mathcal{O}}_{\text{join}}$
$\hat{pk}_i$
$pk_i = \hat{pk}_i \odot H(ovk_i)$
pk_i

Fig. 1. real $\mathcal{O}_{\text{join}}$



query i
$\mathcal{O}_{\text{join}}$
query i
$\hat{\mathcal{O}}_{\text{join}}$
$\hat{pk}_i$
$pk_i = \hat{pk}_i \odot h_i$
pk_i
$\hat{pk}''_i$
$pk_i = \hat{pk}''_i \odot h'$
pk_i
$h_i = pk_i \odot \hat{pk}_i^{-1}$
h_i
choose h' randomly
program $h' = H(ovk_I)$
program $h_i = H(ovk_i)$
Ith query of H

Fig. 2. Simulation of $\mathcal{O}_{\text{join}}$

1.3 Related Work

Ring Signature. Abe et al. [1] 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 [15] 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. [5] 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 [6] proposed the first lattice-based RS scheme from ring trapdoor functions whose public keys are matrices over a group. Libert et al. [16] proposed the first lattice-based RS scheme whose signature size is logarithmic in the number of ring members. The scheme is from zero-knowledge arguments for lattice-based accumulators and the public keys of it 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. Tsang and Wei [26] extended the generic RS constructions in [10] to their linkable version, but their schemes are under a weak security model which does not consider the nonslanderability. Chow et al. [7] proposed escrowed linkability of RS which can be used in spontaneous traceable signature and anonymous verifiably encrypted signature. Yuen et al. [28] proposed a LRS scheme whose signature size is square root of the number of ring members. Sun et al. [24] presented an accumulator-based LRS scheme whose signature size is independent of the number of ring members. Yang et al. [27] presented a construction of weak-PRF from LWR and designed a LRS scheme based on lattice by combining with an accumulator scheme in [16] and the supporting ZKAoKs. Zhang et al. [29] proposed an anonymous post-quantum cryptocash which contains an ideal lattice-based LRS scheme. Baum et al. [4] 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. [25] proposed a post-quantum one-time LRS scheme, which generalized a practical lattice-based digital signature BLISS [11] to LRS and is successfully applied to the privacy protection protocol which is called lattice ringCT v1.0. Recently, Backes et al. [3] proposed the first construction of logarithmic-size ring signatures which do not rely on a trusted setup or the random oracle heuristic and extended their scheme to the setting of linkable ring signatures.

Note 1. 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

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 $\{1, ..., N\}$. 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 the algorithm outputs s. We denote a negligible function by $\mathsf{negl}(\lambda)$, which is a function $g(\lambda) = O(\lambda^c)$ for some constant c. 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 $f(x) = a_0 + a_1X + \cdots + a_{n-1}X^{n-1}$

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. 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 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 following:

- 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 associated 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.

Anonymity. Anonymity can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$:

1. Setup: $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter pp to $\mathcal{A}$.
2. Query: $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$ according to any adaptive strategy.
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 1. *A* RS *scheme is said to be anonymous (resp.anonymous against unbounded adversaries) if for any PPT adversary (unbounded adversary) $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}}$ is negligible in λ.*

Unforgeability. Unforgeability is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$.

1. Setup: $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter pp to $\mathcal{A}$.
2. Query: $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}, \mathcal{O}_{\text{corrupt}}$ and $\mathcal{O}_{\text{sign}}$ according to any adaptive strategy.
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. *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.*

2.3 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 $\mathsf{Vrfy}(T, m, \mathsf{Sign}(sk_s, m, T)) = accept$ where $T = \{pk_i\}_{i\in[N]}$.

Anonymity. Anonymity of LRS is the same as that of RS.

