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 particular, 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 \(PK\) has some group structure \((PK, \odot)\) (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 \(PK\), 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 $\hat{\sigma}$. The signature $\sigma$ is set as $(\hat{\sigma}, \hat{\sigma}, ovk)$. 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 $hk_i$ 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 $(pk, sk)$ and $(ovk, osk)$ of LRS and OTS respectively. Then, he computes the public key $pk = pk \oplus H(ovk)$ and sets the secret key $sk = (sk, osk, ovk)$. 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 $T' = \{pk_i'\}_{i \in [N]}$, where $pk_i'$ is equal to $pk_i$ combined with the same value $H(ovk_s)$. It is easy to see that $\hat{\sigma} = (\hat{\sigma}, T', ovk_s)$, 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 $A$ that break linkability of our LRS scheme. Then, we construct an adversary $B$ that break unforgeability of underlying RS scheme by using $A$. If $A$ succeeds, that is $A$ outputs $N+1$ unlinked valid signatures for the same ring whose size is $N$, then at least one of the signatures, denoted as $\sigma^*$, contains the linkability tag which is not used in the key generation procedure. We set $\sigma^*$ contained in $\sigma^*$ as the output of $B$. The core problem that we face in reduction is how to simulate the public key for $A$ to make $\sigma^*$ is generated on the ring $T'$ which $B$ does not know the secret keys. At a high level, we resolve this problem by fixing every $pk' \in T'$ for $B$ in advance instead of making it generated by $A$. More specifically, we assume $A$ and $B$ have access to the joining oracle $O_{\text{join}}$ and $O_{\text{join}}$ respectively, where $O_{\text{join}}$ and $O_{\text{join}}$ output public keys of LRS and RS at random. For every query to $O_{\text{join}}$ made by $A$, $B$ should query $O_{\text{join}}$ twice to get two public keys $pk, pk''$. $pk$ is used to simulate the response of $O_{\text{join}}$, and $pk''$ is used to fix the elements in $T'$. By the programmability of $H$, we generate $pk$ in two
different ways using $\hat{p}k$, $p\hat{k}''$ respectively:

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

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

![Figure 1: real $O_{\text{join}}$](image1)

![Figure 2: Simulation of $O_{\text{join}}$](image2)

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.
We use the same notation to sample sets then rings is noted. The notation is used to denote that the algorithm outputs and use lower-case bold letters and upper-case bold letters to denote vectors and matrices (e.g. $\mathbb{N}$ real numbers respectively. For a vector $f$ and a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ in variable $X$ as $||x|| = \sqrt{\sum_{i=0}^{n-1} x_i^2}$ and $||f|| = \sqrt{\sum_{i=0}^{n-1} a_i^2}$. For a vector $f = (f_0, \cdots, f_{n-1})$ of polynomials, $||f|| = \sqrt{\sum_{i=0}^{n-1} ||f_i||^2}$. The infinity norm of $f$ is $||f||_\infty = \max_i |a_i|$. Let $q$ be an odd prime integer and assume $q \equiv 5 \mod 8$. We define the rings $R = \mathbb{Z}[X]/(X^d + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^d + 1)$, 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 $\text{negl}(\lambda)$, which is a function $g(\lambda) = O(\lambda^c)$ for some constant $c$. We denote the set of integers

[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 non-slanderability. 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. Yang et al. [YLA+13] proposed a LRS scheme whose signature size is square root of the number of ring members. Sun et al. [SALY17] proposed 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 $\{1, \ldots, N\}$. 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 $\mathbf{x} = (x_0, \ldots, x_{n-1})$ and a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ in variable $X$ as $||x|| = \sqrt{\sum_{i=0}^{n-1} x_i^2}$ and $||f|| = \sqrt{\sum_{i=0}^{n-1} a_i^2}$. For a vector $f = (f_0, \cdots, f_{n-1})$ of polynomials, $||f|| = \sqrt{\sum_{i=0}^{n-1} ||f_i||^2}$. The infinity norm of $f$ is $||f||_\infty = \max_i |a_i|$. Let $q$ be an odd prime integer and assume $q \equiv 5 \mod 8$. We define the rings $R = \mathbb{Z}[X]/(X^d + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^d + 1)$, 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 $\text{negl}(\lambda)$, which is a function $g(\lambda) = O(\lambda^c)$ for some constant $c$. We denote the set of integers
A RS scheme consists of four algorithms (Setup, KeyGen, Sign, Vrfy). We use $D_{v,\sigma}$ to denote the discrete normal distribution centered at $v$ with standard deviation $\sigma$. We write $D_\alpha$ 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 $R_q = \mathbb{Z}_q[X] / (X^d + 1)$. Given $A \leftarrow R_q^{n \times m}$, find $x \in R_q^m$ such that $Ax = 0 \mod q$ and $0 < \|x\| \leq \theta$.

