



**GROUP SIGNATURES WITH ADVANCED
FEATURES FROM LATTICES**

YANHONG XU

**SCHOOL OF PHYSICAL AND MATHEMATICAL
SCIENCES**

2018

GROUP SIGNATURES WITH ADVANCED FEATURES FROM LATTICES

YANHONG XU

School of Physical and Mathematical Sciences

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

To My Father, Mother, and Sisters.

Acknowledgements

First and foremost, I would like to express my gratitude to my main supervisor, Prof. Wang Huaxiong. Without his support, I would not have been a PhD student here. Without his guidance and supervision, I would not have finished my PhD studies. It is a great honor to be his student. He is always a highly motivated researcher, who inspires and instructs me to aim high, target high, and work hard.

I also want to thank my co-supervisors Prof. Ling San and Dr. Nguyen Ta Toan Khoa. I am thankful to Prof. Ling who told me he is always there to support me whenever I have any trouble. I am particularly grateful to Dr. Nguyen Ta Toan Khoa. The first call he dialed gave me a big hope for my PhD studies here. I am deeply thankful to his valuable guidance, support, discussions, and comments. If it were not for him, I would not have finished my PhD on time. Furthermore, he is really enthusiastic about research and enjoys what he is doing, which inspires me and motivates me to study hard.

Special thanks to my cryptographic group members: Dr. Hyung Tae Lee, Dr. Martianus Frederic Ezerman, Dr. Lin Fuchun, Dr. Zhang Juanyang, Dr. Hu Jingwei, Tan Hong Meng Benjamin, Zeng Neng, and Tang Khai Hanh; some visiting professors: Prof. Lin Changlu, Prof. Xiang Can, Prof. Zhang Xiaojun; and some visiting students: Dr. Zhang Yanhua, Dr. Zhang Kai, Zhou Sufang, Wang Shixiong. I have learned a lot from the group meetings and discussions with you.

I would like to thank my seniors Dr. Chen Bocong, Dr. Guo Jian, Dr. Jin Lingfei, Dr. Ma Liming, Dr. Cheng Shantian, Dr. Ivan Tjuawinata, Dr. Yang Yun, and

Dr. Liu Shu, who have offered me a lot of valuable suggestions and help from both academic aspects and daily life. I am also grateful to my friends Peng Mengjiao, Yu Tonghui, Huzhang Guangda, Yuan Bowen, Gao Yang, Liu Xiaoyu, Li Zhe, Huang Rui, Dr. Zhang Han, Dr. Li Chen, and Hong Nankun, with whom I have spent my happiest time in NTU.

Last but not the least, my deepest love goes to my father, my mother, and my sisters, who are always there to support me, to understand me, and to love me. I am deeply indebted for the love you offer. Home is always a term that comforts me whenever I feel tired, frustrated, and even depressed. Now my PhD life has come to an end. However, I hope this is not the end of study.

List of Works

Below is the list of works done during my PhD studies in Nanyang Technological University, in chronological order.

1. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In ACNS 2017, volume 10355 of LNCS, pages 293-312. Springer, 2017.
2. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Constant-size group signatures from lattices. In PKC 2018, volume 10770 of LNCS, pages 58-88. Springer, 2018.
3. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Forward-secure group signatures from lattices. CoRR, abs/1801.08323, 2018. Submitted to PQCrypto 2019.
4. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity (and deniability) with ease. CoRR, abs/1801.08737, 2018. Submitted to Theoretical Computer Science.
5. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Accountable tracing signatures from lattices. To appear in CT-RSA 2019.

Contents

1	Introduction	1
1.1	Group Signatures	1
1.2	Lattice-Based Cryptography	7
1.3	Lattice-Based Group Signatures	15
1.4	Contributions of this Thesis	18
1.5	Organization of this Thesis	28
2	Preliminaries	31
2.1	Notations	31
2.2	Lattices	32
2.2.1	Definition of Lattices	32
2.2.2	Ideal Lattices	34
2.2.3	Some Canonical Lattice Problems	35
2.2.4	SIS and LWE	36
2.2.5	Rings, RSIS and RLWE	37
2.2.6	q -ary Lattices and Gaussian Distributions	39
2.3	Lattice-Based Cryptographic Primitives	41
2.3.1	Trapdoors for Lattices	42
2.3.2	The Bonsai Tree Signatures	45
2.3.3	The Ducas-Micciancio Signatures	46
2.4	Zero-Knowledge Protocols	48

