Raptor: A Practical Lattice-Based (Linkable) Ring Signature

Xingye Lu[1], Man Ho Au[1], and Zhenfei Zhang[2]

[1] The Hong Kong Polytechnic Univerisity, Hong Kong
`xingye.lu@connect.polyu.hk, mhaau@polyu.edu.hk`
[2] Onboard Security, Wilmington, Massachusetts, USA
`zzhang@onboardsecurity.com`

Abstract. We present (linkable) Raptor, the first lattice-based (linkable) ring signature that is practical. Our scheme is as fast as classical solutions; while the size of the signature is roughly 1.3 KB per user. Our designs are based on a completely new generic construction that is provable secure in random oracle model. Prior to our work, all existing lattice-based solutions are analogues of their discrete-log or pairing-based counterparts. We give instantiations to both standard lattice setting, as a proof of concept, and NTRU lattice, as an efficient instantiation. Our main building block is a so called Chameleon Hash Plus (CH+) function, which may be of independent research interest.

1 Introduction

The notion of ring signatures was put forth by Rivest, Shamir and Tauman in 2001 [50]. It is a special type of group signature [19, 17] where

1. a signer is able to produce a signature on behalf of a group of potential signers;
2. there does not exist a group manager (usually a trusted third party) that manages the membership.

In a ring signature, the group can be quite ad hoc. Each user is associated with a public key and a group can be created spontaneously by collecting users' public keys. It is a very attractive property as it enables anonymity: the signer hides its identity within the group, and there is no trusted third party that is capable of revocation.

A side affect of this strong anonymity is that signatures become unlinkable. This is addressed by the notion of linkable ring signature by Liu, Wei and Wong [40]. In such a scheme, the identity of the signer will remain anonymous; in the meantime,

3. two signatures signed by the same signer can be linked.

The properties of linkability and signer anonymity are very desirable in various real world applications, including, but not limited to, e-cash, e-voting, and ad-hoc authentication. For example, in the e-commerce scenario, a linkable ring signature allows the spender to remain anonymous, while making it possible for the merchant/banker to identify double spenders. To date, linkable ring signature has become a mainstream solution to privacy-preserving cryptocurrency.

A major drawback of widely deployed linkable ring signatures is their insecurity against quantum computers [51]. Even though quantum computers are still in their infancy, it is inevitable that general purpose quantum computers will arrive, by when the exiting classical ring signatures will lose their anonymity and/or unforgeability.

Lattice-based cryptography is one of the most promising families of candidates [46] to the quantum apocalypse. It allows us to build cryptographic functions in which breaking a random instance from the family is as hard as solving a worst-case instance in lattice. The connection between average-case and worst-case hardness provides lattice-based cryptographic constructions stronger provable security.

To date, there exist a number of lattice-based ring signature schemes and one lattice-based linkable ring signature scheme, to the best of our knowledge. All those solutions are theoretically sound. However, they are hardly practical and there is no known implementation. At a high level, those schemes are instantiations of existing frameworks, where the underlying discrete-log/pairing-based building blocks are replaced by lattice-based ones. It is highly likely that a breakthrough requires a new framework.

1.1 Related Work

Classical ring signatures We recall the existing frameworks, or generic constructions. For the families of schemes where the signature sizes are linear in the number of group members, we have:

- The framework introduced by Rivest, Shamir and Tauman [50] in 2001. As mentioned earlier, it is the first ring signature scheme. The scheme requires the existence of one-way trapdoor permutations along with a block cipher. This generic construction does not support discrete-log type keys.
- The framework by Abe, Ohkubo and Suzuku [1] in 2004. Their work supports the use of RSA keys and discrete-log type keys. They also show how to transform a hash-and-sign type signature scheme or a three-move sigma-protocol based signature scheme into a ring signature scheme.
- The construction of Bender, Katz and Morselli [13]. It uses a public-key encryption scheme, a signature scheme and a ZAP protocol [26] to construct a ring signature scheme.

The first two schemes are secure in the random oracle model; while the last one is in the standard model. Beyond linear constructions, we have

- The first ring signature with sub-linear signature size without random oracle model proposed by Chandran, Groth and Sahai [18] using pairing-based non-interactive zero knowledge proofs.

– The first ring signature scheme with signature size independent from ring proposed by Dodis et al. in 2004 [21]. It is a generic construction relying on any accumulator with one-way domain and the Fiat-Shamir heuristic.

For non-generic constructions,

– Nguyen [47] presented a identity-based constant-size ring signature scheme in the random oracle model from accumulators, based on pairing-based cryptography.
– Ring signatures introduced by Groth and Kohlweiss [31] in 2015 is instantiated from a sigma-protocol with a sub-linear signature size and secure in the random oracle model.

Lattice-based ring signatures For ring signatures in lattice setting, Brakerski and Kalai [16] proposed a generic ring signature scheme in the standard model. This generic construction is based on a new primitive called ring trapdoor functions. They instantiated this function based on the inhomogeneous short integer solution problem (ISIS). However, this generic construction is only secure under a weak definition; to achieve full security requires a quite inefficient transformation.

Melchor et al. [44] transforms Lyubashevsky's lattice-based signature [42] into a ring signature. As the authors pointed out themselves, this scheme is "pretty unpractical".

In 2016, Libert et al. [38] presented a lattice-based accumulator. With the accumulator and a lattice-based zero-knowledge proof system, they build a ring signature scheme that is with logarithmic size in the cardinality of the ring and secure in the random oracle model. However, Stern type zero-knowledge arguments applied in the accumulator is very inefficient and impractical.

The state-of-the-art lattice-based ring signature scheme is due to Esgin et al. [27]. Esgin et al. [27] extends the short one-out-of-many proof from discrete logarithm setting [31, 15] to lattice setting and leverage the lattice-based one-out-of-many proof system as a building block to design a new lattice-based ring signature scheme. Same as [38], [27] is a logarithmic size ring signature scheme. However, comparing with [38], [27] drastically improves the signature size and is thus far more efficient.

Classical linkable ring signatures Ever since the first build of linkable ring signature [40], we have seen a sequence of work [56, 7, 39, 53] that provide different features.

In 2005, Tsang and Wei [56] extends the genric ring siganture introduced by Dodis et al. [21] to linkable revision, achieve a constant signature size and security in the random oracle model.

Au et al. [7] presented a new security model for linkable ring signatures and a new short linkable ring signature scheme that is secure in this new model.

In 2014, Liu et al. [39] presented the first linkable ring signature scheme achieving unconditional anonymity. Sun et al. [53] proposed a new generic linkable ring signature to construct RingCT 2.0 for Monero.

There are also schemes with special properties such as identity-based linkable ring signatures [55, 9] and certificate-based linkable ring signatures [8].

Lattice-based linkable ring signatures To date, the only lattice-based linkable ring signature scheme was proposed by Torres et al. in 2018 [54]. It transforms the BLISS signature [22] into a linkable ring signature scheme. Their scheme is secure in the random oracle model. The signature size is linear to the number of element in the ring and is reported to be 51 KB per user in the ring. In the same year, Baum, Lin and Oechsner [12] construct another lattice-based linkable ring signature scheme following a very similar ideal to [54]. The signature size for [12] is claimed to be around 10.3KB per user. The main difference between these two work is the way to achieve linkability. We are not aware of any implementation of these work.

1.2 Our Contribution

Table 1. Performance

(a) Raptor-512

Users	5	10	50
KeyGen	29 ms	29 ms	29 ms
Sign	6 ms	9.5 ms	40 ms
verification	3 ms	6.5 ms	32 ms
PK	0.9 KB	0.9 KB	0.9 KB
SK	4.1 KB	4.1 KB	4.1 KB
Signature	6.3 KB	12.7 KB	63.3 KB

(b) Linkable Raptor-512

Users	5	10	50
KeyGen	57 ms	57 ms	57 ms
Sign	10.7 ms	17.4 ms	61 ms
verification	5.2 ms	11 ms	50 ms
PK	0.9 KB	0.9 KB	0.9 KB
SK	9.1 KB	9.1 KB	9.1 KB
Signature	7.8 KB	14.2 KB	64.8 KB

In this paper, we present Raptor, the first lattice-based linkable ring signature that is practical. It gets its name as it is the next generation of Falcon [28] that features a "stealth" mode. Raptor is secure in the random oracle model, based on some widely-accepted lattice assumptions. We also present a less efficient version that is based on standard lattice problems.

We implement Raptor, and its performance on a typical laptop is shown in Table 1(a) and 1(b). Source code will be made available and the experimental setting is presented in Section 5.1.

Our solution is in a sense optimal for the family of solutions where the signatures are linear in terms of users: in our construction, the signature consists of a lattice vector and a randomness nonce of 2λ bits per user. It is unlikely that one is able to reduce the size further within the linear domain. We believe that the only way to get substantially better performance than Raptor is from the family of solutions that are logarithmic/constant in the number of users. Prior to our work, all the existing lattice-based solutions are not implementable. The best theoretical work in linear size is due to [12], where the signature size is claimed to be 82.5KB with a ring size of 8. Comparing with the state-of-the-art [27] with sublinear signature size, our work is still outperformed in a ring size $\lessapprox 1000$. The signature size of [27] is reported to be 930KB with 2^6 users in the ring and 1409KB with 2^{10} users in the ring. The comparison of signature size of our RAPTOR and other existing lattice-based (linkable) ring signature scheme is shown in Table 2.

Table 2. Comparison of lattice-based (linkable) ring signature sizes for a security level $\lambda = 100$. Signature size growth is with respect to ring size (how many users in the ring).

	[38][3]	[54]	[12]	[27]	RAPTOR	(linkable) RAPTOR
Signature size growth	logarithm	linear	linear	logarithm	linear	linear
linkability	×	✓	✓	×	×	✓
Implementation	×	×	×	×	✓	✓
Signature size with 2^6 users	$\approx$ 37 MB	$\approx$ 649 KB	$\approx$ 585 KB	930 KB	80.6 KB	82.7 KB
with 2^8 users	$\approx$ 48.1 MB	$\approx$ 2474 KB	$\approx$ 2340 KB	1132 KB	332.6 KB	326.5 KB
with 2^{10} users	$\approx$ 59.1 MB	$\approx$ 9770 KB	$\approx$ 9360 KB	1409 KB	1290.2 KB	1301.9KB
with 2^{12} users	$\approx$ 70.2 MB	$\approx$ 39 MB	$\approx$ 37.4 MB	1492 KB	5161 KB	5203.3 KB

In terms of security, our (linkable) RAPTOR scheme is backed by a new generic framework that is provably secure in the random oracle model, under the assumption that certain properties of a new defined primitive, namely, Chameleon Hash Plus (CH+), are met. Note that CH+ does not necessarily need to be lattice-based.

Nonetheless, when CH+ is instantiated with a standard lattice problem (i.e., the short integer solution problem), we base the security of (linkable) ring signature on the worst-case lattice problems that are conjectured to be hard against quantum computers.

In practice, one often resorts to NTRU lattices [35] for better efficiency. Our (linkable) RAPTOR scheme is such a case, where the CH+ function is instantiated from the pre-image samplable function of FALCON [28].

We summarize our contributions:

- We propose a new primitive called Chameleon Hash Plus (CH+); a new generic construction for (linkable) ring signature scheme that uses CH+ as a building block. We prove its security in the random oracle model.
- We also give two instantiations of CH+. One is from the standard lattice, based on SIS/ISIS assumption; the other from the NTRU assumption. With those two instantiations, we obtain two (linkable) ring signatures.

[3] The approximate signature size of [38] is from [27].

– We implement our NTRU-based (linkable) signature, a.k.a. Raptor. We achieve both size efficiency and speed efficiency. Although the signature size is linear to the number of members in the ring, the actual size is very practical. For example, in a ring of 5 users, the signature size is roughly 6.3 KB. The signing and verification speed is of the same order of classical solutions.

1.3 Overview of Our Construction

Building block: CH+ The main building block for our generic constructions is a chameleon hash plus (CH+) function. Recall the notion of chameleon hash function, first formalized by Krawczyk and Rabin in 2000 [36]. Chameleon hash functions are randomized collision-resistant hash functions with an additional property that each hash key is equipped with a trapdoor. With the trapdoor, one can easily find collisions for any input. More specifically, on input a trapdoor tr corresponding to some chameleon hash key hk, two messages m, m' and a randomness r, one can efficiently compute another randomness r' such that $\mathsf{Hash}(\mathsf{hk}, m, r) = \mathsf{Hash}(\mathsf{hk}, m', r')$.

Our CH+ consists of four algorithms, namely, SetUp, $\mathsf{TrapGen}$, Hash and Inv. See Section 3.1 for details. Similar to a chameleon hash, without the trapdoor, CH+ needs to be one-way and collision-resistant. There are two main difference in CH+:

– to compute new randomness r' for any given message m', only the hash value, $C = \mathsf{Hash}(\mathsf{hk}, m, r)$, is needed; whereas both the original message m and randomness r are required in a classical Chameleon hash;
– there exists a system parameter $\mathsf{param}^{\mathsf{ch}}$ as an implicit input to all CH+ operations.

The framework: ring signature Now we are going to give an overview of our generic ring signature scheme based on CH+. We assume $\mathsf{param}^{\mathsf{ch}}$ is available at the setup. In the key generation procedure, each signer runs algorithm $\mathsf{TrapGen}$ to obtain a hash key hk and its trapdoor tr. Signer's public key and secret key will be hk and tr, respectively.

Suppose a signer, S_π, with public and secret keys $(\mathsf{hk}_\pi, \mathsf{sk}_\pi)$, tries to sign a message μ on behalf of a group of signer $\{S_1, \cdots, S_\ell\}$ $(\pi \in \{1, \cdots, \ell\})$, S_π first collects all the public keys of the group of signers $\{\mathsf{hk}_1, \cdots, \mathsf{hk}_\ell\}$. Next,

– for $i \neq \pi$, S_π randomly samples message m_i, randomness r_i and computes hash output $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$;
– for $i = \pi$, i.e., the signer himself, S_π samples a C_π.

S_π further sets $C^* = \mathcal{H}(\mu, C_1, \cdots, C_\ell, \mathsf{hk}_1, \cdots, \mathsf{hk}_\ell)$ where μ is the message to be signed and $\mathcal{H}$ is a collision-resistant hash function. It then computes m_π which satisfies $m_1 \oplus \cdots \oplus m_\ell = C^*$ and uses the trapdoor to find an r_π such that $c_\pi = \mathsf{Hash}(\mathsf{hk}_\pi, m_\pi, r_\pi)$. The signature for S_π on μ is $\{(m_1, r_1), \cdots, (m_\ell, r_\ell)\}$. Note that without the trapdoor, it is hard to find such a randomness r_π since CH+ is one-way and collision-resistant.

