CONTRIBUTIONS TO POST-QUANTUM CRYPTOGRAPHIC TECHNIQUES

XINGYE LU

PhD

The Hong Kong Polytechnic University
2020

The Hong Kong Polytechnic University

Department of Computing

Contributions to Post-Quantum Cryptographic Techniques

Xingye Lu

A thesis submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

October 2019

CERTIFICATE OF ORIGINALITY

____________________ (Signed)

Xingye Lu (Name of student)

Abstract

In recent decades, public-key cryptographic techniques have been widely employed to secure digital communication infrastructure. Their security mainly relies on mathematical assumptions that integer factorization problem and discrete logarithm problem are hard. In 1994, Shor's algorithm was proposed. It allows people to solve integer factorization and discrete log problem using a quantum computer efficiently. As such, a quantum computer will make the deployed public-key cryptographic techniques insecure. Facing the rapid development of quantum computing, post-quantum cryptography, the study of cryptographic techniques resilient to attacks from the quantum computer is now receiving growing attention. Upgrading the modern public-key cryptography infrastructure as soon as possible become the best approach to protect individual's privacy under the threats from quantum computers. In this thesis, we focus on designing practical and efficient cryptosystems that are post-quantum secure based on lattice-based and hash-based cryptography, two common approaches in the post-quantum cryptography. Specifically, this thesis develops quantum-safe solutions for three cryptographic primitives.

Firstly, we propose a hash-based ad hoc anonymous identification scheme. Such a scheme allows a participant to identify himself as a member of a group of users in a way that his actual identity is not revealed. We demonstrate a highly efficient construction in the symmetric-key setting based on the idea of program obfuscation. The salient feature of our scheme is that only hash evaluations are needed. Consequently, our scheme outperforms all previous constructions for a reasonably large ad hoc group size (of around 50000 users) since no exponentiation nor pairing operation is involved.

We also present a new signature scheme, PASS$_G$, relying on lattice-based cryptography. It is based on signatures from the partial Fourier recovery problem PASS$_{RS}$ introduced by Hoffstein et al. in 2014. Same as PASS$_{RS}$, security of our construction is based on the hardness of a special kind of Short Integer Solution problem and the hardness of partial Fourier recovery problem. PASS$_G$ improves PASS$_{RS}$ in two aspects. Firstly, it comes with a reduction proof and is thus provably secure. Secondly, we adopt rejection sampling technique introduced by Lyubashevsky in 2012 to reduce the signature size and improve the efficiency. We also present another security parameter set based on best known attack using BKZ 2.0 algorithm introduced by Chen and Nguyen in 2011.

Furthermore, we design two new lattice-based (linkable) ring signature schemes. The first scheme, Raptor, is the first practical construction of its kind that comes with an implementation. We develop a generic construction of (linkable) ring signatures based on the well-known generic construction from Rivest et al., which is not fully compatible with lattices. We give instantiations from both standard lattice, as a proof of concept, and NTRU lattice, as an efficient instantiation. We show that the latter construction, Raptor, is almost as efficient as the classical RST ring signatures and thus may be of practical interest. We also revisit the generic framework of ring signatures based on hash-then-one-way type (Type-H) signatures presented by Abe et al. (AOS) in 2004 and made the following contributions. First, we give a proof for the generic construction, in a strengthened security model. Secondly, we also extend AOS's framework to generically construct one-time linkable ring signatures from Type-H signatures and one-time signatures. Lastly, we instantiate the generic construction with an NTRU-based Type-H signature: FALCON and obtain a post-quantum linkable ring signature scheme. Our analysis suggests that the resulting linkable signature outperforms Raptor.

Publications Arising from the Thesis

1. **Xingye Lu**, Man Ho Au, and Zhenfei Zhang. (Linkable) Ring Signature from Hash-Then-One-Way Signature. In *Proceedings of 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom 2019)*, Rotorua, New Zealand, August 5-8, 2019.

2. **Xingye Lu**, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In *Proceedings of Applied Cryptography and Network Security 17th International Conference (ACNS 2019)*, Bogota, Colombia, June 5-7, 2019, volume 11464 of Lecture Notes in Computer Science, pages 110–130.

3. **Xingye Lu**, Zhenfei Zhang, and Man Ho Au. Practical signatures from the partial fourier recovery problem revisited: A provably-secure and gaussian-distributed construction. In *Proceedings of Information Security and Privacy - 23rd Australasian Conference (ACISP 2018)*, Wollongong, NSW, Australia, July 11-13, 2018, volume 10946 of Lecture Notes in Computer Science, pages 813–820.

4. **Xingye Lu** and Man Ho Au. Anonymous identification for ad hoc group. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (AsiaCCS 2016)*, Xi'an, China, May 30 - June 3, 2016, pages 583–591.

Acknowledgements

First and foremost, I am extremely grateful to my parents for their unconditional support and encouragement. In the past 27 years, they always love me, guide me, protect me and inspire me with all their heart. I am forever indebted to my parents for giving me the experiences that have made me who I am.

I would like to sincerely thank my supervisor, Prof. Man Ho Au for his guidance, support and patience thorough out my PhD study. Without him, this thesis could not become a reality. He always supports me to do the research with my own interest and is willing to discuss every research questions with me in detail. I have benefited a lot from his expertise, vast knowledge and skill in the research area.

I am also deeply grateful to all of my friends and nice people who have willingly help me out with their abilities.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent decades, cryptographic techniques have been widely implemented in global communication digital infrastructure to protect people's security and personal privacy in many ways.

These days, most of the public-key infrastructures leveraged in Internet communication such as digital signature schemes, encryption schemes, and key exchange schemes are constructed from RSA cryptosystem, elliptic curve cryptosystems and Diffie-Hellman key exchange whose security are mainly based on the number theoretic problems such as integer factorization and discrete log problem. In 1994, Shor's algorithms [89], which are quantum algorithms for factoring large number and solving discrete log problem, were proposed. Since Shor's quantum algorithms can be used to solve integer factorization and discrete log problem, it implies that cryptosystem based on RSA and the hardness of discrete log problem can be easily broken in polynomial time by quantum attacks. Ever since the discovery of Shor, the research on quantum algorithm has been developed significantly. Moreover, the quantum computational operations have already been executed on a small number of quantum bits. Facing the rapid development of quantum computing in both practical and theoretical, while a quantum computer with enormous processing power is still in its infancy, it is already a potential threat that cannot be ignored towards the classical cryptography. Once a large quantum computer being built in the near future, it can recover our today's secret by harvest-then-decrypt attack.

To deal with the threats from quantum computers, post-quantum cryptography was proposed. Different from quantum cryptography which uses quantum mechanical properties of the matter for cryptographic application such as secure key distribution, post-quantum cryptography is a cryptography that is resilient to attacks from quantum computers. Currently, post-quantum cryptography mainly has following five categories.

1. **Lattice-based cryptography**: Lattice-based cryptography, which allows people to build families of cryptographic functions in which breaking a random instance from the family is as hard as solving worst-case instance of lattice problem, is currently an important candidate for post-quantum cryptography. The seminal work of Ajtai [2] shows the reduction from an average-case lattice problem, Short Integer Solution (SIS), to the worst-case lattice problem. He also presented the construction of a one-way function whose security is based on SIS. Ever since, numerous constructions of other lattice-based cryptographic primitives such as collision resistant hash functions, identification schemes, encryption schemes and digital signature schemes have been proposed. Constructions from lattice-based cryptography always have the property that easy to be implemented in hardware or software. However, they also suffer from the issue of large user keys.

2. **Code-based cryptography**: Code-based cryptography is one of the promising candidates for post-quantum cryptography. The security of code-based cryptosystem is based on the syndrome-decoding problem which has been proven NP-complete. It puts the adversary in the condition of decoding a randomlike code. Code-based cryptosystems are generally easy to implement and fast. However, their key sizes

are very large and reducing key size may lead to successful attacks on some of the code-based cryptosystems.

3. **Hash-based cryptography**: Hash-based cryptosystems are cryptosystems constructed using hash functions. Hash-based cryptography is mainly concerned with the development of digital signature schemes. Hash-based signatures include Lamport signature scheme [62], Merkle signature scheme [76], XMSS[23] and [17]. Since their security only relies on the collision resistance of the underlying hash functions, so the construction of a secure hash-based scheme is independent of hard number theoretic problems and the security requirements are minimal.

4. **Multivariate polynomial cryptography**: Multivariate polynomial cryptosystems are based on the difficulty of solving Multivariate Quadratic polynomial equations and the Isomorphism of Polynomials problem which have been proven to be NP-hard or NP-complete. Multivariate polynomial cryptography is better as an approach to construct signature schemes because of the short signature size.

5. **Isogeny-based cryptography**: Isogeny-based cryptography, the most recent field in post-quantum cryptography, is originated from elliptic curve cryptography which is a significant part of classical cryptography. The underlying hard problem of isogeny-based cryptosystem states that it is hard to calculate an isogeny between two given elliptic curves . The hard problem itself along with the possible attacks have already been well studied. The main advantages of isogeny-based cryptosystems are their small key sizes and high compatibility with the current elliptic curve primitives.

In 2017, the National Institute of Standards and Technology (NIST) called for post-quantum cryptography standardization. At the end of year 2017, around 23 signature schemes and 59 public-key encryption and key-establishment algorithms were submitted in the first round. In 2019, NIST announced the 2-round candidates which includes 17 public-key encryption and key-establishment algorithms and 9 signature schemes. Among all these 2-round candidates, 12 of which are constructed from lattice-based cryptography covering both PKE/KEM and signature scheme. In the remaining 14 candidates, 8 PKE/KEM schemes are constructed from code-based and isogeny-based cryptography, 6 signature schemes are constructed from hash-based, multivariate polynomial cryptography and post-quantum zero-knowledge proof system.

Digital signature scheme is one of the most widely used cryptographic infrastructures. Besides its fundamental functionality of verifying authenticity of digital message, it can be used to construct protocols such as identification protocol by asking a prover to sign a message and pass the signature to verifier. With growing concerns to user privacy, anonymous identification scheme which allows provers to identify themselves without revealing true identity is attracting increasing attention. Similar to identification scheme constructed from digital signature, anonymous identification scheme can be constructed from a variant of signature scheme, which is called (linkable) ring signature scheme. In this thesis, we focus on designing efficient post-quantum anonymous identification scheme, signature and (linkable) ring signature scheme.

1.1 Thesis Outline

The rest of this thesis is organized as follow:

- Chapter 2 provides some preliminary knowledge for the following content including some notations and definitions;
- Chapter 3 presents the construction of a symmetric-key based lightweight anonymous identification scheme for ad hoc group;
- Chapter 4 gives the construction of a new lattice-based digital signature scheme

which is called PASS_G;

- In Chapter 5, Section 5.1 presents a new generic constructions for (linkable) ring signature scheme along with its standard lattice and NTRU lattice instantiations. Section 5.2 gives a rigorous security proof for an existing generic ring signature framework and extends it to its linkable version. It then instantiates the latter construction to NTRU lattice.

1.2 Anonymous Identification Scheme

Leakage of personal information online is now regarded as a serious issue for people who involved in Internet activities. The discoursed individual data contains not only sensitive identity information but also user's online behaviour which may be collected and analyzed by websites. Anonymous identification scheme which allows a user proving his/her identity in an anonymous way can be applied in scenarios where user privacy is highly concerned.

Identification scheme, a method allowing a prover to prove his/her identity to a verifier, can be applied in various situations. However, with the growing of privacy awareness, a secure identification scheme may not satisfy the privacy demands. In some situations, people simply prefer to authenticated themselves without revealing his/her identity to protect privacy. In other words, anonymous identification is desired on some occasions. For example, when students participate in an online evaluation for their teachers or professors, they would prefer logging into the evaluation system anonymously. Recently, the ABC4trust project[1] has launched a pilot to employ privacy-preserving technologies to allow eligible students to conduct online evaluations without revealing their identities. Achieving a mean of 3.373 and a standard deviation of 1.03 on a 5-point Lickert scale, it

[1] `https://abc4trust.eu`

was concluded that most participants found the system useful for protecting their privacy in the pilot assessment [90].

In the first work of this thesis, we propose an anonymous identification scheme for ad hoc group. Our anonymous identification scheme is symmetric-key based and thus can be easily applied to most of the website with password-based authentication. We also give rigorous proof to demonstrate that our identification scheme satisfies security requirements. The main technique we adopted in our construction is the program obfuscation. An *obfuscator* $\mathcal{O}$, informally, can be considered as a probabilistic 'compiler' which can transform a program into a new program without revealing any secret in the original program. Thus, in the construction of our anonymous identification scheme, we first have a function which takes a list of user password and a string as input, and output a predetermined string β or 0 if the string is in the password list or not respectively. Then we use obfuscator to obfuscate this function and obtain a new function which hides β properly. The new function will be sent to the user trying to authenticate him/herself and the user should return a string β'. The authentication is considered as a success if $\beta' == \beta$. The property of the obfuscation here guarantees that no user without a legitimate password can successfully identifying him/herself. Also, if the verifier in the protocol is an honest-but-curious verifier, then the identification scheme is anonymous.

The main building block in our anonymous identification scheme is hash function. As mentioned previously, the security of hash functions is independent from hard number theoretic problems and thus hash-based cryptographic constructions can be post-quantum. Hence, our anonymous identification is also post-quantum. Even though the running time of our identification scheme is linear to the group size. There is no expensive mathematical operation, for example, pairing operation, involved in our construction. Our construction is still very efficient for a reasonably small group.

1.2.1 Related Work

Anonymous identification scheme which allows participants in a user group to prove their membership without revealing any information about the participants' identity has been successfully implemented in several approaches. The notion of anonymous identification was first proposed in [30]. In previous works, anonymous identification schemes were mainly constructed by group signature or ring signature schemes. Group signatures, whose concept was first introduced in [26], provide users approaches to anonymously sign a message on behalf of some group has been used to implement anonymous identification schemes (for examples [24, 86, 63, 14]). The group formation is performed by a trusted entity known as the group manager who is responsible for registering users into the group. A group signature attests the fact that one of the registered users endorsed the message being signed. Based on group signatures, Boneh and Franklin in [19] first proposed an anonymous identification scheme allowing identity escrow and subset queries.

The notion of ring signatures was formalized in [85] and further studied in [1, 30, 79, 28, 87, 25, 65]. Similar to group signatures, a ring signature allows a signer to endorse a message on behalf of a group of potential signers. Unlike group signatures, however, the formation of the group of the potential signer in a ring signature is spontaneous, meaning that users could be completely unaware of being conscripted into the group. Furthermore, ring signatures support full anonymity in the sense that there is no way to revoke the anonymity and reveal the identity of the signer. Bresson, Stern and Szydlo presented extensions to ring signatures [22]. Ring signatures can be used as an anonymous identification in a typical challenge-response protocol where the verifier challenge the prover to sign a random message. The first constant-size hoc anonymous identification was introduced by Dodis, Kiayias, Nicolosi and Shoup [30]. The scheme, whose security is rest on the strong RSA assumption, is based on the notion of accumulators with one-way domain. The verification cost is time independent of the size of the ad hoc group while the prover's cost is close to constant if the group does not change rapidly. Nguyen [79] proposed a dynamic accumulator scheme for bilinear pairings to construct identity-based ring signatures[2].

To this end, we re-visit the course evaluation scenario above. We observe that several security requirements are desired. Firstly, anonymity is desirable, since the students would be afraid of retribution from the teachers of the course being evaluated. Secondly, security, meaning that only students of the course are eligible to provide feedback, is necessary. Finally, the ability to support ad hoc group identification is needed as students may enroll or withdraw from different subjects throughout the semesters. Ad hoc anonymous identification based on ring signature schemes described above would fulfill these three requirements. Having said that, we believe that there is no need to use a scheme as powerful as a ring signature scheme, which allows a signer to convince any verifier that he is the owner of a public key listed in a group of public keys associated with the ring signature. In the evaluation scenario mentioned above, we observe that there is only a single verifier who can anonymous identifies enrolled students of the course being evaluated. Requiring all students to have their own public/private key pairs, and to have the evaluation system to verify each of these keys might be too expensive. Based on this simplified scenario, the research problem we tackled in Chapter 3 is to find a new and more efficient approach to achieving anonymous identification in ad hoc group that can be applied to scenarios like system login.

1.3 Lattice-Based Signature Scheme

Digital signature scheme, as a broadly used cryptographic scheme, allows individuals to verify the authenticity of a document and guarantee the file not being altered during transit

[2] A flaw of this construction was identified and rectified in [97].

through the Internet. Currently, most of the digital signature schemes applied in practice are based on the hard number theoretic problems, for instance, integer factorization and discrete logarithm problems. As mentioned previously, there are already quantum algorithms developed to solve these problems. Even though there is still no quantum computer can be used in practice, we, however, should prepare in advance.

In the second work of this thesis, we improve a lattice-based signature scheme proposed by Hoffstein et al. in 2014 [53] to obtain a new digital signature scheme with detailed security proof, smaller signature size and better time efficiency. The security of our new digital signature scheme is based on a variant of Short Integer Solution (SIS) problem which we defined as the Vandermonde-SIS problem. To reduce the signature size, we change the distribution of the signature from uniform in [53] to Gaussian and adopting the rejection sampling technique to decouple the signature from signer's secret key. The Gaussian distribution in our signature scheme also guarantees a smaller rejection rate and higher time efficiency comparing with the original work. Besides reduction proof, we also utilize the quantum sieving algorithm to analyze the security of our signature scheme against quantum attackers. For the parameter sets we suggested, our signature scheme is secure against quantum attacks.

1.3.1 Related Work

In 1999, Hoffstein, Lieman and Silverman [52] introduced Polynomial Authentication and Signature Scheme (PASS) based on the hardness of recovering a constrained polynomial from a small number of evaluations of this polynomial. A later version of PASS, called PASS$_2$, was proposed by Hoffstein and Silverman [58]. Compared with PASS, PASS$_2$ reduces the computation and communication costs. Similar to all other "ancient" lattice based signatures such as GGHSign [45] and NTRUSign [51], PASS$_2$ suffers from a kind of transcript attacks known as *learn a parallelepiped* [80, 35], which essentially exploits the fact that the distribution of signatures leaks information of signing key. To decouple the signature from the signing key, Hoffstein et al. [53] adopted the celebrated *rejection sampling* technique from [71] and presented a revised version of PASS$_2$, which is called PASS$_{RS}$. As a candidate of practical post-quantum signature schemes, the security of PASS$_{RS}$ is based on a special hard problem known as partial Fourier recovery. This problem requires recovery of a ring element with small norm given an incomplete description of its Chinese remainder representation. Despite there is no known reduction from lattice-based hard problems to the partial Fourier recovery problem, [53] shows that this problem and the Short Integer Solution (SIS) problem are related in some kind. By assuming the average-case hardness of a special SIS problem which is called Vandermonde-SIS, the security of PASS$_{RS}$ is said to rely on the hardness of Vandermonde-SIS. However, no security reduction between PASS$_{RS}$ and Vandermonde-SIS is provided in [53].

In terms of practicality, PASS$_{RS}$ is efficient in that it enables fast signing and verification operations through number theoretic transform (NTT). A major obstacle from deployment remains the size of the signatures, compared to other lattice-based candidates [31, 40, 33, 57]. In Chapter 4, we present PASS$_G$, an efficient lattice-based signature scheme based on PASS$_{RS}$ that provides provable security along with more secure parameter sets comparing with the original PASS$_{RS}$.

Early candidates of lattice-based signature schemes, such as GGH signature scheme [45] and NTRU signature [51], lack security proofs and have been broken subsequently due to the aforementioned transcript attacks.

The seminal work of Gentry Peikert and Vaikuntanathan [43], known as the GPV framework, opened up a new direction to build secure lattice-based signature schemes. In this framework, one combines a hash-and-sign paradigm with a pre-image sampling function. The signature schemes obtained through this fashion enjoys a provable security based on the hardness of the SIS problem.

In the GPV framework, the efficiency of a signature scheme (in terms of both speed and size) depends heavily on the preimage sampling function and the quality of secret basis produced by the trapdoor generating function. Improving performance of these functions becomes the research objective for the following studies [5, 82, 77]. Alwen and Peikert [5] improves the trapdoor generating function to provide a hard random lattice along with a relatively shorter basis when comparing with [43]. Peikert [82] provides a parallelizable algorithm for preimage sampling which improves the efficiency of sampling signatures. To the best of our knowledge, the most efficient construction following this direction while admitting a security proof is due to Micciancio and Peikert [77]; while in [34, 40], Prest at al. give an efficient instantiation using NTRU lattices, known as FALCON.

Besides GPV framework adopting "hash-and-sign" techniques, there are also lattice-based signature schemes built through Fiat-Shamir heuristics. Lyubashevsky and Micciancio [74] first presented a lattice-based one-time signature scheme based on the ring-SIS problem with key size and computation both linear to the security parameter. This one-time signature scheme can be transformed to a general lattice-based signature scheme by using the standard hash-tree technique. Based on [74], Lyubashevsky [71] then proposed a lattice-based interactive identification scheme and converted the scheme into a signature scheme using Fiat-Shamir heuristics. In the identification scheme, the challenge string from the verifier is an element in some polynomial ring and the prover only needs to respond correctly once to the challenge. However, under some circumstances, the responses from the prover may leak the information about the secret key. So, the prover needs to abort and restart the whole identification protocol. After transforming the identification scheme into a signature scheme, even though the signer still needs to perform abortion during the signing procedure, the signer does not need to output those failure attempts. Thus, the abortion will not increase the number of communications between signer and verifier. This abortion techniques, usually known as rejection sampling, has flourished modern lattice based signatures. For example, by rejecting the generated signature to a Gaussian distribution [72] or a Bimodal Gaussian distribution [31], one is able to reduce both the rejection rate and the size of the signatures. State-of-the-art following this direction is Dilithium [33], whose hardness is based on the learning with error problem over modular lattices.

Different from these previous lattice-based signatures schemes, Hoffstein et al. [53] proposed PASS$_{RS}$ based on the partial Fourier recovery problem. It adopts the same aborting technique used in [71] to decouple the signature from the secret key. Although the time efficiency of PASS$_{RS}$ is comparable with BLISS, we note that there are still rooms for improvement. First of all, PASS$_{RS}$ does not admit a formal reduction proof. Besides, the size of signatures produced by PASS$_{RS}$ is quite large. Moreover, cryptanalysis has been developing very rapidly during the past 2 years due to a new model [4] of analyzing the cost of BKZ 2.0 lattice reduction algorithm [27]. As a consequence, the security level of the original PASS$_{RS}$ will be significantly reduced. It is fair to say PASS$_{RS}$ may not be secure if the originally suggested parameters are adopted. To solve these problems, we apply the rejection sampling technique from [72] to PASS$_{RS}$ to construct a new scheme known as PASS$_G$. We give a formal reduction proof for our new construction, which features reduced signature size thanks to the use of rejection sampling. We further provide several sets of security parameters for our new scheme.

1.4 Lattice-Based (Linkable) Ring Signatures

Ring signature scheme, as a variant of signature schemes, allows a user to sign a message on behalf of a group users in an anonymous way. In another word, when a verifier try to verify a ring signature generated on a given group. The verifier will only acquire the knowledge that whether the ring signature is generated by a member in the group without knowing the true identity of the signer. Due to the anonymity provided by ring signature scheme, it can be applied in scenarios like electronic voting and anonymous identification where user's identity should be protected. Linkable ring signature, besides the property provided by ring signature scheme, has the feature that two signatures sharing the same signer can be linkable. This feature is actually a balance between anonymity and accountability. Nowadays, linkable ring signature scheme has becoming an significant building block used to protect sender privacy in cryptocurrency transaction.

In Section 5.1, we present our new generic constructions of ring signature and linkable ring signature scheme. Our new generic constructions are based on a building block we defined as Chameleon hash plus (CH^+). We give the specific security proof of our generic construction and demonstrate that if the underline CH^+ satisfies its security requirements, the (linkable) ring signature scheme will also be secure. We also give instantiations from both standard lattice, as a proof of concept, and NTRU lattice, as an efficient instantiation. We implement the latter construction, called (linkable) Raptor, and show that it is almost as efficient as the classical ring signatures and thus has practical interest. Moreover, prior to our work, we are not aware of any implementation of lattice-based (linkable) ring signature schemes.

In Section 5.2, we revisit an existing generic framework of ring signature scheme proposed by Abe et al. [1] and extend it to its linkable version. Furthermore, considering in the original paper, security proof only given to concrete examples instead of the generic construction. We thus present a security proof for the original generic ring signature scheme and our linkable version. We instantiate our generic linkable ring signature to its NTRU lattice version and suggest parameters that is secure against quantum attacks. Comparing with our third work (linkable) Raptor, this lattice-based linkable ring signature scheme has a smaller signature size and is more efficient than previous work related to lattice-based linkable ring signature.

1.4.1 Related Work

The notion of ring signatures was put forth by Rivest, Shamir and Tauman in 2001 [85]. It is a special type of group signature [26, 24] where a signer is able to produce a signature on behalf of a group of potential signers. Unlike group signatures, there is no central party to manage group membership nor capable of revealing identity of the generator of the signature. In a typical use case of ring signatures, each user is associated with a public key and a group is formed 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.

Ring signatures offers very strong anonymity. In particular, signatures created by the same signer are unlinkable. Observing that in some real-world applications, such as electronic voting, unlinkability can be undesirable, Liu, Wei and Wong [67] put forth the notion of linkable ring signatures. In such a scheme, the identity of the signer remains anonymous. In the meantime, two signatures created 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-cash scenario, a linkable ring signature allows the spender to remain anonymous, while making it possible for the bank to identify double spenders. To date, linkable ring signature has become a mainstream solution to protect sender privacy in cryptocurrency transaction [81].