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

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

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

The Lemma 2 in [ESS+18] shows the above commitment scheme is statistically hiding if $q > 8\epsilon^2$ and $\chi$ is greater than $n \log q / m + 2\lambda / (md)$, and computationally strong binding if Module-SIS$_{n,m+v,q,\theta}$ problem for $\theta = 2\sqrt{\epsilon^2md + \epsilon_M^2vd}$ 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 $T$, and $h$ be a probability distribution over $V$. Define the following algorithms:

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

where $\sigma = 12T$ and $M = e^{1 + \frac{1}{\sqrt{3}}}$. Then the output of algorithm $\mathcal{A}$ is within statistical distance $2^{-100}/M$ of the output of $\mathcal{F}$. Moreover, the probability that $\mathcal{A}$ outputs something is more than $\frac{1}{1 - 2^{-100}}/M$.

### 2.4 Ring Signature

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

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


• **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.

• **Vrfy**(T, m, σ): 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 Vrfy(T, m, 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** \( pk \stackrel{\$}{\leftarrow} O_{\text{join}}(\bot) \): \( O_{\text{join}} \) generates a new user and returns the public key \( pk \) of the new user.

• **Corruption Oracle** \( sk \stackrel{\$}{\leftarrow} O_{\text{corrupt}}(pk) \): On input a public key \( pk \) which is a output of \( O_{\text{join}} \), returns the corresponding secret key \( sk \).

• **Signing Oracle** \( \sigma \stackrel{\$}{\leftarrow} O_{\text{sign}}(T, m, pk_s) \): 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 \( \sigma \) 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 \( A \) and a challenger \( CH \):

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

\( A \) wins if \( s^* = s \). The advantage of \( A \) is defined by \( \text{Adv}_A^{\text{anon}} = \left| \Pr[s^* = s] - \frac{1}{N} \right| \).

**Definition 2.3.** A RS scheme is said to be anonymous (resp.unconditionally anonymous) if for any PPT adversary (unbounded adversary) \( A \), \( \text{Adv}_A^{\text{anon}} \) is negligible in \( \lambda \).

**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 \( A \) and a challenger \( CH \):

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

\( A \) wins if

• \( \text{Vrfy}(m^*, \sigma^*, T^*) = \text{accept} \);
• all of the public keys in \( T^* \) are query outputs of \( O_{\text{join}} \);
• no public key in \( T^* \) has been input to \( O_{\text{corrupt}} \); and
• \((m^*, T^*)\) has not been queried to \( O_{\text{sign}} \).

The advantage of \( \mathcal{A} \), denoted as \( \text{Adv}^{\text{forge}}_{\mathcal{A}} \), 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 \( \mathcal{A} \), \( \text{Adv}^{\text{forge}}_{\mathcal{A}} \) is negligible in \( \lambda \).

### 2.5 Linkable Ring Signature

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

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

**Correctness.** For any security parameter \( 1^\lambda \), any \( \{pk_i, sk_i\}_{i \in [N]} \) output by \( \text{KeyGen} \), any \( s \in [N] \), and any message \( m \), we have \( \text{Vrfy}(T, m, \text{Sign}(sk_s, m, T)) = \text{accept} \) 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 \( \text{Setup} \) with security parameter \( 1^\lambda \) and sends the public parameter \( pp \) to \( \mathcal{A} \).

2. **Query:** The adversary \( \mathcal{A} \) is allowed to make queries to \( O_{\text{join}} \).

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 \( \sigma \leftarrow \text{Sign}(sk_s, m, T) \). \( \mathcal{CH} \) sends \( \sigma \) 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 \( \text{Adv}^{\text{anon}}_{\mathcal{A}} = |\text{Pr}[s^* = s] - \frac{1}{N}| \).