To verify the signature, one can first compute $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$ for $i = 1, \cdots, \ell$. Then check whether $m_1 \oplus \cdots \oplus m_\ell$ is equivalent to $\mathcal{H}(\mu, C_1, \cdots, C_\ell, \mathsf{hk}_1, \cdots, \mathsf{hk}_\ell)$. If so, the verifier accepts the signature as signed by one of the group member.

The framework: linkable ring signature Linkable ring signature scheme allows others to link two signatures sharing the same signer. At a high level, we will use a tag to achieve this property. The tag is a representative of the signer's identity for each signature. Signatures that share a same tag is linked. It is natural to enforce that each signer only obtains one unique tag; and this tag cannot be forged, or transferred from/to another user. We use a one-time signature[4] to achieve those properties.

During the key generation procedure, in addition to a hk and its trapdoor tr, the signer also generates a pair of public key and secret key $(\mathsf{opk}, \mathsf{osk})$ for a one-time signature. The signer then masks hk by $\mathcal{H}(\mathsf{opk})$ and obtains a masked hash key hk'. The unique tag for the signer will be the public key opk. In the end, the signer sets hk' as public key and $(\mathsf{tr}, \mathsf{opk}, \mathsf{osk})$ as secret key.

When the signer S_π signs a message μ on behalf of a group of signers $\{S_1, \cdots, S_\ell\}$ $(\pi \in \{1, \cdots, \ell\})$, it will collect the public keys of the group $\{\mathsf{hk}'_1, \cdots, \mathsf{hk}'_\ell\}$ as usual. For each public key hk'_i in the group, S_π computes $\mathsf{hk}''_i = \mathsf{hk}'_i \oplus \mathcal{H}(\mathsf{opk})$. A new list of "public keys", $\{\mathsf{hk}''_1, \cdots, \mathsf{hk}''_\ell\}$, is then formed. Note that hk''_π is equivalent to the original hk_π. Next, the signer S_π invokes the (none linkable) ring signature with keys $\{\mathsf{hk}''_1, \cdots, \mathsf{hk}''_\ell\}$, a trapdoor tr_π and a message μ, and obtains a (none linkable) ring signature $\sigma_R = \{(m_1, r_1), \cdots, (m_\ell, r_\ell)\}$ on μ. Finally, S_π signs μ, σ_R using osk_π and gets a one-time signature sig. The linkable ring signature produced by S_π will be $\{\sigma_R, \mathsf{opk}_\pi, sig\}$.

As for verification, in addition to verifying σ_R, one should also check whether sig is a valid signature on μ and σ_R under opk_π.

2 Preliminary

2.1 Notation

Elements in $\mathbb{Z}_q$ are represented by integers in $[-\frac{q}{2}, \frac{q}{2})$. For a ring $\mathcal{R}$ we define $\mathcal{R}_q$ to be the quotient ring $\mathbb{Z}_q[x]/(x^n + 1)$ with n being a power of 2 and q being a prime. Column vectors in $\mathbb{Z}_q^m$ and elements in $\mathcal{R}_q$ are denoted by lower-case bold letters (e.g. $\mathbf{x}$). Matrices are denoted by upper-case bold letters (e.g. $\mathbf{X}$). We use $\hat{\mathbf{x}}$ to denote a column vector with entries from the ring.

For distribution D, $x \leftarrow_\$ D$ means sampling x according to distribution D. $\|\mathbf{v}\|_1$ is the ℓ_1 norm of vector $\mathbf{v}$ and $\|\mathbf{v}\|$ is the ℓ_2 norm of $\mathbf{v}$. For $\hat{\mathbf{v}} = (\mathbf{v}_1, \cdots, \mathbf{v}_n)^T$, we define $\|\hat{\mathbf{v}}\| = \sqrt{\sum_{i=1}^n \|\mathbf{v}_i\|^2}$

[4] We abuse the notion of one-time signature. Here we will only use the public key once; the actual signature scheme does not necessarily need to be a one-time signature scheme.

The continuous normal distribution over $\mathbb{R}^n$ centered at $\mathbf{v}$ with standard deviation σ is defined as $\rho^n_{\mathbf{v},\sigma}(\mathbf{x}) = (\frac{1}{\sqrt{2\pi\sigma^2}})^n e^{\frac{-\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2}}$. For simplicity, when $\mathbf{v}$ is the zero vector, we use $\rho^n_\sigma(\mathbf{x})$.

The discrete normal distribution over $\mathbb{Z}^n$ centered at $\mathbf{v} \in \mathbb{Z}^n$ with standard deviation σ is defined as $D^n_{\mathbf{v},\sigma}(\mathbf{x}) = \frac{\rho^n_{\mathbf{v},\sigma}(\mathbf{x})}{\rho^n_{\mathbf{v},\sigma}(\mathbb{Z}^n)}$.

We define the exclusive-or operation of two matrix $\mathbf{X}^{(1)} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{X}^{(2)} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{X}^{(1)} \oplus \mathbf{X}^{(2)}$, as:

$$\begin{bmatrix} \mathsf{b}_q(x^{(1)}_{11}) \oplus \mathsf{b}_q(x^{(2)}_{11}) & \cdots & \mathsf{b}_q(x^{(1)}_{1m}) \oplus \mathsf{b}_q(x^{(2)}_{1m}) \\ \vdots & \ddots & \vdots \\ \mathsf{b}_q(x^{(1)}_{n1}) \oplus \mathsf{b}_q(x^{(2)}_{n1}) & \cdots & \mathsf{b}_q(x^{(1)}_{nm}) \oplus \mathsf{b}_q(x^{(2)}_{nm}) \end{bmatrix}$$

where $\mathsf{b}_q(x)$ means that transform a value $x \in \mathbb{Z}_q$ to its binary representation. $\mathsf{b}_q(.)$ can be efficiently computed.

2.2 Lattices and Hardness Assumptions

A lattice in m-dimension Euclidean space $\mathbb{R}^m$ is a discrete set

$$\Lambda(\mathbf{b}_1, \cdots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i | x_i \in \mathbb{Z} \right\}$$

of all integral combinations of n linear independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n$ in $\mathbb{R}^m$ ($m \leq n$). We call matrix $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_n] \in \mathbb{R}^{m\times n}$ a basis of lattice Λ. Using matrix notation, a lattice can be defined as

$$\Lambda(\mathbf{B}) = \{\mathbf{B}\mathbf{x} | \mathbf{x} \in \mathbb{Z}^n\}$$

The discrete Gaussian distribution of a lattice Λ, parameter s and center $\mathbf{v}$ is defined as $D_{\Lambda,\mathbf{v},s}(\mathbf{x}) = \frac{\rho_{\mathbf{v},s}(\mathbf{x})}{\rho_{\mathbf{v},s}(\Lambda)}$.

Definition 1 ([11]) *For any lattice $\Lambda \in \mathbb{R}^m$ and positive real number $s > 0$, we have* $\Pr_{\mathbf{x}\leftarrow D_{\Lambda,s}}[\|\mathbf{x}\| \leq \sqrt{m}s] \geq 1 - 2^{-\Omega(m)}$.

Definition 2 *Let $m \geq n \geq 1$ and $q \geq 2$. For arbitrary matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and vector $\mathbf{u} \in \mathbb{Z}_q^n$ define m-dimensional integer lattices:*

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \mod q\},$$

$$\Lambda^\perp_{\mathbf{u}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \mod q\}.$$

Short Integer Solution (SIS) problem and Inhomogeneous Short Integer Solution (ISIS) problem are two average-case hard problems frequently used in lattice-based cryptography constructions.

Definition 3 (SIS$_{q,n,m,\beta}$ problem) *Given a uniformly chosen matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, find* $\mathbf{x} \in \Lambda^{\perp}(\mathbf{A})$ *and* $0 < \|\mathbf{x}\| \leq \beta$.

Definition 4 (ISIS$_{q,n,m,\beta}$ problem) *Given a uniformly chosen matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and vector* $\mathbf{u} \in \mathbb{Z}_q^n$*, find* $\mathbf{x} \in \Lambda_{\mathbf{u}}^{\perp}(\mathbf{A})$ *and* $0 < \|\mathbf{x}\| \leq \beta$.

According to [29], if $q \geq \omega(\sqrt{n \log n})\beta$ and $m, \beta = poly(n)$, then SIS$_{q,n,m,\beta}$ and ISIS$_{q,n,m,\beta}$ is at least as hard as a standard worst-case lattice problem SIVP$_\gamma$ with $\gamma = \tilde{O}(\beta n)$.

In the ring version, there is Ring-SIS (Ring-ISIS) problem as an analogue of SIS (ISIS) problem.

Definition 5 (R-SIS$_{q,m,\beta}$ problem) *Given a uniformly chosen vector* $\hat{\mathbf{a}} \in \mathcal{R}_q^m$*, find* $\hat{\mathbf{x}} \in \mathcal{R}^m$ *such that* $\hat{\mathbf{a}}^T \cdot \hat{\mathbf{x}} = 0$ *and* $0 < \|\hat{\mathbf{x}}\| \leq \beta$.

Definition 6 (R-ISIS$_{q,m,\beta}$ problem) *Given a uniformly chosen vector* $\hat{\mathbf{a}} \in \mathcal{R}_q^m$ *and a ring element* $\mathbf{u} \in \mathcal{R}_q$*, find* $\hat{\mathbf{x}} \in \mathcal{R}^m$ *such that* $\hat{\mathbf{a}}^T \cdot \hat{\mathbf{x}} = \mathbf{u}$ *and* $0 < \|\hat{\mathbf{x}}\| \leq \beta$.

The R-SIS problem was concurrently introduced in [49, 43]. According to [43], the R-SIS$_{q,m,\beta}$ is as hard as the SVP$_\gamma$ problem for $\gamma = \tilde{O}(n\beta)$ in all lattice that are ideals in $\mathcal{R}$ if $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$, where n is a power of 2.

Definition 7 (NTRU assumption) *Let* $\mathbf{a} = \mathbf{g}/\mathbf{f}$ *over* $\mathcal{R}_q$ *where* $\|\mathbf{f}, \mathbf{g}\|_1$ *is bounded by some parameter* $\beta < q$*. The NTRU assumption says it is hard to distinguish* $\mathbf{a}$ *from a uniformly random element from* $\mathcal{R}_q$.

Over the years, there has been a few different versions of the NTRU assumption [35, 52, 41]. Here we use a decisional version that is most convenient for our proof. Note that this assumption holds as long as GapSVP problem is hard for NTRU lattices.

2.3 Preimage Sampleable Functions and Falcon

Generating a 'hard' public basis $\mathbf{A}$ (chosen at random from some appropriate distribution) of some lattice Λ, together with a 'good' trapdoor basis $\mathbf{T}$ has been studied since the work of Ajtai [2]. In 2008, Gentry, Peikert and Vaikuntanathan [30] construct a preimage sampleable function using the 'hard' public basis and trapdoor basis and apply it as a building block to lattice-based signature schemes. This celebrated work (referred to as the GPV framework) is followed by a sequence of improvements. Alwen and Peikert [5] is able to generate a shorter trapdoor, compared to [30]; while Peikert [48] provides a parallelizable algorithm to sample preimages. To the best of our knowledge, the most efficient construction following this direction while maintaining a security proof is due to Micciancio and Peikert [45]. The following Theorem is abstracted from their results.

Theorem 1 ([45], Theorem 5.1). *There exists an efficient algorithm* GenBasis $(1^n,\ 1^m,\ q)$ *that given any integers* $n \leq 1$, $q \leq 2$, *and sufficiently large* $m = O(n \log q)$, *outputs a parity-check matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and a 'trapdoor'* $\mathbf{T}$ *such that the distribution of* $\mathbf{A}$ *is* $\mathsf{negl}(n)$*-far from uniform. Moreover, there is an efficient algorithm* PreSample. *With overwhelming probability over all random choices, for any* $\mathbf{u} \in \mathbb{Z}_q^n$ *and large enough* $s = O(\sqrt{n \log q})$, PreSample$(\mathbf{A},\ \mathbf{T},\ \mathbf{u},\ s)$ *samples from a distribution within* $\mathsf{negl}(n)$ *statistical distance of* $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}), s\cdot\omega(\sqrt{\log n})}$.

On the other hand, the most efficient GPV construction in practice is due to Prest at al. [24, 28] using NTRU lattices [35]. The corresponding signature scheme is named FALCON [28].

FALCON is a candidate lattice-based signature scheme to the NIST post-quantum standardization process [46]. It is the resurrection of NTRUSign [33] with the aforementioned GPV framework for transcript security [30, 24], and a fast Fourier sampling for efficiency [25]. It is by far the most practical candidates among all submitted proposals, in terms of the combined sizes of public keys and signatures; and the only solution that provides a preimage sampleable function. In terms of security,

- FALCON stems from the provable secure GPV construction [29], under the (quantum) random oracle model [14];
- although the parameters in FALCON does not support GPV's security proof, they are robust against best known attacks[5].

2.4 Ring Signatures

In this section, we are going to give the syntax for ring signatures.

Syntax A ring signature scheme usually is a tuple of four algorithms (**Setup**, **KeyGen**, **Signing**, **Verification**):

- **Setup**$(1^\lambda)\rightarrow$ param: On input security parameter 1^λ, this algorithm generates system parameter param. We assume param is an implicit input to all the algorithms listed below.
- **KeyGen**$\rightarrow$ (sk, pk): By taking system parameter param, this key generation algorithm generates a private signing key sk and a public verification key pk.
- **Signing**$(\mathsf{sk}, \mu, L_{\mathsf{pk}}) \rightarrow \sigma$: On input message μ, a list of user public keys L_{pk}, and signing key sk of one of the public keys in L_{pk}, the signing algorithm outputs a ring signature σ on μ.
- **Verification**$(\mu, \sigma, L_{\mathsf{pk}})\rightarrow$ *accept*/*reject*: On input message μ, signature σ and list of user public keys L_{pk}, the verification algorithm outputs *accept* if σ is legitimately created; *reject*, otherwise.