All linkable ring signatures deployed in practice are based on number-theoretic assumptions and thus vulnerable to quantum computers [89]. Even though quantum computers are still in their infancy, many believed that general purpose quantum computers will inevitably 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 [78] to the quantum apocalypse. To date, there exist a number of lattice-based ring signature schemes and lattice-based linkable ring signature schemes [21, 75, 64, 38, 93, 13]. While some of them are asymptotically efficient, they are hardly practical. In particular, to the best of our knowledge, none of these constructions come with an implementation.

Classical ring signatures We review the existing constructions of (linkable) ring signatures. The generic construction introduced by Rivest, Shamir and Tauman [85] in 2001 (RST). This generic construction is based on one-way trapdoor permutations along with a block cipher. It can be instantiated from the RSA assumption. In 2004, Abe, Ohkubo and Suzuku [1] (AOS) proposed a new generic construction which allows discrete-log type of keys. This generic construction can make use of hash-and-sign signature or any three-move sigma-protocol-based signature. It can be instantiated from RSA or discrete-log assumptions. Both of the RST and AOS constructions are secure in the random oracle model and the signature sizes are linear to the ring size. To achieve the security in standard model, Bender, Katz and Morselli [16] (BKM) presented a ring signature scheme which adopts a public-key encryption scheme, a signature scheme and a ZAP protocol for any language in $\mathcal{NP}$ [37]. Even though BKM construction is secure in standard model, the signature size is still linear in the number of group members and the generic ZAPs are quite impractical. Shacham and Waters [87] then proposed a more efficient linear-size ring signature scheme without random oracle from bilinear pairing.

To reduce the signature size, Dodis et al. proposed the first ring signature scheme with constant signature size in 2004 [30]. It relies on accumulator with one-way domain and is secure in the random oracle model. The first ring signature with sub-linear without random oracle model is due to Chandran, Groth and Sahai [25]. This scheme has signature size $\mathcal{O}(\sqrt{\ell})$ where ℓ is the number of users in the ring. All of the above sub-linear size constructions are secure in the common reference string model which requires a trusted setup. The first sub-linear ring signatures without relying on a trusted setup is due to Groth and Kohlweiss [47]. It features logarithmic size signature and is secure in the random oracle model.

Classical linkable ring signatures Since the first proposal of linkable ring signature [67], a sequence of work [95, 7, 66, 92] which provides different features has been proposed. In 2005, Tsang and Wei [95] extends the generic ring signature introduced by Dodis et al. [30] to a linkable version, which also feature constant signature size and is secure 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 strengthened model. In 2014, Liu et al. [66] presented the first linkable ring signature scheme achieving unconditional anonymity. Sun et al. [92] 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 [94, 9] and certificate-based linkable ring signatures [8].

Lattice-based ring signatures For ring signatures in the lattice setting, Brakerski and Kalai [21] 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, the resulting scheme is only secure under a weak definition. To achieve full security, an inefficient transformation is needed. Melchor et al. [75] transforms Lyubashevsky's lattice-based signature [72] into a ring signature. As the authors pointed out themselves, their scheme is "pretty unpractical". In 2016, Libert et al. [64] presented a lattice-based accumulator. With the accumulator and a lattice-based zero-knowledge proof system, they build a ring signature scheme that features logarithmic signature size. However, the zero-knowledge arguments applied in the accumulator is very inefficient. The state-of-the-art is the lattice-based ring signature scheme proposed by Esgin et al. [38]. They adapt the efficient one-out-of-many proof [47, 20] to build a lattice-based ring signature scheme. Same as [64], [38] is also a logarithmic size ring signature scheme and is secure in the random oracle model.

Lattice-based linkable ring signatures The first lattice-based linkable ring signature scheme was proposed by Torres et al. in 2018 [93]. It can be seen and instantiation of the AOS framework from the lattice-based BLISS signature [31], with adaption to introduce linkability. The signature size is linear to the number of members in the ring and is reported to be 51 KB per ring member. In the same year, Baum, Lin and Oechsner [13] construct another lattice-based linkable ring signature scheme following a very similar ideal to [93]. The signature size for [13] 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.

In terms of performance, lattice-based (linkable) ring signatures [75, 93, 13] are all based on the lattice-based sigma-protocol-based signature and thus involve additional overhead in the form of rejection sampling which affects the performance of the signature scheme. As mentioned above, the inefficiency of the underlying zero-knowledge proof system makes [64] quite impractical. It is fair to say constructing linkable ring signatures from lattices (even with linear signature size) is non-trivial. Common framework such as AOS does not give concrete security proof; and the RST framework relies on one-way trapdoor permutation of which no lattice-based realization is known. In Section 5.1, we present a practical and efficient lattice-based (linkable) ring signature scheme and implement it on a typical laptop. We also provide the performance comparison with previous related works. In Section 5.2 we revisit the generic framework of ring signatures based on the Type-H signatures presented in AOS framework and prove the security of the generic construction in a strengthened model. We also instantiate AOS framework to its linkable version and give its NTRU lattice-based instantiation.

Chapter 2

Preliminary

2.1 Notation

Elements in $\mathbb{Z}_q$ are represented by integers in $[-\frac{q}{2}, \frac{q}{2})$. We use cyclotomic polynomial rings $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ with n being a power of 2 and q being a prime. An element $\mathbf{a} \in \mathcal{R}_q$ is represented as a polynomial $\mathbf{a} = a_0 + a_1\mathbf{x} + a_2\mathbf{x}^2 + \cdots + a_{n-1}\mathbf{x}^{n-1}$ with coefficients $a_i \in \mathbb{Z}_q$. We can also use vector $[a_0, a_1, a_2, \cdots, a_{n-1}]^T$ to represent polynomial $\mathbf{a}$. Thus, column vectors 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.

In Chapter 4, we use $\star$ to denote the multiplication on $\mathcal{R}_q$ and $\odot$ to denote component-wise multiplication of vectors. Assume q is a prime number and congruent to $1 \mod 2n$. For any β with $\gcd(\beta, q) = 1$, Fermat's little theorem says $\beta^{q-1} = 1 \pmod q$. Since $q = rn + 1$, we have $\beta^{rn} = 1 \bmod q$. We can define a ring homomorphism mapping $\mathbf{f} \to \mathbf{f}(\beta^{\mathbf{r}})$ for any $\mathbf{f} \in \mathcal{R}_q$. For any $\mathbf{f}_1, \mathbf{f}_2 \in \mathcal{R}_q$,

$$(\mathbf{f}_1 + \mathbf{f}_2)(\beta^r) = \mathbf{f}_1(\beta^r) + \mathbf{f}_2(\beta^r) \text{ and } (\mathbf{f}_1 \star \mathbf{f}_2)(\beta^r) = \mathbf{f}_1(\beta^r) \odot \mathbf{f}_2(\beta^r)$$

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:

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

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.

For distribution D, $x \leftarrow D$ means sampling x according to distribution D. For set S, $x \leftarrow_\$ S$ means that x is chosen uniformly at random from S. $\|\mathbf{v}\|_1$ is the ℓ_1 norm of vector $\mathbf{v}$ and $\|\mathbf{v}\|$ is the ℓ_2 norm of $\mathbf{v}$.

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 $D^n_{\mathbf{v},\sigma}(\mathbf{x}) = \frac{\rho^n_{\mathbf{v},\sigma}(\mathbf{x})}{\rho^n_{\mathbf{v},\sigma}(\mathbb{Z}^n)}$.

Lemma 2.1.1 (Rejection Sampling [31]) *Let V be an arbitrary set, and $h : V \to \mathbb{R}$ and $f : \mathbb{Z}^m \to \mathbb{R}$ be probability distributions. If $g_v : \mathbb{Z}^m \to \mathbb{R}$ is a family of probability distribution indexed by all $v \in V$ with the property that*

$$\exists M \in \mathbb{R} \textit{ such that } \forall v \in V, \forall \mathbf{z} \in \mathbb{Z}^m, \Pr[M \cdot g_v(\mathbf{z}) \geq f(\mathbf{z})] \geq 1 - \varepsilon.$$

Then the output distribution of the following algorithm $\mathcal{A}$:

1. $v \leftarrow h$

2. $\mathbf{z} \leftarrow g_v$

3. *output* $(\mathbf{z}, v)$ *with probability* $\min\left(\frac{f(\mathbf{z})}{M \cdot g_v(\mathbf{z})}, 1\right)$

is within statistical distance $\frac{\varepsilon}{M}$ *of the output distribution of the following algorithm* $\mathbf{F}$*:*

1. $v \leftarrow h$

2. $\mathbf{z} \leftarrow f$

3. *output* $(\mathbf{z}, v)$ *with probability* $\frac{1}{M}$

The probability of algorithm $\mathcal{A}$ *output something is at least* $\frac{1-\varepsilon}{M}$*.*

2.2 Lattice-Based Cryptography

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

Fig.2.1 shows a 2-dimensional example. Both $(\mathbf{b}_1, \mathbf{b}_2)$ and $(\mathbf{b}'_1, \mathbf{b}'_2)$ are $basis$ for this lattice.

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 2.2.1 *Let* $m \geq n \geq 1$ *and* $q \geq 2$*. For arbitrary matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and vector*

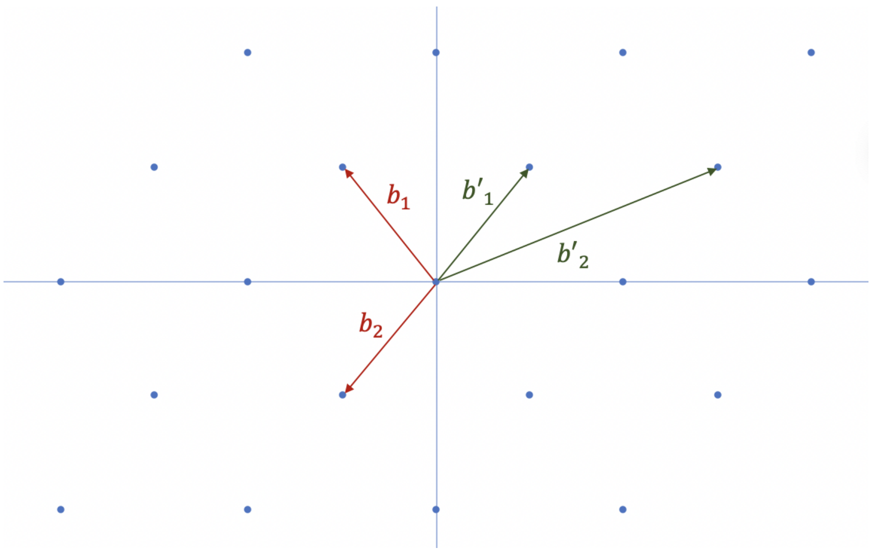

Figure 2.1: A lattice in $\mathbb{R}^2$

$\mathbf{u} \in \mathbb{Z}_q^n$ *define m-dimensional full-rank integer lattices and its shift:*

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

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

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 2.2.2 (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 2.2.3 (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* $\mathbf{x} \in \Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$ *and* $0 < \|\mathbf{x}\| \leq \beta$*.*

According to [43], 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$_\gamma$ (Shortest

Independent Vector Problem) with $\gamma = \beta\tilde{O}(\sqrt{n})$. Similarly, R-SIS (R-ISIS) problems are defined as an analogue of SIS (ISIS) problem in ideal lattices.

Definition 2.2.4 (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 2.2.5 (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 [83, 73]. According to [73], the R-SIS$_{q,m,\beta}$ is as hard as the SVP$_\gamma$ (Shortest Vector 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 2.2.6 (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 [56, 91, 68]. 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.

Lemma 2.2.1 ([72]Lemma 3.3)

1. *For any $k > 0$, $\Pr[\|\mathbf{z}\| > k\sigma\sqrt{n}; \mathbf{z} \leftarrow D_\sigma^n] < k^n e^{\frac{n}{2}(1-k^2)}$.*

2. *For any $\mathbf{z} \in \mathbb{Z}^n$, and $\sigma \geq 3/\sqrt{2\pi}$, $D_\sigma^n(\mathbf{z}) \leq 2^{-n}$.*

3. *For any vector $\mathbf{v} \in \mathbb{R}^n$ and any $\sigma, r > 0$, we have: $\Pr[|\langle \mathbf{z}, \mathbf{v} \rangle| > r; \mathbf{z} \leftarrow D_\sigma^n] \leq 2\exp(-\frac{r^2}{2\|\mathbf{v}\|^2\sigma^2})$*

4. *For any positive real number $s > 0$, we have $\Pr_{\mathbf{x} \leftarrow D_s^m}[\|\mathbf{x}\| \leq 2\sqrt{m}s] \geq 1 - 2^{-m}$.*

2.2.2 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 [44] 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 [44]; while Peikert [82] 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 [77]. Here we re-state one of their results.

Theorem 2.2.1 ([77], Theorem 5.1) *There exists an efficient algorithm* $\mathsf{GenBasis}$ *(*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* $\mathsf{PreSample}$*. With overwhelming probability over all random choices, for any* $\mathbf{u} \in \mathbb{Z}_q^n$ *and large enough* $s = O(\sqrt{n \log q})$*,* $\mathsf{PreSample}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s)$ *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. [34, 40] using NTRU lattices [56]. The corresponding signature scheme is named FALCON [40].

FALCON is a candidate lattice-based signature scheme to the NIST post-quantum standardization process [78]. It is the resurrection of NTRUSign [51] with the aforementioned GPV framework for transcript security [44, 34], and a fast Fourier sampling for efficiency [36]. 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 [43], under the (quantum) random oracle model [18];
- although the parameters in FALCON does not support GPV's security proof, they are robust against best known attacks[1].

The high level description of FALCON is in Section 2.4.2.

2.3 Ad Hoc Anonymous Identification Scheme

An ad hoc anonymous identification scheme consists of four efficient algorithms, namely, Setup, $\mathsf{Register}$, Pr, Vf, where:

- $\mathsf{Setup}(1^\lambda)$. On input a security parameter 1^λ, this algorithm generates the system's parameter param. We assume param is an implicit input to all the algorithms listed below.
- $\mathsf{Register}(\mathcal{I})$. This algorithm allows users to register with the system. On input a new user identity, $\mathcal{I}$, this algorithm outputs the corresponding user secret key $\mathsf{sk}_\mathcal{I}$.
- $\mathsf{Pr}(\mathsf{sk}_\mathcal{I}) \overset{\mathcal{L}_\mathcal{I}}{\longleftrightarrow} \mathsf{Vf}(\mathcal{L}_\mathsf{sk})$. This is the interactive identification protocol runs between PPT Pr and Vf. The common inputs to the algorithms are a list of user identities $\mathcal{L}_\mathcal{I} := \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell\}$ and the list of corresponding secret key $\mathcal{L}_\mathsf{sk} := \{\mathsf{sk}_{\mathcal{I}_1}, \mathsf{sk}_{\mathcal{I}_2}, \ldots, \mathsf{sk}_{\mathcal{I}_\ell}\}$. Upon successful completion of the protocol, Vf outputs $0/1$ to indicate rejection or acceptance.

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

Typically, the identification system consists of one server (verifier) and a number of users (provers). Algorithm Setup is usually executed by the server. $\mathsf{Register}$ could be initiated by the server or a user, depending on the application scenario. When a user wishes to identify anonymously to the server, the two parties engage in an interactive identification protocol where Pr and Vf will be executed by the user and the server respectively. We note that the common input $\mathcal{L}_{\mathcal{I}}$ can be chosen by either the server or the user, depending on the application scenario.

Correctness. We required that an honest verifier will always accept the identification from an honest prover. More formally, we require that the quantity

$$\Pr\left[\begin{array}{c|c} \begin{array}{c} \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_{\mathcal{I}_i} \leftarrow \mathsf{Register}(\mathcal{I}_i) \\ \text{for } i = 1 \text{ to } n \\ \mathcal{L}_{\mathcal{I}} := \{\mathcal{I}_1, \ldots, \mathcal{I}_n\} \\ \mathcal{L}_{\mathsf{sk}} := \{\mathsf{sk}_{\mathcal{I}_1}, \ldots, \mathsf{sk}_{\mathcal{I}_n}\} \\ \mathsf{sk}_{\mathcal{I}} \overset{R}{\leftarrow} \mathcal{L}_{\mathsf{sk}} \end{array} & \mathsf{Pr}(\mathsf{sk}_{\mathcal{I}}) \overset{\mathcal{L}_{\mathcal{I}}}{\longleftrightarrow} \mathsf{Vf}(\mathcal{L}_{\mathsf{sk}}) \rightarrow 1 \end{array}\right]$$

is greater than or equal to $1 - \mathsf{negl}(\lambda)$.

2.3.1 Security Requirements

We consider two security requirements for an ad hoc anonymous identification scheme, namely, anonymity and soundness. In this subsection, we formalize these requirements as games between a challenger and an attacker.

Anonymity

To define the anonymity of an ad hoc anonymous identification scheme, we define the following game, denoted as $\mathsf{Game}_{\mathsf{anon}}$, between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup. Challenger $\mathcal{C}$ runs Setup with security parameter 1^λ and generates system parameter param. Then $\mathcal{C}$ sends param to adversary $\mathcal{A}$.

Challenge. $\mathcal{A}$ sends a list of identity-key pairs $\{(\mathcal{J}_i, \mathsf{sk}_{\mathcal{J}_i})\}_{i=1}^{\ell}$ to $\mathcal{C}$. $\mathcal{A}$ further specifies two identities $\mathcal{I}_0, \mathcal{I}_1$ such that $\mathcal{I}_0 = \mathcal{J}_x$ and $\mathcal{I}_1 = \mathcal{J}_y$ for some $x, y \in \{1, \ldots, \ell\}$. We use $\mathsf{sk}_{\mathcal{I}_0}$ (resp. $\mathsf{sk}_{\mathcal{I}_1}$) to denote $\mathsf{sk}_{\mathcal{J}_x}$ (resp. $\mathsf{sk}_{\mathcal{J}_y}$). Define $\mathcal{L}_{\mathcal{I}}^*$ to be $\{\mathcal{J}_i\}_{j=1}^{\ell}$ and $\mathcal{L}_{\mathsf{sk}}^* := \{\mathsf{sk}_{\mathcal{J}_i}\}_{i=1}^{\ell}$.

Then $\mathcal{C}$ flips a fair coin $b \in \{0, 1\}$ and invokes $\pi^* \leftarrow [\mathsf{Pr}(\mathsf{sk}_{\mathcal{I}_b}) \overset{\mathcal{L}_{\mathcal{I}}^*}{\longleftrightarrow} \mathsf{Vf}(\mathcal{L}_{\mathsf{sk}}^*)]$. The resulting transcript, π^*, is given to $\mathcal{A}$.

Guess. Finally, $\mathcal{A}$ outputs a bit b'. We say that $\mathcal{A}$ wins $\mathsf{Game}_{\mathsf{anon}}$ if $b = b'$.

The advantage of $\mathcal{A}$, $\mathsf{adv}_{\mathcal{A},\mathsf{anon}}$, is defined as the probability that it wins the above game minus $\frac{1}{2}$.

Definition 2.3.1 *An ad hoc anonymous identification scheme is said to offer anonymity if for any adversary $\mathcal{A}$, the advantage* $\mathsf{adv}_{\mathcal{A},\mathsf{anon}}$ *in the above game is negligible.*

In the above definition, $\mathcal{A}$ is not computationally bounded. In other words, a scheme satisfying Definition 2.3.1 offers unconditional anonymity. We also note that the above game allows the adversary to present maliciously chosen keys (i.e., keys that do not follow the distribution of algorithm Register). In other words, anonymity has to be preserved even when the keys are not properly chosen.

Soundness

Soundness of ad hoc anonymous identification scheme captures the requirement that a user without a legitimate secret key for an identity in the list of authenticating users should be rejected in an identification protocol by the verifier. We introduce $\mathsf{Game}_{\mathsf{sec}}$ between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$ to formally capture this intuition. The goal of $\mathcal{A}$ in the game is to prove his identity without valid user ID and secret key pair $(\mathcal{I}, \mathsf{sk})$.

Setup. Challenger $\mathcal{C}$ runs Setup with security parameter 1^λ and generates system parameter param. Then $\mathcal{C}$ sends param to adversary $\mathcal{A}$. $\mathcal{C}$ maintains two lists, namely, the list of honest users $(\mathcal{L}_H)$ and corrupted users $(\mathcal{L}_C)$.

Query. $\mathcal{A}$ can issue three types of queries in an adaptive manner.

- $\mathsf{Register}(\mathcal{I}, \omega)$. $\mathcal{A}$ can issue $\mathsf{Register}$-queries to $\mathcal{C}$ to introduce users into the system. If $\omega = \perp$, $\mathcal{C}$ invokes $\mathsf{Register}$ on input $\mathcal{I}$ and obtains $\mathsf{sk}_\mathcal{I}$. $\mathcal{I}$ is added to $\mathcal{L}_H$. Otherwise, $\mathcal{C}$ sets $\mathsf{sk}_\mathcal{I} := \omega$ adds $\mathcal{I}$ to $\mathcal{L}_C$.

- $\mathsf{Corrupt}(\mathcal{I})$. $\mathcal{A}$ submits a user identity $\mathcal{I}$. If $\mathcal{I}$ is included in $\mathcal{L}_H$, $\mathcal{C}$ returns to $\mathcal{A}$ the corresponding $\mathsf{sk}_\mathcal{I}$ and moves $\mathcal{I}$ from $\mathcal{L}_H$ to $\mathcal{L}_C$.

- $\mathsf{Trans}(\mathcal{I}, \mathcal{L}_\mathcal{I})$. $\mathcal{A}$ chooses a set of users $\mathcal{L}_\mathcal{I} \subset \mathcal{L}_C \cup \mathcal{L}_H$ and a user $\mathcal{I} \in \mathcal{L}_\mathcal{I}$ to obtain an identification transcript. $\mathcal{C}$ first collects the corresponding user secret key $\mathcal{L}_{\mathsf{sk}} := \{\mathsf{sk}_\mathcal{I} | \mathcal{I} \in \mathcal{L}_\mathcal{I}\}$. Next, it executes $\pi \leftarrow [\mathsf{Pr}(\mathsf{sk}_\mathcal{I}) \overset{\mathcal{L}_\mathcal{I}}{\longleftrightarrow} \mathsf{Vf}(\mathcal{L}_{\mathsf{sk}})]$ and returns π to $\mathcal{A}$.

Challenge. $\mathcal{A}$ chooses a set of user identities $\mathcal{L}^* \subset \mathcal{L}_H$ on which it wishes to be challenged. $\mathcal{C}$ parses $\mathcal{L}^*_{\mathsf{sk}} := \{\mathsf{sk}_\mathcal{I} | \mathcal{I} \in \mathcal{L}^*\}$. Next, $\mathcal{C}$ plays the role of the verifier with input $(\mathcal{L}^*, \mathcal{L}^*_{\mathsf{sk}})$ with $\mathcal{A}$ acting as a prover. We says that $\mathcal{A}$ wins the game if and only if

$$\mathcal{A} \overset{\mathcal{L}^*}{\longleftrightarrow} \mathsf{Vf}(\mathcal{L}^*_{\mathsf{sk}}) \rightarrow 1$$

The advantage of $\mathcal{A}$, $\mathsf{adv}_{\mathcal{A},\mathsf{sec}}$, is defined as the probability that it wins the above game.

Definition 2.3.2 *An ad hoc anonymous identification scheme is sound if for any PPT adversary $\mathcal{A}$ the advantage* $\mathsf{adv}_{\mathcal{A},\mathsf{sec}}$ *is negligible* .

We would like to remark that our definitions (Definition 2.3.1 and Definition 2.3.2) only allow the attacker to passively eavesdrop the communications. We note that this is a common security requirement for identification protocols as in [30]. One possible reason is that in most cases, the constructions are Σ-protocol that will be converted generically to its non-interactive form in which the generic construction requires the identification protocol to be passively sound. However, we shall see in Chapter 3 that our protocol is not a Σ-protocol. The implication of the choice this security definition will be discussed after we present our construction.

2.4 Digital Signature Scheme

A digital signature scheme consists of three algorithms, namely, KeyGen, $\mathsf{Signing}$, $\mathsf{Verification}$, described as follows.

- **KeyGen**$(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$: This key generation algorithm generates private signing key sk and public verification key pk.

- **Signing**$(\mathsf{sk}, \mu) \to \sigma$: On input signing key sk and message μ, the signing algorithm outputs signature σ on μ.

- **Verification**$(\mu, \sigma, \mathsf{pk}) \to$ $accept/reject$: On input message μ, signature σ and verification key pk, the verification algorithm outputs accept if σ is a signature on μ. otherwise, it outputs reject.

Security of a digital signature scheme can be defined by a Game held between a challenger $\mathcal{C}$ and a probabilistic polynomial-time adversary $\mathcal{A}$. Game consists of three phases, namely, $Setup$, $Query$ and $Output$.

- $Setup$. The challenger $\mathcal{C}$ runs **KeyGen** algorithm and obtains private signing key and public verification key pair $(\mathsf{sk}, \mathsf{pk})$. $\mathcal{C}$ sends verification key pk to the forger $\mathcal{A}$.

- $Query$. Adversary $\mathcal{A}$ sends message μ_i to challenger $\mathcal{C}$. $\mathcal{C}$ signs μ_i using sk and returns the corresponding signature σ_i to $\mathcal{A}$. Adversary $\mathcal{A}$ repeats the process n times where n is polynomial in λ and finally obtains a list of message and signature pair $((\mu_1, \sigma_1),(\mu_2, \sigma_2),\cdots,(\mu_n, \sigma_n))$.

- $Output$. The adversary $\mathcal{A}$ outputs a forgery (μ^*, σ^*). $\mathcal{A}$ wins Game if

  $(\textbf{Verification}(\mu^*, \sigma^*, \mathsf{pk}) \to accept) \wedge ((\mu^*, \sigma^*) \notin \{(\mu_1, \sigma_1), (\mu_2, \sigma_2), \cdots, (\mu_n, \sigma_n)\})$.