**Definition 2.5.** A LRS scheme is said to be anonymous (resp. unconditionally anonymous) if for any PPT adversary (unbounded adversary) \( \mathcal{A} \), \( \text{Adv}^{\text{anon}}_{\mathcal{A}} \) is negligible in \( \lambda \).
**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 \( \text{Setup} \) and gives \( \mathcal{A} \) public parameter \( pp \).
2. **Query:** The adversary \( \mathcal{A} \) is allowed to make queries to \( O_{\text{join}}, O_{\text{corrupt}}, O_{\text{sign}} \).
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 \( O_{\text{join}} \);
- \( \text{Vrfy}(m_i, \sigma_i, T) = \text{accept} \) for all \( i \in [N + 1] \);
- \( \text{Link}(m_i, m_j, \sigma_i, \sigma_j) = \text{unlinked} \) for all \( i, j \in [N + 1] \) and \( i \neq j \).

The advantage of \( \mathcal{A} \), denoted as \( \text{Adv}_{\mathcal{A}}^{\text{link}} \), 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 \( \mathcal{A} \), \( \text{Adv}_{\mathcal{A}}^{\text{link}} \) is negligible in \( \lambda \).

**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. Nonslanderability can be defined by the following game between an adversary \( \mathcal{A} \) and a challenger \( \mathcal{CH} \):

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

\( \mathcal{A} \) wins if

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

The advantage of \( \mathcal{A} \), denoted as \( \text{Adv}_{\mathcal{A}}^{\text{nlander}} \), 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 \( \mathcal{A} \), \( \text{Adv}_{\mathcal{A}}^{\text{nlander}} \) is negligible in \( \lambda \).
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).

- **Setup(1^λ):** On input the security parameter 1^λ, this algorithm runs \( \hat{pp} \leftarrow \text{RS.Setup}(1^\lambda) \). It also chooses a hash function \( H : \text{OVK} \rightarrow \text{PK} \) modeled as random oracle, where \( \text{OVK} \) and \( \text{PK} \) 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.

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

- **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 \( pk'_i = pk_i \odot H(ovk_s) \) for \( i \in [N] \) and sets \( T' = \{pk'_i\}_{i \in [N]} \). Next, the algorithm runs
  \[
  \hat{\sigma} \leftarrow \text{RS.Sign}(sk_s, m, T'),
  \]
  \[
  \hat{\sigma} \leftarrow \text{OTS.Sign}(osk_s, \hat{\sigma}, T, ovk_s).
  \]
  Finally, it returns the signature \( \sigma = (\hat{\sigma}, \hat{\sigma}, ovk_s) \).

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

- **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 \( \sigma_1, \sigma_2 \), runs \( \text{Vrfy}(m_1, \sigma_1, T_1) \) and \( \text{Vrfy}(m_2, \sigma_2, T_2) \). If \( \text{Vrfy}(m_1, \sigma_1, T_1) = \text{reject} \) or \( \text{Vrfy}(m_2, \sigma_2, T_2) = \text{reject} \), it aborts. Otherwise, the algorithm parses \( \sigma_1 \) and \( \sigma_2 \) as \( \sigma_1 = (\hat{\sigma}_1, \hat{\sigma}_1, ovk_1) \) and \( \sigma_2 = (\hat{\sigma}_2, \hat{\sigma}_2, ovk_2) \), 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 \( A \) with oracle access to \( O_{\text{join}} \) can break the anonymity of LRS, then we can construct an adversary \( B \) with oracles access to \( \hat{O}_{\text{join}}, \hat{O}_{\text{sign}} \) to break the anonymity of RS with the same advantage, where \( \hat{O}_{\text{join}} \) and \( \hat{O}_{\text{sign}} \) 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 \( B \), \( B \) interacts with \( A \) with the aim to guess the signer \( s \).

1. **Setup:** Given the public parameter \( \hat{pp} \), \( B \) selects a hash function \( H : \text{OVK} \rightarrow \text{PK} \), where \( H \) is modeled as random oracle and \( \text{OVK} \) and \( \text{PK} \) are public key spaces of OTS and RS respectively. \( B \) then sends \( pp = \hat{pp} \) to \( A \).
2. **Oracle Simulation**: $A$ is allowed to access the joining oracle $O_{\text{join}}$: $B$ runs $(ovk, osk) \leftarrow \text{OTS}$. KeyGen at first. Upon receiving a joining query, $B$ queries $O_{\text{join}}$ to obtain a public key $pk$ of RS, computes $pk = pk \odot H(ovk)$. Then sends $pk$ to $A$.