[5] In practical lattice-based cryptography, it is common to derive parameters from best known attacks other than security proofs. For example, see [4, 3].

Correctness: the scheme is correct if signatures generated according to above specification are always accepted during verification.

The security requirements for a ring signature scheme have two aspects: unforgeability and anonymity. Before presenting their definitions, we first introduce the following oracles which can be used by adversaries in breaking the security of ring signature schemes:

– *Registration Oracle* $\mathcal{RO}(\perp) \rightarrow \mathsf{pk}_i$: On request, $\mathcal{RO}$ generates a new user and returns the public key of the new user.
– *Corruption Oracle* $\mathcal{CO}(\mathsf{pk}_i) \rightarrow \mathsf{sk}_i$: On input a user public key pk_i that is a query output of $\mathcal{RO}$, $\mathcal{CO}$ returns corresponding secret key sk_i.
– *Signing Oracle* $\mathcal{SO}(\mu, L_{\mathsf{pk}}, \mathsf{pk}_\pi) \rightarrow \sigma$: On input a list of user public keys L_{pk}, message μ and the public key of the signer $\mathsf{pk}_\pi \in L_{\mathsf{pk}}$, $\mathcal{SO}$ returns a valid signature σ on μ and L_{pk}.

Unforgeability The unforgeability of a ring signature scheme is defined via the following game, denoted by $\mathsf{Game_{forge}}$, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

– *Setup.* The challenger $\mathcal{C}$ runs Setup with security parameter 1^λ and generates system parameter param. $\mathcal{C}$ sends param to $\mathcal{A}$.
– *Query.* The adversary $\mathcal{A}$ may query $\mathcal{RO}$, $\mathcal{CO}$ and $\mathcal{SO}$ for a polynomial bounded number of times in an adaptive manner.
– *Output.* The adversary $\mathcal{A}$ outputs a forgery $(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$. $\mathcal{A}$ wins $\mathsf{Game_{forge}}$ if
  - **Verification**$(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) = \mathit{accept}$;
  - $(\mu^*, L^*_{\mathsf{pk}})$ has not been queried by $\mathcal{A}$; and
  - no public key in L^*_{pk} has been input to $\mathcal{CO}$.

The advantage of $\mathcal{A}$, denoted by $\mathbf{adv}^{\mathsf{forge}}_{\mathcal{A}}$, is defined by the probability that $\mathcal{A}$ wins $\mathsf{Game_{forge}}$:

$$\mathbf{adv}^{\mathsf{forge}}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins } \mathsf{Game_{forge}}]$$

Definition 8 (Unforgeability) *A ring signature scheme (**KeyGen**, **Signing**, **Verification**) is said to be unforgeable if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{adv}^{\mathsf{forge}}_{\mathcal{A}}$ is negligible.*

Anonymity For a ring signature scheme, this notion captures that it is impossible for an adversary to identify the actual signer with probability greater than $\frac{1}{n}$ where n is the size of the ring. More specifically, the anonymity of a ring signature scheme can be defined by the following game, denoted by $\mathsf{Game_{anon}}$, between adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

– *Setup.* The challenger $\mathcal{C}$ runs Setup with security parameter 1^λ and sends the system parameter param to $\mathcal{A}$.
– *Query.* The adversary $\mathcal{A}$ may query $\mathcal{RO}$ and $\mathcal{CO}$ in an adaptive manner.

- *Challenge.* $\mathcal{A}$ picks a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \mathsf{pk}_2, \cdots, \mathsf{pk}_n\}$ and a message μ. $\mathcal{A}$ sends (L_{pk}, μ) to $\mathcal{C}$. $\mathcal{C}$ randomly picks $\pi \in \{1, \cdots, n\}$ and runs **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \rightarrow \sigma$. $\mathcal{C}$ sends σ to $\mathcal{A}$.
- *Output.* $\mathcal{A}$ outputs a guess $\pi^* \in \{1, \cdots, n\}$.

$\mathcal{A}$ wins $\mathsf{Game}_{\mathsf{anon}}$ if $\pi^* = \pi$. The advantage of $\mathcal{A}$ is defined by

$$\mathbf{adv}_{\mathcal{A}}^{\mathsf{anon}} = |\Pr[\pi^* = \pi] - \frac{1}{n}|.$$

Definition 9 (Anonymity) *A ring signature scheme (**KeyGen**, **Signing**, **Verification**) is said to be anonymous (resp. unconditionally anonymous) if for any polynomial-time adversary (resp. unbounded adversary) $\mathcal{A}$, $\mathbf{adv}_{\mathcal{A}}^{\mathsf{anon}}$ is negligible.*

2.5 Linkable Ring Signatures

In this section, we are going to present the syntax of linkable ring signatures. We emphasize that the linkable ring signature here is one-time linkable ring signature and the public key for a signer is only supposed to use once.

Syntax A linkable ring signature scheme usually consists of five algorithms, namely, (**Setup**, **KeyGen**, **Signing**, **Verification**, **Link**):

- **Setup**$(1^\lambda) \rightarrow$ param: On input the security parameter 1^λ, this algorithm generates the system parameter param. We assume param is an implicit input to all the algorithms listed below.
- **KeyGen**$\rightarrow$ $(\mathsf{sk}, \mathsf{pk})$: By taking the system parameter param, this key generation algorithm generates a private signing key sk and a public verification key pk.
- **Signing**$(\mathsf{sk}, \mu, L_{\mathsf{pk}}) \rightarrow \sigma$: On input a message μ, a list of user public keys L_{pk}, and a signing key sk of one of the public keys in L_{pk}, the signing algorithm outputs a ring signature σ on μ.
- **Verification**$(\mu, \sigma, L_{\mathsf{pk}}) \rightarrow$ *accept/reject*: On input a message μ, a signature σ and a list of user public keys L_{pk}, the verification algorithm outputs *accept* if σ is legitimately created. otherwise, output *reject*.
- **Link** $(\sigma_1, \sigma_2, \mu_1, \mu_2, L_{\mathsf{pk}}^{(1)}, L_{\mathsf{pk}}^{(2)}) \rightarrow$ *linked/unlinked*: Takes two messages μ_1, μ_2 and their signatures σ_1 and σ_2 as input, output *linked* or *unlinked*.

Correctness: the scheme is correct if

- signatures signed as above is always accepted during verification; and
- two legally signed signatures are linked if and only if they share a same signer.

The security of a linkable ring signature should have four aspects, namely, unforgeability, anonymity, linkability and nonslanderability. Same as the security notions for ring signatures, there are also three oracles, namely, $\mathcal{RO}$, $\mathcal{CO}$ and $\mathcal{SO}$ jointly model the ability of an adversary:

The security definition of unforgeability for linkable ring signatures remains the same as in section **??**. The definitions of anonymity, linkability and nonslanderability are adopted from Liu et al. [39].

Anonymity We require that, for a secure linkable ring signature scheme, it should be impossible for an adversary to identify the actual signer with probability greater than $\frac{1}{n}$ where n is the size of the ring. More specifically, the anonymity of a linkable ring signature scheme can be defined by the following game, $\mathsf{Game}^*_{\mathsf{anon}}$, held between adversary $\mathcal{A}$ and challenger $\mathcal{C}$. The difference between $\mathsf{Game}^*_{\mathsf{anon}}$ and $\mathsf{Game}_{\mathsf{anon}}$ is that, in $\mathsf{Game}^*_{\mathsf{anon}}$, $\mathcal{A}$ is only allowed to query register oracle $\mathcal{RO}$.

- *Setup.* The challenger $\mathcal{C}$ runs Setup with security parameter 1^λ and sends the system parameter param to $\mathcal{A}$.
- *Query.* The adversary $\mathcal{A}$ may query $\mathcal{RO}$ in an adaptive manner.
- *Challenge.* $\mathcal{A}$ picks a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \mathsf{pk}_2, \cdots, \mathsf{pk}_n\}$ and a message μ. $\mathcal{A}$ sends (L_{pk}, μ) to $\mathcal{C}$. $\mathcal{C}$ randomly picks $\pi \in \{1, \cdots, n\}$ and runs **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \rightarrow \sigma$. $\mathcal{C}$ sends σ to $\mathcal{A}$.
- *Output.* $\mathcal{A}$ outputs a guess $\pi^* \in \{1, \cdots, n\}$.

$\mathcal{A}$ wins $\mathsf{Game}^*_{\mathsf{anon}}$ if $\pi^* = \pi$. The advantage of $\mathcal{A}$ is defined by

$$\mathbf{adv}^{\mathsf{anon}}_{\mathcal{A}} = |\Pr[\pi^* = \pi] - \frac{1}{n}|.$$

Definition 10 (Anonymity) *A linkable ring signature scheme is said to be anonymous (resp. unconditionally anonymous) if for any polynomial-time adversary (resp. unbounded adversary) $\mathcal{A}$, $\mathbf{adv}^{\mathsf{anon}}_{\mathcal{A}}$ is negligible.*

Linkability This notion captures that **Link** algorithm always outputs *linked* for two signatures generated by a same signer. We use the following game, $\mathsf{Game}_{\mathsf{link}}$, between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ to define linkability:

- *Setup.* The challenger $\mathcal{C}$ runs Setup and gives $\mathcal{A}$ system parameter param.
- *Query.* The adversary $\mathcal{A}$ is given access to $\mathcal{RO}$, $\mathcal{CO}$, $\mathcal{SO}$ and may query the oracles in an adaptive manner.
- *Output.* $\mathcal{A}$ outputs two sets, $\{L^{(1)}_{\mathsf{pk}}, \mu_1, \sigma_1\}$ and $\{L^{(2)}_{\mathsf{pk}}, \mu_2, \sigma_2\}$, where $L^{(1)}_{\mathsf{pk}}$ and $L^{(2)}_{\mathsf{pk}}$ are two lists of public keys, μ_1 and μ_2 are messages, σ_1 and σ_2 are two signatures.

$\mathcal{A}$ wins the game if

- all public keys in $L^{(1)}_{\mathsf{pk}}$ and $L^{(2)}_{\mathsf{pk}}$ are query output of $\mathcal{RO}$;
- **Verification**$(\mu_1, \sigma_1, L^{(1)}_{\mathsf{pk}})$ = *Accept* ;
- **Verification**$(\mu_2, \sigma_2, L^{(2)}_{\mathsf{pk}})$ = *Accept*;
- $\mathcal{A}$ queried $\mathcal{CO}$ less than two times; and
- **Link**$(\sigma_1, \sigma_2, \mu_1, \mu_2)$ = *unlinked.*

The advantage of $\mathcal{A}$ is defined by the probability $\mathcal{A}$ wins $\mathsf{Game}_{\mathsf{link}}$:

$$\mathbf{adv}^{\mathsf{link}}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}_{\mathsf{link}}]$$

Definition 11 (Linkability) *A linkable ring signature scheme is linkable if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{adv}^{\mathsf{link}}_{\mathcal{A}}$ is negligible.*

Nonslanderability The nonslanderability requires that a signer cannot frame other honest signers for generating a signature linked with another signature not signed by the signer. We use the following game, $\mathsf{Game}_{\mathsf{slander}}$, to define the nonslanderability of a linkable ring signature scheme:

– *Setup.* The challenger $\mathcal{C}$ runs Setup and gives $\mathcal{A}$ system parameter param.
– *Query.* The adversary $\mathcal{A}$ is given access to $\mathcal{RO}$, $\mathcal{CO}$, $\mathcal{SO}$ and may query the oracles in an adaptive manner.
– *Challenge.* $\mathcal{A}$ gives $\mathcal{C}$ a list of public keys L_{pk}, a message μ and a public key $\mathsf{pk}_\pi \in L_{\mathsf{pk}}$. $\mathcal{C}$ runs $\mathbf{Signing}(\mathsf{sk}, \mu, L_{\mathsf{pk}})$ and returns the corresponding signature σ to $\mathcal{A}$. $\mathcal{A}$ still can queries oracles with arbitrary interleaving.
– *Output.* $\mathcal{A}$ outputs a list of public keys L^*_{pk}, message μ^*, and a signature σ^*.

$\mathcal{A}$ wins $\mathsf{Game}_{\mathsf{slander}}$ if the following holds:

– $\mathbf{Verification}(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) = \mathit{accept}$;
– pk_π is not queried by $\mathcal{A}$ to $\mathcal{CO}$;
– pk_π is not queried by $\mathcal{A}$ as an insider to $\mathcal{SO}$;
– all public keys in L^*_{pk} and L_{pk} are query outputs of $\mathcal{RO}$; and
– $\mathbf{Link}(\sigma, \sigma^*, \mu, \mu^*) = \mathit{linked}$.

The advantage of $\mathcal{A}$ is defined by:

$$\mathbf{adv}_{\mathcal{A}}^{\mathsf{slander}} = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}_{\mathsf{slander}}]$$

Definition 12 (Nonslanderability) *A linkable ring signature scheme is non-sladerable if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{adv}_{\mathcal{A}}^{\mathsf{slander}}$ is negligible.*

Theorem 2. *[[10], Sec 3.2] If a linkable ring signature scheme is linkable and nonslanderable, it is also unforgeable.*

3 Generic Construction

3.1 Chameleon Hash Plus

Now we are ready to give the definition to chameleon hash plus CH+. CH+ can be considered as a more powerful variants of Chameleon hash functions. A CH+ usually consists of four algorithms, namely, SetUp, $\mathsf{TrapGen}$, Hash and Inv, as follow:

– $\mathsf{SetUp}(1^\lambda) \to \mathsf{param}^{\mathsf{ch}}$: On input the security parameter 1^λ, this algorithm generates system parameter $\mathsf{param}^{\mathsf{ch}}$. $\mathsf{param}^{\mathsf{ch}}$ will be an implicit input to SamK and Hash.
– $\mathsf{TrapGen}(1^\lambda) \to (\mathsf{hk}, \mathsf{tr})$: This algorithm takes security parameter 1^λ as input and then returns a pair $(\mathsf{hk}, \mathsf{tr})$ where hk is a hash key to CH+, tr is the trapdoor of hk.
– $\mathsf{Hash}(\mathsf{hk}, m, r) \to C$: On input a hash key hk, a message m and a randomness r, this algorithm returns a hash output C.

– $\mathsf{Inv}(\mathsf{hk}, \mathsf{tr}, C, m') \to r'$: On input a hash key hk, a trapdoor tr, a hash output C and a message m', this algorithm returns randomness r' such that $\mathsf{Hash}(\mathsf{hk}, m', r') = C$.