Definition 2.4.1 *A signature scheme (**KeyGen**, **Signing**, **Verification**) is said to be strong unforgeable if for any polynomial-time adversary $\mathcal{A}$, the probability of $\mathcal{A}$ winning* Game *is negligible.*

2.4.1 One-Time Signature Scheme

A digital signature scheme is said to be one-time secure if it satisfies following definition,

Definition 2.4.2 *A signature scheme (**KeyGen**, **Signing**, **Verification**) is said to be one-time secure if for any polynomial-time adversary $\mathcal{A}$, the probability of $\mathcal{A}$ winning* Game^* *is negligible.*

Game^* consists of three phases, namely, $Setup$, $Query$ and $Output$.

- $Setup$. The challenger $\mathcal{C}$ runs **KeyGen** algorithm and obtains private signing key and public verification key pair $(\mathsf{sk}, \mathsf{pk})$. $\mathcal{C}$ sends verification key pk to the forger $\mathcal{A}$.

- $Query$. Adversary $\mathcal{A}$ sends one message μ to challenger $\mathcal{C}$. $\mathcal{C}$ signs μ using sk and returns the corresponding signature σ to $\mathcal{A}$. Adversary $\mathcal{A}$ obtains a message and signature pair (μ, σ).

- $Output$. The adversary $\mathcal{A}$ outputs a forgery (μ^*, σ^*). $\mathcal{A}$ wins Game if

$$(\textbf{Verification}(\mu^*, \sigma^*, \mathsf{pk}) \to accept) \land ((\mu^*, \sigma^*) \neq (\mu, \sigma).$$

2.4.2 FALCON Signature Scheme

This section gives the high level description of FALCON signature scheme. The detail of the scheme can be found in [40]. Here we assume the signature scheme works over a polynomial ring $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^n + 1)$.

- FALCON.$\mathsf{KeyGen}(1^\lambda) \to (\mathbf{a}, \mathbf{T})$: this algorithm takes security parameter 1^λ as input and chooses random $\mathbf{f}$ and g polynomials $(\mathbf{f}, \mathbf{g} \in \mathcal{R}_q)$ using an appropriate distribution. The public key will be set as $\mathbf{a} = \mathbf{g}/\mathbf{f}$ and the secret key $\mathbf{T} := \begin{bmatrix} \mathbf{f} & \mathbf{g} \\ \bar{\mathbf{f}} & \bar{\mathbf{g}} \end{bmatrix}$ is the trapdoor of $\mathbf{a}$. $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}$ satisfy $\mathbf{f}\bar{\mathbf{g}} - \mathbf{g}\bar{\mathbf{f}} = q \mod (x^n + 1)$ and $\mathbf{f}$, $\mathbf{g}$, $\bar{\mathbf{f}}$ $\bar{\mathbf{g}}$ should be short.

- FALCON.$\mathsf{Sign}(\mathbf{a}, \mathbf{T}; \mu) \to (\mathbf{r}_0, \mathbf{r}_1)$: the signing algorithm first hashing the message μ into a polynomial $\mathbf{c} \in \mathcal{R}_q$. Then it uses the short trapdoor $\mathbf{T}$ to produce a pair of short polynomials $(\mathbf{r}_0, \mathbf{r}_1)$ such that $\mathbf{r}_0 + \mathbf{a}\mathbf{r}_1 = \mathbf{c}$.

- FALCON.$\mathsf{Verify}(\mathbf{a}, (\mathbf{r}_0, \mathbf{r}_1), \mu) \to 0/1$: this algorithm verifies that $(\mathbf{r}_0, \mathbf{r}_1)$ is a pair of appropriately short polynomials and $\mathbf{c} = \mathbf{r}_0 + \mathbf{a}\mathbf{r}_1$ where $\mathbf{c}$ is the hash of message μ. If all pass, output 1; otherwise, output 0.

2.5 Ring Signature Scheme

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

2.5.1 Syntax

A ring signature scheme usually is a tuple of four algorithms (**Setup**, **KeyGen**, **Signing**, $\mathbf{Verify}$):

- **Setup**$(1^\lambda) \to$ 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**$\to$ $(\mathsf{sk}, \mathsf{pk})$: By taking system parameter param, this key generation algorithm generates a private signing key sk and a public verification key pk.
- **Signing**$(\mathsf{sk}, \mu, L_{\mathsf{pk}}) \to \sigma$: 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**$(\mu, \sigma, L_{\mathsf{pk}}) \to$ $accept/reject$: 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.

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

2.5.2 Security Notions

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* $\mathcal{RO}(\perp) \to \mathsf{pk}_i$: 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 $\mathsf{Game}_{\mathsf{forge}}$, 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 $\mathsf{Game}_{\mathsf{forge}}$ if
  - $\mathbf{Verification}(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) = accept$;
  - $(\mu^*, L^*_{\mathsf{pk}})$ has not been queried by $\mathcal{A}$;
  - all public keys in L^*_{pk} should be outputs of $\mathcal{RO}$; 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 $\mathsf{Game}_{\mathsf{forge}}$:

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

Definition 1 (Unforgeability) *A ring signature scheme (**KeyGen**, **Signing**, **Verification**) is said to be unforgeable if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{adv}_{\mathcal{A}}^{\mathsf{forge}}$ 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 $\mathsf{Game}_{\mathsf{anon}}$, 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 2 (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.*

2.6 Linkable Ring Signature Scheme

In this section, we are going to present the syntax and security requirements 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.

2.6.1 Syntax

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

- **Setup**$(1^\lambda) \rightarrow \mathsf{param}$: 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.

2.6.2 Security Notions

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 2.5.2. The definitions of anonymity, linkability and nonslanderability are adopted from Liu et al. [66].

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. All public keys in L_{pk} should be query outputs of $\mathcal{RO}$. $\mathcal{A}$ sends (L_{pk}, μ) to $\mathcal{C}$. $\mathcal{C}$ randomly picks $\pi \in \{1, \cdots, n\}$ and runs **Signing**(sk_π, μ, L_{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 3 (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 k sets, $\{L^{(i)}_{\mathsf{pk}}, \mu_i, \sigma_i\}$ for $i \in [1, \cdots, k]$, where $L^{(i)}_{\mathsf{pk}}$ is a list of public keys, μ_i is message, σ_i is signature.

$\mathcal{A}$ wins the game if

- all σ_is are not query output of $\mathcal{SO}$;

- all public keys in $L^{(i)}_{\mathsf{pk}}$ are query output of $\mathcal{RO}$;

- **Verification**(μ_i, σ_i,$L^{(i)}_{\mathsf{pk}}$) = *Accept* ;

- $\mathcal{A}$ queried $\mathcal{CO}$ less than k times; and

- **Link**(σ_i, σ_j, μ_i, μ_j, $L_{\mathsf{pk}}^{(i)}$, $L_{\mathsf{pk}}^{(j)}$) = *unlinked* for $i, j \in [1, \cdots, k]$ and $i \neq j$.

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

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

Definition 2.6.1 (Linkability) *A linkable ring signature scheme is linkable if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{adv}_{\mathcal{A}}^{\mathsf{link}}$ 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 **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:

- **Verification**(μ^*, σ^*,L_{pk}^*) = *accept*;

- pk_π is not queried by $\mathcal{A}$ to $\mathcal{CO}$;

- pk_π is not queried by $\mathcal{A}$ as an insider to $\mathcal{SO}$; and

- **Link**(σ, σ^*, μ, μ^*) = *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 2.6.2 (Nonslanderability) *A linkable ring signature scheme is nonsladerable if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{adv}_{\mathcal{A}}^{\mathsf{slander}}$ is negligible.*

Theorem 2.6.1 *[[10], Sec 3.2] If a linkable ring signature scheme is linkable and nonslanderable, it is also unforgeable.*

Chapter 3

Anonymous Identification for Ad Hoc Group

In this chapter, we propose a conceptually simple and efficient approach to construct an ad hoc anonymous identification scheme. Specifically, we make the following contributions.

- We formalize the notion of symmetric-key based anonymous identifications for ad hoc group and develop security models to capture security requirements.
- We introduce a conceptually simple approach based on program obfuscation and a concrete construction of this primitive. We prove that our proposal satisfies the security definitions.
- We conduct empirical analysis on the efficiency of our proposal and show that our system out-perform existing solutions in practical settings.

3.1 Overview

As mentioned above, we use obfuscation in our scheme to achieve security. Informally, an *obfuscator* $\mathcal{O}$ is an efficient and probabilistic "compiler" that transforms a program

P into a new program $\mathcal{O}(P)$ which still has the same functionality with P and reveals no secrets that may be used by P. This techniques can be very useful and with wide applications, for example, to prevent tampering or protect copyright a software developer needs obfuscation to hide secrets in the code while maintaining its functionality.

The theoretical investigation of obfuscation was initiated by Barak *et al.* [12] in which they discovered several impossibility results. The first positive results in program obfuscation was presented by Lynn, Prabhakaran and Sahai [70]. Before this, none of the proposed program obfuscation schemes had proven its security properties. In [70], several provably-secure obfuscation techniques were presented in the random oracle model including the obfuscation of multi-point functions.

Assume each user is represented by a unique user identity, $\mathcal{I}$, and that he/she shares a secret key, $\mathsf{sk}_{\mathcal{I}}$, with the verifier. Define a function $g_{\mathcal{L}}(\cdot)$ such that

$$g_{\mathcal{L}}(\mathsf{sk}_{\mathcal{I}}) = \begin{cases} 1 & \text{if } \mathsf{sk}_{\mathcal{I}} \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

At an abstract level, an identification for ad hoc group in the symmetric key setting is a mechanism that realises a multi-point function. Specifically, the server specifies $\mathcal{L}$ and accept an identification from a user if and only if $g_{\mathcal{L}}(\mathsf{sk}) = 1$.

We further define a function $f_{\mathcal{L},\beta}(\cdot)$ as $f_{\mathcal{L},\beta}(x) \mapsto g_{\mathcal{L}}(x) * \beta$. Now $f_{\mathcal{L},\beta}(\cdot)$ is a multi-input multi-bit output point function. Below we outline our idea in constructing an anonymous identification for ad hoc group in the symmetric key setting based on function $f_{\mathcal{L},\beta}$. As a warm-up, we first give a construction that possesses anonymity but not soundness. The warm-up construction is illustrated in Figure 3.1. Note that the construction is trivially anonymous, since the output of all legitimate users in the list $\mathcal{L}$ is indistinguishable. The same can be said for users not in the list. However, one challenge remains. In the above

$$\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
& & \beta \xleftarrow{R} \{0,1\}^* \\
& & \text{Choose user list } \mathcal{L} \\
& \xleftarrow{f_{\mathcal{L},\beta}(\cdot)} & \text{Construct function } f_{\mathcal{L},\beta}(\cdot) \\
\beta' = f_{\mathcal{L},\beta}(\mathsf{sk}) & & \\
& \xrightarrow{\beta'} & \text{If } \beta' = \beta\text{, output 1} \\
& & \text{Otherwise, output 0}
\end{array}$$

Figure 3.1: Warm-up construction

protocol, there is no mechanism to prevent an attacker from reading the value β from the implementation of function $f_{\mathcal{L},\beta}$ and returns β to authenticate without the need to use a secret key.

To tackle this challenge, we observe that it suffices if $f_{\mathcal{L},\beta}$ can be implemented as a black-box. So, for the security of our scheme, we need to turn the function $f_{\mathcal{L},\beta}$ into a black box which means no one can get any useful information by reading code after the transition. In our scheme, we use obfuscation to obfuscate function $f_{\mathcal{L},\beta}$ and turn $f_{\mathcal{L},\beta}$ into a black box. Looking ahead, the anonymity of our scheme is unconditional even in the situation that obfuscation is broken while soundness is based on whether or not we can implement function $f_{\mathcal{L},\beta}$ as a black box. This requirement is equivalent to the obfuscation of a multi-input multi-bit output function of which efficient solution exists in the random oracle model.

3.2 Our Construction

In this section, we give the details of our scheme.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the algorithm chooses a hash function $\mathcal{R} : \{0,1\}^* \rightarrow$

$\{0,1\}^{2\lambda+s(\lambda)}$, where $s(\lambda)$ is a quantity polynomial in λ. Set param as $\mathcal{R}$, which will be modelled as a random oracle.

$\mathsf{Register}(\mathcal{I})$: On input a new user identity, $\mathcal{I}$, the algorithm first compares $\mathcal{I}$ with all the identities stored in its database. If $\mathcal{I}$ exists, it returns false and abort. Otherwise, it randomly generates random bit-string $\mathsf{sk}_{\mathcal{I}} \in_R \{0,1\}^{\lambda}$. Then, the tuple $(\mathcal{I}, \mathsf{sk}_{\mathcal{I}})$ is stored in its database.

$\mathsf{Pr}(\mathsf{sk}_{\mathcal{I}}) \overset{\mathcal{L}_{\mathcal{I}}}{\longleftrightarrow} \mathsf{Vf}(\mathcal{L}_{\mathsf{sk}})$: The pair of interactive algorithms are to be executed by the prover and the verifier respectively. The prover and the verifier first agrees on the list of user identities $\mathcal{L}_{\mathcal{I}} := \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell\}$. The anonymous identification protocol between the prover and the verifier is a two-move protocol:

1. From $\mathcal{L}_{\mathcal{I}}$, the verifier obtains the list of corresponding secret keys $\mathcal{L}_{\mathsf{sk}} := \{\mathsf{sk}_{\mathcal{I}_1}, \mathsf{sk}_{\mathcal{I}_2}, \ldots, \mathsf{sk}_{\mathcal{I}_\ell}\}$ from the database. Then, it picks a random $\beta \in_R \{0,1\}^{s(\lambda)}$. The verifier computes $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})$, the obfuscation of the function $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ and sends it to the prover. Recall that $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ is a multi input multi-bit output point function with the following specification:

$$f_{\mathcal{L}_{\mathsf{sk}},\beta}(\mathsf{sk}_{\mathcal{I}}) = \begin{cases} \beta & \text{if } \mathsf{sk} \in \mathcal{L}_{\mathsf{sk}} \\ 0 & \text{otherwise} \end{cases}$$

   Details of $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})$ will be discussed in the next paragraph.

2. Upon receiving $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})$, the prover evaluates $\beta' := \mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})(\mathsf{sk}_{\mathcal{I}})$. The prover returns β' to the verifier if $\beta' \neq 0$.

3. The verifier outputs 1 if and only if $\beta = \beta'$. Otherwise, it outputs 0.

Our construction is shown in Figure 3.2.

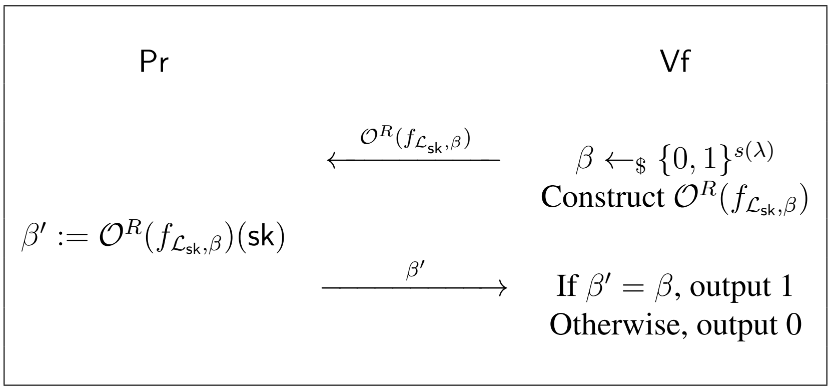
Pr
Vf
$\mathcal{O}^R(f_{\mathcal{L}_{\mathsf{sk}},\beta})$
$\beta \leftarrow_\$ \{0,1\}^{s(\lambda)}$
Construct $\mathcal{O}^R(f_{\mathcal{L}_{\mathsf{sk}},\beta})$
$\beta' := \mathcal{O}^R(f_{\mathcal{L}_{\mathsf{sk}},\beta})(\mathsf{sk})$
β'
If $\beta' = \beta$, output 1
Otherwise, output 0

Figure 3.2: Our anonymous identification protocol based on obfuscations

Details of $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})$ In this paragraph, we discuss an efficient implement of $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})$ using obfuscation of multi-point functions based on the techniques from [70]. The obfuscation of $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ is constructed as follows.

- Denote by $\mathcal{R}_1(\cdot)$ the first 2λ bits output of $\mathcal{R}$ and $\mathcal{R}_2(\cdot)$ the last $s(\lambda)$ bits of $\mathcal{R}$. Choose a random $\delta \in_R \{0,1\}^\lambda$.
- Parse $\mathcal{L}_{\mathsf{sk}}$ as $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell\}$, where $\ell = |\mathcal{L}_{\mathsf{sk}}|$. For $i = 1$ to ℓ, compute $a_i = \mathcal{R}_1(\delta, sk_i)$, $b_i = \mathcal{R}_2(\delta, sk_i)$, $c_i = \beta \oplus b_i$. The obfuscated function $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})$ is defined as $(\delta, \{a_i, c_i\}_{i=1}^{\ell})$.
- To evaluate $\mathcal{O}^{\mathcal{R}}(f_{\mathcal{L}_{\mathsf{sk}},\beta})(x)$, locate i such that $a_i = \mathcal{R}_1(\delta, x)$ and outputs $c_i \oplus \mathcal{R}_2(\delta, x)$. If i cannot be found, output 0.

Discussions Note that the downlink is of complexity $O(\ell)$ while the uplink is of constant complexity. The verifier's computation is $O(\ell)$, while that of the prover can be reduced to $\mathcal{O}(1)$ if the list of identities is used to label the values a_i's so that the prover knows exactly which i should he based his computation on.

3.3 Analysis

In this section we are going to prove the security of our scheme and give the efficiency analysis.

3.3.1 Security Analysis

Theorem 3.3.1 *Our ad hoc anonymous identification scheme is unconditionally anonymous according to Definition 2.3.1.*

Proof 1 *Assume there exists an adversary trying to attack the anonymity of proposed scheme, the adversary chooses two valid user identity and secret key pairs* $(\mathcal{I}_0, \mathsf{sk}_{\mathcal{I}_0}), (\mathcal{I}_1, \mathsf{sk}_{\mathcal{I}_1})$*. To break the anonymity, the adversary is given the transcript between* Pr *and* Vf *generated from one of the two pairs. The goal of the adversary is to guess whether* $(\mathcal{I}_0, \mathsf{sk}_{\mathcal{I}_0})$ *or* $(\mathcal{I}_1, \mathsf{sk}_{\mathcal{I}_1})$ *is used to produce this transcript.*

In our scheme, the transcript π *between* Pr *and* Vf *consists of the obfuscation of function* $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ *and a string* β' *which is the output of the obfuscation. If the input secret key of obfuscated* $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ *is a valid one, the string* β' *should be equal to the string* β *which is preselected by* Vf *and obfuscated in the obfuscation of* $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ *no matter the valid secret key belongs to which user. So in the view of adversary, transcripts generated from* $(\mathcal{I}_b, \mathsf{sk}_{\mathcal{I}_b})$*, for* $b = 0$ *or* 1*, is identical. In other words, the view of the adversary is completely independent to* b*. Thus, the probability for an adversary winning the game is the same as random guessing. So the advantage for the adversary in game* $\mathsf{Game}_{\mathit{anon}}$ *is always negligible. In other words, our scheme is secure according to Definition 2.3.1.*

Theorem 3.3.2 *Our ad hoc anonymous identification scheme is sound according to Definition 2.3.2.*

Proof 2 *We describe the proof using the game-hoping technique with two games, where* *the first game is the original soundness game defined in Section 2.3.1. We prove that for a polynomial time adversary $\mathcal{A}$, the probability that the advantage of $\mathcal{A}$ in the first game is close to that in the second game. Next, we show that $\mathcal{A}$ wins the second game with negligible probability. The two games are defined below:*

Game 1: *The first game is identical to the original soundness game described in 2.3.1.*

Game 2: *The second game has the following steps:*

Setup. *This is the same as in* ***Game 1****. System parameter* param *is generated by algorithm* Setup *and passed to adversary $\mathcal{A}$.*

Query. *$\mathcal{A}$ will issue three types of query:*

- $\mathsf{Register}(\mathcal{I}, \omega)$. *When $\mathcal{A}$ issues* $\mathsf{Register}$*-queries, if $\omega = \perp$, a string will be randomly sampled from secret key space as $\mathsf{sk}_{\mathcal{I}}$. $\mathcal{I}$ is added to $\mathcal{L}_H$. Otherwise, $\mathcal{C}$ sets $\mathsf{sk}_{\mathcal{I}} := \omega$ adds $\mathcal{I}$ to $\mathcal{L}_C$.*

- $\mathsf{Corrupt}(\mathcal{I})$. *$\mathcal{A}$ submits user identity $\mathcal{I}$. If $\mathcal{I}$ is included in $\mathcal{L}_H$, $\mathcal{C}$ returns to $\mathcal{A}$ the corresponding $\mathsf{sk}_{\mathcal{I}}$ and moves $\mathcal{I}$ from $\mathcal{L}_H$ to $\mathcal{L}_C$. And meanwhile, $\mathcal{R}(\delta, \mathsf{sk}_{\mathcal{I}})$ will be programmed as $a_{\mathcal{I}}||(c_{\mathcal{I}} \oplus \beta)$ for $(\delta, a_{\mathcal{I}}, c_{\mathcal{I}})$ from all queries of π in* $\mathsf{Trans}(\mathcal{I}, \mathcal{L}_{\mathcal{I}})$ *which $\mathcal{L}_{\mathcal{I}}$ includes $\mathcal{I}$.*

- $\mathsf{Trans}(\mathcal{I}, \mathcal{L}_{\mathcal{I}})$. *$\mathcal{A}$ chooses a set of users $\mathcal{L}_{\mathcal{I}} \subset \mathcal{L}_C \cup \mathcal{L}_H$ and user $\mathcal{I} \in \mathcal{L}_{\mathcal{I}}$ to obtain an identification transcript. Parse $\mathcal{L}_{\mathcal{I}} := \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell\}$. When $\mathcal{A}$ queries for the transcription, β is randomly chosen from $\{0,1\}^{s(\lambda)}$ and obfuscation of $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ is constructed by randomly choosing $a_{\mathcal{I}_1}, a_{\mathcal{I}_2}..., a_{\mathcal{I}_\ell}$ from $\{0,1\}^{2\lambda}$, $c_{\mathcal{I}_1}, c_{\mathcal{I}_2}..., c_{\mathcal{I}_\ell}$ from $\{0,1\}^{s(\lambda)}$ and δ from $\{0,1\}^{\lambda}$. Transcript π is constructed by β and the obfuscation of $f_{\mathcal{L}_{\mathsf{sk}},\beta}$. π then will be sent to $\mathcal{A}$. $\mathcal{R}(\delta, \mathsf{sk}_{\mathcal{I}_i})$ will be programmed as $a_{\mathcal{I}_i}||(c_{\mathcal{I}_i} \oplus \beta)$ for $\{\mathsf{sk}_{\mathcal{I}_i} | i = 1...\ell, \mathcal{I}_i \in \mathcal{L}_C\}$*

and $\mathcal{R}(\delta, \mathsf{sk}_{\mathcal{I}_i})$ will remain unprogrammed for the rest $\mathsf{sk}_{\mathcal{I}_i}$.

Challenge. *$\mathcal{A}$ chooses a set of user identities $\mathcal{L}^* \subset \mathcal{L}_H$, $\mathcal{L}^* := \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell\}$ on which it wishes to be challenged. $\mathcal{A}$ will receive the obfuscation of function $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ in which δ, $a_{\mathcal{I}_i}$ and $c_{\mathcal{I}_i}$ are randomly chosen from $\{0,1\}^\lambda$, $\{0,1\}^{2\lambda}$ and $\{0,1\}^{s(\lambda)}$ $(i = 1 \ldots \ell)$. Next, $\mathcal{A}$ players the role of prover when receiving the obfuscation. We says that $\mathcal{A}$ wins the game if and only if the output string from $\mathcal{A}$ equals to the randomly chosen β in $\{0,1\}^{s(\lambda)}$ by $\mathcal{C}$.*

*To adversary $\mathcal{A}$, **Game 2** is identical to the original game. In original obfuscation of $f_{\mathcal{L}_{\mathsf{sk}},\beta}$, $a_{\mathcal{I}_i}$, $b_{\mathcal{I}_i}$ are all generated from random oracle and $c_{\mathcal{I}_i} = b_{\mathcal{I}_i} \oplus \beta$, of which the distribution to adversary $\mathcal{A}$ is the same as randomly choosing $a_{\mathcal{I}_i}$, $c_{\mathcal{I}_i}$ from $\{0,1\}^{2\lambda}$, $\{0,1\}^{s(\lambda)}$. Since in the original obfuscation, $a_{\mathcal{I}_i}$ is computed by $\mathcal{R}_1(\delta, \mathsf{sk}_{\mathcal{I}_i})$ and $c_{\mathcal{I}_i}$ is computed by $\mathcal{R}_2(\delta, \mathsf{sk}_{\mathcal{I}_i}) \oplus \beta$. After programming the output of $\mathcal{R}(\delta, \mathcal{I}_i)$ to $a_{\mathcal{I}_i}||(c_{\mathcal{I}_i} \oplus \beta)$, adversary $\mathcal{A}$ will not notice the secret key $\mathsf{sk}_{\mathcal{I}_i}$ he/she owing is not used to construct obfuscation.*

The only possible way for $\mathcal{A}$ to behalf differently in Game 1 and Game 2 is that $\mathcal{A}$ queries the random oracle $\mathcal{R}(.)$ on any secret key in $\mathcal{L}_H$. Assume $\mathcal{A}$ queries the random oracle q times, the probability for this happening is:

$$\Pr \leqslant \frac{q\ell}{2^\lambda},$$

which is negligible (ℓ is the number of user identity in $\mathcal{L}_H$).