2.4.1	Definition	48
2.4.2	Stern-Like Protocols	51
2.4.3	Decompositions	55
3	Group Signatures from Lattices: Achieving Full Dynamicity (and Deniability) with Ease	58
3.1	Overview of our Techniques	59
3.2	Fully Dynamic Group Signatures	61
3.2.1	Syntax of fully dynamic group signatures.	62
3.2.2	Notions of Correctness and Security requirements	63
3.3	Updatable Lattice-Based Merkle Hash Trees	68
3.3.1	Cryptographic Accumulators	68
3.3.2	The LLNW Merkle-tree Accumulator	69
3.3.3	An Efficient Updating Algorithm	71
3.4	Fully Dynamic Group Signatures from Lattices	72
3.4.1	Description of Our FDGS Scheme	74
3.4.2	Analysis of the Scheme	80
3.4.3	The Main Zero-Knowledge Argument System	88
3.4.4	Zero-Knowledge Argument of Correct Decryption	93
3.5	Achieving Deniability with Ease	95
3.5.1	Fully Dynamic Group Signatures with Deniability	96
3.5.2	Our Construction	100
3.5.3	Zero-Knowledge Argument of “Denial Opening”	105
4	Constant-size Group Signatures from Lattices	107
4.1	Overview of Our Techniques	108
4.2	Dynamic Group Signatures	109
4.2.1	Syntax of Dynamic Group Signatures	109
4.2.2	Notions of Correctness and Security Definitions	110

4.3	ZKAoK for the Ducas-Micciancio Signature	113
4.3.1	A Refined Permuting Technique	114
4.3.2	Zero-Knowledge Protocol for the Ducas-Micciancio Signature . .	118
4.4	Constant-Size Group Signatures from Lattices	123
4.4.1	Description of Our Scheme	124
4.4.2	Analysis of the Scheme	129
4.4.3	The Underlying Zero-Knowledge Argument System	138
5	Forward-Secure Group Signatures from Lattices	142
5.1	Overview of Our Techniques	143
5.2	Forward-Secure Group Signatures	146
5.2.1	Syntax of Forward-Secure Group Signatures	146
5.2.2	Notions of Correctness and Security Requirements	147
5.3	Forward-Secure Group Signatures from Lattices	150
5.3.1	Our Lattice-Based FSGS Scheme	151
5.3.2	Analysis of the Scheme	157
5.3.3	The Main Zero-Knowledge Argument System	165
6	Accountable Tracing Signatures from Lattices	170
6.1	Overview of Our Techniques	171
6.2	Accountable Tracing Signatures	174
6.2.1	Syntax of Accountable Tracing Signatures	174
6.2.2	Definitions of Correctness and Security Requirements	175
6.3	Key-Oblivious Encryption from Lattices	179
6.3.1	Key-Oblivious Encryption	179
6.3.2	Description of Our KOE Scheme	182
6.3.3	Security Analysis of Our KOE Scheme	184
6.4	Proving the RLWE Relation with Hidden Vector	186
6.4.1	An Extended Permutation Technique	187

6.4.2	Proving the RLWE Relation with Hidden Vector	189
6.5	Accountable Tracing Signatures from Lattices	193
6.5.1	Description of Our ATS Scheme	195
6.5.2	Security Analysis of Our Scheme	201
6.5.3	The Main Zero-Knowledge Argument System	210
7	Conclusions and Open Problems	216
	Bibliography	218
	Appendices	238
	Appendix A Proof of Theorem 2.4.4	239

List of Figures

2.1	Stern-type statistical ZKAoK for the NP relation $\varrho_{\text{abstract}}$	54
3.1	An illustration of Merkle tree with $2^3 = 8$ leaves.	72

List of Tables

1.1 Comparison of efficiency and functionality for known group signatures
from lattice assumptions. 29

Abstract

Building cryptographic schemes only on the number-theoretic assumptions such as the hardness of factoring or computing discrete logarithms seems to put us at the risk of the emerging technologies such as the quantum computers. Therefore it is better to design cryptographic schemes based on as many fundamentally different hardness problems as possible. Many lattice problems, such as finding the shortest non-zero vector problem, seem to withstand the quantum adversaries. Furthermore, the worst-case/average-case reduction discovered by Ajtai (STOC 1996) makes lattice-based cryptography outstanding among other fields of cryptography; instead of assuming the hardness of a random instance, it suffices for lattice-based cryptosystems to assume the hardness of lattice problems in the worst case.

Recently, lattice-based cryptography has witnessed a great improvement with many remarkable results such as realization of fully homomorphic encryption by Gentry (STOC 2009). In this thesis, we are particularly interested in lattice-based group signatures. Since the pioneering work by Gordon, Katz, and Vaikuntanathan (Asiacrypt 2010), many other schemes have been suggested. In contrast to their counterparts that are based on number-theoretic assumptions, however, lattice-based group signatures are not fully developed. In particular, most of the existing proposals are designed for static groups and there are only three of them with some advanced features/functionalities, which are motivated by special needs of applications. The three exceptions are the schemes proposed by Langlois et al. (PKC 2014) that is motivated by the desire to revoke some misbehaving/leaving users, by Libert et al. (Asiacrypt 2016) that is designed to admit new users, and by Libert et al. (ACNS 2016) that is driven by the need to open all signatures related to a specific message.