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

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

4. **Output**: $A$ outputs the index $s^*$.

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

**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 $A$ succeeds in Game $i$.

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

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

2. **Oracle Simulation**: $A$ 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 $O_{\text{join}}$ and $O_{\text{sign}}$. $CH$ initializes an empty set $RO$. Upon receiving a random oracle query $i$, if it has been queried, $CH$ returns corresponding output in $RO$; else, $CH$ picks $h_i \leftarrow PK$ at random, sends $h_i$ to $A$ and stores the pair of $(i, h_i)$ in $RO$.

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

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

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

3. **Outputs**: $A$ outputs $N + 1$ message/signature pairs $\{m_i, \sigma_i\}_{i \in [N+1]}$ on the same set of public keys $T = \{pk_i\}_{i \in [N]}$. $A$ wins if
• all public keys in $T$ are query outputs of $O_{\text{join}}$;
• $\text{Vrfy}(m_i, \sigma_i, T) = \text{accept}$ for all $i \in [N + 1]$;
• $\text{Link}(m_i, m_j, \sigma_i, \sigma_j) = \text{unlinked}$ for all $i, j \in [N + 1]$ and $i \neq j$.

According to the definition, we have

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

**Game 1.** Same as Game 0 except that in $O_{\text{join}}$ of Oracle Simulation stage, $\mathcal{CH}$ additionally choose $h' \leftarrow PK$ 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 $I \leftarrow [1, \ldots, q_h]$, where $q_h$ is the maximum number of times $\mathcal{A}$ directly queries $H$, then $\mathcal{CH}$ programs the output of the $I$th 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 $O_{\text{join}}$ of Oracle Simulation stage, $\mathcal{CH}$ additionally runs $(pk'', sk'') \leftarrow \text{RS.KeyGen}$ and computes $h$ such that $pk \odot h = pk'' \odot h'$ upon receiving a joining query. $\mathcal{CH}$ then sends $pk = pk \odot h$ to $\mathcal{A}$. Due to the distribution of $pk$ is close to the uniform distribution over $PK$, 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 $O_{\text{join}}$) 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 $\lambda$.

If $\mathcal{A}$ wins in Game 4, then we can construct an adversary $\mathcal{B}$ with oracles access to $O_{\text{join}}$, $O_{\text{corrupt}}$ and $O_{\text{sign}}$ to break the unforgeability of RS with the advantage $q_h \cdot \text{Adv}_{A}^{\text{forge}}$, implying $\Pr[S_4]$ must be negligible, where $O_{\text{join}}, O_{\text{corrupt}}$ and $O_{\text{sign}}$ 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 $H : OV K \rightarrow PK$, where $\hat{H}$ is modeled as random oracle and $OV K$ and $PK$ 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 $O_{\text{join}}$ and $O_{\text{sign}}$. $\mathcal{B}$ initializes an empty set $RO$, chooses a index $I \leftarrow [1, \ldots, q_h]$, where $q_h$ is the maximum number of times $\mathcal{A}$ directly queries $H$. $\mathcal{A}$ then programs the output of the $I$th 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 $O_{\text{join}}$), else $\mathcal{B}$ picks $h_i \leftarrow PK$, sends $h_i$ to $\mathcal{A}$ and stores the pair $(i, h_i)$ in $RO$. 

12
Joining Oracle $O_{\text{join}}$: $B$ initializes an empty set $JO$ and chooses $h' \leftarrow PK$. Upon receiving a joining query from $A$, $B$ queries $O_{\text{join}}$ twice to get two public keys $pk$, $pk''$, computes $h$ such that $pk \odot h = pk'' \odot h'$, runs $(ovk, osk) \leftarrow \text{OTS.KeyGen}$. $B$ then sends $pk = pk \odot h$ to $A$ and stores $pk$ in $JO$.