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: $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter pp to $\mathcal{A}$.
2. Query: $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$, $\mathcal{O}_{\text{corrupt}}$, $\mathcal{O}_{\text{sign}}$ according to any adaptive strategy.
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 $\mathcal{O}_{\text{join}}$;
- $\mathsf{Vrfy}(m_i, \sigma_i, T) = accept$ for all $i \in [N+1]$;
- $\mathsf{Link}(m_i, m_j, \sigma_i, \sigma_j) = unlinked$ 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 3. *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. Nonsladerabiliy can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{CH}$:

1. Setup: $\mathcal{CH}$ runs Setup with security parameter 1^λ and sends the public parameter pp to $\mathcal{A}$.
2. Query: $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{\text{join}}$, $\mathcal{O}_{\text{corrupt}}$, $\mathcal{O}_{\text{sign}}$ according to any adaptive strategy.
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 signature σ to $\mathcal{A}$.
4. Output: $\mathcal{A}$ outputs a messages/signature pair (m^*, σ^*, T^*).

$\mathcal{A}$ wins if

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

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 4 (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.*

We adopt the same definitions of linkability and nonslanderability as in [2]. Typically, the linkability definition in [2] allows the adversary to make polynomially many queries to $\mathcal{O}_{\text{corrupt}}$ which is necessary because in the definition of unforgeability the adversary has the same ability.

Hence, the *unforgeability* of LRS can be implied by the linkability and the nonslanderability according to [2].

2.4 Assumption and Rejection Sampling

For the sake of completeness, we state the following lattice assumption, commitment scheme in [13] and rejection sampling lemma.

Definition 5 (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 $\mathbf{Ax} = \mathbf{0}$ mod q and $0 < ||\mathbf{x}|| \leq \theta$.*

Lemma 1 ([20]). *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:*

$\mathcal{A}$: *$\boldsymbol{v} \leftarrow h$; $\mathbf{z} \leftarrow D^d_{v,\sigma}$; output $(\mathbf{z},\mathbf{v})$ with probability* $\min\{\frac{D^d_\sigma(\mathbf{z})}{\mathcal{M}D^d_{\mathbf{v},\sigma}(\mathbf{z})}, 1\}$
$\mathcal{F}$: *$\boldsymbol{v} \leftarrow h$; $\mathbf{z} \leftarrow D^d_{\sigma}$; 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 $\frac{1-2^{-100}}{\mathcal{M}}$.

2.5 Commitment Scheme

Definition 6. *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*

$$\text{Com}_{ck}(\mathbf{m};\mathbf{r}) = \mathbf{G}\cdot(\mathbf{r}; m_1, \cdots, m_v)^{\text{T}} \in R_q^n$$

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

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 $\hat{pp} \leftarrow \mathrm{RS}.\mathsf{Setup}(1^\lambda)$. It also chooses a hash function $H : OVK \rightarrow \hat{PK}$, 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 $(\hat{pk}, \hat{sk}) \leftarrow \mathrm{RS}.\mathsf{KeyGen}(\hat{pp})$. Then, the algorithm runs $(ovk, osk) \leftarrow \mathrm{OTS}.\mathsf{KeyGen}$. 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 $\hat{pk}'_i = pk_i \odot (H(ovk_s))^{-1}$ for $i \in [N]$ and sets $\hat{T}' = \{\hat{pk}'_i\}_{i \in [N]}$. Next, the algorithm runs
$$\hat{\sigma} \leftarrow \mathrm{RS}.\mathsf{Sign}(\hat{sk}_s, m, \hat{T}'),$$
$$\tilde{\sigma} \leftarrow \mathrm{OTS}.\mathsf{Sign}(osk_s, \hat{\sigma}, T, ovk_s).$$
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 $\hat{pk}'_i = pk_i \odot (H(ovk))^{-1}$ for $i \in [N]$. Next, it runs $\mathrm{RS}.\mathsf{Vrfy}(\hat{T}' = \{\hat{pk}'_i\}_{i \in [N]}, m, \hat{\sigma})$ and $\mathrm{OTS}.\mathsf{Vrfy}(ovk, (\hat{\sigma}, T, ovk), \tilde{\sigma})$. Finally, it outputs *accept* if $\mathrm{RS}.\mathsf{Vrfy}$ returns *accept* and $\mathrm{OTS}.\mathsf{Vrfy}$ 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 1. *Our LRS scheme is anonymous (resp. anonymous against unbounded adversary) if the underlying RS scheme is anonymous (resp. anonymous against unbounded adversary).*