Observing the fact that the ordinary group signatures are often not enough for specific requirements of real-life applications, in this thesis, we construct several lattice-based group signature schemes with different functionalities that can find applications in different situations. Specifically, we design the following lattice-based group signatures with some advanced features.

- We construct the first lattice-based group signature scheme that achieves full dynamicity, where users are able to join and leave the group at any time. Prior to our work, this feature is only achieved from number-theoretic assumptions. In this thesis, we thus solve a prominent open question posed by previous works. In addition, we achieve this non-trivial feat in a relatively simple manner.
- Incorporating a new property - deniability - into the above scheme, we obtain a lattice-based fully dynamic group signature scheme with deniability. Group signature with deniability was first introduced by Ishida et al. (CANS 2016), motivated by a situation where we are more interested in whether a suspect generated a specific signature. In the case the signature was not generated by the suspect, we should be able to generate an evidence showing that this suspect did not produce that signature without revealing the actual signer, thus serving as a way to protect the privacy of the real signer.
- We construct the first lattice-based group signature scheme that features constant-size signatures, which means that the size of the group signatures is independent of the group size N . In particular, all the parameters in our scheme do not depend on N . This result is appealing to applications in which group size is too large.
- We design the first lattice-based forward-secure group signature scheme. Forward-security was put forward by Anderson (Technical report 2002) to minimize the damage caused by the key exposure problem. It preserves the security of the

scheme in previous time periods even when the keys are compromised at current time period. This feature is attractive to group signatures for the following reasons. Once a break in occurs, all prior signed signatures are rendered invalid since it is not clear how to distinguish signatures generated by honest users from those generated by malicious adversaries. In addition, a misbehaving user could repudiate his illegally signed signatures by exposing his secret key somewhere in the internet and then claiming to be the victim of the key exposure problem. What is worse, the damage escalates when the group size increases. However, these problems would not arise if the underlying group signature scheme is forward-secure. Thus group signature with forward-security is imperative in applications that are more susceptible to key exposure attacks.

- We propose the first accountable tracing signatures from lattices. This notion was initially proposed by Kohlweiss et al. (PoPETs 2015) to ensure the accountability of the group manager. In all other group signatures, the group manager is granted too much power, i.e., he is able to open any signature and we do not have any mechanism to verify whether this trust is well placed. This would seriously violate the privacy of all group members. However, in an accountable tracing signature scheme, there is an “accounting” mechanism to check the behaviour of the group manager and thus keeping him accountable.

Chapter 1

Introduction

1.1 Group Signatures

Overview of Group Signatures. Group signature is a fundamental privacy primitive in cryptography. It was first introduced by Chaum and van Heyst [62] in 1991. In a group signature scheme, there is a group manager and a set of group members. Each group member is able to sign messages on behalf of the group. All the generated signatures are anonymous in the sense that it only tells that the signer is one of the group members but nothing about the identity of the signer. However, should the need arise, group manager is able to identify the signer of a message using a tracing mechanism, thus ensuring the accountability of the group members. As a motivated example proposed by Chaum and van Heyst [62], let us consider the following scenario. Suppose a company has several departments and each department has its own printer. It is required that any employee is only allowed to use the printer of his own department. Therefore, each employee should convince the printer that he is from that particular department before printing. From the employees' viewpoint, they would like to hide their identities when using the printer. While unconditional privacy is attractive, everyone understands that privacy is only for those who obey the rules. Therefore, it is desirable that there exists a way to revoke such privacy if

someone misuses the printer. For example, if an employee uses the printer too much, then the director of that department should be able to identify who had misused it and sends him a bill. We can see that the group signature scheme suits this scenario quite well. The director of a particular department is the group manager while all the employees of that department form the group. Before printing, each employee generates a signature proving that he is from that department. Once seeing a valid signature, the printer should be convinced that the one who is using the printer is indeed from that department. Since the signatures do not reveal any identity information, the employee is anonymous to the printer. At the same time, if someone uses the printer too often, the director can identify who misused it using the embedded tracing mechanism.

As another application of group signature scheme, let us consider electronic auction. Suppose a seller offers an item to be sold to some buyers. The buyers submit their bids for the item, attempting to outbid the others'. The winner will be the bidder with the highest bid. We require that only authorized parties can be the potential buyers; otherwise someone might bid for the item but later refuse to buy it. In addition, the buyers do not want their bids to be known to others, which might reveal some sensitive information about them. Furthermore, the seller needs to know the identity of the highest bid (the winner) so that the transaction can be made. In this situation, group signature scheme performs very well. The seller plays the role of the group manager while all the authorized buyers are group members. When the auction starts, only the buyers in the group can generate signatures on their bids for the selling item. By the anonymity property of group signature scheme, all the bids do not reveal their corresponding buyers. Furthermore, the seller can identify the buyer with the highest bid by the opening mechanism.