We require CH+ to satisfy following requirements:

1. CH+ should be one-way and collision resistant. In other words, for all PPT $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that
$$\Pr[\{(m_0, r_0), (m_1, r_1)\} \leftarrow \mathcal{A}(1^\lambda, \mathsf{hk}, \mathsf{param}^{\mathsf{ch}}) : (m_0, r_0) \neq$$
$$(m_1, r_1) \wedge \mathsf{Hash}(\mathsf{hk}, m_0, r_0) = \mathsf{Hash}(\mathsf{hk}, m_1, r_1)] = \mathsf{negl}(\lambda);$$
$$\Pr[(m, r) \leftarrow \mathcal{A}(1^\lambda, C, \mathsf{hk}, \mathsf{param}^{\mathsf{ch}}) : \mathsf{Hash}(\mathsf{hk}, m, r) = C] = \mathsf{negl}(\lambda).$$
2. For hash key hk generated from $\mathsf{TrapGen}$, assuming the range of hk is R_{hk}, the distribution of hk should be either statistically close to uniform in R_{hk}; or computationally close to the uniform distribution with an additional property that the probability a randomly sampled $\bar{\mathsf{hk}} \leftarrow_\$ R_{\mathsf{hk}}$ has a trapdoor is negligible.
3. For r' generated from Inv, the distribution of r' should be with $\mathsf{negl}(\lambda)$ distance from the distribution where r is sampled from.

Looking ahead, in the next section, we will show how to build CH+ from standard lattice problems and from NTRU assumptions.

3.2 A new framework for ring signatures

Our ring signature is constructed as follows:

– **Setup**$(1^\lambda) \to \mathsf{param}$: On input the security parameter 1^λ, this algorithm chooses a hash function $\mathcal{H} : \{*\} \to D_b$. It also runs $\mathsf{SetUp}(1^\lambda) \to \mathsf{param}^{\mathsf{ch}}$.
– **KeyGen** $\to (\mathsf{sk}, \mathsf{pk})$: This algorithm generates $(\mathsf{hk}, \mathsf{tr}) \leftarrow \mathsf{TrapGen}(1^\lambda)$. Then it sets public key $\mathsf{pk} = \mathsf{hk}$ and secret key $\mathsf{sk} = \mathsf{tr}$.
– **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \to \sigma$: On input a message μ, a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and a signing key $\mathsf{sk}_\pi = \mathsf{tr}_\pi$ of $\mathsf{pk}_\pi = \mathsf{hk}_\pi \in L_{\mathsf{pk}}$, the signing algorithm runs as follow:
  1. For $i \in [1, \cdots, \ell]$ and $i \neq \pi$, pick m_i and r_i at random. Compute $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$. For $i = \pi$, pick C_π at random.
  2. Compute m_π such that
  $$m_1 \oplus \cdots \oplus m_\pi \oplus \cdots \oplus m_n = \mathcal{H}(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}}).$$
  3. Given m_π and C_π, invoke $\mathsf{Inv}(\mathsf{hk}_\pi, \mathsf{tr}_\pi, C_\pi, m_\pi) \to r_\pi$.

  The ring signature of μ and L_{pk} is $\sigma = \{(m_1, r_1), \cdots, (m_\ell, r_\ell)\}$.
– **Verification**$(\mu, \sigma, L_{\mathsf{pk}}) \to$ *accept/reject*: On input a message μ, a signature σ and a list of user public keys L_{pk}, the verification algorithm first phrases $\sigma = \{(m_1, r_1), \cdots, (m_\ell, r_\ell)\}$. It then checks whether each pair of (m_i, r_i) satisfies $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$ for all $i \in [1, \cdots, \ell]$ and whether $m_1 \oplus \cdots \oplus m_\ell = \mathcal{H}(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}})$. If yes, output *accept*. Otherwise, output *reject*.

Security proof

Theorem 3 (Unforgeability). *Our generic ring signature scheme is unforgeable in random oracle model if CH+ is collision resistant.*

Proof. Assume there is an adversary $\mathcal{A}$ who can successfully forge a ring signature with probability δ by making at most q_r queries to $\mathcal{RO}$ oracle, q_c queries to $\mathcal{CO}$ oracle, q_s queries to $\mathcal{SO}$ oracle, and q_h queries to random oracle $\mathcal{H}$. We define the number of possible values in the output range of $\mathcal{H}$ as $\mathcal{D}_{\mathcal{H}}$. Then we can construct a simulator $\mathcal{S}$ who can break the collision resistance of CH+ with a non-negligible probability.

$\mathcal{S}$ is given an instance as following: Given CH+ hash key hk_c and CH+ system parameter ${\mathsf{param}^{\mathsf{ch}}}_c$, it is asked to output $\{(m', r'), (m'', r'')\}$ such that $(m', r') \neq (m'', r'')$ and $\mathsf{Hash}(\mathsf{hk}', m', r') = \mathsf{Hash}(\mathsf{hk}', m'', r'')$ for ${\mathsf{param}^{\mathsf{ch}}}'$. In order to use $\mathcal{A}$ to solve this problem instance, the simulator $\mathcal{S}$ needs to simulate the challenger $\mathcal{C}$ and oracles to play $\mathsf{Game}_{\mathsf{forge}}$ with $\mathcal{A}$. $\mathcal{S}$ runs as follow:
Setup. Simulator $\mathcal{S}$ picks a hash function $\mathcal{H}$. $\mathcal{H}$ will be modeled as a random oracle. $\mathcal{S}$ picks $\{h_1, h_2, \cdots, h_p\} \leftarrow_{\$} \mathcal{D}_{\mathcal{H}}$ as the q_h responses of the random oracle. $\mathcal{S}$ gives random coin ϕ to $\mathcal{A}$. Hash function $\mathcal{H}$ and ${\mathsf{param}^{\mathsf{ch}}}_c$ are set as system parameter.
Oracle Simulation. $\mathcal{S}$ simulates the oracles as follow:

- $\mathcal{RO}(\perp)$: Assume the adversary $\mathcal{A}$ can only queries $\mathcal{RO}$ q_r times ($q_r \geq 1$). $\mathcal{S}$ randomly picks an index $\mathcal{I} \in [1, \cdots, q_r]$. For index $\mathcal{I}$, $\mathcal{S}$ assigns hk_c to index $\mathcal{I}$ as the public key. For other indexes, $\mathcal{S}$ generates the public key and secret key according to the **KeyGen** algorithm. Upon the jth query, $\mathcal{S}$ returns the corresponding public key.
- $\mathcal{CO}(\mathsf{pk})$: On input a public key pk returned by $\mathcal{RO}$ oracle, $\mathcal{S}$ first checks whether it corresponds to the index $\mathcal{I}$. If yes, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ returns the corresponding secret key sk.
- $\mathcal{SO}(\mu, L_{\mathsf{pk}}, \mathsf{pk}_\pi)$: When $\mathcal{A}$ queries $\mathcal{SO}$ on message μ, a list of public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$ and the public key for the signer pk_π where $\mathsf{pk}_\pi \in L_{\mathsf{pk}}$, $\mathcal{S}$ simulates $\mathcal{SO}$ as follow:
  - If $\mathsf{pk}_\pi \neq \mathsf{pk}_{\mathcal{I}}$, $\mathcal{S}$ runs **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}})$ where the output of the random oracle will be programmed as the first $h_i \in \{h_1, h_2, \cdots, h_p\}$ that has not been used yet. $\mathcal{S}$ returns the signature σ to $\mathcal{A}$;
  - If $\mathsf{pk}_\pi = \mathsf{pk}_{\mathcal{I}}$, for $i \in [1, \cdots, \ell]$ and $\mathsf{pk}_i = \mathsf{hk}_i$, $\mathcal{S}$ samples m_i, r_i and computes $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$. $\mathcal{S}$ then programs random oracle as $\mathcal{H}(\mu, C_1, \cdots, C_\ell) = m_1 \oplus \cdots \oplus m_\ell$.
- *Random Oracle* $\mathcal{H}$: For a query input that has already been programmed, $\mathcal{S}$ returns the corresponding output. Otherwise, the output of the random oracle will be the first $h_i \in \{h_1, h_2, \cdots, h_p\}$ that has not been used yet. $\mathcal{S}$ will record all the queries to the random oracle in a table, in case same query is issued twice.

Output. Finally, the adversary $\mathcal{A}$ will finish running and output a forgery (μ^*, σ^*, L^*_{pk}) with probability δ such that **Verification**$(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) = $ *accept*; $(\mu^*, L^*_{\mathsf{pk}})$

has not been queried by $\mathcal{A}$ for signature; and no public key in L^*_{pk} has been input to $\mathcal{CO}$. If $\mathsf{pk}_{\mathcal{I}} \notin L^*_{\mathsf{pk}}$, $\mathcal{S}$ aborts.

Simulator $\mathcal{S}$ then uses the forgery $(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$ to solve the problem instance. $\mathcal{S}$ phrases σ^* to $\{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell)\}$ and denotes $m^*_1 \oplus \cdots \oplus m^*_\ell$ by h^*. Notice that with probability $1 - \frac{1}{|\mathcal{D}_{\mathcal{H}}|}$, h^* will be one of the $h_i \in \{h_1, \cdots, h_p\}$ or the hash outputs from the $\mathcal{SO}$ queries. Since if the random oracle was not queried or programmed on some input, the probability for $\mathcal{A}$ to produce a $\{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell)\}$ such that $m^*_1 \oplus \cdots \oplus m^*_\ell = \mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}})$ where $C^*_i = \mathsf{Hash}(\mathsf{pk}^*_i, m^*_i, r^*_i)$ is $\frac{1}{|\mathcal{D}_{\mathcal{H}}|}$. The probability for $\mathcal{A}$ to produce a forgery is δ. Thus, the probability for $\mathcal{A}$ outputs a forgery $(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$ and $h^* = \mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}})$ has been queried in $\mathcal{SO}$ or $\mathcal{RO}$ is $\delta - \frac{1}{|\mathcal{D}_{\mathcal{H}}|}$.

Type 1 forgery: The first type of forgery is that, for the forgery $(\mu^*, \sigma^* = \{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell)\}, L^*_{\mathsf{pk}})$, h^* is a response of random oracle $\mathcal{H}$ on $\mathcal{H}(\mu', C'_1, \cdots, C'_{\ell'}, L'_{\mathsf{pk}})$ during a $\mathcal{SO}$ query. Then, we have

$$\mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}}) = \mathcal{H}(\mu', C'_1, \cdots, C'_{\ell'}, L'_{\mathsf{pk}}).$$

If $\mu^* \neq \mu'$, $(C^*_1, \cdots, C^*_\ell) \neq (C'_1, \cdots, C'_{\ell'})$ or $L'_{\mathsf{pk}} \neq L^*_{\mathsf{pk}}$, we find a collision to the hash function. Thus, we must have $\mu^* = \mu'$, $(C^*_1, \cdots, C^*_\ell) = C'_1, \cdots, C'_{\ell'}$ and $L'_{\mathsf{pk}} = L^*_{\mathsf{pk}}$. Since we require that $(\mu^*, L^*_{\mathsf{pk}})$ has not been queried by $\mathcal{A}$ for signature. Type 1 forgery is not a valid forgery.

Type 2 forgery: The second type of forgery is that, $h^* = m_1 \oplus \cdots \oplus m_\ell$ is a response of a $\mathcal{RO}$ query issued by $\mathcal{A}$. We store the forgery $(\mu^*, \sigma^* = \{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell)\}, L^*_{\mathsf{pk}})$. Assume $h^* = h_i$ where $h_i \in \{h_1, \cdots, h_p\}$, picks new $h'_i, \cdots, h'_p \leftarrow_{\$} D_H$. $\mathcal{S}$ then run $\mathsf{Game}_{\mathsf{forge}}$ again on $(\mathsf{hk}_c, \mathsf{param^{ch}}_c, \psi, \phi, h_1, \cdots, h_{i-1}, h'_i, \cdots, h'_p)$. According to the General Forking Lemma, we obtain that $h'_i \neq h_i$ and the adversary $\mathcal{A}$ uses the new random oracle response h'_i in its forgery is at least

$$\Pr = acc(\frac{acc}{q_s + q_h} - \frac{1}{|\mathcal{D}_{\mathcal{H}}|}),$$

where

$$acc = \frac{1}{q_r - q_c}(\delta - \frac{1}{|\mathcal{D}_{\mathcal{H}}|} - \frac{1}{q_r}).$$

That is, with the same probability, $\mathcal{A}$ will output a forgery $\{\mu', \sigma' = \{(m'_1, r'_1), \cdots, (m'_{\ell'}, r'_{\ell'})\}, L'_{\mathsf{pk}}\}$ and $\mu^* = \mu'$, $(C^*_1, \cdots, C^*_\ell) = (C'_1, \cdots, C'_{\ell'})$, $L'_{\mathsf{pk}} = L^*_{\mathsf{pk}}$. Thus, $\ell = \ell'$. Assuming $\mathsf{pk}_{\mathcal{I}} = \mathsf{pk}_j \in L^*_{\mathsf{pk}}(L'_{\mathsf{pk}})$, at least with probability $\frac{1}{\ell}$, $\mathcal{S}$ has $m^*_j \neq m'_j$. Since $C^*_j = C_j$, $\mathcal{S}$ finds a collision $(m^*_j, r^*_j), (m'_j, r'_j)$.

The probability for $\mathcal{S}$ aborting during $\mathcal{SO}$ is no more than $\frac{1}{q_r}$. The probability for $\mathcal{S}$ not aborting during *output* is no less than $\frac{1}{q_r - q_c}$. Thus, the probability for $\mathcal{S}$ solving problem instance is no less than

$$(1 - \frac{1}{q_r})(\frac{1}{q_r - q_c}) \cdot \Pr$$

which is non-negligible.

Theorem 4 (Anonymity). *Our ring signature scheme is unconditional anonymous.*

Proof. When $\mathsf{Game}_{\mathsf{anon}}$ is played between challenger $\mathcal{C}$ and adversary $\mathcal{A}$, for each $\mathcal{RO}$ query, **KeyGen** algorithm runs and public key $\mathsf{pk} = \mathsf{hk}$ is returned. For each $\mathcal{CO}(\mathsf{pk})$ query, secret key $\mathsf{sk} = \mathsf{tr}$ corresponding to hk will be returned. If adversary $\mathcal{A}$ ask for a signature on message μ and ring $\{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, $\mathcal{C}$ will random samples $\pi \leftarrow_\$ [1, \cdots, \ell]$ and signs μ using sk_π.