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

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

3. Output: $A$ 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 $(m_i, \sigma_i, T = \{pk_j\}_{j \in [N]}; i \in [N + 1])$, $B$ parses every signature $\sigma_i$ as $\sigma_i = (\tilde{\sigma}_i, \hat{\sigma}_i, ovk_i)$. Since $A$ outputs $N + 1$ unlinked signatures on $N$ public keys, so there exits at least one of $ovk_i$ in $\sigma_i$ which is not produced by $O_{\text{join}}$. We assume it is $ovk^*$. Hence, the probability of $pk^* = pk_j \odot H(ovk^*)$ has been produced by $O_{\text{corrupt}}$ and $O_{\text{sign}}$ is negligible for all $j \in [N]$. Furthermore, if $ovk^*$ is the $I$th query of $H$, which happens with probability at least $\frac{1}{N^2}$, then $\{pk^*_j\}_{j \in [N]}$ are all query outputs of $O_{\text{join}}$. Hence, $B$ can outputs a successful forgery $(m^*, \hat{\sigma}^*, T^* = \{pk^*_j\}_{j \in [N]})$ if $H(ovk^*) = h'$; else it returns $\bot$.

It is straightforward to verify that $B$’s simulation for Game 4 is perfect, we can conclude

$$\Pr[S_4] = q_h \cdot \text{Adv}^\text{forge}_B$$

Putting all the above together, the theorem immediately follows. \hfill \square

**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 $A$ with oracle access to $O_{\text{join}}, O_{\text{corrupt}}$ and $O_{\text{sign}}$ can break the nonslanderability of LRS, then we can construct an adversary $B$ allowed to query the signature once for any message of his choosing to break the unforgeability of OTS with the same advantage, implying that $\text{Adv}^\text{slander} \leq \text{Adv}^\text{forge}_B$ must be negligible.

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

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

2. Oracle Simulation:

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

This change cannot be distinguished from the real game since $pk = \tilde{pk} \odot H(ovk)$ in real game and $H$ is the random oracle.
We show an instantiation of our construction by using the RS in [ESS+18] and the signature Dilithium [DKL+18].

- **Setup(1^λ):** On input 1^λ, select the commitment key ck = G ← R_q^{n(m+kβ)}, two hash functions H : {0,1}^* → C and H' : {0,1}^* → R_q^n, where H, H' are modeled as random oracles and C = {X^ω : 0 ≤ ω ≤ 2d – 1} is the challenge space.

- **KeyGen(pp):** On input public parameter pp, select ri ← {-M,...,M}^d for i ∈ [m] and set r = (r_1,...,r_m), compute c = Com_{ck} (0;r), where 0 is the all-zero vector, set pk = c, sk = r, run (ovk, osk) ← Dilithium.KeyGen(1^λ). Output pk = pk + H'(ovk), sk = (sk, osk, ovk).
• \textbf{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 \( pk'_i = pk_i - H'(ovk_s) \) for each \( i \in [N] \) and set \( \hat{T}' = \{ pk'_i \}_{i \in [N]} \).

2. Sample \( a_{0,1}, ..., a_{k-1, \beta-1} \leftarrow D^d_{12\sqrt{\tau}} \); compute \( a_{j,0} = -\sum_{i=1}^{\beta-1} a_{j,i} \) for \( j = 0, ..., k - 1 \), select \( r_{b,1}, r_{c,i} \leftarrow \{-M, ..., M\}^d \) for \( i \in [m] \) and set \( r_b = (r_{b,1}, ..., r_{b,m}) \), \( r_c = (r_{c,1}, ..., r_{c,m}) \), sample \( r_{a,1}, r_{d,i} \leftarrow D^d_{12M/\sqrt{2md}} \) for \( i \in [m] \) and set \( r_a = (r_{a,1}, ..., r_{a,m}) \), \( r_d = (r_{d,1}, ..., r_{d,m}) \), compute

\[
A = \text{Com}_{ck}(a_{0,0}, ..., a_{k-1, \beta-1}; r_a) \\
B = \text{Com}_{ck}(\delta_{0,0}, ..., \delta_{k-1, \beta-1}; r_b) \\
C = \text{Com}_{ck}(\{a_{j,i}(1-2\delta_{j,i})\}_{j,i=0}^{k-1, \beta-1}; r_c) \\
D = \text{Com}_{ck}(a_{0,0}^2, ..., a_{k-1, \beta-1}^2; r_d),
\]