Group signature scheme has further applications in anonymous credentials [52], electronic cash [58], trusted computing platforms [46], traitor tracing [97], digital rights management [50], and so on.

Formalization of Group Signatures. Since its introduction [62], numerous works on group signatures have emerged. One line of research focuses on augmenting or refining the security requirements, such as framing resistance [64], exculpability [16], coalition resistance [15], and unlinkability [20]. Even though there is a growing list of security requirements for group signatures, they are still not yet clearly defined until 2003, when Bellare, Micciancio, and Warinschi [20] formalized the security notions of group signatures and showed that all previous requirements are implied by only two requirements: full anonymity and full traceability. This conceptual simplification makes it easier to give formal security proofs when there are new group signatures.

Observing that only static groups were considered by Bellare et al. [20], Kiayias and Yung [98], Bellare, Shi, and Zhang [21] extended their work to capture dynamic groups, where users are allowed to join the group at any time, but once they joined the group, they are not allowed to leave. Later, Sakai et al. [165] observed that there is a potential weakness in dynamic group signatures [21, 98]. Specifically, the adversary may claim the ownership of a signature generated by an honest user by forging a proof of ownership. Therefore, they strengthened the models for dynamic group signatures by introducing a new notion of *opening soundness*, ensuring that a valid signature can only be opened to the original signer. Efficient schemes satisfying these models have been constructed both in the random oracle model (e.g., [68, 149]) and the standard model (e.g., [90, 113]).

One concerning problem of group signature scheme is the revocation of group members. For instance, misbehaving members who issue signatures on controversial documents should be revoked from the group. If the scheme does not have a suitable mechanism to handle this case, then the whole system has to be re-initialized, which is obviously an undesirable solution in practice. To address this problem, several different methods have been suggested [28, 30, 44, 51, 55, 144, 148, 171, 174] and some remarkable construction from pairing supporting both revocation and dynamic join were proposed [111, 112, 143]. More recently, Bootle et al. [31] identified some

limitations on previous models, i.e., some users who joined the group at current time is able to sign messages at previous time, which can be seen as an attack against traceability. This motivated them to design a stringent security definition for fully dynamic group signatures, in which users are able to join and leave the group at any time. They also presented a construction satisfying these requirements based on the Decisional Diffie-Hellman (DDH) assumption through a generic transformation from a secure accountable ring signature scheme [32].

Group Signatures with Enhanced Privacy. Notice that in the aforementioned group signature schemes, the group manager has to open all signatures to identify the misbehaving users, which seriously violates the privacy of other rule-abiding group members. Thus it is encouraging if we have some additional mechanism that can selectively open signatures and hence enhancing privacy of the well-behaved group members. In 2004, Kiayias, Tsiounis, and Yung [96] designed a traceable signature scheme, in which the group manager can release a user-specific trapdoor such that all the signatures signed by that user can be identified. This mechanism thus allows selectively linking (hence opening) signatures of a specific (misbehaving) user, which is helpful for protecting privacy of other users. Yet it is important to note that the group manager still has the full power to identify the signer of any message. Hence, the privacy of other users is preserved only against a third party, not the group manager. In 2012, Sakai et al. [164] proposed a group signature scheme with message dependent opening (MDO), which can be seen as providing a dual functionality of traceable signatures by Kiayias et al. [96]. In a group signature scheme with MDO, the group manager can only identify the signers of messages that are admitted by an additional authority named *admitter*. More specifically, when receiving request of opening a signature related to a message from the group manager, the *admitter* produces a message-specific token. Using this token and his secret key, the group manager can identify the signers of all signatures related to that specific message. In other words, group signatures with MDO allow selectively opening of signatures

related to a specific message, without violating privacy of other users that did not sign that message. It can also be seen as a way to restrict the power of the group manager, who may otherwise open signatures indiscriminately. However, we remark that colluding with the admitter, the group manager is able to open all signatures again. Another group signature scheme with additional functionality to enhance privacy is the one with deniability suggested by Ishida et al. [91]. It allows the group manager to generate a digital evidence showing that a given user did not generate a signature in question, thus protecting the privacy of the actual signer. As in the traceable signature scheme [96], the privacy of the actual signer is only against a third party, not the group manager.

Group Signatures Secure Against Key Exposure. Another extension is forward secure group signature (FSGS), which was first introduced by Song [171] in 2001. Forward secure mechanism, first introduced by Anderson [12], aims to minimize the damages caused by key exposures. It requires that the security of the scheme at the previous time is preserved even when the secret key is exposed at current time. As pointed out by Song [171], exposure of signing keys of group members is quite damaging. This motivated Song to use the concept of forward security to minimize the damages caused by key exposure problem. Concretely, she constructed two FSGS schemes in the random oracle model, where group members are able to update their secret keys autonomously at each time period in such a way that it is infeasible to compute the prior keys from the current one. Note that Song’s work [171] appeared at the time when no rigorous security definitions of group signatures were available.