*Since in **Game 2**, adversary $\mathcal{A}$ gains no knowledge related to secret keys of users he/she chose to attack during queries for transcript π and user secret keys. Besides, $a_{\mathcal{I}_i}$ and $c_{\mathcal{I}_i}$ in the challenge obfuscation are chose at random and has no relation with either $\mathsf{sk}_{\mathcal{I}_i}$ or*

β. Thus, the probability for $\mathcal{A}$ to successfully win Game 2 is:

$$\Pr = \frac{1}{2^{s(\lambda)}},$$

which is negligible.

To sum up, the probability for adversary $\mathcal{A}$ to win Game 1 is no more than $\frac{1}{2^{s(\lambda)}} + \frac{ql}{2^\lambda}$ which is also negligible. In other words, our scheme is secure according to Definition 2.3.2.

Discussions In our construction, the set of eligible users, $\mathcal{L}_\mathcal{I}$, has to be agreed by both the prover and the verifier. For maximum user privacy protection, $\mathcal{L}_\mathcal{I}$ can be chosen as the set of all eligible users. We would like to remark that our protocol is not a 3-move Σ-protocol. Consequently, we cannot employ the classical results that turns an honest verifier zero-knowledge protocol into a full zero-knowledge protocol [46]. In particular, a malicious server could obfuscate a "wrong" program so that each user secret key will output a different β. In other words, an active malicious verifier could break the anonymity of the scheme. We outline a solution to mitigate this attack. Specifically, the server also publishes $H(\beta)$ for each authentication request. User can abort if the hash of the output from the obfuscated program is not equivalent to the published hash value. This, however, still do not prevent the selective-failure attack as the server can just put one valid secret key inside the program to be obfuscated. This appears to be an inherent limitation in the symmetric key setting, since an honest user has no way to ensure other user identities included in the group are legitimate. Our protocol is, thus, suitable for applications where the verifier is "honest-but-curious".

3.3.2 Efficiency Analysis

We provide an empirical analysis of the efficiency of our proposal and compare it with existing anonymous identification schemes. We first provide a breakdown of the time cost of the schemes based on the number of exponentiation operations (EXP) and pairing operations (PAIR). We remark that PAIR is usually regarded as an expensive operation in comparison with EXP, which in turn takes significantly more time compared with the evaluation of a hash function[1]. Assuming there are ℓ members in the ad hoc group, the breakdown of the major operations for various schemes is summarized in Table 3.1. In this table, $\mathbf{EXP}_p$ and $\mathbf{EXP}_v$ represents the number of exponentiation operation done by prover and verifier. $\mathbf{PAIR}_p$ and $\mathbf{PAIR}_v$ represents the number of pairing operation done by prover and verifier. We do not consider hash operations as a part of the time complexity since compared with exponentiation and pairing operations, the time of evaluating a hash function on a short input is rather insignificant. Since our scheme does not include any pairing and exponentiation operation, the row of our scheme in Table 3.1 has 0 operation. Comparing with [96], which also do not require exponentiation or pairing operations, we are still more efficient. In [96], there are $n - 1$ modular squaring, addition and hash operations, plus 1 square-root operation on the prover side, n modular squaring and hash operations on the verifier side. In our scheme there are n hash operations on the prover side and n hash and addition operations on the verifier side. Since modular squaring is an expensive operation comparing with hash evaluation with short input. In other words, our scheme is still more efficient compared with the state-of-the-art.

To give an estimate of the actual running time of these schemes, we instantiate the numbers based on the benchmark results from [50]. The numbers are obtained, as per [50], from a PIV 3-GHZ processor with 512-MB memory and a Windows XP operation system. Running time for these operations is obtained by using a standard cryptographic library MIRACL [88]. The benchmark results of each major operation is shown in Table 3.2.

Based on these numbers, we can calculate the approximate time for each scheme. The

[1] The exception is to hash an arbitrary string into a point on an elliptic curve group, which could be expensive or impossible to achieve depending on the underlying group structure.

results are presented in Table 3.1.

It is obvious that, as long as the ad hoc group size of our scheme is not large (say, less than 50000), our scheme is more efficient than all the other schemes in the literature. The threshold can be easily calculated from Table 3.3. The space complexity of our scheme is linearly in the group size too, since the obfuscation of $f_{\mathcal{L}_{\mathsf{sk}},\beta}$ should contain all the obfuscated user secret keys in the ad hoc group.

We argue that in practice, however, our scheme is more space-efficient than most existing schemes. For a list of ℓ user identities, the total number of bits to be transmitted is $\ell * (4\lambda) + \lambda$, assuming we use $\mathcal{I}$ to label each a_i, c_i in the obfuscated point function[2]. This number is, in fact, less than the public-key based scheme if we take into account the list of public keys to be transmitted. Specifically, for a ring signature of ℓ ring members, we need to transmit ℓ public keys and ℓ certificates. This is almost certainly larger than that of $2\ell\lambda$. Concrete numbers are illustrated in Table 3.3.

We would like to remark again that our construction is applicable to a less general scenario. Specifically, anonymous identification scheme based on ring signatures allows a signer to convince all verifiers that the former belongs to a group while in our system, we consider a single verifier. Having said that, the efficiency of our scheme allows it to be used on lightweight devices and applied in system login scenario we mentioned above in which a user only needs to confirm his/her identity with a single server. Lastly, we would like to remark that similar to the other systems in the random oracle model, our protocol is also of two rounds.

[2] This will allow a prover to identify directly which index should he use, and thus reduce the evaluation of the obfuscated function to the computation of a single hash value.

Scheme	$\mathbf{EXP}_p$	$\mathbf{EXP}_v$	$\mathbf{PAIR}_p$	$\mathbf{PAIR}_v$
Rivest-Shamir-Tauman [85]	ℓ	ℓ	0	0
Abe-Ohkubo-Suzuki [1]	ℓ	ℓ	0	0
Dodis-Kiayias-Nicolosi-Shoup [30]	8	14	0	0
Nguyen [79]	13	10	0	2
Chow-Liu-Wei-Yuen [28]	ℓ	ℓ	0	0
Shacham-Waters [87]	$4\ell + 3$	0	0	$2\ell + 3$
Chandran-Groth-Sahai [25]	$\frac{\ell+1}{3} + 6\sqrt{\ell}$ $+5$	1	0	$l + 7\sqrt{\ell}$ $+5$
Liu-Au-Susilo-Zhou [65]	offline $\ell - 1$ online 1	0 0	0 0	ℓ ℓ
Xiu-Wu-Liu-Chen[96]	0	0	0	0
Our scheme	0	0	0	0

Table 3.1: Number of operations for each scheme ($\mathbf{EXP}_p$ and $\mathbf{EXP}_v$ represents the number of exponentiation operation done by Pr and Vf. $\mathbf{PAIR}_p$ and $\mathbf{PAIR}_v$ represents the number of pairing operation done by Pr and Vf. ℓ is the group size)

operation	pairing	exponentiation	hash
time	20.04	5.31	< 0.001

Table 3.2: Running time for each operation (in millisecond) [50]

scheme	Prover	Verifier
[85]	5.31ℓ	5.31ℓ
[1]	5.31ℓ	5.31ℓ
[30]	42.48	74.34
[79]	69.03	93.18
[28]	5.31ℓ	5.31ℓ
[87]	21.24ℓ+15.93	40.08ℓ+60.12
[25]	$1.77\ell + 31.86\sqrt{\ell} + 28.32$	$20.04\ell + 140.28\sqrt{\ell} + 105.51$
[65]	offline:5.31(ℓ-1) online:5.31	20.04ℓ 20.04ℓ
ours	$< 0.001\ell$ < 0.001 (optimized version)	$< 0.001\ell$ $< 0.001\ell$

Table 3.3: Operation time for each scheme (in millisecond)

Chapter 4

Practical Signatures from the Partial Fourier Recovery Problem Revisited

In this chapter, we present a lattice-based signature scheme, PASS$_G$, based on PASS$_{RS}$ which was introduced by Hoffstein et al. [54]. Same as PASS$_{RS}$, security of our construction is based on the hardness of a variant of SIS problem and partial Fourier recovery problem. PASS$_G$ improves PASS$_{RS}$ in two aspects. Firstly, unlike PASS$_{RS}$, PASS$_G$ comes with a reduction proof and is thus provably secure. Secondly, we adopt rejection sampling technique introduced by Lyubashevsky [72] to reduce the signature size and improve the efficiency. The generated signatures of PASS$_G$ are Gaussian-distributed, not uniform-distributed like PASS$_{RS}$, and is more space efficient. We also present another security parameter set based on best known attack using BKZ 2.0 algorithm introduced by Chen and Nguyen [27].

4.1 Overview

We briefly sketch our proposed PASS$_G$ and discuss its improvements over the original scheme PASS$_{RS}$. In the following we will be using a similar notation as in [53].

We employ a polynomial ring $\mathcal{R}_q := \mathbb{Z}[x]/(x^n + 1)$. The secret key of the proposed scheme is a polynomial $\mathbf{f} \in \mathcal{R}_q$ with small coefficients, and the public key $\hat{\mathbf{f}}|_\Omega = \mathcal{F}_\Omega \mathbf{f}$ is a partial discrete Fourier transform of $\mathbf{f}$ where $\mathcal{F}_\Omega$ is the restriction of the Fourier transform matrix. To sign a padded message μ, the signer first samples a polynomial $\mathbf{y} \in \mathcal{R}_q$ according to a discrete Gaussian distribution $\mathcal{D}$. Then the signer computes a short challenge polynomial $\mathbf{c} \in \mathcal{R}_q = \mathsf{Hash}(\mathcal{F}_\Omega \mathbf{y}, \mu)$ and the potential signature $\mathbf{z} = \mathbf{f} \star \mathbf{c} + \mathbf{y}$ where $\star$ is the polynomial multiplication over the ring. The pair of $\mathbf{z}$ and $\mathbf{c}$ will be output as the signature. However, the distribution of $(\mathbf{z}, \mathbf{c})$ is affected by, and may leak information of, secret key $\mathbf{f}$. To decouple the distribution of signature from secret key, we apply rejection sampling technique and output $(\mathbf{z}, \mathbf{c})$ with some probability.

Before we proceed further, let us briefly recall the notion of rejection sampling. The rejection sampling method allows one to obtain a target probability distribution f from another probability distribution g. Specifically, instead of directly sampling from f, one can draw sample x from g and accept it with probability $\frac{f(x)}{Mg(x)}$, where M is a constant satisfying $1 \leq M < \infty$ and for all valid x, $f(x) \leqslant Mg(x)$. The distribution produced by rejection sampling will be the same as f and the expected number of iterations to successfully generate a sample will be M.

In our scheme, we use a discrete Gaussian distribution to generate $\mathbf{y}$. This is in analogy to the target probability distribution f as in the above example. On the other hand, we view $\mathbf{z} = \mathbf{f} \star \mathbf{c} + \mathbf{y}$ as the proposal distribution g. By choosing some appropriate constant value M and accepting $\mathbf{z}$ with probability $\frac{f(x)}{Mg(x)}$, the signing algorithm will eventually produce a signature pair $(\mathbf{z}, \mathbf{c})$ which is independent of signing key $\mathbf{f}$. We remark that in order to achieve a meaningful scheme, the quantity M needs to be polynomially bounded. In practice, it is common to set it as a constant.

To verify signature $(\mathbf{z}, \mathbf{c})$ of message μ, the verifier first checks whether the norm bound of $\mathbf{z}$ is "small". This ensures that the signature is genuinely generated from the secret key. Then the verifier need to be convinced that the signature is binded to the message. This step is performed via computing $\mathsf{Hash}(\mathcal{F}_\Omega \mathbf{z} - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega, \mu)$ and checking whether the output of the hash function equals to $\mathbf{c}$. $\odot$ here is the component-wise multiplication.

We highlight the differences between our PASS$_G$ and PASS$_{RS}$. PASS$_{RS}$ samples polynomial $\mathbf{y}$ uniformly with a ℓ_∞ norm $\leqslant t$ and requires $||\mathbf{f} \star \mathbf{c}||_1 \leq b$ for some parameter b. The signing algorithm only outputs $\mathbf{z}$ when $||\mathbf{z}||_\infty \leqslant t - b$. To guarantee the acceptance rate and the security of the scheme, t is between 2^{12} and 2^{15} which leads to quite a large signature size. In our work, we show that, by sampling $\mathbf{y}$ from a discrete Gaussian distribution and choosing parameters appropriately, we can both reduce the size of $\mathbf{z}$ and increase the acceptance rate. Besides, in section 4.4, we prove the strong unforgeability of the proposed scheme and reduce the security of it to the hardness of a special kind of SIS problem, namely, Vandermonde-SIS problem. We remark no reduction proof is given for PASS$_{RS}$. Moreover, as mentioned above, the security level of PASS$_{RS}$ using its suggested parameter is questionable thanks to improved cryptanalysis techniques in [4], where the core sub-routing, known as the enumeration methods with extreme pruning [42], is replaced by a quantum sieving algorithm [11] that drastically improves the cost of the BZK 2.0 algorithm [27]. To this end, we give new parameter sets that are robust against this new analysis.

4.2 Hardness Assumption

Before introducing the hard problem in our construction, we first introduce the partial Fourier recovery problem which requires recovering a signal from a restricted number of its Fourier coefficients.

Let ω be the primitive Nth root of -1 module q. We define the discrete Fourier transform over $\mathbb{Z}_q$ to be the linear transformation $\mathbf{F}\mathbf{f} = \hat{\mathbf{f}} : \mathbb{Z}_q^n \to \mathbb{Z}_q^n$ given by

$$(\mathbf{F})_{i,j} = \omega^{ij}$$

The Fourier transform matrix $\mathbf{F}$ is a Vandermonde matrix

$$\mathbf{F} = \begin{bmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \dots & \omega^{0\cdot(n-1)} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & \dots & \omega^{1\cdot(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{(n-1)\cdot 0} & \omega^{(n-1)\cdot 1} & \dots & \omega^{(n-1)\cdot(n-1)} \end{bmatrix}$$

Let $\mathbf{F}_\Omega$ be the restriction of $\mathbf{F}$ to the set of t rows specified by an index set, Ω,

$$(\mathbf{F}_\Omega)_{ij} = \omega^{\Omega_i j}$$

The partial Fourier recovery problem is that, given an evaluation $\hat{\mathbf{f}}|_\Omega \in \mathbb{Z}_q^t$, find $\mathbf{x}$ with small norm such that $\hat{\mathbf{x}}|_\Omega = \hat{\mathbf{f}}|_\Omega \pmod q$. The solution $\mathbf{x}$ is required to be small since one can easily find a large $\mathbf{x}$ such that $\hat{\mathbf{x}}|_\Omega = \hat{\mathbf{f}}|_\Omega$. This problem has been well studied and considered to be hard in general. Since partial transform matrix $\mathbf{F}_\Omega$ is a restriction of Fourier transform matrix $\mathbf{F}$, the image $\hat{\mathbf{f}}|_\Omega$ is expected to contain little information about the hidden Fourier coefficients. The only possible way to recover $\mathbf{f}$ with small norm is by combinatorial optimization procedure which is very expensive. The best way to solve partial Fourier recovery problem is by solving the closest vector problem in the lattice $\Lambda^\perp(\mathbf{F}_\Omega)$, where

$$\Lambda^\perp(\mathbf{F}_\Omega) = \{\mathbf{a} \in \mathbb{Z}_q^n : \mathbf{F}_\Omega \mathbf{a} = \mathbf{0}\}.$$

The closest vector problem is defined as given vector $\mathbf{y} \in \mathbb{Z}_q^n$, find a lattice point

$\mathbf{x} \in \Lambda^{\perp}(\mathbf{F}_{\Omega})$ such that $\|\mathbf{y} - \mathbf{x}\|_{\infty} \leq \beta$. Since we have $\mathbf{F}_{\Omega}(\mathbf{y} - \mathbf{x}) = \mathbf{F}_{\Omega}\mathbf{y} = \hat{\mathbf{y}}|_{\Omega}$, if we can find such lattice point $\mathbf{x}$, we can find vector $\mathbf{y}' = \mathbf{y} - \mathbf{x}$ with small norm such that $\mathbf{F}_{\Omega}\mathbf{y}' = \hat{\mathbf{y}}|_{\Omega}$.

We note that to date, there is no known reduction from lattice-based hard problem to *partial Fourier recovery* problem. However, finding a short preimage by a given evaluation and a transform matrix $\mathcal{F}_{\Omega}$ is known to be related to solving the Short Integer Solution (SIS) problem and the Inhomogeneous Short Integer Solution (ISIS) problem, two average-case hard problems which are frequently used in lattice-based cryptography constructions.

The main difference between SIS (ISIS) and the partial Fourier recovery problem is that public matrix $\mathbf{A}$ in SIS and ISIS problems are uniformly chosen, while public matrix $\mathcal{F}_{\Omega}$ in *partial Fourier recovery* problem is a restriction of discrete Fourier transform. So we define a new average-case problem called Vandermonde-SIS problem.

Definition 4.2.1 ($\mathbf{Vandermonde - SIS}^{\mathcal{K}}_{q,t,n,\beta}$ problem) *Given a Vandermonde matrix $\mathcal{F}_{\Omega} \in \mathbb{Z}_q^{t \times n}$ drawn according to some distribution $\mathcal{K}$, find a non-zero $\mathbf{v} \in \mathbb{Z}_q^n$ such that $\mathcal{F}_{\Omega}\mathbf{v} = \mathbf{0}$ and $\|\mathbf{v}\| \leq \beta$.*

The distribution $\mathcal{K}$ here refers to randomly samples t rows from discrete Fourier transform matrix $\mathbf{F}$.

We emphasize that we are not aware if there exists any reduction from a worst-case lattice problem to Vandermonde-SIS. Here we assume that the hardness of SIS problem is not relied on the structure of the public matrix $\mathbf{A}$ and the Vandermonde-SIS problem is hard in average-case as in SIS and ISIS problem. In the following section, we are going to prove that the security of our proposed signature scheme is based on the assumed average-case hardness of the Vandermonde-SIS problem.

Parameter	Definition
n	Dimension
q	Prime ($\equiv -1 \mod n$)
ω	A primitive n^{th} root of -1 in $\mathbb{Z}_q$
Ω	A subset of $\{\omega^i : 1 \leq i \leq n-1\}$
t	$\lvert\Omega\rvert$ (size of Ω)
σ	Standard deviation of discrete normal distribution
κ	s.t. $2^\kappa \cdot \begin{pmatrix} n \\ \kappa \end{pmatrix} \geq 2^{128}$

Table 4.1: Parameter List for PASS$_G$

4.3 Our Construction

In this section, we are going to present the construction of PASS$_G$. Parameters and their definitions are listed in Table 1.

KeyGen: This algorithm generates polynomial $\mathbf{f} \in \mathcal{R}_q$ with each coefficient independently and uniformly sampled from $\{-1, 0, 1\}$ as the secret key. The corresponding public key is $\hat{\mathbf{f}}|_\Omega = \mathcal{F}_\Omega \mathbf{f}$. As described in section 4.2, $\mathcal{F}_\Omega$ is the restriction of $\mathcal{F}$ to the set of t rows. Thus, $\mathcal{F}_\Omega$ can be generated by randomly picking t rows from the original Fourier transform matrix $\mathcal{F}$.

Signing($\mathbf{f}, \mu$): To sign message μ, the signer first randomly samples polynomial $\mathbf{y}$ from discrete Gaussian distribution D_σ^n and computes $\hat{\mathbf{y}}|_\Omega = \mathcal{F}_\Omega \mathbf{y}$. The signer then computes challenge $\mathbf{c} = \mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{y}}|_\Omega, \mu))$ where FormatC and Hash are two public algorithms such that:

$$\mathsf{Hash} : \mathbb{Z}_q^t \times \{0,1\}^* \to \{0,1\}^\ell,$$

Original Signing algorithm

```
Signing($\mathbf{f}, \mu, \mathcal{F}_\Omega$)
1: $\mathbf{y} \leftarrow D_\sigma^n$
2: $\mathbf{c} = \mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{y}}|_\Omega, \mu))$
3: $\mathbf{z} \leftarrow \mathbf{f} \star \mathbf{c} + \mathbf{y}$
4: with probability $\min(\frac{D_\sigma^n(\mathbf{z})}{MD_{\mathbf{f}\star\mathbf{c},\sigma}^n(\mathbf{z})}, 1)$
5:   output $(\mathbf{z}, \mathbf{c})$
```

Figure 4.1: Original Signing algorithm in PASS$_G$

$$\mathsf{FormatC} : \{0,1\}^\ell \to \{\mathbf{v} : \mathbf{v} \in \{-1,0,1\}^n, \|\mathbf{v}\|_1 \le \kappa\}.$$

Finally, the signer computes $\mathbf{z} = \mathbf{f} \star \mathbf{c} + \mathbf{y}$ and outputs $(\mathbf{z}, \mathbf{c})$ with probability $\min(\frac{D_\sigma^n(\mathbf{z})}{MD_{\mathbf{f}\star\mathbf{c},\sigma}^n(\mathbf{z})}, 1)$ where $M = \exp(\frac{28\alpha+1}{2\alpha^2})$ and $\sigma = \alpha \cdot \kappa\sqrt{N}$.

Verification($\mu, \mathbf{z}, \mathbf{c}, \mathcal{F}_\Omega, \hat{\mathbf{f}}|_\Omega$): The verifier accepts the signature if and only if $\|\mathbf{z}\| \le k\sigma\sqrt{n}$ and $\mathbf{c} = \mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega, \mu))$.

4.4 Security Analysis

The security of PASS$_G$ is based on the average-case hardness of the **Vandermonde -SIS**$_{q,t,n,\beta}^{\mathcal{K}}$ problem in the random oracle model. In another words, if there is any forger against the PASS$_G$ signature scheme, we can use the forger to solve **Vandermonde-SIS**$_{q,t,n,\beta}^{\mathcal{K}}$ problem for $\beta = 2k\sigma\sqrt{n} + 2\kappa\sqrt{n}$.

Theorem 4.4.1 *Assume there is a polynomial-time forger who can successfully forge a PASS$_G$ signature with non-negligible probability δ by making at most s queries to the signing oracle and h queries to the random oracle* $\mathsf{FormatC} \circ \mathsf{Hash}$. *Then, there exits a polynomial-time algorithm which can solve the* $\mathbf{Vandermonde} - \mathbf{SIS}_{q,t,n,\beta}^{\mathcal{K}}$ *problem for*

```
Hybrid 1
Signing($\mathbf{f}, \mu, \mathcal{F}_\Omega$)
1: $\mathbf{y} \leftarrow D_\sigma^n$
2: $\mathbf{c} \leftarrow_\$ \{\mathbf{v} : \|\mathbf{v}\|_1 \le \kappa\}$
3: $\mathbf{z} \leftarrow \mathbf{f} \star \mathbf{c} + \mathbf{y}$
4: with probability $\min(\frac{D_\sigma^n(\mathbf{z})}{MD^n_{\mathbf{f}\star\mathbf{c},\sigma}(\mathbf{z})}, 1)$
5:   output $(\mathbf{z}, \mathbf{c})$
6:   Program FormatC(Hash($\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega, \mu$)) $= \mathbf{c}$
```

Figure 4.2: Hybrid 1

```
Hybrid 2
Signing($\mathbf{f}, \mu, \mathcal{F}_\Omega$)
1: $\mathbf{c} \leftarrow_\$ \{\mathbf{v} : \|\mathbf{v}\|_1 \le \kappa\}$
2: $\mathbf{z} \leftarrow D_\sigma^N$
3: with probability $\frac{1}{M}$
4:   output $(\mathbf{z}, \mathbf{c})$
5:   Program FormatC(Hash($\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega, \mu$)) $= \mathbf{c}$
```

Figure 4.3: Hybrid 2

$\beta = 2k\sigma\sqrt{n} + 2\kappa\sqrt{n}$ with probability

$$\frac{1}{2}(1-2^{-128})\left(\delta - \frac{1}{2^\kappa \cdot \begin{pmatrix} n \\ \kappa \end{pmatrix}}\right)\left(\frac{\delta - 1/\left(2^\kappa \cdot \begin{pmatrix} n \\ \kappa \end{pmatrix}\right)}{s+h} - \frac{1}{2^\kappa \cdot \begin{pmatrix} n \\ \kappa \end{pmatrix}}\right) \approx \frac{\delta^2}{2(h+s)}$$

Proof 3 *This theorem will be proved by Lemma 4.4.2 and 4.4.3. Lemma 4.4.2 shows that the statistical distance between the original signing algorithm in Fig.4.1 and Hybrid 2 in Fig.4.3 is negligible. In another words, the signing algorithm in PASS$_G$ can be replaced by Hybrid 2 and the statistical distance of the two outputs is negligible. Lemma 4.4.3 shows that if there is a forger who can produce a valid signature for PASS$_G$ with probability δ when the signing algorithm is from Hybrid 2, with probability $\approx \frac{\delta^2}{2(h+s)}$, we can use this forger to solve the* $\mathbf{Vandermonde}-\mathbf{SIS}^{\mathcal{K}}_{q,t,n,\beta}$ *for $\beta = 2k\sigma\sqrt{n} + 2\kappa\sqrt{n}$.*

Lemma 4.4.1 *For any Vandermonde matrix $\mathcal{F}_\Omega \in \mathbb{Z}_q^{t\times n}$ where $n > 81 + t \cdot \frac{\log q}{\log 3}$,* *for randomly choose* $\mathbf{f} \xleftarrow{\$} \{-1,0,1\}^n$*, with probability* $1-2^{-128}$ *there exists another* $\mathbf{f}' \in \{-1,0,1\}^n$ *such that* $\mathcal{F}_\Omega \mathbf{f} = \mathcal{F}_\Omega \mathbf{f}'$*.*

Proof 4 *We consider* $\mathcal{F}_\Omega$ *as a linear transformation whose output range has size* q^t*. Thus, there are at most* $q^t - 1$ *elements in* $\{-1,0,1\}^n$ *do not collide with other elements. The probability of randomly choosing an element from* $\{-1,0,1\}^n$ *which does not collide with the remaining elements is:*

$$\frac{q^t-1}{3^n} < \frac{q^t}{3^{81+t\cdot\frac{\log q}{\log 3}}} = \frac{1}{3^{81}} < 2^{-128}$$

.