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. This simulation is indistinguishable from the real game. According to the distribution of $\hat{pk}$ is close to the uniform distribution over $\hat{PK}$, we can get that $\hat{pk} \odot H(ovk)$ is also close to the uniform distribution no matter which ovk is chosen since $\hat{pk} \in \hat{PK}$. Hence, 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 $\hat{pk}'_i = pk_i \odot (H(ovk))^{-1}$ for all $i \in [N]$ and sets $\hat{T}' = \{\hat{pk}'_i\}_{i\in[N]}$. $\mathcal{B}$ then sends $(\hat{T}', m)$ to $\mathcal{CH}$ 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 $\mathcal{CH}$. $\mathcal{A}$ 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 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. $\mathcal{CH}$ 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 associated 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 associated 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 $\hat{pk}'_i = pk_i \odot (H(ovk_s))^{-1}$ 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 $T = \{pk_i\}_{i\in[N]}$.

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 $\mathcal{O}_{\text{join}}$) as corresponding h which is computed in Game 3. By the programmability of H, we have $\Pr[S_4] = \Pr[S_3]$.

Lemma 2. *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 associated output in RO; otherwise, $\mathcal{B}$ programs the output as associated 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 $(\hat{T}' = \{\hat{pk}'_i = pk_i \odot h_i^{-1}\}_{i \in [N]}, m, \hat{pk}_s \in \hat{T}')$ 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 $\hat{pk}_j^* = pk_j \odot (H(ovk^*))^{-1}$ 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. *Our LRS is nonslanderable in the random oracle model if the underlying one-time signature is unforgeable .*

Proof. The proof of nonslanderability is trivial. Due to the page limitation, we omit the details of this proof and just provide a brief sketch here. We embedded the challenge ovk of unforgeability into the challenge $\sigma = (\hat{\sigma}, \tilde{\sigma}, ovk)$ of nonslanderability by querying the one-time signature $\tilde{\sigma}$ on input the message $(\hat{\sigma}, T, ovk)$ and programming the random oracle H. If the unforgeability is broken, that is the adversary outputs (T^*, m^*, σ^*) which satisfies $\mathsf{Link}(\sigma^*, \sigma, m^*, m, T^*, T) =$ *linked*. Then, $((\hat{\sigma}^*, T^*, ovk^*), \tilde{\sigma^*})$ is a valid forgery of OTS, where $\hat{\sigma}^*, \tilde{\sigma}^*, ovk^*$ are contained in $\sigma^* = (\hat{\sigma}^*, \tilde{\sigma}^*, ovk^*)$.

4 Instantiation

We give an instantiation of our construction by using the RS in [13] and the (one-time) signature Dilithium [12].

- $\mathsf{Setup}(1^\lambda)$: On input 1^λ, select the commitment key $ck = \mathbf{G} \leftarrow R_q^{n\times(m+k\beta)}$, two hash functions $H : \{0,1\}^* \to \mathcal{C}$ and $H' : \{0,1\}^* \to R_q^n$, where $k = \log_\beta N$, $\mathcal{C} = \{X^\omega : 0 \le \omega \le 2d-1\}$ 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 $\mathbf{c} = \mathrm{Com}_{ck}(\mathbf{0}; \mathbf{r})$, where $\mathbf{0}$ is the all-zero vector, set $\hat{pk} = \mathbf{c}$, $\hat{sk} = \mathbf{r}$, run $(ovk, osk) \leftarrow \text{Dilithium}.\mathsf{KeyGen}(1^\lambda)$. 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 $A = \mathrm{Com}_{ck}(a_{0,0}, ..., a_{k-1,\beta-1}; \mathbf{r}_a)$, $B = \mathrm{Com}_{ck}(\delta_{s_0,0}, ..., \delta_{s_{k-1},\beta-1}; \mathbf{r}_b)$, $C = \mathrm{Com}_{ck}(\{a_{j,i}(1-2\delta_{j,i})\}_{j,i=0}^{k-1,\beta-1}; \mathbf{r}_c)$, $D = \mathrm{Com}_{ck}(-a^2_{0,0}, ..., -a^2_{k-1,\beta-1}; \mathbf{r}_d)$, 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 $\hat{\sigma} \leftarrow \text{Dilithium}.\mathsf{Sign}(osk, (\hat{\sigma}, T, ovk_s))$.
  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 $j = 0, ..., k-1$
     - $xB + A = \mathrm{Com}_{ck}(f_{0,0}, ..., f_{k-1,\beta-1}; \mathbf{z}_b)$
     - $xC + D = \mathrm{Com}_{ck}(f_{0,0}(x - f_{0,0}), ..., f_{k-1,\beta-1}(x - f_{k-1,\beta-1}); \mathbf{z}_c)$
     - $||f_{j,i}|| \le 60\sqrt{dk}, \forall j, \forall i \neq 0$ and $||f_{j,0}|| \le 60\sqrt{dk(\beta-1)}, \forall j$