In 2009, Nakanishi, Hira, Funabiki [145] defined a rigorous security model of FSGS for static group, in an analogue of the Bellare et al.’s model [20], and constructed a scheme satisfying their model while also being the first to achieve logarithmic overhead in total number of time periods T for sizes of all parameters. In 2010, Libert and Yung [114] extended Nakanishi et al.’s work [145] to capture the setting of the dynamically growing groups, in an analogue of the Kiayias and Yung’s model [98], and

proposed a scheme that achieved better efficiency than all previous schemes [145,171]. Specifically, in their construction [114], the signature size, signing and verification costs do not depend on T , and other metrics have at most log-squared complexity in T .

Group Signatures with Accountable Tracing. For all the group signatures we have discussed so far, it is implicitly assumed that the group manager is honest and will only revoke anonymity and identify the signer of any message should the need arise. While such a tracing mechanism is necessary to ensure user accountability, it has granted too much power to the group manager, i.e., the group manager can revoke anonymity whenever he wants, and we do not have any method to verify whether this trust is well placed or not. To tackle this issue, Kohlweiss and Miers [99] suggested a new model, named as accountable tracing signature (ATS), which provides an additional mechanism to ensure the accountability of the group manager.

Efficient Group Signatures. Along the line of development of group signatures, numerous works aiming to improve efficiency (and functionality) were also proposed. Ateniese, Camenisch, Joye, and Tsudik (ACJT) [15] presented the first practical scheme, which also offers coalition resistance property. Several works extended the ACJT scheme [15] to achieve better efficiency and functionality, e.g., [51,55,174]. Note that these schemes are based on the strong RSA assumption [17,76]. It is known that group signatures from bilinear maps (pairings) are usually more efficient than those based on the RSA assumption. The first two constructions from pairings were independently discovered by Boneh, Boyen, and Shacham [28] along with by Camenisch and Lysyanskaya [56]. Inspired by these two works, more efficient pairing based constructions were suggested, e.g. [14,25,36,37,68,90,113], some of which achieve constant size signatures. Here constant means that the sizes of group signatures only contain $\mathcal{O}(1)$ number of group elements and are independent of the group size N .

All the group signature schemes mentioned in this section all rely on number-theoretic assumptions, which are vulnerable to the emerging quantum computers.

Hence, it is appealing to have instantiations based on different, post-quantum foundations, e.g., lattice assumptions. In light of this, let us take a closer look at lattices.

1.2 Lattice-Based Cryptography

Overview of Lattices and Its Related Problems. A lattice Λ is a discrete subgroup of $\mathbb{R}^n$ that contains all integer linear combination of n linearly independent vectors $\mathbf{s}_1, \dots, \mathbf{s}_n$. The properties of lattices have been studied for several centuries. For example, in 1842, Dirichlet formulated the problem of finding a short vector in a lattice in the form of simultaneous Diophantine approximation problem; in 1896, Minkowski's theorem about convex, symmetric, bounded bodies can be seen as an estimation for the norm of the shortest nonzero vector [4]. More recently, in 1982, Lenstra, Lenstra, Lovász [104] developed a basis reduction algorithm, which finds a short vector in any lattice over $\mathbb{R}^n$. Since the introduction of the LLL algorithm, lattices have been widely used as tools for cryptanalysis on public key cryptosystems, e.g., [93, 151, 169]. Moreover, many more works have been proposed to improve the LLL algorithm, e.g., [8, 100, 150, 166–168], motivated by their applications to cryptanalysis.

As mentioned above, lattices are employed as tools for cryptanalysis. Indeed, lattices hold great promise for building cryptographic constructions in the post-quantum era, when the quantum computer becomes a reality. It contains a lot of computational problems that are conjectured to be hard to solve even for quantum polynomial-time algorithms. The most well-known lattice problems are Shortest Vector Problem (SVP), the Closest Vector Problem (CVP), and the Shortest Independent Vectors Problem (SIVP). In cryptography, one usually considers the approximation variants of these problems, which are denoted by employing an additional subscript γ as the approximation factor and the goal of which is to output some result whose length is at most γ multiples of the optimal one. These problems are proven to be NP-hard to be exactly solved or to approximate to within small factors, e.g., [5, 49, 69, 70, 95, 133, 158].

Furthermore, it is generally accepted that there is no polynomial-time algorithms for lattice problems even when the magnitude of the approximation factor is polynomial. In fact, the fastest known polynomial-time algorithms can only solve these problems to within sub-exponential approximation factors, e.g., [104], while for polynomial approximation factors, the running time is exponential, e.g., [8, 9]. However, all these results do not justify the potential uses of lattice problems in the post-quantum world. In 1994, Shor [170] gave some algorithms to solve the integer factoring problem and the discrete logarithm problem on a quantum computer with polynomial running time in the input size. Therefore, all the cryptographic constructions based on these problems will become insecure in the post-quantum world. Since Shor’s pioneering discovery, attempts to solve those computational lattice problems have been made using quantum algorithms, but little success has been observed so far and there is no evidence indicating that quantum algorithms outperform the classical ones for lattice problems. Therefore it is reasonable to have the following conjecture.