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 \( \rho_1 = (\rho_{1,1}, ..., \rho_{1,m}) \), compute \( E_j = \sum_{i=0}^{N-1} p_i j c_i \) + \( \text{Com}(0; \rho_j) \) for \( j = 0, ..., k - 1 \), where \( p_i j c_i \) is computed by \( p_i(x) = \prod_{j=0}^{k-1}(x \cdot \delta_{j,i} + a_{j,i}) = \prod_{j=0}^{k-1} x \cdot \delta_{j,i} + \sum_{j=0}^{k-1} p_j x^j , i \in [N] \). Compute \( z = H'(ck, m, T', A, B, C, D, \{ E_j \}_{j=0}^{k-1}, \{ f_{j,i} \}_{j,i=0}^{k-1}, \{ r_{b,i} \}_{i=0}^{m}, \{ r_{c,i} \}_{i=0}^{m}, \{ r_{d,i} \}_{i=0}^{m}, \{ z_{c,i} \}_{i=0}^{m}, \) \( z = x^k \cdot \hat{sk}_s - \sum_{j=0}^{k-1} x^j \cdot \rho_j \). Set \( \text{CMT} = (A, B, C, D, \{ E_j \}_{j=0}^{k-1}) \) and \( \text{RSP} = (\{ f_{j,i} \}_{j=0}^{k-1}, \{ r_{b,i} \}_{i=0}^{m}, \{ r_{c,i} \}_{i=0}^{m}, \{ z_{c,i} \}_{i=0}^{m}) \).

3. Repeat step 2 \( L \) times in parallel and get \( \{ \text{CMT}_l \}_{l \in [L]}, \{ x_l \}_{l \in [L]} \) and \( \{ \text{RSP}_l \}_{l \in [L]} \).

If \( \text{RSP}_l \neq \perp \) for all \( l \in [L] \), set \( \hat{\sigma} = (\{ \text{CMT}_l \}_{l \in [L]}, \{ x_l \}_{l \in [L]}, \{ \text{RSP}_l \}_{l \in [L]} \). Otherwise, go to step 2 (repeat at most \( \frac{\log_{1/\delta}(\tau)}{L} \)).

4. Run \( \hat{\sigma} \leftarrow \text{Dilithium.Sign}(ovk_s, (\hat{\sigma}, T, ovk_s)) \).

5. Output \( \sigma = (\hat{\sigma}, \sigma, ovk_s) \).

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

1. For every \( \{ \text{CMT}_l, x_l, \text{RSP}_l \}, l \in [L] \) check whether

   - \( f_{j,0} = x - \sum_{i=1}^{k-1} f_{j,i} \) for \( j = 0, ..., k - 1 \)
   - \( xB + \text{A} = \text{Com}_{ck}(f_{0,0}, ..., f_{k-1, \beta-1}; z_b) \)
   - \( xC + D = \text{Com}_{ck}(f_{0,0}(x - f_{0,0}), ..., f_{k-1, \beta-1}(x - f_{k-1, \beta-1}); z_c) \)
   - \( \| f_{j,0} \| \leq 60/\sqrt{dk}, \forall j, i = 0 \)
   - \( \| f_{j,0} \| \leq 60/\sqrt{dk}(\beta-1), \forall j, i \neq 0 \)
   - \( \| z \|, \| z_b \|, \| z_c \| \leq 24\sqrt{3Md} \)
   - \( \sum_{i=0}^{N-1} (\prod_{j=0}^{k-1} f_{j,i})c_i - \sum_{j=0}^{k-1} E_j x^j = \text{Com}_{ck}(0; z) \) for \( i = (i_0, ..., i_{k-1}) \)

if not, return reject.