The signature will be in the form of $\sigma = \{(m_1, r_1), \cdots, (m_\ell, r_\ell)\}$. Assume the signer of the signature is s_π, for $i \neq \pi$, m_i and r_i are sampled by $\mathcal{S}$. For $i = \pi$, $m_\pi = m_1 \oplus \cdots \oplus m_{\pi-1} \oplus m_{\pi+1} \oplus \cdots \oplus m_\ell \oplus \mathcal{H}(\mu, C_0, \cdots, C_\ell)$, r_π is generated by $\mathsf{Inv}(\mathsf{hk}_\pi, \mathsf{tr}_\pi, C_\pi, m_\pi)$. Since m_i ($i \neq \pi$) is uniformly sampled and $\mathcal{H}$ is a hash function, the distribution of m_π should be also uniform over $\{0,1\}^k$. According to the requirements of CH+, the distribution of r_π is within $\mathsf{negl}(\lambda)$ statistical distance from the distribution $\mathcal{S}$ used to sample other randomness. Thus, the best way for an adversary to win this game is to make a guess. The probability for adversary to make a successful guess is no more than $\frac{1}{\ell}$. Thus, the advantage $\mathbf{adv}_{\mathcal{A}}^{\mathsf{anon}}$ of an adversary should be negligible. Our ring signature scheme is unconditional anonymous.

3.3 Linkable ring signatures

Our linkable ring signature is constructed as follows:

- **Setup**$(1^\lambda) \to \mathsf{param}$: On input the security parameter 1^λ, this algorithm chooses two hash functions $\mathcal{H}_0$ and $\mathcal{H}_1$. It also runs $\mathsf{SetUp}(1^\lambda) \to \mathsf{param}^{\mathsf{ch}}$ and selects a one-time signature scheme $\Pi^{OTS} = \{\mathsf{OKeygen}, \mathsf{OSign}, \mathsf{OVer}\}$.
- **KeyGen** $\to (\mathsf{sk}, \mathsf{pk})$:
  1. This algorithm first generates $(\mathsf{hk}, \mathsf{tr}) \leftarrow \mathsf{TrapGen}(1^\lambda)$.
  2. It also generates a pair of Π^{OTS} public key and secret key $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OKeygen}(1^\lambda)$ and computes $\mathsf{mk} = \mathcal{H}_1(\mathsf{opk})$.
  3. It computes $\mathsf{hk}' = \mathsf{hk} \oplus \mathsf{mk}$.
  4. It sets public key $\mathsf{pk} = \mathsf{hk}'$ and secret key $\mathsf{sk} = \{\mathsf{tr}, \mathsf{opk}, \mathsf{osk}\}$.
- **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \to \sigma$: On input a message μ, a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and a signing key $\mathsf{sk}_\pi = \{\mathsf{tr}_\pi, \mathsf{opk}_\pi, \mathsf{osk}_\pi\}$ of $\mathsf{pk}_\pi = \mathsf{hk}'_\pi \in L_{\mathsf{pk}}$, the signing algorithm runs as follow:
  1. Compute $\mathsf{mk}_\pi = \mathcal{H}_1(\mathsf{opk}_\pi)$.
  2. For $i \in [1, \cdots, n]$ and $i \neq \pi$, pick m_i and r_i at random. Compute $\mathsf{hk}_i = \mathsf{hk}'_i \oplus \mathsf{mk}_\pi$ and $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$. For $i = \pi$, pick C_π at random.
  3. Compute m_π such that
  $$m_1 \oplus \cdots \oplus m_\pi \oplus \cdots \oplus m_\ell = \mathcal{H}_0(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}}).$$
  4. Given m_π and C_π, compute $r_\pi \leftarrow \mathsf{Inv}(\mathsf{hk}_\pi, \mathsf{tr}_\pi, C_\pi, m_\pi)$.
  5. Compute one-time signature sig=OSign (osk_π; $(m_1, r_1), \cdots, (m_\ell, r_\ell)$,$L_{\mathsf{pk}}$, opk_π).

The linkable ring signature of μ and L_{pk} is σ=$\{(m_1, r_1), \cdots, (m_\ell, r_\ell), \mathsf{opk}_\pi, sig\}$.

- **Verification**$(\mu, \sigma, L_{\mathsf{pk}}) \rightarrow$ *accept/reject*: On input a message μ, a signature σ and a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{hk}'_1, \cdots, \mathsf{hk}'_\ell\}$, the verification algorithm first phrases $\sigma = \{(m_1, r_1), \cdots, (m_\ell, r_\ell), \mathsf{opk}, sig\}$. This algorithm runs as follow:
  1. It first computes $\mathsf{mk} = \mathcal{H}_1(\mathsf{opk})$. It also computes $\mathsf{hk}_i = \mathsf{hk}'_i \oplus \mathsf{mk}$ and $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$ for all $i \in [1, \cdots, \ell]$;
  2. It checks whether $m_1 \oplus \cdots \oplus m_\ell = \mathcal{H}_0\ (\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}})$;
  3. Verify the signature via $\mathsf{OVer}(\mathsf{opk}; sig; (m_1, r_1), \cdots, (m_\ell, r_\ell), L_{\mathsf{pk}}, \mathsf{opk})$.

  If all pass, output *accept*. Otherwise, output *reject*.
- **Link**$(\sigma_1, \sigma_2, \mu_1, \mu_2, L_p^{(1)}k, L_p^{(2)}k) \rightarrow$ *linked/unlinked*: On input two message signature pairs (μ_1, σ_1) and (μ_2, σ_2), this algorithm first checks the validity of signatures σ_1 and σ_2. If **Verification**$(\mu_1, \sigma_1, L_{\mathsf{pk}}^{(1)}) \rightarrow$ *accept* and **Verification**$(\mu_2, \sigma_2, L_{\mathsf{pk}}^{(2)}) \rightarrow$ *accept*, it phrases σ_1=$\{(m_1^{(1)}$,$r_1^{(1)})$, $\cdots$,$(m_\ell^{(1)}$, $r_\ell^{(1)})$, opk_1,$sig_1\}$ and $\sigma_2 = \{(m_1^{(2)}, r_1^{(2)}), \cdots, (m_\ell^{(2)}, r_\ell^{(2)}), \mathsf{opk}_2, sig_2\}$. The algorithm outputs *linked* if $\mathsf{opk}_1 = \mathsf{opk}_2$. Otherwise, output *unlinked*.

Security proof

Theorem 5 (Anonymity). *Our linkable ring signature scheme is anonymous in random oracle model if the second requirement in section 3.1 holds for CH+.*

Proof. Assume there is a simulator $\mathcal{S}$ who plays $\mathsf{Game}^*_{\mathsf{anon}}$ with adversary $\mathcal{A}$ as follow:
Setup. Simulator $\mathcal{S}$ runs **Setup**$(1^\lambda) \rightarrow \mathsf{param}$ and passes system parameter param to adversary $\mathcal{A}$.
Oracle Simulation. For registration oracle $\mathcal{RO}(\perp)$, when adversary queries $\mathcal{RO}$, $\mathcal{S}$ samples pk uniformly at random from its possible range.
Challenge. $\mathcal{A}$ picks a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \mathsf{pk}_2, \cdots, \mathsf{pk}_\ell\}$ and a message μ. $\mathcal{A}$ sends (L_{pk}, μ) to $\mathcal{S}$. $\mathcal{S}$ randomly picks $\pi \in \{1, \cdots, \ell\}$. $\mathcal{S}$ also generates a pair of Π^{OTS} public key and secret key $(\mathsf{opk}_\pi, \mathsf{osk}_\pi) \leftarrow \mathsf{OKeygen}$ for pk_π. For $i = \{1, \cdots, \ell\}$, $\mathcal{S}$ first computes $\mathsf{pk}'_i = \mathsf{pk}_i \oplus \mathcal{H}_1(\mathsf{opk}_\pi)$. It also picks m_i, r_i and computes $C_i = \mathsf{Hash}(\mathsf{pk}'_i, m_i, r_i)$. $\mathcal{S}$ programs $\mathcal{H}_0(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}}) = m_1 \oplus \cdots \oplus m_\ell$. Finally, it computes one-time signature $sig = \mathsf{OSign}(\mathsf{osk}_\pi; (m_1, r_1), \cdots, (m_\ell, r_\ell)$, L_{pk}, $\mathsf{opk}_\pi)$ and returns σ=$\{(m_1, r_1), \cdots, (m_\ell, r_\ell), \mathsf{opk}_\pi, sig\}$ as signature.

For adversary $\mathcal{A}$, $\mathcal{A}$ can not distinguish this game from the original one. Since in the scheme, the signer public key pk_i is the result of the exclusive or of hk_i and $\mathcal{H}_1(\mathsf{opk}_i)$ where $\mathcal{H}_1(\mathsf{opk}_i)$ is a hash output. Thus, $\mathcal{A}$ can not distinguish pk generated following the rule from pk sampled uniformly at random from its possible range.
Case 1: In case 1, we have the distribution of hk statistically close to uniform over R_{hk}. Thus for $i = 1, \cdots, \ell$, all the $\mathsf{pk}'_i = \mathsf{pk}_i \oplus \mathcal{H}_1(\mathsf{opk}_\pi)$ are indistinguishable from a true hash key for $\mathcal{A}$. The best way for $\mathcal{A}$ to win this game is to guess a $\pi^* \in \{1, \cdots, \ell\}$. The probability for $\pi^* = \pi$ is no more $\frac{1}{\ell}$.

$\mathsf{Case\ 2}$: In case 2, we have the distribution of the distribution of hk computationally close to the uniform distribution and the probability for $\bar{\mathsf{hk}} \leftarrow_{\$} R_{\mathsf{hk}}$ and $\bar{\mathsf{hk}}$ existing trapdoor is negligible. Thus for $i = 1, \cdots, \ell$, all $\mathsf{pk}'_i = \mathsf{pk}_i \oplus \mathcal{H}_1(\mathsf{opk}_\pi)$ are computationally indistinguishable from a true hash key for $\mathcal{A}$. Since pk'_i can be considered as sampled uniformly at random from R_{hk}, the probability for pk'_i having trapdoor is negligible. Thus for $\mathcal{A}$, the best way wining this game is to guess a $\pi^* \in \{1, \cdots, \ell\}$. The probability for $\pi^* = \pi$ is no more $\frac{1}{\ell}$.

The advantage $\mathbf{adv}^{\mathsf{anon}}_{\mathcal{A}}$ in this game is negligible. Our scheme is anonymous.

Theorem 6 (Linkability). *Our linkable ring signature is linkable in random oracle model if CH+ is collision resistant.*

Proof. Assume there is an adversary $\mathcal{A}$ who can successfully forge a linkable ring signature with probability δ by making at most q_r queries to $\mathcal{RO}$ oracle, q_c queries to $\mathcal{CO}$ oracle, q_s queries to $\mathcal{SO}$ oracle, and q_h queries to random oracle $\mathcal{H}_0$. We define the number of possible values in the output range of $\mathcal{H}_0$ as $|\mathcal{D}_{\mathcal{H}}|$. Then we can construct a simulator $\mathcal{S}$ who can break the collision resistance of CH+ with a non-negligible probability.

$\mathcal{S}$ is given an instance as following: Given CH+ hash key hk_c and CH+ parameter $\mathsf{param^{ch}}_c$, it is asked to output $\{(m', r'), (m'', r'')\}$ such that $(m', r') \neq (m'', r'')$ and $\mathsf{Hash}(\mathsf{hk}', m', r') = \mathsf{Hash}(\mathsf{hk}', m'', r'')$ for $\mathsf{param^{ch}}_c$. In order to use $\mathcal{A}$ to solve this problem instance, the simulator $\mathcal{S}$ needs to simulate the challenger $\mathcal{C}$ and oracles to play $\mathsf{Game_{forge}}$ with $\mathcal{A}$. $\mathcal{S}$ runs as follow:
Setup. Simulator $\mathcal{S}$ picks two hash functions $\mathcal{H}_0$, $\mathcal{H}_1$ and sets as system parameter. $\mathcal{H}_0$ will be modeled as random oracle. $\mathcal{S}$ picks random coins ψ, ϕ for $\mathcal{S}$ and $\mathcal{A}$ respectively. Besides, $\mathcal{S}$ also picks $\{h_1, h_2, \cdots, h_p\} \xleftarrow{\$} \mathcal{D}_{\mathcal{H}_0}$ as the q_h responses of the random oracle $\mathcal{H}_0$. $\mathcal{S}$ gives random coin ϕ to $\mathcal{A}$. $\mathcal{S}$ sets $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathsf{param^{ch}}_c$ as public parameter.
Oracle Simulation. $\mathcal{S}$ simulates the oracles as follow:

- $\mathcal{RO}(\perp)$: Assume adversary $\mathcal{A}$ can only queries $\mathcal{RO}$ q_r times ($q_r \geq 1$). $\mathcal{A}$ random picks an index $\mathcal{I} \leftarrow_{\$} [1, \cdots, q_r]$. For index $\mathcal{I}$, $\mathcal{S}$ runs $\mathsf{OKeygen}(1^\lambda) \rightarrow (\mathsf{opk}_{\mathcal{I}}, \mathsf{osk}_{\mathcal{I}})$ and set $\mathsf{pk}_{\mathcal{I}} = \mathsf{hk}'_{\mathcal{I}} = \mathsf{hk}_c \oplus \mathcal{H}_1(\mathsf{opk})$. For other index, $\mathcal{S}$ generates the public key and secret key according to the **KeyGen** algorithm. Upon the jth query, $\mathcal{S}$ returns the corresponding public key.
- $\mathcal{CO}(\mathsf{pk})$: On input a public key pk returned by $\mathcal{RO}$ oracle, $\mathcal{S}$ first checks whether it corresponds to index $\mathcal{I}$. If yes, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ returns the corresponding secret key sk. According to the requirements, $\mathcal{A}$ is allowed to query this oracle no more than once.
- $\mathcal{SO}(\mu, L_{\mathsf{pk}}, \mathsf{pk}_\pi)$: When $\mathcal{A}$ queries $\mathcal{SO}$ on message μ, a list of public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$ and the public key for the signer pk_π where $\mathsf{pk}_\pi \in L_{\mathsf{pk}}$, $\mathcal{S}$ simulates $\mathcal{SO}$ as follow:
  - If $\mathsf{pk}_\pi \neq \mathsf{pk}_{\mathcal{I}}$,$\mathcal{S}$ runs **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}})$ where the output of the random oracle will be the first $h_i \in \{h_1, h_2, \cdots, h_p\}$ that has not been used yet. $\mathcal{S}$ returns the signature σ to $\mathcal{A}$;

- If $\mathsf{pk}_\pi = \mathsf{pk}_\mathcal{I}$, for $i \in [1, \cdots, \ell]$, $\mathsf{pk}_i = \mathsf{hk}'_i$, $\mathcal{S}$ computes $\mathsf{hk}_i = \mathcal{H}_1(\mathsf{opk}_\pi) \oplus \mathsf{hk}'_i$. $\mathcal{S}$ samples m_i, r_i and computes $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$. $\mathcal{S}$ then programs random oracle $\mathcal{H}_0$ as $\mathcal{H}_0(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}}) = m_1 \oplus \cdots \oplus m_\ell$. $\mathcal{S}$ also computes one-time signature $sig = \mathsf{OSign}(\mathsf{osk}_\pi;\ (m_1, r_1), \cdots, (m_\ell, r_\ell), L_{\mathsf{pk}}, \mathsf{opk})$. $\mathcal{S}$ returns signature $\sigma = \{(m_1, r_1), \cdots, (m_\ell, r_\ell), \mathsf{opk}_\pi, sig\}$.