- $||\mathbf{z}||, ||\mathbf{z}_s||, ||\mathbf{z}_c|| \leq 24\sqrt{3}Mmd$
- $\sum_{i=0}^{N-1}(\prod_{j=0}^{k-1} f_{j,i_j})c_i - \sum_{j=0}^{k-1} E_j x^j = \mathrm{Com}_{ck}(\mathbf{0};\mathbf{z})$ for $i = (i_0, ..., i_{k-1})$

   if not, return *reject*.
2. Run $accept/reject \leftarrow$ 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 Implementation

5.1 Comparison

We compare the size of public key and signature of existing lattice-based LRS in Table 1. Like [13], 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
[27]	$n\log p$	$2m(\log q)^2 \cdot \boxed{\log N}$	SIS/LWR
[29]	$md\log q$	$m^2 d\log q \cdot \boxed{\log N}$	I(f)-SVP$_\gamma$
[4]	$nd\log q$	$m\log(2\sigma\sqrt{d}) \cdot \boxed{N}$	M-SIS/M-LWE
[25]	$d\log q$	$m\log(\eta\sigma\sqrt{d}) \cdot \boxed{N}$	R-SIS
[19]	$d\log q$	$(256 + 2d\log q) \cdot \boxed{N}$	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 [13] based on the NTL library and the source code of Dilithium.

Parameter Setting and Experimental Results. We set the parameters in the part of RS as in Table 2 and adopt the very high version of Dilithium.

Table 2. Experimental Parameter

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

6 Discussion

In this paper, we adds linkability to any compatible ring signature scheme with one-time signature scheme. Essentially, linkability in this paper is only one-time linkability, which means the linkability tag is not bound to a general linkability context such as the ring of possible signers nor the message being signed, but only bound to a signer. Linkability with variable restrictions are available for different applications. One-time linkability may not be applied to some scenarios such as e-voting but it is vital in constructing cryptocurrencies, because a sum of money can be spent by the owner only once no matter to any ring or any transaction.

Acknowledgments. We thank the anonymous reviewers of ISC 2019 for their helpful comments. We are grateful to Rupeng Yang and Man Ho Allen Au for helpful discussions and advices. This work is supported by National Natural Science Foundation of China (Grant No. 61772522), Youth Innovation Promotion Association CAS, Key Research Program of Frontier Sciences, CAS (Grant No. QYZDB-SSW-SYS035).