2. Run \( \text{accept/reject} \leftarrow \text{Dilithium.Vrfy}(ovk_s, (\hat{\sigma}, T, pk)) \).

3. If neither 1 and 2 return reject, return accept.

• \textbf{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 \( \sigma_1, \sigma_2 \), run Vrfy(\( m_1, \sigma_1, T_1 \)) and Vrfy(\( m_2, \sigma_2, T_2 \)). Parse \( \sigma_1 \) and \( \sigma_2 \) as \( \sigma_1 = (\hat{\sigma}_1, \sigma_1, ovk_1) \) and \( \sigma_2 = (\hat{\sigma}_2, \sigma_2, ovk_2) \). Compare \( ovk_1 \) and \( ovk_2 \). Return linked if \( \text{Vrfy}(m_1, \sigma_1, T_1) = \text{Vrfy}(m_2, \sigma_2, T_2) = \text{accept} \) 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>
<thead>
<tr>
<th>Scheme</th>
<th>Public Key Size</th>
<th>Signature Size</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[YAL⁺17]</td>
<td>nlogp</td>
<td>$2m(\log q)^2 \cdot \log N$</td>
<td>SIS/LWR</td>
</tr>
<tr>
<td>[ZZTA17]</td>
<td>mndlogq</td>
<td>$m^2d\log q \cdot \log N$</td>
<td>I(f)-SVP$_\gamma$</td>
</tr>
<tr>
<td>[BLO18]</td>
<td>ndlogq</td>
<td>$m\log(2\sigma \sqrt{d}) \cdot N$</td>
<td>M-SIS/M-LWE</td>
</tr>
<tr>
<td>[TSS⁺18]</td>
<td>dlogq</td>
<td>$m\log(\eta \sigma \sqrt{d}) \cdot N$</td>
<td>R-SIS</td>
</tr>
<tr>
<td>[LAZ18]</td>
<td>dlogq</td>
<td>$(256 + 2d\log q) \cdot N$</td>
<td>NTRU</td>
</tr>
<tr>
<td>Ours</td>
<td>ndlogq</td>
<td>$(n\log q + \beta d\log \sqrt{144L\log q}N) \cdot \log_f N$</td>
<td>M-SIS</td>
</tr>
</tbody>
</table>

1. Constant terms are omitted.
2. n and m denote the row and column of matrix on $\mathbb{Z}_q$ or $\mathbb{R}_q$, d denotes the dimension of polynomials, $\beta$ denotes the base representations, $\sigma$ 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>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>M, n, q, m, l, d, L, k</td>
<td>100, 9, 2^60, 71, 100, 76, 17, 2</td>
</tr>
</tbody>
</table>

Table 3: Experimental Results

<table>
<thead>
<tr>
<th>Size(KB)</th>
<th>Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyGen</td>
<td>Sign</td>
</tr>
</tbody>
</table>

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


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 $pp \leftarrow \text{Setup}(1^\lambda)$ and sends $pp$ to $\mathcal{A}$.
2. Query. $\mathcal{A}$ is given access to $O_{\text{join}}, O_{\text{corrupt}}, O_{\text{sign}}$ 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, $\sigma_1$ and $\sigma_2$ are signatures.

$\mathcal{A}$ wins the game if

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

The advantage of $\mathcal{A}$, denoted as $\text{Adv}^{\text{link}}_\mathcal{A}$, 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 $\mathcal{A}$, $\text{Adv}^{\text{link}}_\mathcal{A}$ is negligible in $\lambda$.

**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 \( A \) against unforgeability is allowed to make polynomial many \( O_{\text{corrupt}} \) queries in the unforgeability game, whereas the adversary \( B \) against linkability is restricted to make at most one \( O_{\text{corrupt}} \) query in the linkability game. This means \( B \) cannot simulate \( O_{\text{corrupt}} \) for \( A \) and thus \( B \) cannot run \( A \) 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 \( h k_c \) into one of the public keys \( p k_I \) by computing \( p k_I = h k_c \oplus H(o v k_I) \). Second, the adversary \( A \) outputs two signatures and they concluded that at least one of the output signatures should be generated from the secret key that \( A \) does not obtain because \( A \) is allowed to make at most one \( O_{\text{corrupt}} \) query. The signature is denoted as \( (m^*, \sigma^*, T^*) \), where \( \sigma^* = \{(m_1^*, r_1^*), \ldots, (m_N^*, r_N^*), o v k^*, \tilde{\sigma}^*\} \). Finally, they assumed \( p k_I \in T^* \) and used \( (m^*, \sigma^*, T^*) \) to find a collision of \( h k_c \) according to the General Forking Lemma.

However, the collision resistance challenge may not be embedded into the output signatures of \( A \). This means that \( h k_c \) is not used to generate the signature \( (m^*, \sigma^*, T^*) \) although \( p k_I \in T^* \). The reason is that \( o v k^* \) may not equal to \( o v k_I \) and thus \( h k_c \neq h k_i = p k_i \oplus H(o v k^*) \) for every \( i \in [N] \). According to the signing algorithm of the LRS in [LAZ18], we can conclude that \( h k_c \) is independent of \( \sigma^* \) if \( o v k^* \neq o v k_I \). Thus, the collision resistance of CH+ cannot be broken although \( A \) has broken the linkability of the LRS.