Lemma 4.4.2 *Assume there is a distinguisher* $\mathcal{A}$ *who can query either the original signing algorithm from PASS*$_G$ *or the signing algorithm from Hybrid 2 and the random oracle* $\mathsf{FormatC} \circ \mathsf{Hash}$*. If* $\mathcal{A}$ *queries the random oracle at most* h *times and signing algorithm at most* s *times, the advantage for* $\mathcal{A}$ *to distinguish the original signing algorithm from Hybrid 2 is at most* $s(s+h)2^{-t} + s \cdot \frac{2^{-128}}{M}$*.*

Proof 5 *Different from the original signing algorithm, in Hybrid 1, we first randomly choose* $\mathbf{c}$ *from* $\{\mathbf{v} : \|\mathbf{v}\|_1 \le \kappa\}$ *and then program* $\mathbf{c}$ *as the answer to the random oracle* $\mathsf{FormatC} \circ \mathsf{Hash}$ *query* $\mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega, \mu)) = \mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{y}}|_\Omega, \mu))$*. Since we program the random oracle without checking whether the value for* $(\hat{\mathbf{y}}|_\Omega, \mu)$ *has already been programmed yet. So, the only chance for a distinguisher* $\mathcal{A}$ *distinguish the original signing algorithm from the one in Hybrid 1 is that, Hybrid 1 generates a* $\mathbf{y}$ *such that* $\hat{\mathbf{y}}|_\Omega$ *already showed in the previous queries. Since* $\mathcal{F}_\Omega$ *is a Vardemonde matrix and its columns are linearly independent. Thus,* $\mathcal{F}_\Omega$ *can always be written in the "Hermite*

Normal Form" as $\mathcal{F}_\Omega = [\bar{\mathcal{F}}_\Omega \| \mathbf{I}]$. According to Lemma.2.2.1, for any $\mathbf{v} \in \mathbb{Z}_q^t$, we have

$$\begin{aligned}\Pr[\mathcal{F}_\Omega \mathbf{y} = \mathbf{v}; \mathbf{y} \leftarrow D_\sigma^n] &= \Pr[\mathbf{y}_1 = (\mathbf{v} - \bar{\mathcal{F}}_\Omega \mathbf{y}_0); \mathbf{y} \leftarrow D_\sigma^n] \\ &\leq \max_{\mathbf{v}' \in \mathbb{Z}_q^M} \Pr[\mathbf{y}_1 = \mathbf{v}'; \mathbf{y}_1 \leftarrow D_\sigma^t] \\ &\leq 2^{-t}.\end{aligned}$$

Thus, the probability for Hybrid 1 sampling $\mathbf{y}$ such that $\hat{\mathbf{y}}|_\Omega$ already showed in the previous queries is at most 2^{-t}. Assume Hybrid 1 is queried s times and the random oracle is queried h times by distinguisher $\mathcal{A}$, the probability for such a collision happening is at most $s(s+h)2^{-t}$. So the advantage for $\mathcal{A}$ to distinguish Hybrid 1 from the original signing algorithm is also at most $s(s+h)2^{-t}$.

We then prove the statistical distance between Hybrid 1 and Hybrid 2 is at most $s \cdot \frac{2^{-128}}{M}$ when we set $M = \exp(\frac{28\alpha+1}{2\alpha^2})$. Since the vector representation of $\mathbf{y}$ is distributed according to D_σ^n, in Hybrid 1, $\mathbf{z}$ is distributed according to $D_{\mathbf{f}\star\mathbf{c},\sigma}^n$, for any $\mathbf{z}^* \in \mathbb{R}^n$, we have:

$$\begin{aligned}\Pr[\mathbf{z} = \mathbf{z}^*] &= D_{\mathbf{f}\star\mathbf{c},\sigma}^n(\mathbf{z}^*) \\ &= \frac{\rho_{\mathbf{f}\star\mathbf{c},\sigma}(\mathbf{z}^*)}{\rho_\sigma(\mathbb{Z}^n)} \\ &= \frac{1}{\rho_\sigma(\mathbb{Z}^n)} \exp(-\frac{\|\mathbf{z}^* - \mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}) \\ &= \frac{1}{\rho_\sigma(\mathbb{Z}^n)} \exp(-\frac{\|\mathbf{z}^*\|^2}{2\sigma^2} - \frac{-2\langle \mathbf{z}^*, \mathbf{f}\star\mathbf{c}\rangle + \|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}) \\ &= D_\sigma^n(\mathbf{z}^*) \exp(-\frac{-2\langle \mathbf{z}^*, \mathbf{f}\star\mathbf{c}\rangle + \|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}).\end{aligned}$$

We have:

$$\frac{D_\sigma^n(\mathbf{z}^*)}{D_{\mathbf{f}\star\mathbf{c},\sigma}^n(\mathbf{z}^*)} = \frac{D_\sigma^n(\mathbf{z}^*)}{D_\sigma^n(\mathbf{z}^*)\exp(-\frac{-2\langle\mathbf{z}^*,\mathbf{f}\star\mathbf{c}\rangle+\|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2})}$$
$$= \exp(\frac{-2\langle\mathbf{z}^*,\mathbf{f}\star\mathbf{c}\rangle+\|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}).$$

According to Lemma.2.2.1, when $r = 14\|\mathbf{v}\|\sigma$, with probability at least $1 - 2^{-128}$ we have $|\langle\mathbf{z}^*, \mathbf{f}\star\mathbf{c}\rangle| < 14\|\mathbf{f}\star\mathbf{c}\|\sigma$. Then, with probability at least $1 - 2^{-128}$, we have:

$$\exp(\frac{-2\langle\mathbf{z}^*,\mathbf{f}\star\mathbf{c}\rangle+\|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}) < \exp(\frac{28\|\mathbf{f}\star\mathbf{c}\|\sigma+\|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}).$$

Since the coefficients of $\mathbf{f}$ are randomly picked from $\{-1, 0, 1\}$ and $\|\mathbf{c}\|_1 \le \kappa$, we have

$$\|\mathbf{f}\star\mathbf{c}\| \le \sqrt{n\cdot\kappa^2} = \kappa\sqrt{N}.$$

Assume $\sigma = \alpha\cdot\kappa\sqrt{n}$. Then,

$$\exp(\frac{28\|\mathbf{f}\star\mathbf{c}\|\sigma+\|\mathbf{f}\star\mathbf{c}\|^2}{2\sigma^2}) \le \exp(\frac{28\kappa\sqrt{n}\sigma+(\kappa\sqrt{n})^2}{2\sigma^2}) = \exp(\frac{28\alpha+1}{2\alpha^2})$$

Since $\frac{D_\sigma^n}{D_{\mathbf{f}\star\mathbf{c},\sigma}^n} < \exp(\frac{28\alpha+1}{2\alpha^2})$ happens with probability at least $1 - 2^{-128}$, when $M = \exp(\frac{28\alpha+1}{2\alpha^2})$, the statistical distance between the outputs of Hybrid 1 and Hybrid 2 is at most $\frac{2^{-128}}{M}$ according to Lemma 2.1.1. Thus, after calling the signing oracle for s times, the statistical distance is no more than $s\cdot\frac{2^{-128}}{M}$.

Lemma 4.4.3 *Assume there is a polynomial-time forger $\mathbf{F}$ who can successfully forge a PASS$_G$ signature with non negligible probability δ by making at most s queries to the* *signer in Hybrid 2 and h queries to the random oracle* FormatC $\circ$ Hash,. *Then, there exits a polynomial-time algorithm who can solve the* $\mathbf{Vandermonde\text{-}SIS}^{\mathcal{K}}_{q,t,n,\beta}$ *problem for* $\beta = 2k\sigma\sqrt{n} + 2\kappa\sqrt{n}$ *with probability at least*

$$\frac{1}{2}(1-2^{-128})\left(\delta - \frac{1}{2^{\kappa}\cdot\begin{pmatrix} n \\ \kappa \end{pmatrix}}\right)\left(\frac{\delta - 1/\left(2^{\kappa}\cdot\begin{pmatrix} n \\ \kappa \end{pmatrix}\right)}{s+h} - \frac{1}{2^{\kappa}\cdot\begin{pmatrix} n \\ \kappa \end{pmatrix}}\right).$$

Proof 6 *The output range of the random oracle* FormatC $\circ$ Hash *is denoted by* $|D_H| = \{\mathbf{v} : \mathbf{v} \in \{-1,0,1\}^N, \|\mathbf{v}\|_1 \le \kappa\} = 2^{\kappa}\cdot\begin{pmatrix} n \\ \kappa \end{pmatrix}$. *Let* $p = s + h$ *be the total number of queries that* $\mathbf{F}$ *makes to the signer and random oracle. Given Vandermonde matrix* $\mathcal{F}_{\Omega} \in \mathbb{Z}_q^{t\times n}$ *drawn from distribution* $\mathcal{K}$*, we pick signing key* $\mathbf{f} \xleftarrow{\$} \{-1,0,1\}^n$ *and release the corresponding public key* $\hat{\mathbf{f}}|_{\Omega} \leftarrow \mathcal{F}_{\Omega} \odot \mathbf{f}$.

We also pick random coins ψ *and* ϕ *for the signer and forger respectively. Besides, we pick* $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_p\} \xleftarrow{\$} D_H$ *as the* t *responses of the random oracle* FormatC $\circ$ Hash. *Assume there is a subroutine denoted by* $\mathcal{S}$ *which takes* ($\mathcal{F}_{\Omega}$,$\hat{\mathbf{f}}|_{\Omega}$, ψ,ϕ,$\mathbf{c}_1$, $\mathbf{c}_2$,...,$\mathbf{c}_p$) *as input. Subroutine* $\mathcal{S}$ *runs as follow:*

- $\mathcal{S}$ *gives public matrix* $\mathcal{F}_{\Omega}$ *and public key* $\hat{\mathbf{f}}|_{\Omega}$ *along with random coins* ϕ *to forger* $\mathbf{F}$ *to initialize* $\mathbf{F}$ *and run* $\mathbf{F}$.

- $\mathcal{S}$ *uses random coins* ψ *to run the signing algorithm in Hybrid 2.*

- *When* $\mathbf{F}$ *issues a query to* $\mathcal{S}$ *to sign a message or to the random oracle* FormatC $\circ$ Hash, *the random oracle will be programmed and the output of the random oracle will be the first* $\mathbf{c}_i \in \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_p\}$ *that has not been used yet.* $\mathcal{S}$ *will record all the queries to the random oracle in a table to maintain the consistency.*

- *Finally, forger* $\mathbf{F}$ *outputs a forgery* $\{(\mathbf{z}, \mathbf{c}), \mu\}$ *such that* $\|\mathbf{z}\| \le k\sigma\sqrt{n}$ *and* $\mathbf{c} =$

$\mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega, \mu))$ *with probability* δ. $\mathcal{S}$ *will output identical* $\{(\mathbf{z}, \mathbf{c}), \mu\}$.

Notice that with probability $1 - \frac{1}{|D_H|}$, $\mathbf{c}$ *output by* $\mathcal{S}$ *will be one of the* $\mathbf{c}_i \in \{\mathbf{c}_1, \cdots, \mathbf{c}_p\}$. *Since if the random oracle was not queried or programmed on some input* $\mathbf{w} = \hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}|_\Omega$, *the probability for* $\mathbf{F}$ *to produce a* $\mathbf{c}$ *such that* $\mathbf{c} = \mathsf{FormatC}(\mathsf{Hash}(\mathbf{w}, \mu))$ *is* $\frac{1}{|D_H|}$. *The probability for* $\mathbf{F}$ *to produce a forgery is* δ. *Thus, the probability for* $\mathbf{F}$ *outputs a forgery* $\{(\mathbf{z}, \mathbf{c}), \mu\}$ *and* $\mathbf{c} = \mathbf{c}_i$ *for some* i *is* $\delta - \frac{1}{|D_H|}$.

Type 1 forgery*: The first type of forgery is that,* $\mathbf{c}_i$ *is a response of random oracle* $\mathsf{FormatC} \circ \mathsf{Hash}$ *on input* $(\mathbf{w}', \mu') = (\hat{\mathbf{z}}|'_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega, \mu')$ *during a signing query. Then we have*

$$\mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega, \mu)) = \mathsf{FormatC}(\mathsf{Hash}(\hat{\mathbf{z}}|'_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega, \mu'))$$

If $\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega \neq \hat{\mathbf{z}}|'_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega$ *or* $\mu \neq \mu'$, *we find a collision of the hash function. Thus, we must have* $\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega = \hat{\mathbf{z}}|'_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega$ *and* $\mu = \mu'$. *We have* $\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{z}}|'_\Omega = \mathcal{F}_\Omega(\mathbf{z} - \mathbf{z}') = \mathbf{0} \mod q$. *Since* $\mathbf{z} \neq \mathbf{z}'$ *and* $\|\mathbf{z}\|, \|\mathbf{z}'\| \leq k\sigma\sqrt{n}$. *We have* $\|\mathbf{z} - \mathbf{z}'\| \leq 2k\sigma\sqrt{n}$.

Type 2 forgery*: The second type of forgery is that,* $\mathbf{c}_i$ *is a response of a random oracle query issued by* $\mathbf{F}$. *We first store the forgery* $\{(\mathbf{z}, \mathbf{c}_i), \mu\}$ *and then picks new* $\mathbf{c}'_i, \cdots, \mathbf{c}'_p \xleftarrow{\$} D_H$. *We then run the subroutine* $\mathcal{S}$ *again on input* $(\mathcal{F}_\Omega, \hat{\mathbf{f}}|_\Omega, \psi, \phi, \mathbf{c}_1, ..., \mathbf{c}_{i-1}, \mathbf{c}'_i, \cdots, \mathbf{c}'_p)$. *According to the General Forking Lemma in [15], we obtain that* $\mathbf{c}'_i \neq \mathbf{c}_i$ *and the forger uses the random oracle response* $\mathbf{c}'_i$ *in its forgery is at least*

$$(\delta - \frac{1}{|D_H|})(\frac{\delta - \frac{1}{|D_H|}}{p} - \frac{1}{|D_H|}).$$

In another words, with the same probability $\mathbf{F}$ *will output a forgery* $\{(\mathbf{z}', \mathbf{c}_i'), \mu\}$ *and* $\hat{\mathbf{z}}|_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|_\Omega = \hat{\mathbf{z}}|'_\Omega - \hat{\mathbf{f}}|_\Omega \odot \hat{\mathbf{c}}_i|'_\Omega$ *. Now we have*

$$\mathcal{F}_\Omega(\mathbf{z} - \mathbf{z}' + \mathbf{f} \star (\mathbf{c}_i' - \mathbf{c}_i)) = 0$$

and $\|\mathbf{z} - \mathbf{z}' + \mathbf{f} \star \mathbf{c}_i' - \mathbf{f} \star \mathbf{c}_i\| \leq 2k\sigma\sqrt{n} + 2\kappa\sqrt{n}$*.*

We can prove that with probability at least $1/2$*,* $(\mathbf{z} - \mathbf{z}' + \mathbf{f} \star (\mathbf{c}_i' - \mathbf{c}_i)) \neq 0$*. According to Lemma 4.4.1, if* $\mathbf{f}$ *is randomly chosen, with probability at least* $1 - 2^{-128}$ *there exists another different* $\mathbf{f}'$ *such that* $\mathcal{F}_\Omega \mathbf{f} = \mathcal{F}_\Omega \mathbf{f}'$*. Which means that for every distinct key* $\mathbf{f}$ *such that* $(\mathbf{z} - \mathbf{z}' + \mathbf{f} \star (\mathbf{c}_i' - \mathbf{c}_i)) = 0$*, there is a distinct key* $\mathbf{f}'$ *such that* $(\mathbf{z} - \mathbf{z}' + \mathbf{f}' \star (\mathbf{c}_i' - \mathbf{c}_i)) \neq 0$*. Since both the subroutine* $\mathcal{S}$ *and the forger* $\mathbf{F}$ *do not know whether we generate a secret key like* $\mathbf{f}$ *or* $\mathbf{f}'$*, with probability at least 1/2 we can obtain a non-zero answer with length at most* $2k\sigma\sqrt{n} + 2\kappa\sqrt{n}$*.*

4.5 Practical Instantiation

In section 4.4, we show the security of our signature scheme is based on the average-case hardness of Vandermonde-SIS problem. In this section, we are going to present a practical instantiation with parameters choosing according to the lattice reduction algorithm BKZ 2.0. This gives us an approach to analyse the security of PASS$_G$ under best known attack. Two sets of parameters with 128-bit security will be presented. Based on the two sets of parameters, we can estimate the rejection rate and signature size of our PASS$_G$.

4.5.1 Parameters

In the following we will give two sets of parameters. Both sets provides 128 bit security against quantum attackers. The first set of parameters provides a similar security level as the original PASS$_{RS}$ signature scheme, and is performance oriented. The second set is security oriented and has a larger build in margin. This is to account for future advance in cryptanalysis.

In the following we will be using the first set of parameters as an example. We use polynomial ring $x^{512} + 1$ with $q = 2^{16} + 1$ that allows for very efficient NTT transformations. This choice of $q > 2\tau\sigma$ ensures that there will be no wraparound when we sample a Gaussian vector.

Then, we choose $\kappa = 44$. That is, there are 44 non-zero coefficients in challenge $\mathbf{c}$ for the first parameter set. The space of challenges will be over 2^{256} for both parameter sets to avoid any (quantum) searching algorithm [48].

The next task is to give a practical bound of $\mathbf{f} \star \mathbf{c}$. Here we further assume that the coefficients of $\mathbf{f} \star \mathbf{c}$ behave as if they are independent. Notice that each coefficient is a sum of κ different ± 1s. As shown in [4], the distribution will be very close to a Gaussian distribution (in terms of Renyi divergence) with a deviation of $\sqrt{\kappa}$. And since we have assumed that each coefficient of $\mathbf{f} \star \mathbf{c}$ behaves independent from the other, we obtain that

$$\|\mathbf{f} \star \mathbf{c}\| \approx \sqrt{\kappa n}.$$

In practice, we can expect $\|\mathbf{f} \star \mathbf{c}\| < 2\sqrt{\kappa N}$ with high probability. Therefore we set the bound for $\|\mathbf{f} \star \mathbf{c}\|$ to $2\sqrt{\kappa n}$, and re-sample a $\mathbf{c}$ if this isn't satisfied. As a result if we set $\sigma = k\sqrt{N\kappa} \approx 2000$, we can expect a repetition rate of $\exp(\frac{2k\kappa\sigma + \kappa^2}{2\sigma^2}) \approx e^2 = 7.4$.

Now we need to estimate the size of the keys and signatures. The public key is $\hat{\mathbf{f}}|_{\Omega}$, a t dimensional vector with coefficients between 0 and q. On the other hand, the signature is a pair $(\mathbf{z}, \mathbf{c})$, where the first vector $\mathbf{z}$ is a Gaussian vector with variance σ, we can use Hoffman encoding, as suggested by [31], that effectively stores the vector with roughly $(\log_2 \sigma + 2)N$ bits. The second vector $\mathbf{c}$ is a sparse binary vector with κ non-zero coefficients, and can be stored efficiently with $\min(\kappa \log_2 n, n)$ bits.

	Parameter 1	Parameter 2
n	512	1024
$q \equiv 1 \mod 2n$	$2^{16}+1$	$2^{16}+1$
$t = \|\omega\|$	256	512
k	13.3	13.3
σ	2000	1800
κ s.t. $2^\kappa \cdot \binom{n}{\kappa} \geq 2^{256}$	44	36
$M = \exp(\frac{2\tau\kappa\sigma+\kappa^2}{2\sigma^2})$	≈ 7.4	≈ 7.4
Lattice strength	1.0035	1.0017
public key size $(\log_2 q + 2)t$	832 Bytes	1664 Bytes
signature length $\approx (\log_2 \sigma + 2)n + \min(\kappa \log_2 n, n)$	882 Bytes	1709 Bytes

Table 4.2: PASS$_{RS}$ Signature Scheme Parameter

4.5.2 Best known attacks

The best known lattice attack against our scheme is to look for the unique shortest vector within a lattice spanned via the basis:

$$\mathbf{B} = \begin{bmatrix} q\mathbf{I}_t & 0 & 0 \\ \mathcal{F}_\Omega & \mathbf{I}_n & 0 \\ \hat{\mathbf{f}}|_\Omega & 0 & 1 \end{bmatrix}$$

where $\mathbf{I}_t$ is a t dimensional identity matrix. This lattice has a unique shortest vector $\langle 0, \mathbf{f}, 1\rangle$ with an l_2 norm of approximately $\sqrt{2n/3+1}$. On the other hand, it has been shown in [41] that the ability to locate a unique shortest vector in a lattice depends on the root Hermite factor of the lattice, which is the N-th root of

$$\frac{\text{Gaussian expected length}}{l_2 \text{ norm of the target vector}}$$

where $N = (n+t+1)$ is the dimension of the lattice. We known that the Gaussian expected length of this lattice is $\sqrt{\frac{n+t+1}{2\pi e}} q^{\frac{t}{n+t+1}}$. This results in

$$\left(\frac{\sqrt{\frac{n+t+1}{2\pi e}} q^{\frac{t}{n+t+1}}}{\sqrt{2n/3+1}} \right)^{\frac{1}{n+t+1}}$$

With $t \approx n/2$, this quantity is

$$\approx \left(\sqrt{9/(8\pi e)}q^{\frac{1}{3}}\right)^{\frac{2}{3n}}.$$

For the parameter sets that we are suggesting, this yields 1.0035 and 1.0017, respectively. Applying the latest results of estimating the cost of the BKZ 2.0 algorithm with (quantum) sieving [27, 4, 11], we estimate the cost to recover this shortest vector requires at least 2^{129} and 2^{198} operations.

Chapter 5

Practical Lattice-Based (Linkable) Ring Signature Schemes

In this chapter, we present two generic constructions for linkable ring signature scheme. In Section 5.1, we give a new generic construction for (linkable) ring signature scheme inspired by RST framework. In Section 5.2, we revisit an existing generic ring signature framework and proposes its linkable version. Both of these two generic constructions can be instantiated to standard lattice and NTRU lattice. The latter NTRU instantiations provide us practical lattice-based (linkable) ring signature schemes.

5.1 Raptor: Practical Lattice-Based (Linkable) Ring Signature Schemes

Before presenting a high-level description of our construction, we first discuss the subtlety of instantiating the RST generic construction from lattices. The main building block of RST ring signatures is one-way trapdoor permutation. While trapdoor functions can be built from lattices, they are not permutations by themselves and therefore cannot be applied directly. Consequently, all existing linear-size lattice-based constructions opt for the AOS framework, which can be built from any sigma-protocol based signatures.

Indeed, [93, 13] can be seen as instantiations from this framework. On the other hand, we identify essential properties required by the underlying building blocks in the RST framework.The result is a type of trapdoor function that we called Chameleon hash plus (CH^+), a construction that is similar to Chameleon hash functions.

Building block: CH^+

The main building block for our generic constructions is a chameleon hash plus (CH^+) function. We recall the notion of chameleon hash function which was first formalized by Krawczyk and Rabin in 2000 [60]. 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 5.1.1 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;
- optionally, there exists a system parameter $\mathsf{param}^{\mathsf{ch}}$ as an implicit input to all CH^+ operations.

Our Generic Construction of Ring Signatures

We describe how we can built a ring signature from CH^+. We assume $\mathsf{param}^{\mathsf{ch}}$ is available at the setup. In the key generation procedure, a signer runs algorithm $\mathsf{TrapGen}$

to obtain 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 $(\mathsf{hk}_\pi, \mathsf{tr}_\pi)$, tries to sign 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 members.

Our Generic Construction of Linkable Ring Signatures

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 are 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[1] to achieve those

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

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 and list of public keys 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, σ_R and $\{\mathsf{hk}'_1, \cdots, \mathsf{hk}'_\ell\}$ under opk_π.

Our Contribution

We present RAPTOR, the first lattice-based (linkable) ring signature with implementation. It gets its name as it is the next generation of FALCON [40] 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 5.1(a) and 5.1(b). The experimental setting is presented in Section 5.1.3.

Table 5.1: 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

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 random 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. The best theoretical work in linear size is due to [13], where the signature size is claimed to be 82.5KB with a ring size of 8. Comparing with the state-of-the-art [38] with sublinear signature size, our work is still comparing favourably for ring signature size $\lessapprox 1000$. As a remark, a common use case of linkable ring signature, privacy protection for cryptocurrency, often uses a ring size less than 20 and thus Raptor is more preferable in this setting. The signature size of [38] 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

Table 5.2: Comparison of lattice-based (linkable) ring signature at security level $\lambda = 100$. Signature size increases with ring size (i.e. number of public keys in the ring).

	[64]	[93]	[13]	[38]	RAPTOR	(linkable) RAPTOR
Signature size growth	logarithm	linear	linear	logarithm	linear	linear
linkability	$\times$	✓	✓	$\times$	$\times$	✓
Implementation	$\times$	$\times$	$\times$	$\times$	✓	✓
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

our RAPTOR and other existing lattice-based (linkable) ring signature scheme is shown in Table 5.2.

In terms of security, (linkable) RAPTOR is backed by a new generic framework that is provably secure in the random oracle model, under the assumption based on RST construction. Instead of relying on one-way trapdoor permutation, the new generic framework is based on a new primitive called Chameleon Hash Plus (CH^+) which can be instantiated from lattice setting (e.g. NTRU). Our generic construction can additionally transform any ring signature into a one-time linkable ring signature.

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 [56] 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 [40].

5.1.1 Our Generic Constructions

In this section, we present our generic construction of CH^+ and our (linkable) ring signature scheme based on CH^+.