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: ASIACRYPT. pp. 415–432 (2002)
2. Au, M.H., Susilo, W., Yiu, S.: Event-oriented k-times revocable-iff-linked group signatures. In: ACISP. pp. 223–234 (2006)
3. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: Logarithmic-size, no setup - from standard assumptions. In: EUROCRYPT. pp. 281–311 (2019)
4. Baum, C., Lin, H., Oechsner, S.: Towards practical lattice-based one-time linkable ring signatures. In: ICICS. pp. 303–322 (2018)
5. Bose, P., Das, D., Rangan, C.P.: Constant size ring signature without random oracle. In: ACISP. pp. 230–247 (2015)
6. Brakerski, Z., Kalai, Y.T.: A framework for efficient signatures, ring signatures and identity based encryption in the standard model. IACR Cryptology ePrint Archive **2010**, 86 (2010), http://eprint.iacr.org/2010/086
7. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: VIETCRYPT. pp. 175–192 (2006)
8. Chow, S.S.M., Wei, V.K., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: ASIACCS. pp. 297–302 (2006)
9. Chow, S.S.M., Yiu, S., Hui, L.C.K.: Efficient identity based ring signature. In: ACNS. pp. 499–512 (2005)
10. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: EUROCRYPT. pp. 609–626 (2004)
11. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: CRYPTO. pp. 40–56 (2013)
12. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(1), 238–268 (2018)
13. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: ACNS. pp. 67–88 (2019)
14. Franklin, M.K., Zhang, H.: Unique ring signatures: A practical construction. In: FC. pp. 162–170 (2013)
15. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: EUROCRYPT. pp. 253–280 (2015)
16. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: EUROCRYPT. pp. 1–31 (2016)
17. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. IEEE Trans. Knowl. Data Eng. **26**(1), 157–165 (2014)
18. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In: ACISP. pp. 325–335 (2004)
19. Lu, X., Au, M.H., Zhang, Z.: Raptor: A practical lattice-based (linkable) ring signature. IACR Cryptology ePrint Archive **2018**, 857 (2018), https://eprint.iacr.org/2018/857
20. Lyubashevsky, V.: Lattice signatures without trapdoors. In: EUROCRYPT. pp. 738–755 (2012)
21. Noether, S.: Ring signature confidential transactions for monero. IACR Cryptology ePrint Archive **2015**, 1098 (2015), http://eprint.iacr.org/2015/1098

22. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT. pp. 552–565 (2001)
23. van Saberhagen, N.: Cryptonote v 2.0. https://cryptonote.org/whitepaper.pdf (2013)
24. Sun, S., Au, M.H., Liu, J.K., Yuen, T.H.: Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: ESORICS. pp. 456–474 (2017)
25. Torres, W.A.A., Steinfeld, R., Sakzad, A., Liu, J.K., Kuchta, V., Bhattacharjee, N., Au, M.H., Cheng, J.: Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0). In: ACISP. pp. 558–576 (2018)
26. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: ISPEC. pp. 48–60 (2005)
27. Yang, R., Au, M.H., Lai, J., Xu, Q., Yu, Z.: 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), http://eprint.iacr.org/2017/781
28. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient linkable and/or threshold ring signature without random oracles. Comput. J. **56**(4), 407–421 (2013)
29. Zhang, H., Zhang, F., Tian, H., Au, M.H.: Anonymous post-quantum cryptocash. IACR Cryptology ePrint Archive **2017**, 716 (2017), http://eprint.iacr.org/2017/716

A Comment on [19]

Lu et al. [19] adopted the definitions of anonymity, linkability and nonslanderability from [17]. Then, they gave a theorem which shows that the unforgeability is implied by linkability and nonslanderability. first review the definition of linkability and the theorem as follows:

The linkability in [19] 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 pairs $\{T_1, m_1, \sigma_1\}$ and $\{T_2, m_2, \sigma_2\}$.

$\mathcal{A}$ wins the game if

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

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 7 ([19], 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 4 ([19], Theorem 2). *If a LRS scheme is linkable and nonslanderable, it is also unforgeable.*

Issue 1. Theorem 4 does not hold for the definition of linkability in [19]. The content of theorem 4 was introduced in [2] which towards the security definitions in [2]. However, the definition of linkability in [19] is different from the definition in [2]. In [19], the adversary $\mathcal{A}$ against unforgeability is allowed to make polynomially 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 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 $\sigma^* = (\{(m_i^*, r_i^*)\}_{i\in[N]}, \tilde{\sigma}^*, ovk^*)$. 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 [19], 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.