“There is no polynomial-time (possibly quantum) algorithm that can approximate lattice problems to within polynomial factors.” [141]

Interestingly, some problems on the ideal lattices can be solved by some quantum algorithms while they remain hard on the general lattices (see, e.g., [71]). Nevertheless, the above conjecture is still plausible when restricted on ideal lattices.

Even with the conjectured hardness of lattice problems, lattice-based constructions did not become a reality until the ground-breaking works of Ajtai [4] and Regev [160], where they reduced the worst-case hardness of lattice problems to average-case Short Integer Solution (SIS) problem and Learning With Errors (LWE) problem, respectively. The latter two problems are so versatile and have now served as the basis for numerous lattice-based protocols, which are resistant to quantum computers. Ajtai and Regev’s discoveries are so celebrated also in the way that the security of the cryptographic constructions based on SIS and LWE problems is guaranteed by the *worst-case* hardness

of lattice problems. In other words, breaking the security of those cryptographic constructions would imply efficient (possibly quantum) algorithm to approximate some lattice problems to within polynomial factors in arbitrary general lattice (hence in the worst case). Such strong security guarantee is what makes lattice-based cryptography distinguished among other cryptographic constructions, almost all of which are based on *average-case* hardness. For example, security of constructions based on factoring lies in the assumption that it is hard to factor *some* numbers chosen according to a certain distribution [141]. But a natural question is which distribution we should choose. Obviously, we should avoid some small numbers or even numbers; however, it is not clear whether there exists other numbers we should also avoid. In cryptographic constructions that enjoy the worst-case hardness, such questions would never arise.

Hardness Results of the SIS Problem and Its Ring Variant. Now let us describe in more details about Ajtai’s work [4]. The underlying average-case problem is the **SIS** problem, which was later formalized by Micciancio and Regev [140]. Parameterized by integers n, q, m and a real number β , in the ℓ_p norm ($1 \leq p \leq +\infty$), the **SIS** problem is described as follows: given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\|\mathbf{x}\|_p \leq \beta$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{0} \bmod q$. This problem served as the foundation of the family of one-way functions [4], which was later shown to also be collision resistant by Golderich, Goldwasser, and Halevi [84]. The security of Ajtai’s one-way functions is based on the hardness of the **SIS** problem, which in turn relies on the hardness of the approximation problems SVP_γ and SIVP_γ on arbitrary general lattice (hence in the worst case) with $\gamma = n^c$ for some constant c . As estimated by Cai and Nerurkar [48], the approximation factor in [4] is even larger than n^8 . It is well known that the approximation problems become easier when the approximation factors become larger. Therefore, a lot of subsequent work aiming at decreasing the approximation factors (hence weakening the underlying security assumptions) were proposed [48, 81, 135, 140]. Eventually, the work by Micciancio and Regev [140] obtained approximation factor quasi-linear in n , i.e., $\tilde{\mathcal{O}}(n)$, and the work by Gentry,

Peikert, and Vaikuntanathan [81] obtained approximation factor $\beta \cdot \tilde{\mathcal{O}}(\sqrt{n})$. Another related work by Micciancio and Peikert [139] showed that the hardness of the SIS problem is retained for small modulus, improving the previous result by Gentry et al. [81]. Inspired by Ajtai's pioneering work, the SIS problem has been employed to design collision resistant hash functions [84, 135], commitment scheme [94], identification schemes [94, 123, 142], and digital signature schemes [34, 61, 81, 125, 138, 162].

In addition to the above SIS problem, Micciancio [134] introduced a variant of this problem, known as RSIS problem. Let f be a monic irreducible polynomial of degree n and $q \geq 2$ be a positive integer. Define the rings $R = \mathbb{Z}[X]/(f(X))$ and $R_q = R/(qR)$. Specified by parameters n, q, m, β , in the ℓ_p norm ($1 \leq p \leq +\infty$), the RSIS problem is described as follows: given a uniformly random matrix $\mathbf{A} = [a_1 | \dots | a_m] \in R_q^{1 \times m}$, find a non-zero vector $\mathbf{x} = (x_1, \dots, x_m)^\top \in R^m$ such that $\|\mathbf{x}\|_p \leq \beta$ and $\mathbf{A} \cdot \mathbf{x} = \sum_{i=1}^m a_i \cdot x_i = 0$ over R_q . As in Ajtai's work [4], Micciancio [134] showed that solving the RSIS problem with small non-negligible probability is at least as hard as the approximation problem SIVP_γ on *cyclic* lattices in the worst case with $\gamma = \tilde{\mathcal{O}}(n)$. The RSIS problem was then utilized to construct more efficient one-way functions [134] than the ones based on the SIS problem. Subsequently, Lyubashevsky and Micciancio [126] as well as Peikert and Rosen [155] showed that the one-way functions [134] can be made collision resistant by relying on worst-case hardness of the approximation problem SIVP_γ on *ideal* lattices (a generalization of cyclic lattices). Furthermore, the RSIS problem is exploited to construct collision resistant hash function [128], identification scheme [124], and digital signature schemes [72, 73, 124, 127]. More recently, as observed by Langlois and Stehlé [103, Theorem 3.4], the hardness of the RSIS problem can be based on the SIVP_γ problem on ideal lattice, with $\gamma = \beta \cdot \tilde{\mathcal{O}}(\sqrt{n})$.