Chameleon Hash Plus

CH^+ can be considered as a variant of Chameleon hash functions. A CH^+ consists of four algorithms, namely, Setup, $\mathsf{TrapGen}$, Hash and Inv, as follow:

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{param}^{\mathsf{ch}}$: On input security parameter 1^λ, this algorithm generates system parameter $\mathsf{param}^{\mathsf{ch}}$. $\mathsf{param}^{\mathsf{ch}}$ will be an implicit input to Hash and Inv.
- $\mathsf{TrapGen}(1^\lambda) \rightarrow (\mathsf{hk}, \mathsf{tr})$: This algorithm takes security parameter 1^λ as input and returns a pair $(\mathsf{hk}, \mathsf{tr})$ where hk and tr are respectively a hash key and a trapdoor.
- $\mathsf{Hash}(\mathsf{hk}, m, r) \rightarrow C$: On input hash key hk, message m and randomness r, this algorithm returns hash output C.
- $\mathsf{Inv}(\mathsf{hk}, \mathsf{tr}, C, m') \rightarrow r'$: On input hash key hk, trapdoor tr, hash output C and message m', this algorithm returns randomness r' s.t. $\mathsf{Hash}(\mathsf{hk}, m', r') = C$.

We require CH^+ to satisfy following requirements:

1. CH^+ should be one-way and collision resistant. In other words, for all PPT $\mathcal{A}$ without a trapdoor, there exists a negligible function $\mathsf{negl}(l)$ such that

$$\Pr[\{(m_0, r_0), (m_1, r_1)\} \leftarrow \mathcal{A}(1^\lambda, \mathsf{hk}, \mathsf{param}^{\mathsf{ch}}) : (m_0, r_0) \neq$$

$$(m_1, r_1) \wedge \mathsf{Hash}(\mathsf{hk}, m_0, r_0) = \mathsf{Hash}(\mathsf{hk}, m_1, r_1)] = \mathsf{negl}(l);$$

$$\Pr[(m, r) \leftarrow \mathcal{A}(1^\lambda, C, \mathsf{hk}, \mathsf{param}^{\mathsf{ch}}) : \mathsf{Hash}(\mathsf{hk}, m, r) = C] = \mathsf{negl}(l).$$

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 $\mathsf{negl}(l)$ distance from the distribution where r is sampled from.

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 $\mathcal{H} : \{*\} \to R_C$. 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 from its possible range R_C.

    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 for Ring Signature

Theorem 5.1.1 (Unforgeability) *Our generic ring signature scheme is unforgeable in random oracle model if* CH^+ *is collision resistant.*

Proof 7 *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* $\mathsf{param^{ch}}_c$*, 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 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* $\mathsf{param^{ch}}_c$ *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* $\mathbf{Verification}(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) =$ *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*

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

where

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

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

Proof 8 *When* $\mathsf{Game_{anon}}$ *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* $\mathsf{negl}(l)$ *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* $\mathbf{adv}^{\mathsf{anon}}_{\mathcal{A}}$ *of an adversary should be negligible. Our ring signature scheme is unconditional anonymous.*

A New Framework for Linkable Ring Signatures

Our linkable ring signature is constructed as follows:

- **Setup**$(1^\lambda) \rightarrow \mathsf{param}$: On input the security parameter 1^λ, this algorithm chooses two hash functions $\mathcal{H}$ and $\mathcal{H}_1$. It also runs $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{param}^{\mathsf{ch}}$ and selects a one-time signature scheme $\Pi^{OTS} = \{\mathsf{OKeygen}, \mathsf{OSign}, \mathsf{OVer}\}$.

- **KeyGen** $\rightarrow (\mathsf{sk}, \mathsf{pk})$: This algorithm first generates $(\mathsf{hk}, \mathsf{tr}) \leftarrow \mathsf{TrapGen}(1^\lambda)$. It also generates a pair of Π^{OTS} public key and secret key $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OKeygen}(1^\lambda)$ and computes $\mathsf{mk} = \mathcal{H}_1(\mathsf{opk})$. It then computes $\mathsf{hk}' = \mathsf{hk} \oplus \mathsf{mk}$. Finally, it sets public key $\mathsf{pk} = \mathsf{hk}'$ and secret key $\mathsf{sk} = \{\mathsf{tr}, \mathsf{opk}, \mathsf{osk}\}$.

- **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \rightarrow \sigma$: 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 $m_1 \oplus \cdots \oplus m_\pi \oplus \cdots \oplus m_\ell = \mathcal{H}(\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}})$.

  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=$\mathsf{OSign}(\mathsf{osk}_\pi;$ (m_1, r_1), $\cdots$, (m_ℓ, r_ℓ),L_{pk}, $\mathsf{opk}_\pi)$.

  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 $m_1 \oplus \cdots \oplus m_\ell = \mathcal{H}\ (\mu, C_1, \cdots, C_\ell, L_{\mathsf{pk}})$;

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**$(\sigma_1, \sigma_2, \mu_1, \mu_2, L^{(1)}_{\mathsf{pk}}, L^{(2)}_{\mathsf{pk}}) \to$ *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 $\mathbf{Verify}(\mu_1, \sigma_1, L^{(1)}_{\mathsf{pk}}) \to$ *accept* and $\mathbf{Verify}(\mu_2, \sigma_2, L^{(2)}_{\mathsf{pk}}) \to$ *accept*, it phrases σ_1=$\{(m^{(1)}_1$,$r^{(1)}_1)$, $\cdots$,$(m^{(1)}_\ell$, $r^{(1)}_\ell)$, opk_1,$sig_1\}$ and $\sigma_2 = \{(m^{(2)}_1, r^{(2)}_1), \cdots, (m^{(2)}_\ell, r^{(2)}_\ell), \mathsf{opk}_2, sig_2\}$. The algorithm outputs *linked* if $\mathsf{opk}_1 = \mathsf{opk}_2$. Otherwise, output *unlinked*.

Our generic ring signature scheme and linkable signature scheme are both secure under random oracle model.

Security Proof for Linkable Ring Signature Scheme

Theorem 5.1.3 (Anonymity) *Our linkable ring signature scheme is anonymous in random oracle model if the second requirement in section 5.1.1 holds for* CH^+.

Proof 9 *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***$(1^\lambda) \to \mathsf{param}$ *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 $sig = \mathsf{OSign}(\mathsf{osk}_\pi; (m_1, r_1), \cdots, (m_\ell, r_\ell), L_{\mathsf{pk}}, \mathsf{opk}_\pi)$ 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 $\frac{1}{\ell}$.*

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

Theorem 5.1.4 (Linkability) *Our linkable ring signature is linkable in random oracle model if* CH^{+} *is collision resistant.*

Proof 10 *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* $\mathsf{Hash}(\mathsf{hk}_c, m', r') = \mathsf{Hash}(\mathsf{hk}_c, m'', r'')$ *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* $\mathcal{H}_0$ *and* $\mathcal{S}$ *also picks* $\{h'_1, h'_2, \cdots, h'_t\} \xleftarrow{\$} \mathcal{D}_{\mathcal{H}_1}$ *as the* q'_h *responses of the random oracle* $\mathcal{H}_1$*.* $\mathcal{S}$ *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* $\mathcal{I}$*,* $\mathcal{S}$ *sets* $\mathsf{pk}_{\mathcal{I}} = \mathsf{hk}'_{\mathcal{I}} = \mathsf{hk}_c \oplus h'_{Q}$ *where* $h'_{Q} \leftarrow_{\$} \{h'_1, h'_2, \cdots, h'_t\}$*. For other index,* $\mathcal{S}$ *generates the public key and secret key* *according to the **KeyGen** algorithm where the output of the random oracle $\mathcal{H}_1$ will be the first $h'_i \in \{h'_1, h'_2, \cdots, h'_t\}$ that has not been used yet. 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* $k-1$ *times.*

- $\mathcal{SO}(\mu, L_{\mathsf{pk}}, \mathsf{pk}_\pi)$*: 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* $\mathcal{H}_0$ *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* $i \in [1, \cdots, \ell]$*,* $\mathsf{pk}_i = \mathsf{hk}'_i$*,* $\mathcal{S}$ *runs* $\mathsf{OKeygen}(1^\lambda) \rightarrow (\mathsf{opk}_\pi, \mathsf{osk}_\pi)$ *and programs* $\mathcal{H}_1(\mathsf{opk}_\pi)$ *as the first* $h'_i \in \{h'_1, h'_2, \cdots, h_t\}$ *that has not been used yet. computes* $\mathsf{hk}_i = \mathcal{H}_1(\mathsf{opk}_\pi) \oplus \mathsf{hk}'_i$ *for* $i \in [1, \cdots, \ell]$*.* $\mathcal{S}$ *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 $\mathcal{H}_0$ *(*$\mathcal{H}_1$*): 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* $h_i \in \{h_1, h_2, \cdots, h_p\}$*(*$h'_i \in \{h'_1, \cdots, h'_t\}$*) 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 k sets $\{L_{\mathsf{pk}}^{(i)}, \mu_i, \sigma_i\}$ for $i \in [1, \cdots, k]$. These k sets should satisfy that **Verification**$(\mu_i, \sigma_i, L_{\mathsf{pk}}^{(i)})$ = accept ; $\mathcal{A}$ queried $\mathcal{CO}$ less than k times; and **Link**$(\sigma_i, \sigma_j, \mu_i, \mu_j, L_{\mathsf{pk}}^{(i)}, L_{\mathsf{pk}}^{(j)})$ = unlinked for $i \neq j$ and $i, j \in [1, \cdots, k]$. Since $\mathcal{A}$ is allowed query $\mathcal{CO}$ less than k times. At least one of the output signatures should be generated from the secret key that $\mathcal{A}$ does not obtain. Assume σ_j, $j \in \{1, \cdots, k\}$ is not produced by the secret key $\mathcal{A}$ obtaining. If $\mathsf{pk}_{\mathcal{I}} \notin L_{\mathsf{pk}}^{(j)}$ and $H_1(opk_j) = h'_Q$, abort.*

The probability for $\mathsf{pk}_{\mathcal{I}} \in L_{\mathsf{pk}}^{(j)}$ is no less than $\frac{1}{q_r}$ and the probability for $H_1(opk_j) = h'_Q$ is no less than $\frac{1}{q'_h}$. In the following we use $(\mu^*, \sigma^*, L_{\mathsf{pk}}^*)$ to denote $(\mu^j, \sigma^j, L_{\mathsf{pk}}^{(j)})$. 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 $h^* = \mathcal{H}_0(\mu^*, C_1^*, \cdots, C_\ell^*, L_{\mathsf{pk}}^*)$ 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 $(\mu^*, \sigma^* = \{(m_1^*, r_1^*), \cdots, (m_\ell^*, r_\ell^*), \mathsf{opk}^*, sig^*\}, L_{\mathsf{pk}}^*)$, $m_1^* \oplus \cdots \oplus m_\ell^* = \mathcal{H}(\mu^*, C_1^*, \cdots, C_\ell^*, L_{\mathsf{pk}}^*)$ is a response of random oracle $\mathcal{H}_0$ on $\mathcal{H}_0(\mu', C_1', \cdots, C_{\ell'}', L_{\mathsf{pk}}')$ during a $\mathcal{SO}$ query. Then, we have*

$$\mathcal{H}_0(\mu^*, C_1^*, \cdots, C_\ell^*, L_{\mathsf{pk}}^*) = \mathcal{H}_0(\mu', C_1', \cdots, C_{\ell'}', L_{\mathsf{pk}}')$$

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.* 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* $(\mu^*, \sigma^* = \{(m^*_1, r^*_1), \cdots, (m^*_\ell, r^*_\ell), \mathsf{opk}^*, sig^*\}, L^*_{\mathsf{pk}})$*. 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*

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

where $acc = (1 - \frac{1}{q_r})\frac{1}{q_r \cdot q'_h}(\delta - \frac{1}{|\mathcal{D}_\mathcal{H}|})$*. 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* $\mathcal{CO}$ *is no more than* $\frac{1}{q_r}$*. The probability for* $\mathcal{S}$ *not aborting during* output *is no less than* $\frac{1}{q_r} \cdot \frac{1}{q'_h}$*. Thus, the probability for* $\mathcal{S}$ *solving problem instance is no less than* $(1 - \frac{1}{q_r})\frac{1}{q_r \cdot q'_h} \cdot \Pr$ *which is non-negligible.*

Theorem 5.1.5 (Nonslanderability) *Our linkable ring signature is nonslanderable in random oracle model if the one-time signature scheme* Π^{OTS} *is one-time unforgeable.*

Proof 11 *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 $\mathsf{param}^{\mathsf{ch}} \leftarrow \mathsf{Setup}(1^\lambda)$. $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathsf{param}^{\mathsf{ch}}$ 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* $\mathbf{Verify}(\mu^*, \sigma^*, L^*_{\mathsf{pk}})$ = 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 2.6.1, our linkable ring signature scheme has nonsladerability and linkability. Thus, it is also unforgeable.

For the two security requirements of our ring signature scheme, both of them achieve the strongest security notions. Specifically, the unforgeability of our ring signature scheme allows chosen key attack to the signing oracle and chosen subring attack to the target signature. The anonymity of our ring signature scheme is unconditional and allow chosen key attack. For our linkable ring signature scheme, the unforgeability and linkability allows chosen key attack to the signing oracle and chosen subring attack to the target signature. The scheme is anonymous under chosen subring attack and the slanderability of our linkable ring signature scheme allows chosen key attack.

5.1.2 Instantiations from Lattice

In this section, we will show how to build CH^+ from standard lattice problems and from NTRU assumptions.

Instantiation of CH^+ from Standard Lattice

Here we present our first instantiation of CH^+ from standard lattice.

$\mathsf{Setup}(1^\lambda) \rightarrow \mathbf{H}$: 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.

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

$\mathsf{Hash}(\mathbf{A}, \mathbf{b}, \mathbf{r}) \rightarrow \mathbf{c}$: 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 $\mathbf{c} = \mathbf{Hb} + \mathbf{Ar}$ and returns $\mathbf{c}$.

$\mathsf{Inv}(\mathbf{A}, \mathbf{T}, \mathbf{c}, \mathbf{b}') \rightarrow \mathbf{r}'$: On input 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 $\mathbf{u} = \mathbf{c} - \mathbf{Hb}'$ and $\mathbf{r}' = \mathsf{PreSample}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s')$ where $s = s'\omega(\sqrt{\log n})$.

Now we argue that this instantiation satisfies our requirements of CH^+ in Section 5.1.1.

- Our instantiation is collision resistant and one-way if $\text{SIS}_{q,n,m',\beta}$ and $\text{ISIS}_{q,n,m',\beta}$ are hard for $m' = m + k$, $\beta = \sqrt{8ms^2 + 2k}$ and $\beta = \sqrt{4ms^2 + k}$ respectively.
- For the second requirement, according to Theorem 2.2.1, 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 Theorem 2.2.1, if we set deviation s appropriately (i.e., greater than the smooth parameter of $\mathbf{T}$, see [43]), 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.

Instantiation of CH^+ from NTRU

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

$\mathsf{Setup}(1^\lambda) \rightarrow (\mathbf{h}, D_\mathbf{b}, D_\mathbf{r})$: 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 $\mathcal{R}_q \times \mathcal{R}_q$.

$\mathsf{TrapGen}(1^\lambda) \rightarrow (\mathbf{a}, \mathbf{T})$: 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 $\mathbf{a}$.

$\mathsf{Hash}(\mathbf{a}, \mathbf{b}, \mathbf{r}) \rightarrow \mathbf{c}$: 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 $\mathbf{c} := \mathbf{r}_0 + \mathbf{a}\mathbf{r}_1 + \mathbf{h}\mathbf{b} \in \mathcal{R}_q$.

$\mathsf{Inv}(\mathbf{a}, \mathbf{T}, \mathbf{c}, \mathbf{b}') \rightarrow \mathbf{r}'$: On input hash key $\mathbf{a}$, its trapdoor $\mathbf{T}$, a ring element $\mathbf{c}$ and a binary message $\mathbf{b}'$, this algorithm first computes $\mathbf{u} = \mathbf{c} - \mathbf{b}'\mathbf{h}$. It then generates 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^+ in Section 5.1.1.

- 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[2] $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, 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.

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

- 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 2.2.1, if the deviation s 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.

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 security level, where $D_{\mathbf{b}} := \{0,1\}^{256}$, $D_{\mathbf{r}} := D^2_{\mathcal{R},\eta}$, 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**$\to$ $(\mathsf{sk}, \mathsf{pk})$: 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**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \to \sigma$: On input message μ, list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and signing key $\mathsf{sk}_\pi = \{\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi\}$ 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 $(\mathbf{r}_{\pi,0}, \mathbf{r}_{\pi,1}) = \text{FALCON}.\mathsf{sign}(\mathbf{a}_\pi, \mathsf{sk}_\pi; \mathbf{u}_\pi)$ 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**$(\mu, \sigma, L_{\mathsf{pk}}) \to$ $accept/reject$: On input message μ, signature σ and a list of user public keys L_{pk}, the verification algorithm performs as follows:

  1. phrases $\sigma = \{(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i)\}_{i=1}^{\ell}$;

  2. checks whether for each tuple of $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i)$, $\|\mathbf{r}_{i,0}\|, \|\mathbf{r}_{i,1}\| \leq B_1$ and $\mathbf{b}_i \in D_{\mathsf{b}}$; 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 $accept$.

For a legitimately produced ring signature σ, each $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1})$ pair should be distributed according to $\mathcal{D}^2_{\mathcal{R},\eta}$, thus the acceptance bound B_1 of $\mathbf{r}_{i,0}, \mathbf{r}_{i,1}$ should be $\nu\eta\sqrt{n}$ where ν is set such that $\|\mathbf{r}_{i,0}\|, \|\mathbf{r}_{i,1}\| \leq B_1$ with probability $1 - 2^{-l}$ Lemma 2.2.1.

Full Description of Linkable RAPTOR

Here we present the full description of linkable RAPTOR. 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. For easiness of implementation, we will also use FALCON to instantiate Π^{OTS}.

- **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 $\mathcal{R}_q$ and distributions $D_{\mathbf{b}}, D_{\mathbf{r}}$ for the security level, where $D_{\mathbf{b}} := \{0,1\}^{256}$, $D_{\mathbf{r}} := \mathcal{D}^2_{\mathcal{R}_q,\eta}$, $\mathcal{D}_{\mathcal{R}_q,\eta}$ is a discrete Gaussian distribution over $\mathcal{R}_q$ 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 $\mathsf{param}^{\mathsf{ch}}$. It also chooses a hash function $\mathcal{H}_1$: $\{*\} \to \mathcal{R}_q$.

- **KeyGen**$\to (\mathsf{sk}, \mathsf{pk})$: This algorithm runs as follow

  1. $(\mathbf{a}, \mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}) \leftarrow$ FALCON.KeyGen (param);

  2. $(\mathbf{a}_{ots}, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}) \leftarrow$ FALCON.$\mathsf{KeyGen}(\mathsf{param})$;

  3. set $\mathbf{a}' := \mathbf{a} + \mathcal{H}_1(\mathbf{a}_{ots}) \bmod q$.

  The public key $\mathsf{pk} = \mathbf{a}'$ and secret key $\mathsf{sk} = \{\mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}, \mathbf{a}_{ots}\}$.

- **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \to \sigma$: On input message μ, list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and signing key $\mathsf{sk}_\pi = \{\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}, \mathbf{a}_{ots}\}$ of

$\mathsf{pk}_\pi = \mathbf{a}'_\pi$, and the system parameter param, the signing algorithm runs as follow:

1. For $i \in [1, \cdots, \ell]$, 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 $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}) \leftarrow \mathcal{D}^2_{\mathcal{R}_q,\eta}$. 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 $(\mathbf{r}_{\pi,0}, \mathbf{r}_{\pi,1}) = \text{FALCON}.\mathsf{sign}(\mathbf{a}_\pi, (\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi); \mathbf{u}_\pi)$ such that $\mathbf{r}_{\pi,0} + \mathbf{r}_{\pi,1}\mathbf{a}_\pi = \mathbf{u}_\pi$.

7. Compute $sig := \text{FALCON}.\mathsf{sign}\ (\mathbf{a}_{ots}, (\mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}); (\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}_{i=1}^{\ell}, \{\mathbf{a}'_i\}_{i=1}^{\ell}, \mathbf{a}_{ots}))$.

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

  1. phrases $\sigma = \{\{\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i\}_{i=1}^{\ell}, \mathbf{a}_{ots}, sig\}$;

  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 for each tuple of $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{b}_i)$, $\|\mathbf{r}_{i,0}\|, \|\mathbf{r}_{i,1}\| \leq B_1$ and $\mathbf{b}_i \in D_{\mathsf{b}}$; 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. verifies 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 $accept$.

- **Link**$(\sigma_1, \sigma_2, \mu_1, \mu_2, L_{\mathsf{pk}}^{(1)}, L_{\mathsf{pk}}^{(2)}) \to$ *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. It then phrases σ_1=$\{\{\mathbf{r}_{i,0}^{(1)}, \mathbf{r}_{i,1}^{(1)}, \mathbf{b}_i^{(1)}\}_{i=1}^{\ell}, \mathbf{a}_{ots}^{(1)}, sig_1\}$ and σ_2= $\{\{\mathbf{r}_{i,0}^{(2)}, \mathbf{r}_{i,1}^{(2)}, \mathbf{b}_i^{(2)}\}_{i=1}^{\ell'}, \mathbf{a}_{ots}^{(2)}, sig_2\}$. This algorithm outputs *linked* if $\mathbf{a}_{ots}^{(1)} = \mathbf{a}_{ots}^{(2)}$. Otherwise, output *unlinked*.

For a legitimately produced ring signature σ, each $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1})$ pair should be distributed according to $\mathcal{D}_{\mathcal{R}_q,\eta}^2$, thus the acceptance bound B_1 of $\mathbf{r}_{i,0}, \mathbf{r}_{i,1}$ should be $\nu\eta\sqrt{n}$ where ν is set such that $\|\mathbf{r}_{i,0}\|, \|\mathbf{r}_{i,1}\| \leq B_1$ with probability $1 - 2^{-100}$ according to Lemma 2.2.1.

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.

5.1.3 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 [49]. This parameter set yields 114 bits security against classical attackers, and 103 bits security against quantum attackers, under the BKZ2.0 framework [27] with (quantum) sieving algorithm [4, 61].

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 5.1(a) and 5.1(b). Our source code is available at [98]. 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.

5.1.4 Known Attacks of RAPTOR

The NTRU assumption and the security of FALCON signature has been extensively studied in the literature [55, 27, 3, 32, 34, 40]. 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.

5.2 (Linkable) Ring Signature from Hash-Then-One-Way Signature

In this section, we first revisit AOS generic method for ring signatures. While they provide a generic approach to construct ring signatures, in their paper, security proofs are only given to the concrete examples. In other words, if one is to instantiate the AOS generic method from other cryptographic setting, a new security proof is needed. Observing this limitation, we give a security proof for the generic AOS transformation from Type-H signature scheme to ring signature schemes. Moreover, in the original paper, the security of its RSA instantiation is based on the one-wayness of the RSA trapdoor function. In our new proof, we instead rely on the unforgeability of the underlying Type-H signature. Also, different from the unforgeability security model in [1], we use a strengthened model that allows for both a corruption oracle and a signing oracle with adversarially chosen keys.

We then extend AOS framework to its linkable variant. Borrowing the idea from [69], we adopt a one-time signature scheme (Π^{OTS}). We build a generic method of constructing linkable ring signature based on Π^{OTS} and Type-H signature with uniform distributed public key. During the key generation procedure, in addition to the public key and secret key pair $(\mathsf{pk}, \mathsf{sk})$, each signer also generates a pair of public key and secret key $(\mathsf{opk}, \mathsf{osk})$ of a one-time signature. The signer then computes $\mathsf{PK} = \mathsf{pk} \oplus \mathsf{Hash}(\mathsf{opk})$ for some appropriate hash function $\mathsf{Hash}(\cdot)$. The new public key is PK and the secret key is $\mathsf{SK} = (\mathsf{sk}, \mathsf{opk}, \mathsf{osk})$.

Suppose a signer with public key PK_π wants to sign a message μ on behalf of a group of signers $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_\ell\}$ ($\pi \in \{1, \cdots, \ell\}$). For each public key PK_i in the group, the signer computes $\mathsf{pk}'_i = \mathsf{PK}_i \oplus \mathsf{Hash}(\mathsf{opk})$. The signer then obtains a new list of "public keys", $\{\mathsf{pk}'_1, \cdots, \mathsf{pk}'_\ell\}$. Note that, for the signer, pk'_π is equivalent to the original ring signature public key pk_π. The signer then runs the ring signature's signing algorithm with inputs μ, sk_π and $\{\mathsf{pk}'_1, \cdots, \mathsf{pk}'_\ell\}$ to obtain a ring signature σ_R. Next, it signs $\{L_{\mathsf{PK}}, \mu, \sigma_R, \mathsf{opk}_\pi\}$ using the one-time signature scheme. Denote by sig the one-time signature. The linkable ring signature is $\sigma = \{\sigma_R, \mathsf{opk}_\pi, sig\}$.

The verification is similar to the non-linkable version, with an additional step to verify the one-time signature.

Similar to [69], we also instantiate this generic linkable ring signature with NTRU lattice using a NTRU-based Type-H signature scheme: FALCON [40].

Comparison with Other Lattice-Based (Linkable) Ring Signature Scheme

In this section, we give a brief overview of the difference between this work, Raptor [69] and schemes from [13, 93]. We observe that at a high level, [13, 93] are both instantiations of AOS framework from the lattice-based signature BLISS [31], a Three-move type (Type-T) according to AOS's terminology. There are mainly two drawbacks from the underlying BLISS signature. First of all, for a lattice-based Type-T signature, a technique called rejection sampling is always required to prevent the leakage of secret key from signatures. Thus, ring signatures adopting BLISS require multiple iterations. Secondly, BLISS is vulnerable to side-channel attacks [39] due to the usage of a Gaussian sampler.