– *Random Oracle* $\mathcal{H}_0$: For query input that has already been programmed, $\mathcal{S}$ returns the corresponding output. Otherwise, the output of the random oracle will be the first $h_i \in \{h_1, h_2, \cdots, h_p\}$ that has not been used yet. $\mathcal{S}$ will record all the queries to the random oracle in a table, in case same query is issued twice.

Output. Adversary $\mathcal{A}$ outputs two sets $\{L^{(1)}_{\mathsf{pk}},\ \mu_1,\ \sigma_1\}$ and $\{L^{(2)}_{\mathsf{pk}},\ \mu_2,\ \sigma_2\}$ where $L^{(1)}_{\mathsf{pk}}$ and $L^{(2)}_{\mathsf{pk}}$ are two lists of public keys, μ_1 and μ_2 are messages, σ_1 and σ_2 are two signatures. Also these two sets should satisfy that $\mathbf{Verification}(\mu_1, \sigma_1, L^{(1)}_{\mathsf{pk}}) = accept$ and $\mathbf{Verification}(\mu_2, \sigma_2, L^{(2)}_{\mathsf{pk}}) = accept$; $\mathcal{A}$ queried $\mathcal{CO}$ less than two times; and $\mathbf{Link}(\sigma_1, \sigma_2, \mu_1, \mu_2, L^{(1)}_{\mathsf{pk}}, L^{(2)}_{\mathsf{pk}}) = unlinked$. Since $\mathcal{A}$ is allowed query $\mathcal{CO}$ less than two times. At least one of the output signatures should be generated from the secret key that $\mathcal{A}$ does not obtain. Assume σ_j, $j \in \{1, 2\}$ is not produced by the secret key $\mathcal{A}$ obtaining. If $\mathsf{pk}_\mathcal{I} \notin L^{(j)}_{\mathsf{pk}}$, abort. Otherwise, $\mathcal{S}$ accepts σ_1 and σ_2.

The probability for $\mathsf{pk}_\mathcal{I} \in L^{(j)}_{\mathsf{pk}}$ is no less than $\frac{1}{q_r}$. In the following we use $(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$ to denote $(\mu^j, \sigma^j, L^{(j)}_{\mathsf{pk}})$. Simulator $\mathcal{S}$ then uses the set $(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$ to break the collision resistance of CH+. $\mathcal{S}$ phrases σ^* to $\{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell), \mathsf{opk}^*, sig^*\}$ and denotes $m^*_1 \oplus \cdots \oplus m^*_\ell$ by h^*. Notice that with probability $1 - \frac{1}{|\mathcal{D}_\mathcal{H}|}$, h^* will be one of the $h_i \in \{h_1, \cdots, h_p\}$ or the hash outputs from the $\mathcal{SO}$ queries. Since if the random oracle was not queried or programmed on some input, the probability for $\mathcal{A}$ to produce a $\{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell)\}$ such that $m^*_1 \oplus \cdots \oplus m^*_\ell = \mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}})$ is $\frac{1}{|\mathcal{D}_\mathcal{H}|}$. The probability for $\mathcal{A}$ to produce a forgery is δ. Thus, the probability for $\mathcal{A}$ outputs a forgery $(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$ and $h^* = \mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}})$ has been queried in $\mathcal{SO}$ or $\mathcal{RO}$ is $\delta - \frac{1}{|\mathcal{D}_\mathcal{H}|}$.

$\mathsf{Type\ 1\ forgery}$: The first type of forgery is that, for the forgery $(\mu^*, \sigma^* = \{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell), \mathsf{opk}^*, sig^*\}, L^*_{\mathsf{pk}})$, $m^*_1 \oplus \cdots \oplus m^*_\ell = \mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}})$ is a response of random oracle $\mathcal{H}$ on $\mathcal{H}(\mu', C'_1, \cdots, C'_{\ell'}, L'_{\mathsf{pk}})$ during a $\mathcal{SO}$ query. Then, we have

$$\mathcal{H}(\mu^*, C^*_1, \cdots, C^*_\ell, L^*_{\mathsf{pk}}) = \mathcal{H}(\mu', C'_1, \cdots, C'_{\ell'}, L'_{\mathsf{pk}})$$

If $\mu^* \neq \mu'$, $(C^*_1, \cdots, C^*_\ell) \neq (C'_1, \cdots, C'_{\ell'})$ or $L'_{\mathsf{pk}} \neq L^*_{\mathsf{pk}}$, we find a collision of the hash function. Thus, we must have $\mu^* = \mu'$, $(C^*_1, \cdots, C^*_\ell) = C'_1, \cdots, C'_{\ell'}$ and $L'_{\mathsf{pk}} = L^*_{\mathsf{pk}}$. Since we require that $(\mu^*,\ L^*_{\mathsf{pk}})$ has not been queried by $\mathcal{A}$ for signature. $\mathsf{Type\ 1\ forgery}$ is not a valid forgery.

$\mathsf{Type\ 2\ forgery}$: The second type of forgery is that, $h^* = m^*_1 \oplus \cdots \oplus m^*_\ell$ is a response of a $\mathcal{RO}$ query issued by $\mathcal{A}$. We store the forgery $(\mu^*,\ \sigma^* = \{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell), \mathsf{opk}^*,\ sig^*\}, L^*_{\mathsf{pk}})$. Assume $h^* = h_i$ where $h_i \in \{h_1, \cdots, h_p\}$, picks new $h'_i, \cdots, h'_p \leftarrow_\$ D_H$. $\mathcal{S}$ then run $\mathsf{Game}_{\mathsf{forge}}$ again on $(\mathsf{hk}_c, \mathsf{param^{ch}}_c, \psi, \phi, h_1, \cdots, h_{i-1}, h'_i, \cdots, h'_p)$. According to the General Forking Lemma, we obtain that $h'_i \neq h_i$ and the adversary $\mathcal{A}$ uses the random oracle response h'_i in its forgery is at least

$$\Pr = acc\left(\frac{acc}{q_s + q_h} - \frac{1}{|\mathcal{D_H}|}\right),$$

where

$$acc = \frac{1}{q_r}\left(\delta - \frac{1}{|\mathcal{D_H}|} - \frac{1}{q_r}\right)$$

Which means that with the same probability, $\mathcal{A}$ will output a forgery $\{\mu', \sigma' = \{(m'_1, r'_1), \cdots, (m_{\ell'}, r_{\ell'}), \mathsf{opk}', sig'\}, L'_{\mathsf{pk}}\}$ and $\mu^* = \mu'$, $(C^*_1, \cdots, C^*_\ell) = (C'_1, \cdots, C'_{\ell'})$, $L'_{\mathsf{pk}} = L^*_{\mathsf{pk}}$, and $\mathsf{opk}' = \mathsf{opk}^*$. Thus, $\ell = \ell'$. At least with probability $\frac{1}{\ell}$, $m^*_{\mathcal{I}} \neq m'_{\mathcal{I}}$. Since $C^*_{\mathcal{I}} = C'_{\mathcal{I}}$, $\mathcal{S}$ has $\mathsf{Hash}(\mathsf{hk}_c, m^*_{\mathcal{I}}, r^*_{\mathcal{I}}) = \mathsf{Hash}(\mathsf{hk}_c, m'_{\mathcal{I}}, r'_{\mathcal{I}})$. $(m^*_{\mathcal{I}}, r^*_{\mathcal{I}})$ and $(m'_{\mathcal{I}}, r'_{\mathcal{I}})$ is a collision for hash key hk_c.

The probability for $\mathcal{S}$ aborting during $\mathcal{SO}$ is no more than $\frac{1}{q_r}$. The probability for $\mathcal{S}$ not aborting during *output* is no less than $\frac{1}{q_r}$. Thus, the probability for $\mathcal{S}$ solving problem instance is no less than

$$(1 - \frac{1}{q_r})(\frac{1}{q_r}) \cdot \Pr$$

which is non-negligible.

Theorem 7 (Nonslanderability). *Our linkable ring signature is nonslanderable in random oracle model if the one-time signature scheme Π^{OTS} is one-time unforgeable.*

Proof. Assume there is an adversary $\mathcal{A}$ who can win $\mathsf{Game}_{\mathsf{slander}}$ with probability δ. Then we can construct a simulator $\mathcal{S}$ who can break the unforgeability of the one-time signature Π^{OTS} used in our construction also with probability δ.

$\mathcal{S}$ is given a Π^{OTS} public key opk' and is allowed to query the signature sig' of a message m' once for any message of its choosing. $\mathcal{S}$ is said breaking the unforgeability of Π^{OTS} if it can produce (m'', sig'') such that $(m'', sig'') \neq (m', sig')$ and $\mathsf{OVer}(\mathsf{opk}'; sig''; m'') = accept$. In order to use $\mathcal{A}$ to break the unforgeability of Π^{OTS}, the simulator $\mathcal{S}$ needs to simulate the challenger $\mathcal{C}$ and oracles to play $\mathsf{Game}_{\mathsf{slander}}$ with $\mathcal{A}$. $\mathcal{S}$ runs as follow:
Setup. Simulator $\mathcal{S}$ picks two hash functions $\mathcal{H}_0$, $\mathcal{H}_1$. It also generates $\mathsf{param^{ch}} \leftarrow \mathsf{SetUp}(1^\lambda)$. $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathsf{param^{ch}}$ and Π^{OTS} will be set as system parameter.$\mathcal{H}_0$ and $\mathcal{H}_1$ will be modeled as random oracles.
Oracle Simulation. $\mathcal{S}$ simulates the oracles as follow:

- $\mathcal{RO}(\perp)$: $\mathcal{S}$ uniformly samples hk' and returns hk' as the public key.
- $\mathcal{CO}(\mathsf{pk})$: On input a public key $\mathsf{pk} = \mathsf{hk}'$ returned by $\mathcal{RO}$ oracle, $\mathcal{S}$ first checks whether it is an output of $\mathcal{RO}$ query. If yes, $\mathcal{S}$ runs $\mathsf{OKeygen}(1^\lambda) \rightarrow (\mathsf{opk}, \mathsf{osk})$. $\mathcal{S}$ runs $\mathsf{TrapGen}(1^\lambda) \rightarrow (\mathsf{hk}, \mathsf{tr})$. $\mathcal{S}$ returns $(\mathsf{tr}, \mathsf{opk}, \mathsf{osk})$ as secret key and programs $\mathcal{H}_1(\mathsf{opk}) = \mathsf{hk} \oplus \mathsf{hk}'$.

- $\mathcal{SO}(\mu, L_{\mathsf{pk}}, \mathsf{pk}_\pi)$: When $\mathcal{A}$ queries $\mathcal{SO}$ on message μ, a list of public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$ and the public key for the signer pk_π where $\mathsf{pk}_\pi = \mathsf{hk}' \in L_{\mathsf{pk}}$, $\mathcal{S}$ simulates $\mathcal{SO}$ as follow:
    - If pk_π has been queried to $\mathcal{CO}$ oracle,$\mathcal{S}$ runs **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}})$ and returns the signature σ to $\mathcal{A}$;
    - If pk_π has not been queried to $\mathcal{CO}$, $\mathcal{S}$ runs $\mathsf{OKeygen}(1^\lambda) \rightarrow (\mathsf{opk}, \mathsf{osk})$. For $i \in [1, \cdots, \ell]$, $\mathsf{pk}_i = \mathsf{hk}'_i$, $\mathcal{S}$ computes $\mathsf{hk}_i = \mathsf{hk}'_i \oplus \mathcal{H}_1(\mathsf{opk})$. $\mathcal{S}$ samples m_i, r_i and computes $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$. $\mathcal{S}$ then programs random oracle $\mathcal{H}_0$ as $\mathcal{H}_0(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}}) = m_1 \oplus \cdots \oplus m_\ell$. $\mathcal{S}$ Computes one-time signature $sig = \mathsf{OSign}(\mathsf{osk}; (m_1, r_1), \cdots, (m_\ell, r_\ell), L_{\mathsf{pk}}, \mathsf{opk})$. $\mathcal{S}$ returns signature $\sigma = \{(m_1, r_1), \cdots, (m_\ell, r_\ell), \mathsf{opk}, sig\}$.
- *Random Oracle* $\mathcal{H}_0$: For input that has already been programmed, $\mathcal{S}$ returns the corresponding output. Otherwise, $\mathcal{S}$ randomly samples h_0 and outputs h_0. $\mathcal{S}$ will record all the queries to the random oracle in a table, in case same query is issued twice.
- *Random Oracle* $\mathcal{H}_1$: For input that has already been programmed, $\mathcal{S}$ returns the programmed output. Otherwise, $\mathcal{S}$ randomly samples h_1 and output h_1. $\mathcal{S}$ will record all the queries to the random oracle in a table, in case same query is issued twice.