On a different but related research line, several public key encryption schemes were proposed [6, 7, 159], based on the worst-case hardness of a specific approximation problem, named as unique-SVP $_\gamma$ problem, in which the shortest non-zero vector of the input lattice is significantly shorter than any other vector that is not parallel to

it. However, this problem is not as well understood as other lattice problems such as SVP_γ and SIVP_γ . Furthermore, the constructions are quite inefficient.

As we already see, the SIS and RSIS problems have served as the basis for one way functions, collision resistant hash functions, commitment schemes, identifications schemes, digital signature schemes, and other “minicrypt” primitives (but not public key encryption). Even though some provably secure public key encryption were proposed [6, 159], they are based on worst-case hardness of some not well-studied lattice problems and rather impractical. Therefore, an interesting open problem is that whether we can construct efficient public encryption schemes from the problems SVP_γ , CVP_γ , SIVP_γ , and other standard and well-studied lattice problems.

Hardness Results of the LWE Problem and Its Ring Variant. In 2005, Regev [160] made a big step in the progress of lattice-based cryptography by introducing the average-case problem LWE and constructed a much more efficient public key cryptosystem (compared to the constructions in [6, 159]) that enjoys the worst-case hardness of the approximation problem SIVP_γ and the decision version of the approximation problem SVP_γ , denoted as GapSVP_γ , on general lattices. The LWE problem, parameterized by positive integers n, q, m , and an error distribution χ over integers $\mathbb{Z}$, is defined as follows: given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector computed as $\mathbf{b} = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} \bmod q$ with $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \chi^m$, the search version of the LWE problem asks to find the secret $\mathbf{s}$ while the decision version of the LWE problem asks to distinguish whether a given pair $(\mathbf{A}, \mathbf{b})$ is computed as above or is uniformly random from $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$. Hereafter, we focus on the decision version of the LWE problem. The LWE problem is so versatile and has served as the foundation for many advanced cryptographic constructions, such as public key encryption schemes that are secure under chosen plaintext attack [80, 115, 156, 160] as well as chosen ciphertext attack [138, 153, 157], oblivious transfer [156], identity based encryption [1, 2, 61, 63, 81, 147], fully homomorphic encryption [38–40, 42, 82], different forms of leakage-resilient schemes [10, 13, 85], attribute based encryption [3, 29, 87, 175],

and predicate encryption [88].

However, a drawback of Regev’s result is that his worst-case to average-case reduction is quantum. In other words, if there exists an efficient (possibly quantum) algorithm that solves the **LWE** problem, then there exists an efficient *quantum* algorithm that solves the worst-case problems SIVP_γ and GapSVP_γ on general lattices. This in turn results in the security of the public key encryption depending on the quantum hardness of the worst-case lattice problems SIVP_γ and GapSVP_γ . This result is meaningful since the underlying worst-case lattice problems are conjectured to be hard even for polynomial-time quantum algorithms. Nevertheless, it would be exciting if the reduction can be made to be classical (non-quantum), giving further confidence on the hardness of the **LWE** problem.

In 2009, Peikert [153] made a further progress in the development of lattice-based cryptography by giving a classical polynomial-time reduction that employs an oracle for solving the **LWE** problem to solve the worst-case lattice GapSVP_γ problem. This naturally gave rise to a public key cryptosystem whose security is based on the worst-case hardness of the problem GapSVP_γ on general lattices. We remark that in Peikert’s classical reduction, it additionally requires an exponentially large modulus q . Therefore, an interesting open question is whether the **LWE** problem is classically hard for small modulus. In 2013, Brakerski et al. [41] gave the first classical hardness reduction for the **LWE** problem with polynomial bounded modulus. This result solves the previous open question in some sense. However, there is a significant loss in the **LWE** parameter. Specifically, Brakerski et al. [41] showed that breaking the **LWE** problem with dimension n^2 (which is also the size of the secret $\mathbf{s}$) and polynomial bounded modulus q implies an efficient algorithm for solving the worst-case lattice problem GapSVP_γ on general lattices with dimension n . This leads to another interesting open question: does there exist a classical reduction for the **LWE** problem (with polynomial bounded modulus) without quadratic loss in the **LWE** parameter? In other words, can we show a classical reduction from some worst-case lattice problem with lattice

dimension n to the LWE problem with the same dimension n and polynomial bounded modulus q ?