Raptor [69] is a new generic framework for (linkable) ring signature based on RST framework. Instead of relying on one-way trapdoor permutation as in the RST framework, [69] firstly introduces a new primitive, named CH+, and then uses CH+ to construct (linkable) ring signature scheme. They also showed how to build CH+ from one-way trapdoor functions. Thus, their work allows instantiations from lattice-based one-way

Table 5.3: Comparison of lattice-based (linkable) ring signature at security level $\lambda = 100$.

	[64]	[93]	[13]	[38]	[69]	Our
Signature size growth	logarithm	linear	linear	logarithm	linear	linear
linkability	$\times$	✓	✓	$\times$	✓	✓
Sig size for 2^6 users	37 MB	649 KB	585 KB	930 KB	82.7 KB	82.0 KB
2^8 users	48.1 MB	2.47 MB	2.34 MB	1132 KB	326.5 KB	318.9 KB
2^{10} users	59.1 MB	9.77 MB	9.36 MB	1409 KB	1301.9 KB	1266.6 KB
2^{12} users	70.2 MB	39 MB	37.4 MB	1492 KB	5203.3 KB	5057.5 KB

trapdoor function (rather than permutation). In this paper, we focus on the AOS framework with Type-H signature. Note that there are already lattice-based Type-H signatures in the literature. Comparing with Type-H signature, Type-H does not require rejection sampling. Comparing with linkable Raptor, we are able to achieve a slightly smaller signature size under the same assumption.

Contribution

We summarize our contributions as follows.

- We present a new security proof for the AOS generic construction of ring signatures from Type-H signature schemes. In the original paper, proofs are only given for concrete instantiations.

- Our proof is in a stronger security model which allows corruptions and a signing oracle with adversarially chosen keys. As a side note, we reduce the security of the generic construction to the unforgeability of the underlying Type-H signature (instead of the one-wayness of the underling trapdoor function), which allows generic constructions from any given Type-H digital signatures.

- We give a generic method of constructing linkable ring signature from Type-H signatures with uniformly distributed public key. We also provide the security proofs for the generic construction based on the security of underlying Type-H signature scheme.

- We instantiate the generic linkable ring signature from NTRU lattice and obtain a post-quantum and efficient linkable ring signature scheme. Our scheme has the shortest signature size when the ring size is reasonably small (i.e., less than 1024).

5.2.1 AOS Ring Signature Revisited

Recall that AOS's generic method builds ring signature schemes from hash-then-one-way type (Type-H) signature schemes. We first review the concept of (Type-H) digital signatures. Specifically, in a Type-H signature, pk and sk, created from key generation algorithm $\mathcal{G}_{sig}$, are associated with a one-way trapdoor function and its trapdoor respectively. Let F be a trapdoor one-way function and I be its inverse function. For any c from appropriate domain, compute $e = F_{\mathsf{pk}}(c)$ is easy. However, given any e', one cannot find the preimage c' such that $e' = F_{\mathsf{pk}}(c')$ in polynomial time without trapdoor. Secret key sk can be considered as the trapdoor which allows one to efficiently compute one of the preimages of e'. In signing algorithm **Signing**, Hash : $\{0,1\}^* \rightarrow \Delta$ is a hash function that hashes message μ and auxiliary information *aux* and I is the inverse function. Domain Δ is supposed to be an abelian group. A Type-H digital signature has the following structure.

```
Signing(μ, sk)
1: $c = \mathsf{Hash}(\mu, aux)$
2: $s = I_{\mathsf{sk}}(c)$
3: return $\sigma = (s, aux)$.
```

```
Verification(σ, μ, pk)
1: $\sigma \xrightarrow{parsing} (s, aux)$
2: $c = \mathsf{Hash}(\mu, aux)$
3: $e = F_{\mathsf{pk}}(s)$
4: return 1 if $c = e$,Otherwise, 0.
```

We first recall the generic method of constructing ring signature schemes.

Ring Signatures from $\mathsf{Type\text{-}H}$ Digital Signatures

Let $\mathsf{Hash}_i : \{0,1\}^* \to \Delta_i$ be a hash function where Δ_i is an abelian group. For $a, b \in \Delta_i$, let $a + b$ denote the group operation and $a - b$ be the group operation with inverse of b. Δ_i depends on pk_i in user public key list L_{pk}.

- **Setup**$(1^\lambda) \to$ param: On input security parameter 1^λ, this algorithm generates system parameter param which includes hash function H. We assume param is an implicit input to all the algorithms listed below.

- **KeyGen**$\to$ $(\mathsf{sk}, \mathsf{pk})$: This key generation algorithm generates the key pair using the key generation function of a signature scheme of his choice $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{G}^{sig}(\mathsf{param})$.

- **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \to \sigma$: On input message μ, a list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and signing key sk_π. the signing algorithm runs as follow:

  $\mathbf{G-1}$ (Initialization): Compute $e_\pi = \beta$, $\beta \leftarrow_\$ \Delta_\pi$. Then compute $c_{\pi+1} = \mathsf{Hash}_{\pi+1}(L_{\mathsf{pk}}, \mu, e_\pi)$.

  $\mathbf{G-2}$ (Forward Sequence): For $i = \pi + 1, \cdots, \ell$,1,$\cdots$,$\pi - 1$, compute $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$,where s_i is randomly chosen. Then compute $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$.

  $\mathbf{G-3}$ (Forming the Ring): $s_\pi = I_\pi(\beta - c_\pi, \mathsf{sk}_\pi)$. Output signature $\sigma = \{c_1, s_1, \cdots, s_\ell\}$ for message μ and public key list L_{pk}.

- **Verification**$(\mu, \sigma, L_{\mathsf{pk}}) \to$ $accept/reject$: On input message μ, signature σ and list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_n\}$, the verification algorithm runs as follow:

  For $i = 1, \cdots, \ell$, compute $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$ and then computes $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$ if $i \neq \ell$. Accept if $c_1 = \mathsf{Hash}_1(L_{\mathsf{pk}}, \mu, e_\ell)$. Otherwise, reject.

Security Analysis

In this section, we prove that the above generic construction is unconditional anonymous and is unforgeable if the underlying signature scheme is unforgeable.

Theorem 5.2.1 (Anonymity) *AOS ring signature scheme is unconditional anonymous.*

Here we roughly sketch the proof below. $\mathcal{RO}$ can be perfectly simulated with properly generated keys. The challenge signature will be created by programming the random oracle without using the corresponding signing key. Specifically, the challenge signature will be of the form $\sigma = \{c_1, s_1, \cdots, s_\ell\}$. In the actual signing algorithm, s_π is generated by $I_\pi(\beta - c_\pi, \mathsf{sk}_\pi)$ while in the simulation, all s_i are sampled according to the output distribution o I_i, and that $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$, $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$, with $\mathsf{Hash}_1(L_{\mathsf{pk}}, \mu, e_\ell)$ programmed to be c_1 in the random oracle model. It is straightforward to prove that the distribution of the simulated signature is the same as a real signature, and that it is independent of the signer's key. Therefore, the probability for adversary to make a successful guess is no more than $\frac{1}{\ell}$, meaning that the scheme is unconditionally anonymous.

Theorem 5.2.2 (Unforgeability) *AOS framework is unforgeable in random oracle model if the underlying* $\mathsf{Type\text{-}H}$ *signature scheme is unforgeable.*

Proof 12 *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 all the random oracles Hash_i. Then we can construct a simulator $\mathcal{S}$ who can break the unforgeability of the underlying* $\mathsf{Type\text{-}H}$ *signature scheme with a non-negligible probability.*

$\mathcal{S}$ is given a $\mathsf{Type\text{-}H}$ *signature scheme public key pk_c, it is asked to output σ_c such that σ_c is a valid forgery for pk_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 $\mathsf{Game}_{\mathsf{forge}}$ *with $\mathcal{A}$. $\mathcal{S}$ runs as follow:*

Setup. *Simulator $\mathcal{S}$ picks hash functions Hash_i: $\{0,1\}^* \to \Delta_i$. Hash_is will be modelled as random oracles. Assume $\mathcal{A}$ queries random oracles in the form of $Q = (k, L_{\mathsf{pk}}, \mu, e_{k-1})$ where k is a index in L_{pk}. $\mathcal{S}$ returns $\mathsf{Hash}_k(L_{\mathsf{pk}}, \mu, e_{k-1})$ to $\mathcal{A}$.*

$\mathcal{S}$ then randomly picks message μ_c and queries the challenger from the underlying forge game for its hash value (or uses the underlying signature's hash function to compute the hash value). $\mathcal{S}$ receives hash value h'_c.

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 pk_c to index $\mathcal{I}$ as the public key. For other indexes, $\mathcal{S}$ generates the public key and secret key according to* ***KeyGen***. *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***$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}})$*$\mathcal{S}$ returns the signature σ to $\mathcal{A}$;*

  – *If $\mathsf{pk}_\pi = \mathsf{pk}_{\mathcal{I}}$, $\mathcal{S}$ randomly choose c_1 from its range. For $i \in [1, \cdots, \ell]$ and pk_i, $\mathcal{S}$ samples s_i and computes $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$, $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$. $\mathcal{S}$ then programs random oracle as $\mathsf{Hash}_1(L_{\mathsf{pk}}, \mu, e_\ell) = c_1$. $\sigma = \{c_1, s_1, \cdots, s_\ell\}$.*

- Random Oracle Q: *At beginning of the simulation, $\mathcal{S}$ randomly picks $v, u \leftarrow_{\$} [1, \cdots, q_h]$ ($1 \leq v \leq u \leq q_h$). During simulation, if $Q_v = (k+1, L_{\mathsf{pk}}, \mu, e_k)$, $Q_u = (k, L_{\mathsf{pk}}, \mu, e_{k-1})$. $\mathcal{S}$ programs $\mathsf{Hash}_k(L_{\mathsf{pk}}, \mu, e_{k-1}) = e_k - h'_c$. For other query, if a query input that has already been programmed, $\mathcal{S}$ returns the corresponding output. Otherwise, the output of the random oracle will be randomly sampled from its range. $\mathcal{S}$ will record all the queries to the random oracle in a table, in case same query is issued twice.*

Output. *Finally, $\mathcal{A}$ will output a forgery (μ^*, σ^*, L^*_{pk}) with probability δ such that* $\mathbf{Verification}(\mu^*, \sigma^*, L^*_{\mathsf{pk}}) = accept$*; $(\mu^*, L^*_{\mathsf{pk}})$ has not been queried by $\mathcal{A}$ for signature; and no public key in L^*_{pk} has been input to $\mathcal{CO}$. If $\mathcal{A}$ wants to successfully forge such a ring signature, $\mathcal{A}$ must close a gap in $e^*_t - c^*_t$ for some $\mathsf{pk}_t \in L^*_{\mathsf{pk}}$ by first querying $Q^*_1 = (t+1, L^*_{\mathsf{pk}}, \mu^*, e^*_t)$, then querying $Q^*_2 = (t, L^*_{\mathsf{pk}}, \mu^*, e^*_{t-1})$. The probability for $\mathcal{S}$ successfully guessing $Q_v = Q^*_1$ and $Q_u = Q^*_2$ during random oracle simulation should be at least $\frac{1}{{q_h}^2}$. The probability for $\mathsf{pk}_t = \mathsf{pk}_{\mathcal{I}}$ should be at least $\frac{1}{q_r}$. $\mathcal{S}$ then have $e^*_t - c^*_t = F_t(\mathsf{pk}_t, s^*_t) = F_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, s^*_t)$. Since $Q^*_1 = Q_v$ and $Q^*_2 = Q_u$, we have $e^*_t - c^*_t = h'_c$ where h'_c is the hash output of μ_c. $\mathcal{S}$ outputs (μ_c, s^*_t) as a forgery.The probability for $\mathcal{S}$ to output such a forgery is at least $\frac{\delta}{{q_h}^2 \cdot q_r}$.*

5.2.2 Our Generic Ring Signature with Linkability

In this section, we give our generic construction of linkable ring signatures. Our generic construction mainly has two building blocks, namely, a $\mathsf{Type\text{-}H}$ signature scheme and a one-time signature scheme Π^{OTS}.

Same as AOS ring signature scheme, we have $\mathsf{Hash}_i : \{0,1\}^* \rightarrow \Delta_i$ being a hash function where range Δ_i is an abelian group. For $a, b \in \Delta_i$, let $a + b$ denote the group operation and $a - b$ be the group operation with inverse of b. Δ_i depends on pk_i in user public key list L_{pk}. Also, the distribution of public key pk of the underlying $\mathsf{Type\text{-}H}$

signature scheme should be uniformly distributed in its possible range. We emphasize that, for RSA and one-way trapdoor functions in lattice [44, 40], their public keys are all uniformly distributed.

- **Setup**$(1^\lambda)\to$ param: On input security parameter 1^λ, this algorithm generates system parameter param which includes a hash function H. We assume param is an implicit input to all the algorithms listed below. It also selects a one-time signature scheme $\Pi^{OTS} = \{\mathsf{OKeygen}, \mathsf{OSign}, \mathsf{OVer}\}$

- **KeyGen**$\to$ $(\mathsf{sk}, \mathsf{pk})$: This key generation algorithm generates the key pairs using the key generation function of a signature scheme of his choice $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{G}^{sig}(\mathsf{param})$. It also generates a pair of Π^{OTS} public key and secret key $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OKeygen}(1^\lambda)$ and computes $\mathsf{mk} = \mathsf{Hash}^*(\mathsf{opk})$. It sets public key $\mathsf{PK} = \mathsf{pk} \oplus \mathsf{mk}$ and secret key $\mathsf{SK} = \{\mathsf{sk}, \mathsf{opk}, \mathsf{osk}\}$.

- **Signing**$(\mathsf{SK}_\pi, \mu, L_{\mathsf{PK}}) \to \sigma$: On input message μ, a list of user public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_\ell\}$, and signing key $\mathsf{SK}_\pi = \{\mathsf{sk}_\pi, \mathsf{opk}_\pi, \mathsf{osk}_\pi\}$. For $i \in [1, \cdots, \ell]$, compute $\mathsf{pk}_i = \mathsf{PK}_i \oplus \mathsf{mk}_\pi$ where $\mathsf{mk}_\pi = \mathsf{Hash}^*(\mathsf{opk}_\pi)$. Signer obtains a new public key list $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$. The signing algorithm runs as follow:

  G − 1 (Initialization): Compute $e_\pi = \beta$ where $\beta \leftarrow_\$ \Delta_\pi$. Then compute $c_{\pi+1} = \mathsf{Hash}_{\pi+1}(L_{\mathsf{pk}}, \mu, e_\pi)$.

  G − 2 (Forward Sequence): For $i = \pi+1, \cdots, \ell, 1, \cdots, \pi-1$, compute $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$ where s_i is randomly chosen. Then compute $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$.

  G − 3 (Forming the Ring): $s_\pi = I_\pi(\beta - c_\pi, sk_\pi)$ Compute one-time signature $sig = \mathsf{OSign}(\mathsf{osk}_\pi; (c_1, s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk}_\pi))$. Output signature $\sigma = \{c_1, s_1, \cdots, s_\ell, sig, \mathsf{opk}_\pi\}$ for message μ and public key list L_{PK}.

- **Verification**$(\mu, \sigma, L_{\mathsf{PK}}) \rightarrow$ *accept/reject*: On input message μ, signature σ and list of user public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_n\}$. Parse σ to $\{c_1, s_1, \cdots, s_\ell, sig, \mathsf{opk}\}$. For $i \in [1, \cdots, \ell]$, compute $\mathsf{pk}_i = \mathsf{PK}_i \oplus \mathsf{mk}_\pi$ where $\mathsf{mk}_\pi = \mathsf{Hash}^*(\mathsf{opk})$. The verification algorithm runs as follow: for $i = 1, \cdots, \ell$, compute $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$ and then computes $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$ if $i \neq \ell$. Continue if $c_1 = \mathsf{Hash}_1(L_{\mathsf{pk}}, \mu, e_\ell)$. Otherwise, reject.

  Check whether $\mathsf{OVer}(\mathsf{opk}; (c_1, s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk})) = 1$. If not,output reject. If all pass, output accept.

- **Link**$(\sigma_1, \sigma_2, \mu_1, \mu_2, L_{\mathsf{PK}}^{(1)}, L_{\mathsf{PK}}^{(2)}) \rightarrow$ *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 parses σ_1=$\{c_1^{(1)}, s_1^{(1)}, \cdots, s_\ell^{(1)}$, opk_1,$sig_1\}$ and $\sigma_2 = \{c_1^{(2)}, s_1^{(2)}, \cdots, s_\ell^{(2)}$, $\mathsf{opk}_2, sig_2\}$. The algorithm outputs *linked* if $\mathsf{opk}_1 = \mathsf{opk}_2$. Otherwise, output *unlinked*.

Security Analysis

Theorem 5.2.3 (Anonymity) *The linkable ring signature is anonymous in random oracle model if the underlying* Type-H *signature scheme and* Π^{OTS} *are unforgeable.*

Proof 13 *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 $\mathsf{pk}_i = \mathsf{PK}_i \oplus \mathsf{Hash}^*(\mathsf{opk}_\pi)$. $\mathcal{S}$ randomly choose c_1 from its range. For $i \in [1, \cdots, \ell]$, $\mathcal{S}$ samples s_i and computes $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$, $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$. $\mathcal{S}$ then programs random oracle as $\mathsf{Hash}_1(L_{\mathsf{pk}}, \mu, e_\ell) = c_1$. $\mathcal{S}$ also computes one-time signature $sig = \mathsf{OSign}(\mathsf{osk}_\pi; s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk})$. $\mathcal{S}$ returns signature $\sigma = \{ s_1, \cdots, s_\ell,$ opk_π, $sig\}$.*

For adversary $\mathcal{A}$, $\mathcal{A}$ can not distinguish this game from the original one. Since in the scheme, the signer public key PK_π is the result of the exclusive or of pk_π and $\mathsf{Hash}^*(\mathsf{opk}_\pi)$ where $\mathsf{Hash}^*(\mathsf{opk}_\pi)$ 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. Here we have the distribution of pk uniformly over its possible range. Thus all the pk_i generated from $\mathsf{PK}_i \oplus \mathsf{Hash}^*(\mathsf{opk}_\pi)$ are indistinguishable from a true signature public key for adversary $\mathcal{A}$. 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 than $\frac{1}{\ell}$.

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

Theorem 5.2.4 (Linkability) *The linkable ring signature is linkable in random oracle model if the underlying $\mathsf{Type\text{-}H}$ signature scheme and Π^{OTS} are unforgeable.*

Proof 14 *Let $\mathcal{A}$ be an adversary 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 Hash^*, q_h queries to all the random oracles Hash_i. We show how to construct simulator $\mathcal{S}$ who can break unforgeability of the underlying $\mathsf{Type\text{-}H}$ signature scheme with a non-negligible probability.*

Given public key pk_c *of a* ***Type-H*** *signature scheme,* $\mathcal{S}$*'s task is to output a forged signature* σ_c *on any message of its choice. 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. $\mathcal{S}$ *picks hash functions* Hash^*, Hash_i*s and sets as system parameter.* Hash^*, Hash_i*s will be modeled as random oracle. For* Hash_i*,* $\mathcal{A}$ *queries it in the form of* $Q(k, L_{\mathsf{PK}}, \mu, e_{k-1})$*.* $\mathcal{S}$ *returns* $\mathsf{Hash}_k(L_{\mathsf{PK}}, \mu, e_{k-1})$ *to* $\mathcal{A}$*. For* Hash^**,* $\mathcal{S}$ *picks* $\{h_1^*, h_2^*, \cdots, h_{p^*}^*\} \leftarrow_{\$} \Delta_i^*$ *as the* q_h^* *responses of random oracle* Hash^**.*$\mathcal{S}$ *then random picks a message* μ_c *and queries the challenger from the forge game of the underlying signature scheme for its hash value (or compute the hash value by the given hash function).* $\mathcal{S}$ *receives hash value* h_c'*.*

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* $\mathcal{I}$*,* $\mathcal{S}$ *sets* $\mathsf{PK}_{\mathcal{I}} = \mathsf{pk}_c \oplus h_Q^*$ *where* $h_Q^* \leftarrow_{\$} \{h_1^*, h_2^*, \cdots, h_{p^*}^*\}$*. For other index,* $\mathcal{S}$ *samples* PK *uniformly random from its possible range. Upon the* j*th 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,* $\mathcal{S}$ *runs* $\mathsf{OKeygen}(1^\lambda) \rightarrow (\mathsf{opk}, \mathsf{osk})$*.* $\mathcal{S}$ *runs* $\mathcal{G}^{sig}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$*.* $\mathcal{S}$ *returns* $(\mathsf{sk}, \mathsf{opk}, \mathsf{osk})$ *as secret key and programs* $\mathsf{Hash}^*(\mathsf{opk}) = \mathsf{PK} \oplus \mathsf{pk}$.

- $\mathcal{SO}(\mu, L_{\mathsf{PK}}, \mathsf{PK}_\pi)$*: 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:*

  1). If PK_π *has been queried to* $\mathcal{CO}$*,* $\mathcal{S}$ *runs* ***Signing***$(\mathsf{SK}_\pi, \mu, L_{\mathsf{PK}})$*.* $\mathcal{S}$ *returns the* *signature σ to $\mathcal{A}$;*

2). If $\mathsf{PK}_\pi = \mathsf{PK}_\mathcal{I}$*, $\mathcal{S}$ runs* $\mathsf{OKeygen}(1^\lambda) \to (\mathsf{opk}_\pi, \mathsf{osk}_\pi)$ *and assigns the first $h_i^* \in \{h_1^*, h_2^*, \cdots, h_{p^*}^*\}$ that has not been used yet to* $\mathsf{Hash}^*(\mathsf{opk}_\pi)$*. $\mathcal{S}$ computes* $\mathsf{pk}_i = \mathsf{Hash}^*(\mathsf{opk}_\pi) \oplus \mathsf{PK}_i$ *for $i \in [1, \cdots, \ell]$. $\mathcal{S}$ randomly choose c_1 from its range. For $i \in [1, \cdots, \ell]$, $\mathcal{S}$ samples s_i and computes $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$, $c_{i+1} =$* $\mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$*. $\mathcal{S}$ then programs random oracle as* $\mathsf{Hash}_1(L_{\mathsf{pk}}, \mu, e_\ell) = c_1$*. $\mathcal{S}$ also computes one-time signature $sig =$* $\mathsf{OSign}(\mathsf{osk}_\pi; s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk})$*. $\mathcal{S}$ returns signature $\sigma = \{ s_1, \cdots, s_\ell,$ opk_π, $sig\}$;*

3). for other PK*, $\mathcal{S}$ runs* $\mathsf{OKeygen}(1^\lambda) \to (\mathsf{opk}, \mathsf{osk})$*. $\mathcal{S}$ runs $\mathcal{G}^{sig}(1^\lambda) \to$* $(\mathsf{pk}, \mathsf{sk})$*. $\mathcal{S}$ returns* $(\mathsf{sk}, \mathsf{opk}, \mathsf{osk})$ *as secret key and programs* $\mathsf{Hash}^*(\mathsf{opk}) = \mathsf{PK} \oplus \mathsf{pk}$*. $\mathcal{S}$ runs* ***Signing***$(\mathsf{SK}_\pi, \mu, L_{\mathsf{PK}})$*. $\mathcal{S}$ returns the signature σ to $\mathcal{A}$*

- Random Oracle Hash^**: 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.*

- Random Oracle Q*: At beginning of the simulation, $\mathcal{S}$ randomly picks $v, u \leftarrow_\$ [1, \cdots, q_h]$ ($1 \le v \le u \le q_h$). During simulation, if $Q_v = (k+1, L_{\mathsf{PK}}, \mu, e_k)$, $Q_u = (k, L_{\mathsf{PK}}, \mu, e_{k-1})$. $\mathcal{S}$ programs* $\mathsf{Hash}_k(L_{\mathsf{PK}}, \mu, e_{k-1}) = e_k - h'_c$*. For other query, if a query input that has already been programmed, $\mathcal{S}$ returns the corresponding output. Otherwise, the output of the random oracle will be randomly sampled from its range. $\mathcal{S}$ will record all the queries to the random oracle in a table, in case same query is issued twice.*

Output. *Adversary $\mathcal{A}$ outputs k sets $\{L_{\mathsf{PK}}^{(i)}, \mu_i, \sigma_i\}$ for $i \in [1, \cdots, k]$. These k sets should satisfy that* ***Verification***$(\mu_i, \sigma_i, L_{\mathsf{PK}}^{(i)}) =$ accept *; $\mathcal{A}$ queried $\mathcal{CO}$ less than k times; and*

Link$(\sigma_i, \sigma_j, \mu_i, \mu_j, L_{\mathsf{PK}}^{(i)}, L_{\mathsf{PK}}^{(j)})$ = unlinked *for $i \neq j$ and $i, j \in [1, \cdots, k]$. Since $\mathcal{A}$ is allowed query $\mathcal{CO}$ less than k times. At least one of the output signatures should be generated from the* opk *that $\mathcal{A}$ does not obtain from $\mathcal{CO}$ or $\mathcal{SO}$. If any of the* opk *from the returned signatures are obtained from $\mathcal{SO}$, $\mathcal{A}$ breaks the unforgeability of Π^{OTS}. Assume σ_j, $j \in \{1, \cdots, k\}$ is not produced by the* opk *obtaining from $\mathcal{CO}$ or $\mathcal{SO}$.* $\mathsf{Hash}^*(opk_j) \neq h_Q^*$*, abort.*

The probability for $\mathsf{Hash}^*(opk_j) = h_Q^*$ *is no less than $\frac{1}{q_h^*}$. In the following we use $(\mu^*, \sigma^*, L_{\mathsf{PK}}^*)$ to denote $(\mu^j, \sigma^j, L_{\mathsf{PK}}^{(j)})$. If $\mathcal{A}$ wants to successfully forge such a linkable ring signature, $\mathcal{A}$ must close a gap in $e_t^* - c_t^*$ by first querying $Q_1^* = (t+1, L_{\mathsf{PK}}^*, \mu^*, e_t^*)$, then querying $Q_2^* = (t, L_{\mathsf{PK}}^*, \mu^*, e_{t-1}^*)$. The probability for $\mathcal{S}$ successfully guessing $Q_v = Q_1^*$ and $Q_u = Q_2^*$ during random oracle simulation should be at least $\frac{1}{{q_h}^2}$. The probability for* $\mathsf{PK}_t = \mathsf{PK}_{\mathcal{I}}$ *should be at least $\frac{1}{q_r}$. Since* $\mathsf{Hash}^*(opk^*) = h_Q^*$*, we have* $\mathsf{Hash}^*(opk^*) \oplus \mathsf{PK}_{\mathcal{I}} = \mathsf{pk}_c$. *$\mathcal{S}$ then have $e_t^* - c_t^* = F_t(\mathsf{pk}_c, s_t^*)$. Since $Q_1^* = Q_v$ and $Q_2^* = Q_u$, we have $e_t^* - c_t^* = h_c'$ where h_c' is the hash output of μ_c. $\mathcal{S}$ outputs (μ_c, s_t^*) as a forgery.The probability for $\mathcal{S}$ to output such a forgery is at least $\frac{\delta}{{q_h}^2 \cdot q_h^* \cdot q_r}$.*

Theorem 5.2.5 (Nonslanderability) *Our linkable ring signature is nonslanderable in random oracle model if the one-time signature scheme Π^{OTS} is one-time unforgeable.*

Proof 15 *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. *$\mathcal{S}$ picks hash functions* Hash^*, Hash_i*s and sets as system parameter.* Hash^*, Hash_i*s will be modeled as random oracle.*

Oracle Simulation. *$\mathcal{S}$ simulates the oracles as follow:*

- $\mathcal{RO}(\perp)$*: Upon request, $\mathcal{S}$ samples* PK *uniformly random from its possible range. $\mathcal{S}$ returns the* PK *as the public key.*
- $\mathcal{CO}(\mathsf{PK})$*: On input a public key* PK *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* $\mathcal{G}^{sig}(\mathsf{param}) \rightarrow (\mathsf{sk}, \mathsf{pk})$*. $\mathcal{S}$ returns* $(\mathsf{sk}, \mathsf{opk}, \mathsf{osk})$ *as secret key and programs* $\mathsf{Hash}^*(\mathsf{opk}) = \mathsf{PK} \oplus \mathsf{pk}$.
- $\mathcal{SO}(\mu, L_{\mathsf{PK}}, \mathsf{PK}_\pi)$*: 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* 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 $i \in [1, \cdots, \ell]$, compute* $\mathsf{pk}_i = \mathsf{PK}_i \oplus \mathsf{mk}_\pi$ *where* $\mathsf{mk}_\pi = \mathsf{Hash}^*(\mathsf{opk}_\pi)$. *$\mathcal{S}$ obtains a new public key list* $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$. *The signing algorithm runs as follow:*

    $\mathbf{G-1}$ *(Initialization): Compute $e_\pi = \beta$ where $\beta \leftarrow_\$ \Delta_\pi$. Then compute* $c_{\pi+1} = \mathsf{Hash}_{\pi+1}(L_{\mathsf{pk}}, \mu, e_\pi)$.