Challenge. $\mathcal{A}$ sends a list of public keys $L'_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, message μ and public key $\mathsf{pk}_\pi \in L_{\mathsf{pk}}$. According to the requirements, pk_π should not been queried to $\mathcal{CO}$ or as an insider to $\mathcal{SO}$. Thus, there is no one-time signatures keys chosen for pk_π yet. $\mathcal{S}$ takes opk' as the one-time signature public key for pk_π. For $i \in [1, \cdots, \ell]$, $\mathsf{pk}_i = \mathsf{hk}'_i$, $\mathcal{S}$ computes $\mathsf{hk}_i = \mathsf{hk}'_i \oplus \mathcal{H}_1(\mathsf{opk}')$. $\mathcal{S}$ samples m_i,r_i and computes $C_i = \mathsf{Hash}(\mathsf{hk}_i, m_i, r_i)$. $\mathcal{S}$ then programs random oracle $\mathcal{H}_0$ as $\mathcal{H}_0(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}}) = m_1 \oplus \cdots \oplus m_\ell$. Then, $\mathcal{S}$ queries for the one-time signature sig' of message $\nu' = \{(m_1, r_1), \cdots, (m_\ell, r_\ell), L'_{\mathsf{pk}}, \mathsf{opk}')\}$. $\mathcal{S}$ returns $\sigma' = \{(m_1, r_1), \cdots, (m_\ell, r_\ell), \mathsf{opk}', sig'\}$ to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs a list of public keys L^*_{pk}, message μ^*, and a signature σ^* such that **Verification**$(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) =$ *accept*, **Link**$(\sigma, \sigma^*, \mu, \mu^*, L'_{\mathsf{pk}}, L^*_{\mathsf{pk}}) =$ *linked.*

Simulator $\mathcal{S}$ then use $(L^*_{\mathsf{pk}}, \mu^*, \sigma^*)$ to break the unforgeability of Π^{OTS}. $\mathcal{S}$ phrases $\sigma^* = \{(m_1^*, r_1^*), \cdots, (m^*_{\ell'}, r^*_{\ell'}), \mathsf{opk}^*, sig^*\}$. Since **Link**$(\sigma, \sigma^*, \mu, \mu^*, L'_{\mathsf{pk}}, L^*_{\mathsf{pk}}) =$ *linked*, we must have $\mathsf{opk}' = \mathsf{opk}^*$ and $\mathsf{OVer}(\mathsf{opk}^*; sig^*; (m_1^*, r_1^*), \cdots, (m^*_{\ell'}, r^*_{\ell'}), L^*_{\mathsf{pk}}, \mathsf{opk}^*) =$ *accept.* Since $\sigma^*, L^*_{\mathsf{pk}}$ must be different from $\sigma', L'_{\mathsf{pk}}$. $\mathcal{S}$ obtains a one-time message signature pair where message is $\nu^* = \{(m_1^*, r_1^*), \cdots, (m^*_{\ell'}, r^*_{\ell'}), L^*_{\mathsf{pk}}, \mathsf{opk}^*\} \neq \nu'$ in challenge. sig^* is a valid one-time signature for opk' and ν^*. $\mathcal{S}$ breaks the unforgeability of Π^{OTS}.

According to Theorem 2, our linkable ring signature scheme has nonsladerability and linkability. Thus it is also unforgeable.

4 Instantiation

4.1 Instantiation of CH+ from Standard Lattice

In this section, we are going to present our first instantiation of CH+ from standard lattice.

$\mathsf{SetUp}(1^\lambda) \to \mathbf{H}$: On input the security parameter 1^λ, this algorithm randomly samples a matrix $\mathbf{H} \leftarrow_\$ \mathbb{Z}_q^{n\times k}$. The matrix $\mathbf{H}$ will be an implicit input to Hash and Inv algorithm.

$\mathsf{TrapGen}(1^\lambda) \to (\mathbf{A}, \mathbf{T})$: This algorithm runs $\mathsf{GenBasis}$ $(1^n, 1^m, q) \to (\mathbf{A}, \mathbf{T})$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is a parity-check matrix and $\mathbf{T}$ is a 'good' trapdoor basis of $\Lambda^\perp(\mathbf{A})$.

$\mathsf{Hash}(\mathbf{A}, \mathbf{b}, \mathbf{r}) \to \mathbf{c}$: On input hash key $\mathbf{A}$, binary message vector $\mathbf{b} \in \{0,1\}^k$ and randomness vector $\mathbf{r} \leftarrow D_s^m$, this algorithm computes $\mathbf{c} = \mathbf{H}\mathbf{b} + \mathbf{A}\mathbf{r}$ and returns $\mathbf{c}$.

$\mathsf{Inv}(\mathbf{A}, \mathbf{T}, \mathbf{c}, \mathbf{b}') \to \mathbf{r}'$: On hash key $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and its trapdoor $\mathbf{T}$, a vector $\mathbf{c} \in \mathbb{Z}_q^n$, a binary vector $\mathbf{b}' \in \{0,1\}^k$, it computes $\mathbf{u} = \mathbf{c} - \mathbf{H}\mathbf{b}'$ and $\mathbf{r}' = \mathsf{PreSample}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s)$.

Now we argue that this instantiation satisfies our requirements of CH+.

- Our instantiation is collision resistant and one-way if $\text{SIS}_{q,n,m',\beta}$ and $\text{ISIS}_{q,n,m',\beta}$ is hard for $m' = m + k$, $\beta = \sqrt{2ms^2 + 2k}$ and $\beta = \sqrt{ms^2 + k}$ respectively.
- For the second requirement, according to Lemma 1, we have the distribution of parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ generated from $\mathsf{GenBasis}$ algorithm is within $\mathsf{negl}(n)$ far from uniform. Thus, the distribution of $\mathbf{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$. Our instantiation satisfies the second requirement.
- For the third requirement, this instantiation requires that randomness vector $\mathbf{r}$ is sampled from Gaussian distribution D_s^m. According to Lemma 1, if we set deviation s appropriately (i.e., greater than the smooth parameter of $\mathbf{T}$, see [30]), the random vector $\mathbf{r}'$ sampled by algorithm Inv is within $\mathsf{negl}(n)$ statistical distance of D_s^m. Thus our instantiation satisfies the third requirement.

4.2 Instantiation of CH+ from NTRU

The FALCON-based CH+ scheme consists of following algorithms:

$\mathsf{SetUp}(1^\lambda) \to (\mathbf{h}, D_\mathbf{b}, D_\mathbf{r})$: On input the security parameter 1^λ, this algorithm firstly sets up the polynomial ring $\mathcal{R}_q$ and samples $\mathbf{h} \leftarrow_\$ \mathcal{R}_q$. It also sets related distributions:

- $D_\mathbf{b}$: a uniform distribution over $\mathcal{R}_q$ with binary coefficients;
- $D_\mathbf{r}$: a discrete Gaussian distribution over $\mathcal{R}_q \times \mathcal{R}_q$.

$\mathsf{TrapGen}(1^\lambda) \to (\mathbf{a}, \mathbf{T})$: This algorithm takes security parameter 1^λ as input and then runs FALCON key generation function to obtain a tuple $(\mathbf{a}, \mathbf{T})$ where the public description of CH+, namely, $\mathbf{a} = \mathbf{g}/\mathbf{f}$ is computationally indistinguishable from uniform over $\mathcal{R}_q$ under NTRU assumption; $\mathbf{T} := \begin{bmatrix} \mathbf{f} & \mathbf{g} \\ \bar{\mathbf{f}} & \bar{\mathbf{g}} \end{bmatrix}$ is the trapdoor of $\mathbf{a}$.

$\mathsf{Hash}(\mathbf{a}, \mathbf{b}, \mathbf{r}) \to \mathbf{c}$: On input a hash key $\mathbf{a}$, a binary message string $\mathbf{b} \in D_{\mathbf{b}}$ and randomness $\mathbf{r} := (\mathbf{r}_0, \mathbf{r}_1) \in D_{\mathbf{r}}$, this algorithm returns a hash output $\mathbf{c} := \mathbf{r}_0 + \mathbf{a}\mathbf{r}_1 + \mathbf{h}\mathbf{b} \in \mathcal{R}_q$.

$\mathsf{Inv}(\mathbf{a}, \mathbf{T}, \mathbf{c}, \mathbf{b}') \to \mathbf{r}'$: On input hash key $\mathbf{a}$, its trapdoor $\mathbf{T}$, a value $\mathbf{c} \in D_{\mathbf{c}}$ and a binary message $\mathbf{b}'$, this algorithm

– Compute $\mathbf{u} = \mathbf{c} - \mathbf{b}'\mathbf{h}$;
– Generate a falcon signature $\mathbf{r}' := (\mathbf{r}'_0, \mathbf{r}'_1)$ on $\mathbf{u}$ such that $\mathbf{r}'_0 + \mathbf{r}'_1\mathbf{a} = \mathbf{u}$.

It returns $\mathbf{r}' \in D_{\mathbf{r}}$ such that $\mathsf{Hash}(\mathbf{a}, \mathbf{b}', \mathbf{r}') = \mathbf{c}$. The distribution of $\mathbf{r}'$ will be identical to the distribution of $\mathbf{r}$ used in Hash due to the property of GPV sampler.

This instantiation satisfies our requirements of CH+.

– The one-wayness and collision resistance of this instantiation is based on NTRU assumption, R-SIS and R-ISIS. According to NTRU assumption, $\mathbf{a}$ is computationally close to uniform. For a R-SIS$_{3,q,\beta}$ problem instance[6] $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, we can compute $\{1, \mathbf{a}', \mathbf{h}'\} = \{\frac{\mathbf{e}_1}{\mathbf{e}_1}, \frac{\mathbf{e}_2}{\mathbf{e}_1}, \frac{\mathbf{e}_3}{\mathbf{e}_1}\}$. $\mathbf{a}'$ should be indistinguishable with a real hash key $\mathbf{a}$. By obtaining a collision $\{\mathbf{r}_0^{(0)}, \mathbf{r}_1^{(0)}, \mathbf{b}^{(0)}\}$, $\{\mathbf{r}_0^{(1)}, \mathbf{r}_1^{(1)}, \mathbf{b}^{(1)}\}$ on hash key $\mathbf{a}'$ and public parameter $\mathbf{h}'$. We have

$$((\mathbf{r}_0^{(0)} - \mathbf{r}_0^{(1)}) + \mathbf{a}'(\mathbf{r}_1^{(0)} - \mathbf{r}_1^{(1)}) + \mathbf{h}(\mathbf{b}^{(0)} - \mathbf{b}^{(1)})) = 0.$$

We find a solution to the problem instance $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. We can use the similar way to argue the one-wayness of NTRU instantiation.
– Under NTRU assumption, Falcon public key is computationally indistinguishable from uniform; and the probability that a uniform sampled ring element $\bar{\mathbf{a}} \leftarrow_{\$} \mathcal{R}_q$ having a Falcon trapdoor is negligible.
– Falcon is essentially a GPV sampler over NTRU. Therefore, according to Theorem 1, if the deviation of $D_{\mathbf{r}}$ is greater than the smoothing parameter, then $\mathbf{r}'$ generated by algorithm Inv will be within $\mathsf{negl}(n)$ statistical distance of $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{a}),s}$. Thus our instantiation satisfies the third requirement.

4.3 Full description of Raptor

Now we are ready to present our instantiation. Falcon works over a polynomial ring $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^n + 1)$ for $n \in \{512, 1024\}$ and $q = 12289$. There is a third parameter set with a different, more complicated polynomial ring. For simplicity, we omit this parameter set.
Setup$(1^\lambda) \to \mathsf{param}$: On input the security parameter 1^λ, this algorithm chooses a hash function $\mathcal{H}$: $\{*\} \to \{0, 1\}^n$, a suitable $\mathcal{R}$ and distributions $D_{\mathbf{b}}, D_{\mathbf{r}}$ for the

[6] We require at least one of the three elements is invertible over $\mathcal{R}_q$. For Falcon-512, the probability is $(1 - 1/q)^N \approx 96\%$.

security level, where $D_{\mathbf{b}} := \{0,1\}^{256}$, $D_{\mathbf{r}} := \mathcal{D}^2_{\mathcal{R},\eta}$, $\mathcal{D}$ is a discrete Gaussian distribution over $\mathcal{R}$ with deviation η, and $\eta \approx 1.17\sqrt{q}$ is the smooth parameter. It also picks a public polynomial $\mathbf{h} \leftarrow_{\$} \mathcal{R}_q$ at random as $\mathsf{param}^{\mathsf{ch}}$.
KeyGen$\rightarrow$ $(\mathsf{sk}, \mathsf{pk})$: This algorithm firstly generates $(\mathbf{a}, \mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}) \leftarrow$ Falcon.KeyGen (param) where

1. $\mathbf{a} = \mathbf{g}/\mathbf{f} \in \mathcal{R}_q$,
2. $\mathbf{f} \times \bar{\mathbf{g}} - \mathbf{g} \times \bar{\mathbf{f}} = q$,
3. $\|(\mathbf{f}, \mathbf{g})\|$ and $\|(\bar{\mathbf{f}}, \bar{\mathbf{g}})\|$ are small.

Then it sets public key $\mathsf{pk} = \{\mathbf{a}\}$ and secret key $\mathsf{sk} = \{\mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}\}$.
Signing$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}, \mathsf{param}) \rightarrow \sigma$: On input message μ, list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and signing key $\mathsf{sk}_\pi = \{\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi\}$ of $\mathsf{pk}_\pi = \{\mathbf{a}_\pi\}$, and the system parameter param, the signing algorithm runs as follow:

1. For $i \in [1, \cdots, \ell]$ and $i \neq \pi$, picks $\mathbf{b}_i \leftarrow_{\$} \{0,1\}^{256}$ and $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}) \leftarrow \mathcal{D}^2_{\mathcal{R},\eta}$. Compute $\mathbf{c}_i = \mathbf{r}_{i,0} + \mathbf{a}_i \mathbf{r}_{i,1} + \mathbf{h}\mathbf{b}_i$.
2. For $i = \pi$, pick $\mathbf{c}_\pi \leftarrow_{\$} \mathcal{R}_q$.
3. Compute $\mathbf{b}_\pi$ such that
$$\mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_\pi \oplus \cdots \oplus \mathbf{b}_\ell = \mathcal{H}(\mu, \mathbf{c}_1, \cdots, \mathbf{c}_\ell).$$
4. Set $\mathbf{u}_\pi = \mathbf{c}_\pi - \mathbf{h}\mathbf{b}_\pi$.
5. Compute $(\mathbf{r}_{\pi,0}, \mathbf{r}_{\pi,1}) =$ Falcon.sign$(\mathsf{sk}_\pi; \mathbf{u}_\pi)$ such that $\mathbf{r}_{\pi,0} + \mathbf{r}_{\pi,1}\mathbf{a}_\pi = \mathbf{u}_\pi$.