    $\mathbf{G-2}$ *(Forward Sequence): For $i = \pi + 1, \cdots, \ell, 1, \cdots, \pi - 1$, compute*

$e_i = c_i + F_i(s_i, \mathsf{pk}_i)$ where s_i is randomly chosen. Then compute $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$.

G − 3 *(Forming the Ring): For $i = \pi$, randomly choose s_π and computes $c_\pi = e_\pi - F_\pi(s_\pi, \mathsf{pk}_\pi)$. Program $c_\pi = \mathsf{Hash}_\pi(L_{\mathsf{pk}}, \mu, e_{\pi-1})$. Compute one-time signature $sig = \mathsf{OSign}(\mathsf{osk}_\pi; (c_1, s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk}_\pi))$. Output signature $\sigma = \{c_1, s_1, \cdots, s_\ell, sig, \mathsf{opk}_\pi\}$ for message μ and public key list L_{PK}.*

- Random Oracle*: For input that has already been programmed, $\mathcal{S}$ returns the corresponding output. Otherwise, $\mathcal{S}$ randomly samples hash output and returns the value. $\mathcal{S}$ 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 $i \in [1, \cdots, \ell]$, compute $\mathsf{pk}_i = \mathsf{PK}_i \oplus \mathsf{mk}_\pi$ where $\mathsf{mk}_\pi = \mathsf{Hash}^*(\mathsf{opk}')$. $\mathcal{S}$ obtains a new public key list $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$. $\mathcal{S}$ then computes $e_\pi = \beta$ where $\beta \leftarrow_\$ \Delta_\pi$ and $c_{\pi+1} = \mathsf{Hash}_{\pi+1}(L_{\mathsf{pk}}, \mu, e_\pi)$. For $i = [\pi + 1, \cdots, \ell, 1, \cdots, \pi - 1]$, compute $e_i = c_i + F_i(s_i, \mathsf{pk}_i)$ where s_i is randomly chosen. Then compute $c_{i+1} = \mathsf{Hash}_{i+1}(L_{\mathsf{pk}}, \mu, e_i)$. For $i = \pi$, $\mathcal{S}$ randomly chooses s_π and computes $c_\pi = e_\pi - F_\pi(s_\pi, \mathsf{pk}_\pi)$. Program $c_\pi = \mathsf{Hash}_\pi(L_{\mathsf{pk}}, \mu, e_{\pi-1})$. Then, $\mathcal{S}$ queries for the one-time signature sig' of message $\nu' = (c_1, s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk}')$. $\mathcal{S}$ returns $\sigma = \{c_1, s_1, \cdots, s_\ell, L_{\mathsf{PK}}, \mathsf{opk}', sig'\}$ 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

$\sigma^* = \{c_1^*, s_1^*, \cdots, s_{\ell'}^*, L_{\mathsf{PK}}, \mathsf{opk}^*, sig^*\}$. *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^*; c_1^*, s_1^*, \cdots, s_{\ell'}^*, L_{\mathsf{PK}}, \mathsf{opk}^*) = $ accept. *Since* $\sigma^*, L_{\mathsf{PK}}^*$ *must be different from* σ, L_{PK}. $\mathcal{S}$ *obtains a one-time message signature pair where message is* $\nu^* = \{c_1^*, s_1^*, \cdots, s_{\ell'}^*, L_{\mathsf{PK}}, \mathsf{opk}^*\} \neq \nu'$ *in challenge.* sig^* *is a valid one-time signature for* opk' *and* ν^*. $\mathcal{S}$ *breaks the unforgeability of* Π^{OTS}.

According to Theorem 2.6.1, our linkable ring signature scheme is nonsladerable and linkable. Thus, it is also unforgeable.

5.2.3 Instantiations from NTRU

In this section, we are going to instantiate the linkable ring signature to NTRU lattice using a NTRU based Type-H signature scheme: FALCON. In 2008, Gentry, Peikert and Vaikuntanathan [44] construct a one-way trapdoor function using 'hard' basis and 'good' basis of a lattice and apply it to construct a lattice-based Type-H signature schemes. Prest at al. [34, 40] then use NTRU lattices [56] to instantiate the GPV construction. The corresponding NTRU-based Type-H signature scheme is named FALCON [40].

5.2.4 Linkable Ring Signature from Falcon

FALCON signature scheme is a Type-H signature scheme. Thus, we can apply the generic method of transforming Type-H signature scheme to linkable ring signature scheme to FALCON.

- **Setup**$(1^\lambda) \to \mathsf{param}$: On input security parameter 1^λ, this algorithm chooses Hash^*, $\mathsf{Hash}_i : \{0,1\}^* \to R_q$, $D_{\mathbf{b}}$, $D_{\mathbf{r}}$ and η.

- **KeyGen**$\to (\mathsf{sk}, \mathsf{pk})$: This algorithm firstly generates

  - $(\mathbf{a}, \mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}) \leftarrow$ FALCON.$\mathsf{KeyGen}(\mathsf{param})$, and

– $(\mathbf{a}_{ots}, \mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}) \leftarrow \text{FALCON}.\mathsf{KeyGen}(\mathsf{param})$

Then it sets $\mathbf{a}' := \mathbf{a} + \mathsf{Hash}^*(\mathbf{a}_{ots}) \bmod q$. The public key $\mathsf{pk} = \{\mathbf{a}'\}$ and secret key $\mathsf{sk} = \{\mathsf{sk}^0 = \{\mathbf{f}, \mathbf{g}, \bar{\mathbf{f}}, \bar{\mathbf{g}}\}, \mathsf{sk}^1 = \{\mathbf{f}_{ots}, \mathbf{g}_{ots}, \bar{\mathbf{f}}_{ots}, \bar{\mathbf{g}}_{ots}\}, \mathbf{a}_{ots}\}$.

- **Signing**$(\mathsf{sk}_\pi, \mu, L_{\mathsf{pk}}) \to \sigma$: On input message μ, list of user public keys $L_{\mathsf{pk}} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_\ell\}$, and signing key $\mathsf{sk}_\pi = \{\mathsf{sk}^0_\pi = \{\mathbf{f}_\pi, \mathbf{g}_\pi, \bar{\mathbf{f}}_\pi, \bar{\mathbf{g}}_\pi\}, \mathsf{sk}^1_\pi = \{\mathbf{f}_{ots}, \mathbf{g}_{ots}$, $\bar{\mathbf{f}}_{ots}$, $\bar{\mathbf{g}}_{ots}$ $\}$, $\mathbf{a}_{ots}\}$ of $\mathsf{pk}_\pi = \{\mathbf{a}'_\pi\}$, and the system parameter param, the signing algorithm runs as follow:

  1. for $i \in [1, \cdots, \ell]$, compute $\mathbf{a}_i = \mathbf{a}'_i - \mathsf{Hash}^*(\mathbf{a}_{ots}) \bmod q$. Signer then obtains a new list $L = \{\mathbf{a}_1, \cdots, \mathbf{a}_\ell\}$.
  2. it then randomly samples a polynomial $\mathbf{e}_\pi \leftarrow_\$ R_q$ and computes $\mathbf{c}_{\pi+i} = \mathsf{Hash}_{\pi+1}(L, \mu, \mathbf{e}_\pi)$;
  3. for $i = \pi + 1, \cdots, \ell, 1, \cdots, \pi - 1$, compute $\mathbf{e}_i = \mathbf{c}_i + \mathbf{x}_{i,0} + \mathbf{a}_i \mathbf{x}_{i,1}$ where $\mathbf{x}_i = (\mathbf{x}_{i,0}, \mathbf{x}_{i,1}) \leftarrow \mathcal{D}^2_{\mathcal{R},\eta}$ and $\mathbf{c}_{i+1} = \mathsf{Hash}_{i+1}(L, \mu, \mathbf{e}_i)$;
  4. compute $(\mathbf{x}_{\pi,0}, \mathbf{x}_{\pi,1}) = \text{FALCON}.\mathsf{sign}(\mathsf{sk}^0_\pi; \mathbf{e}_\pi - \mathbf{c}_\pi)$ such that $\mathbf{x}_{\pi,0} + \mathbf{a}_\pi \mathbf{x}_{\pi,1} = \mathbf{e}_\pi - \mathbf{c}_\pi$;
  5. it then generates a signature *sig* on $\mu' = \{\{\mathbf{x}_i\}^i_{i=1}, \mathbf{c}_1, L_{\mathsf{pk}}, \mathbf{a}_{ots}\}$ by computing *sig*=FALCON.sign$(\mathsf{sk}^1_\pi; \mu')$.

  The linkable ring signature of μ and L_{pk} is σ =$\{\{\{\mathbf{x}_i\}^\ell_{i=1}$,$\mathbf{c}_1$,$\mathbf{a}_{ots}$, $sig\}$.

- **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. parses σ =$\{\mathbf{x}_1, \cdots, \mathbf{x}_\ell, \mathbf{c}_1$,$\mathbf{a}_{ots}$, $sig\}$;

2. For $i \in [1, \cdots, \ell]$, compute $\mathbf{a}_i = \mathbf{a}'_i - \mathsf{Hash}^*(\mathbf{a}_{ots}) \bmod q$;

3. checks whether for all $i \in [1, \cdots, \ell]$, $\|\mathbf{x}_i\|_\infty \leq \beta$; outputs $reject$ if not;

4. for all $i \in [1, \cdots, \ell]$, computes $\mathbf{y}_i = \mathbf{x}_{i,0} + \mathbf{a}_i \mathbf{x}_{i,1}$, $\mathbf{e}_i = \mathbf{c}_i + \mathbf{y}_i$. Then compute $\mathbf{c}_{i+1} = \mathsf{Hash}_{i+1}(L, \mu, \mathbf{e}_i)$ if $i \neq \ell$. Continue if $\mathbf{c}_1 = \mathsf{Hash}_1(L, \mu, \mathbf{e}_\ell)$. Otherwise, reject.

5. verify whether sig is a FALCON signature for $(\{\mathbf{x}_i\}_{i=1}^{\ell}, \mathbf{c}_1, \{\mathbf{a}'_i\}_{i=1}^{\ell}, \mathbf{a}_{ots})$ with public key $\mathbf{a}_{ots}$; outputs $reject$ if fails.

6. outputs $accept$.

- **Link**$(\sigma_1, \sigma_2, \mu_1, \mu_2, L_{\mathsf{PK}}^{(1)}, L_{\mathsf{PK}}^{(2)}) \rightarrow$ *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 parses σ_1=$\{\{\{\mathbf{x}_i^{(1)}\}_{i=1}^{\ell}$,$\mathbf{c}^{(1)}$,$\mathbf{a}_{ots}^1, sig_1\}$ and σ_2 = $\{\{\{\mathbf{x}_i^{(2)}\}_{i=1}^{\ell}$,$\mathbf{c}^{(2)}$,$\mathbf{a}_{ots}^2, sig_2\}$. The algorithm outputs *linked* if $\mathbf{a}_{ots}^1 = \mathbf{a}_{ots}^2$. Otherwise, output *unlinked*.

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.

5.2.5 Efficiency Analysis

Here we give some estimated performance of instantiating generic construction with FALCON-512. The estimated linkable signature size is around $617 \times 2(\ell + 1) + 2 * 897 \approx 1.23(\ell + 1) + 2 * 0.897$ kilo bytes, where ℓ is the number of users in a signature. For a signature of FALCON-512, the size is around 2×617 bytes. Besides, 897 bytes is the size of a ring element in FALCON-512.

We give comparison with some other (linkable) ring signature schemes in Table 5.3. Comparing with other linear-size linkable ring signature, we have the smallest signature size. Comparing with scheme with logarithmic signature size, our linkable ring signature scheme has a better performance with a small ring (ring size $\leq 2^{10}$). Thus, our linkable ring signature is practical and can be applied in scenarios with small groups.

Chapter 6

Conclusion

Post-quantum cryptography is now attracting more and more attentions because of its property of resistance to quantum attacks. Even though there is still no practical large quantum computers. Considering the possibility that someone stores sensitive communication data and information from Internet and analyzes it when quantum computer really appears. Designing post-quantum cryptosystems and applying them in the real world as soon as possible is meaningful. In particular, quantum computational operations have already been executed on a small number of quantum bits. In this thesis, our works mainly focus on designing cryptosystems that is resistance to quantum attacks.

Firstly, we propose a symmetric-key based post-quantum anonymous identification protocol for ad hoc group based on hash-based cryptography. We also develop security models to capture the security requirements of anonymous identification scheme. Our protocol is simple in concept and concrete in construction which can be easily compatible with widely used password-based authentication. Additionally, we show that our protocol is more efficient then related works priori to it.

Secondly, we present a new lattice-based signature scheme based on a signature scheme proposed by Hoffstein et al. [53]. Our new signature scheme out-performs the original one in both running time and signature size. Moreover, to prove the security of our construction, we give specific reduction proof, which was lack in [53], and reduce the security to a variant of lattice-based hard problems. We also analyze the security of our signature scheme using quantum sieving algorithm and show that our construction is post-quantum.

The last two works of this thesis are both focusing on generic frameworks of (linkable) ring signature and their lattice instantiations. In Section 5.1, we propose a new generic construction of (linkable) ring signature schemes. The new generic construction is then instantiated to both standard lattice and NTRU lattice setting. We also give our NTRU lattice-based (linkable) ring signature scheme a name called (linkable) Raptor. Furthermore, we implement (linkable) Raptor and show the efficiency of it comparing with lattice-based (linkable) ring signature proposed before it. In Section 5.2, we revised an existing generic ring signature [1] to its linkable version and gives the security proofs for both the original and linkable version of the generic framework. The security proof for generic construction is lack in the original paper. We instantiate our generic linkable ring signature to NTRU lattice and obtains a lattice-based linkable ring signature slightly more efficient than our work in Section 5.1.

6.1 Future Work

In our first work, anonymous identification for ad hoc group, we require that verifier in the scheme should be honest but curious. However, in the real world, we must consider the possibility that verifier is dishonest and try to actively gain personal information from prover. Thus, we could improve the scheme to maintain its anonymity even if verifier does not act following the scheme. At the same time, we should still keep the advantages of high efficiency and compatibility of our scheme.

Besides, we will continuously work on the thesis from following aspects. Our two lattice-based linkable ring signature schemes, while are practical enough to be applied in the real world scenarios, still need improvement. The signature size of our linkable ring signature schemes are linear to the number of members in the ring. Thus, a large ring size can increase the signature size rapidly. Despite our linkable ring signature schemes have a signature size roughly 1.6 KB per member. It is, nevertheless, an important job for us to try to shrink the signature size in various ways. One solution is that we can reduce the size of signature for each user. However, the core problem has not yet been addressed. The best approach is that, we should let the linkable ring signature size be logarithmic or even be constant to the ring size. Thus, our future target in improving the efficiency of linkable ring signature scheme is that, we should design a logarithmic size linkable ring signature scheme that is practical enough to be implemented. One possible way to construct a logarithmic size linkable ring signature scheme is by adopting zero-knowledge proof techniques. The major obstacle of applying existing lattice-based zero-knowledge proof systems to constructing a practical ring signature scheme is that, most of the existing lattice-based zero-knowledge proof systems have large overhead or are not efficient enough to be implemented. Thus we could focus on constructing practical and efficient lattice-based zero-knowledge proof system.

Bibliography

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

[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] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! Cryptology ePrint Archive, Report 2018/331, 2018. `https://eprint.iacr.org/2018/331`.

[4] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter 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] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[6] Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors. *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*. Springer, 2007.

[7] Man Ho Au, Sherman S. M. Chow, Willy Susilo, and Patrick 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] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon 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] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theor. Comput. Sci.*, 469:1–14, 2013.

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

[11] Shi Bai, Thijs Laarhoven, and Damien Stehle. Tuple lattice sieving. Cryptology ePrint Archive, Report 2016/713, 2016. `https://eprint.iacr.org/2016/713`.

[12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[13] Carsten Baum, Huang Lin, , and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. Cryptology ePrint Archive, Report 2018/107, 2018. `https://eprint.iacr.org/2018/107`.

[14] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.

[15] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on*

Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, pages 390–399. ACM, 2006.

[16] Adam Bender, Jonathan Katz, and Ruggero 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.

[17] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 368–397. Springer, 2015.

[18] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *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*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69. Springer, 2011.

[19] Dan Boneh and Matthew K. Franklin. Anonymous authentication with subset queries (extended abstract). In Juzar Motiwalla and Gene Tsudik, editors, *CCS '99, Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, November 1-4, 1999.*, pages 113–119. ACM, 1999.

[20] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe 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.

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

[22] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara,*

California, USA, August 18-22, 2002, Proceedings, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer, 2002.

[23] Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2011.

[24] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Jr. [59], pages 410–424.

[25] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Arge et al. [6], pages 423–434.

[26] David Chaum and Eugène van Heyst. Group signatures. In Davies [29], pages 257–265.

[27] Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.

[28] Sherman S. M. Chow, Joseph K. Liu, Victor K. Wei, and Tsz Hon Yuen. Ring signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:317, 2005.

[29] Donald W. Davies, editor. *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*. Springer, 1991.

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

[31] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013.

[32] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013.

[33] Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - dilithium: Digital signatures from module lattices. *IACR Cryptology ePrint Archive*, 2017:633, 2017.

[34] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *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*, volume 8874 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2014.

[35] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, number 7658 in Lecture Notes in Computer Science, pages 433–450. Springer Berlin Heidelberg, January 2012.

[36] Léo Ducas and Thomas 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.

[37] Cynthia Dwork and Moni 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.

[38] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. Cryptology ePrint Archive, Report 2018/773, 2018. `https://eprint.iacr.org/2018/773`.

[39] Thomas Espitau, Pierre-Alain Fouque, Benoit Gerard, and Mehdi Tibouchi. Side-channel attacks on bliss lattice-based signatures – exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. Cryptology ePrint Archive, Report 2017/505, 2017. `https://eprint.iacr.org/2017/505`.

[40] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU.

[41] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.

[42] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, 2010.

[43] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.

[44] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, page 197–206, New York, NY, USA, 2008. ACM.

[45] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Jr. [59], pages 112–131.

[46] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 399–408. ACM, 1998.

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

[48] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[49] Lov 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.

[50] Debiao He, Jianhua Chen, and Jin Hu. An id-based proxy signature schemes without bilinear pairings. *Annales des Télécommunications*, 66(11-12):657–662, 2011.

[51] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: digital signatures using the NTRU lattice. In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2003.

[52] Jeffrey Hoffstein, Daniel Lieman, and Joseph H. Silverman. Polynomial rings and efficient public key authentication. In *City University of Hong Kong*. Press.

[53] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Practical signatures from the partial fourier recovery problem. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, volume 8479 of *Lecture Notes in Computer Science*, pages 476–493. Springer, 2014.

[54] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 142–159, 2014.

[55] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. *IACR Cryptology ePrint Archive*, 2015:708, 2015.

[56] Jeffrey Hoffstein, Jill Pipher, and Joseph 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.

[57] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. A signature scheme from learning with truncation. Cryptology ePrint Archive, Report 2017/995, 2017. `http://eprint.iacr.org/2017/995`.

[58] Jeffrey Hoffstein and Joseph H. Silverman. *Polynomial Rings and Efficient Public Key Authentication II*, pages 269–286. Birkhäuser Basel, Basel, 2001.

[59] Burton S. Kaliski Jr., editor. *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*. Springer, 1997.

[60] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*, 2000.

[61] Thijs Laarhoven and Artur 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.

[62] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979.

[63] Chan H. Lee, Xiaotie Deng, and Huafei Zhu. Design and security analysis of anonymous group identification protocols. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 188–198. Springer, 2002.

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

[65] Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Online/offline ring signature scheme. In Sihan Qing, Chris J. Mitchell, and Guilin Wang, editors, *Information and Communications Security, 11th International Conference, ICICS 2009, Beijing, China, December 14-17, 2009. Proceedings*, volume 5927 of *Lecture Notes in Computer Science*, pages 80–90. Springer, 2009.

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

[67] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer, 2004.

[68] Adriana López-Alt, Eran Tromer, and Vinod 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.

[69] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. Cryptology ePrint Archive, Report 2018/857, 2018. `https://eprint.iacr.org/2018/857`.

[70] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 20–39. Springer, 2004.

[71] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 598–616, 2009.

[72] Vadim Lyubashevsky. Lattice signatures without trapdoors. In Pointcheval and Johansson [84], pages 738–755.

[73] Vadim Lyubashevsky and Daniele 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.

[74] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2008.

[75] Carlos Aguilar Melchor, Slim Bettaieb, Xavier Boyen, Laurent Fousse, and Philippe 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.

[76] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.

[77] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [84], pages 700–718.

[78] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization, 2017.

[79] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005.

[80] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.

[81] Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. `https://eprint.iacr.org/2015/1098`.

[82] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2010.

[83] Chris Peikert and Alon 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.

[84] David Pointcheval and Thomas Johansson, editors. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*. Springer, 2012.

[85] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 552–565, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[86] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Communication-efficient anonymous group identification. In Li Gong and Michael K. Reiter, editors, *CCS '98, Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, USA, November 3-5, 1998.*, pages 73–82. ACM, 1998.

[87] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2007.

[88] Shamus Software Ltd. Miracl library. `http://www.shamus.ie/index.php?page=home`.

[89] Peter 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.

[90] Souheil Bcheri, Erik Bjork, Daniel Deibler, Goran Hanell, Jimm Lerch, Maksym Moneta, Monika Orski, Eva Schlehahn, Welderufael Tesfay. D6.3 evaluation of the school pilot. `https://abc4trust.eu/download/Deliverable%20D6.3.pdf`.

[91] Damien Stehlé and Ron 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.

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

[93] Wilson Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum

one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0). Cryptology ePrint Archive, Report 2018/379, 2018. `https://eprint.iacr.org/2018/379`.

[94] Patrick P. Tsang, Man Ho Au, Joseph K. Liu, Willy Susilo, and Duncan S. Wong. A suite of non-pairing id-based threshold ring signature schemes with different levels of anonymity (extended abstract). In Swee-Huay Heng and Kaoru Kurosawa, editors, *Provable Security - 4th International Conference, ProvSec 2010, Malacca, Malaysia, October 13-15, 2010. Proceedings*, volume 6402 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2010.

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

[96] Xu Yang, Wei Wu, Joseph K. Liu, and Xiaofeng Chen. Lightweight anonymous authentication for ad hoc group: A ring signature approach. In Man Ho Au and Atsuko Miyaji, editors, *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, volume 9451 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2015.

[97] Fangguo Zhang and Xiaofeng Chen. Cryptanalysis and improvement of an id-based ad-hoc anonymous identification scheme at ct-rsa 05. *Information Processing Letters*, 109(15):846 – 849, 2009.

[98] Z. Zhang. Raptor source code. online. available from `https://github.com/zhenfeizhang/raptor`.