The ring signature of μ and L_{pk} is $\sigma = \{(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i)\}_{i=1}^{\ell}$.
Verification$(\mu, \sigma, L_{\mathsf{pk}}) \rightarrow$ *accept/reject*: On input message μ, signature σ and a list of user public keys L_{pk}, the verification algorithm performs as follows:

1. phrases $\sigma = \{(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i)\}_{i=1}^{\ell}$;
2. checks whether each tuple of $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{d}_i) \in \mathcal{D} \times \mathcal{D} \times D_{\mathbf{b}}$; outputs *reject* if not.
3. computes $\mathbf{c}_i = \mathbf{r}_{i,0} + \mathbf{a}_i \mathbf{r}_{i,1} + \mathbf{h}\mathbf{b}_i$ for all $i \in [1, \cdots, \ell]$ and checks whether $\mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_\ell = \mathcal{H}(\mu, \mathbf{c}_1, \cdots, \mathbf{c}_\ell)$; outputs *reject* if not.
4. outputs *accept*.

4.4 Full description of the linkable Raptor

As shown in Section 3.3, one can convert Raptor into a one-time linkable one with a one-time signature scheme. For easiness of implementation, we will use also use Falcon to instantiate this signature scheme.
Setup$(1^\lambda) \rightarrow \mathsf{param}$: On input the security parameter 1^λ, this algorithm chooses $\mathcal{H}_0$, $\mathcal{H}_1$, $D_{\mathbf{b}}$, $D_{\mathbf{r}}$ and η as in Raptor.
KeyGen$\rightarrow$ $(\mathsf{sk}, \mathsf{pk})$: This algorithm firstly generates

– $(\mathbf{a}, \mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}) \leftarrow$ Falcon.KeyGen(param), and
– $(\mathbf{a}_{ots}, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}) \leftarrow$ Falcon.KeyGen(param)

Then it sets $\mathbf{a}' := \mathbf{a} + \mathcal{H}_1(\mathbf{a}_{ots}) \bmod q$. The public key $\mathsf{pk} = \{\mathbf{a}', \mathbf{a}_{ots}\}$ and secret key $\mathsf{sk} = \{\mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}\}$.

Signing$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}, \mathsf{param}) \to \sigma$: On input message μ, list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and signing key $\mathsf{sk}_\pi = \{\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}\}$ of $\mathsf{pk}_\pi = \{\mathbf{a}'_\pi, \mathbf{a}_{ots}\}$, and the system parameter param, the signing algorithm runs as follow:

1. For $i \in [1, \cdots, \ell]$, compute $\mathbf{a}_i = \mathbf{a}'_i - \mathcal{H}_1(\mathbf{a}_{ots}) \bmod q$.
2. For $i \in [1, \cdots, \ell]$ and $i \neq \pi$, picks $\mathbf{b}_i \leftarrow_\$ \{0,1\}^{256}$ and $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}) \leftarrow \mathcal{D}^2_{\mathcal{R},\eta}$. Compute $\mathbf{c}_i = \mathbf{r}_{i,0} + \mathbf{a}_i \mathbf{r}_{i,1} + \mathbf{h}_i \mathbf{b}_i$.
3. For $i = \pi$, pick $\mathbf{c}_\pi \leftarrow_\$ \mathcal{R}_q$.
4. Compute $\mathbf{b}_\pi$ such that
$$\mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_\pi \oplus \cdots \oplus \mathbf{b}_\ell = \mathcal{H}(\mu, \mathbf{c}_1, \cdots, \mathbf{c}_\ell).$$
5. Set $\mathbf{u}_\pi = \mathbf{c}_\pi - \mathbf{h}\mathbf{b}_\pi$.
6. Set $(\mathbf{r}_{\pi,0}, \mathbf{r}_{\pi,1}) = \text{FALCON}.\mathsf{sign}((\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi); \mathbf{u}_\pi)$ such that $\mathbf{r}_{\pi,0} + \mathbf{r}_{\pi,1}\mathbf{a}_\pi = \mathbf{u}_\pi$.
7. Compute $sig := \text{FALCON}.\mathsf{sign}\ ((\mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}); (\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}_{i=1}^{\ell}, \{\mathbf{a}'_i\}_{i=1}^{\ell}, \mathbf{a}_{ots}))$.

The ring signature of μ and L_{pk} is $\sigma = \{\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}_{i=1}^{\ell}, \mathbf{a}_{ots}, sig\}$.

Verification$(\mu, \sigma, L_{\mathsf{pk}}) \to$ *accept/reject*: On input message μ, signature σ and a list of user public keys L_{pk}, the verification algorithm performs as follows:

1. phrases $\sigma = \{\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}_{i=1}^{\ell}, \mathbf{a}_{ots}, sig\}$;
2. For $i \in [1, \cdots, \ell]$, compute $\mathbf{a}_i = \mathbf{a}'_i - \mathcal{H}_1(\mathbf{a}_{ots}) \bmod q$;
3. checks whether each tuple of $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{d}_i) \in \mathcal{D} \times \mathcal{D} \times D_{\mathbf{b}}$; outputs *reject* if not.
4. computes $\mathbf{c}_i = \mathbf{r}_{i,0} + \mathbf{a}_i \mathbf{r}_{i,1} + \mathbf{h}_i \mathbf{b}_i$ for all $i \in [1, \cdots, \ell]$ and checks whether $\mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_\ell = \mathcal{H}(\mu, \mathbf{c}_1, \cdots, \mathbf{c}_\ell)$; outputs *reject* if not.
5. verify sig is a signature for $(\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}_{i=1}^{\ell}, \{\mathbf{a}'_i\}_{i=1}^{\ell}, \mathbf{a}_{ots})$ with public key $\mathbf{a}_{ots}$; outputs *reject* if fails.
6. outputs *accept*.

Note that in this implementation we use additions and subtractions over the $\mathcal{R}_q$ instead of bit-wise XOR operations. Under the random oracle model $\mathcal{H}_1(\mathbf{a}_{ots})$ will output a random ring element. This creates a perfect one-time mask that assures $\mathbf{a}'$ is indistinguishable from random.

5 Concrete Parameters

5.1 Parameters and implementation

Here we give some parameter figures for RAPTOR-512, instantiated with FALCON-512. Our RAPTOR-512 uses a signature size of $(617 \times 2 + 32)\ell \approx 1.26\ell$ kilo bytes, where ℓ is the number of users in a signature. This is because, for each tuple $\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}$ within a ring signature, we need a pair of $\mathbf{r}_{i,0}$ and $\mathbf{r}_{i,1}$, each of 617 bytes, and an additional 32 bytes for $\mathbf{b}_i$ to avoid any search attacks [32]. This parameter set yields 114 bits security against classical attackers, and 103 bits security against quantum attackers, under the BKZ2.0 framework [20] with (quantum) sieving algorithm [4, 37].

As for linkable Raptor-512, we need an additional Falcon public key and signature which is of size $897 + 617 \approx 1.5$ kilo bytes. This accounts for a total of $(1.3\ell + 1.5)$ kilo bytes.

For conservative purpose, one may also choose Falcon-1024 for better security, which results in a signature size of 2.5ℓ kilo bytes for Raptor-1024, and $(2.5\ell + 3)$ kilo bytes for linkable Raptor-1024. The security level for both schemes will be over 256 bits.

We implemented Raptor-512 on a typical laptop with an Intel 6600U processor. The performance is shown in Tables 1(a) and 1(b). Our source code is available at [6][7]. This is a proof-of-concept implementation. We did not take into account potential optimizations such as NTT-based ring multiplication and AVX-2 instructions. We leave those to future work.

5.2 Known attacks of Raptor

The NTRU assumption and the security of Falcon signature has been extensively studied in the literature [34, 20, 3, 23, 24, 28]. Here we consider the hardness of inverting the CH+. We note that the attack described here does not work for the Falcon parameters. Indeed, this attack is strictly less efficient than forging a Falcon signature, or recovering the secret keys directly.

Our CH+ is defined as $\mathbf{c} = \mathbf{r}_0 + \mathbf{a}\mathbf{r}_1 + \mathbf{h}\mathbf{b} \bmod q$. Therefore, one may build a lattice with basis $\begin{bmatrix} q\mathbf{I} & & \\ \mathbf{a} & \mathbf{I} & \\ \frac{1}{\alpha}\mathbf{h} & 0 & \frac{1}{\alpha}\mathbf{I} \end{bmatrix}$ where the vector $(\mathbf{r}_0, \mathbf{r}_1, \alpha\mathbf{b})$ is a close vector to $(\mathbf{c}, 0, 0)$; α is a scaling factor of roughly $\sim \eta$. Note that solving the CVP here is not equivalent to finding a pre-image. Our $\mathbf{b}$ is a binary vector, therefore, to have a successful forgery we will also require the third part of the output to be in the form of α multiplying a binary vector.

It is easy to see that, even if we relax above the requirement, solving this CVP is still harder than forging a Falcon signature, i.e., solving some CVP for $\begin{bmatrix} q\mathbf{I} & \\ \mathbf{a} & \mathbf{I} \end{bmatrix}$ where the root Hermite factor is a lot larger than that of attacks on the CH+ scheme.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on*

[7] Link omitted due to anonymous submission. Will be provided upon PC chair's request.

the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings, pages 415–432, 2002.
2. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on theory of computing*, STOC 1996, pages 99–108. ACM, 1996.
3. M. R. Albrecht, B. R. Curtis, A. Deo, A. Davidson, R. Player, E. W. Postlethwaite, F. Virdia, and T. Wunderer. Estimate all the LWE, NTRU schemes! Cryptology ePrint Archive, Report 2018/331, 2018. `https://eprint.iacr.org/2018/331`.
4. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343, 2016.
5. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 75–86, 2009.
6. Anonymous. Raptor source code. online. available from TBD.
7. M. H. Au, S. S. M. Chow, W. Susilo, and P. P. Tsang. Short linkable ring signatures revisited. In *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006, Proceedings*, pages 101–115, 2006.
8. M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Certificate based (linkable) ring signature. In *Information Security Practice and Experience, Third International Conference, ISPEC 2007, Hong Kong, China, May 7-9, 2007, Proceedings*, pages 79–92, 2007.
9. M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theor. Comput. Sci.*, 469:1–14, 2013.
10. M. H. Au, W. Susilo, and S. Yiu. Event-oriented k-times revocable-iff-linked group signatures. In *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, pages 223–234, 2006.
11. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–636, 1993.
12. C. Baum, H. Lin, , and S. Oechsner. Towards practical lattice-based one-time linkable ring signatures. Cryptology ePrint Archive, Report 2018/107, 2018. `https://eprint.iacr.org/2018/107`.
13. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 60–79, 2006.
14. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 41–69, 2011.
15. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. Short accountable ring signatures based on DDH. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, pages 243–265, 2015.
16. Z. Brakerski and Y. T. Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive*, 2010:86, 2010.

17. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 410–424, 1997.
18. N. Chandran, J. Groth, and A. Sahai. Ring signatures of sub-linear size without random oracles. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 423–434, 2007.
19. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, pages 257–265, 1991.
20. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.
21. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 609–626, 2004.
22. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 40–56, 2013.
23. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013.
24. L. Ducas, V. Lyubashevsky, and T. Prest. Efficient identity-based encryption over NTRU lattices. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 22–41, 2014.
25. L. Ducas and T. Prest. Fast fourier orthogonalization. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*, pages 191–198, 2016.
26. C. Dwork and M. Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 283–293, 2000.
27. M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. Cryptology ePrint Archive, Report 2018/773, 2018. `https://eprint.iacr.org/2018/773`.
28. P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU.
29. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206, 2008.
30. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, page 197206, New York, NY, USA, 2008. ACM.
31. J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual Inter-*

national Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II, pages 253–280, 2015.

32. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.
33. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN: digital signatures using the NTRU lattice. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, pages 122–140, 2003.
34. J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte, and Z. Zhang. Choosing parameters for ntruencrypt. *IACR Cryptology ePrint Archive*, 2015:708, 2015.
35. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 267–288, 1998.
36. H. Krawczyk and T. Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*, 2000.
37. T. Laarhoven and A. Mariano. Progressive lattice sieving. In *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, pages 292–311, 2018.
38. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 1–31, 2016.
39. J. K. Liu, M. H. Au, W. Susilo, and J. Zhou. Linkable ring signature with unconditional anonymity. *IEEE Trans. Knowl. Data Eng.*, 26(1):157–165, 2014.
40. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, pages 325–335, 2004.
41. A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234, 2012.
42. V. Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 738–755, 2012.
43. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 144–155, 2006.
44. C. A. Melchor, S. Bettaieb, X. Boyen, L. Fousse, and P. Gaborit. Adapting lyubashevsky's signature schemes to the ring signature setting. In *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings*, pages 1–25, 2013.

45. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.
46. National Institute of Standards and Technology. Post-Quantum Cryptography Standardization, 2017.
47. L. Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 275–292, 2005.
48. C. Peikert. An efficient and parallel gaussian sampler for lattices. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 80–97, 2010.
49. C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 145–166, 2006.
50. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 552–565, 2001.
51. P. W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, page 289, 1994.
52. D. Stehlé and R. Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 27–47, 2011.
53. S. Sun, M. H. Au, J. K. Liu, and T. H. Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, pages 456–474, 2017.
54. W. A. Torres, R. Steinfeld, A. Sakzad, J. K. Liu, V. Kuchta, N. Bhattacharjee, M. H. Au, and J. Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0). Cryptology ePrint Archive, Report 2018/379, 2018. `https://eprint.iacr.org/2018/379`.
55. P. P. Tsang, M. H. Au, J. K. Liu, W. Susilo, and D. S. Wong. A suite of non-pairing id-based threshold ring signature schemes with different levels of anonymity (extended abstract). In *Provable Security - 4th International Conference, ProvSec 2010, Malacca, Malaysia, October 13-15, 2010. Proceedings*, pages 166–183, 2010.
56. P. P. Tsang and V. K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *Information Security Practice and Experience, First International Conference, ISPEC 2005, Singapore, April 11-14, 2005, Proceedings*, pages 48–60, 2005.