On a related research line, several other works showed that the hardness of the LWE problem still holds for the secret drawn from the error distribution [13], as opposed to uniform distribution over $\mathbb{Z}_q^n$, and for error distribution [139] that is uniform over $\{0, 1\}$, as opposed to Gaussian-like distribution. More recently, Peikert, Regev, and Stephens-Davidowitz [154] observed that previous reductions for the LWE problem either impose some constraints on the modulus [13, 137–139, 153, 160] or bear some significant loss in the LWE parameter [41]. This motivated them to give a polynomial-time reduction from worst-case approximation problems SIVP_γ and GapSVP_γ to the average-case LWE problem on general lattices for *any* modulus. Their proof idea is nearly identical to that of Regev [160] and thus there is no parameter loss as opposed to the work by Brakerski et al. [41]. Nevertheless, the reduction is *quantum*, inherited from Regev’s work [160]. A natural question is that whether we can have a classical reduction while maintaining the nice properties in [154]. Will we encounter the problems of exponential modulus as in [153] or parameter loss as in [41] again?

Apart from the LWE problem, a compact variant known as RLWE problem was also considered concurrently and independently by Lyubashevsky, Peikert, and Regev [130] and by Stehlé, Steinfeld, Tanaka, and Xagawa [172]. This variant was then used to construct fully homomorphic encryption [40, 43, 79] and public key encryption scheme that is secure under chosen plaintext attack [130]. Let the rings R and R_q be defined as before. Specified by positive integers n, q, m , and an error distribution χ over the ring R , the RLWE is defined as follows: given a uniformly random matrix $\mathbf{A} \in R_q^{1 \times m}$ and a vector computed as $\mathbf{b} = \mathbf{A}^\top \cdot s + \mathbf{e}$ with $s \xleftarrow{\$} R_q$ and $\mathbf{e} \leftarrow \chi^m$, the search version of the RLWE problem asks to find s while the decision version asks to distinguish between the above tuple $(\mathbf{A}^\top, \mathbf{A}^\top \cdot s + \mathbf{e})$ and a tuple uniformly drawn from $R_q^m \times R_q^m$. From now on, we will use the term RLWE problem to refer to the decision version of this problem. Lyubashevsky, Peikert, and Regev [130] first showed a polynomial-time

quantum reduction from the worst-case approximation problems SVP_γ and SIVP_γ on ideal lattices to the average-case RLWE problem. However, their reduction relies on the properties of a specific modulus q and cyclotomic field or Galois field (instead of R_q that is generally defined above). The condition on q can be waived via an additional “modulus switching” reduction by Langlois and Stehlé [103], which in turn relies on the technique from previous work by Brakerski et al. [41]. As in the case for the reduction to LWE problem by Brakerski et al. [41], this additional gain in [103] is at the cost of loss in the RLWE parameter. More recently, Peikert, Regev, and Stephens-Davidowitz [154] showed that the hardness of the RLWE problem can be retained without the constraints on the modulus q or on the Galois field, with the same or even better approximation factor compared to [41, 130]. This is currently the best result about the hardness of the RLWE problem. However, classical reduction from some worst-case lattice problem on ideal lattices to the average-case RLWE problem is unknown and is left as an open question.

As we have seen above, lattices enjoy the conjectured resistance to quantum computers. Since the seminal works by Ajtai [4] and Regev [160], lattices not only allow to build powerful primitives (e.g., fully homomorphic encryption [40, 78, 82] and predicate encryption [88]) that have no feasible instantiations in conventional number-theoretic cryptography, but they also provide several advantages over the latter, such as provable security under the worst-case hardness assumptions and faster arithmetic operations (that only involve addition and multiplication of integers or polynomials). With all these attractive features, lattice-based cryptography has received noticeable attention in recent years and is moving closer to practice with all the hard work of cryptographers. Along with other primitives, lattice-based group signatures have also attracted the interests of researchers recently. The next section is dedicated for a brief introduction on known lattice-based group signatures.

1.3 Lattice-Based Group Signatures

Lattice-Based Group Signatures in the Static Setting. The first lattice-based group signature scheme was introduced by Gordon, Katz, and Vaikuntanathan (GKV) [89] in 2010. Its construction follows the usual strategy of designing group signatures where it uses the signature scheme of Gentry, Peikert, and Vaikuntanathan (GPV) [81] to produce membership certificate, a variant of Regev encryption scheme [161] to encrypt the certificate, and a zero-knowledge protocol of Micciancio and Vadhan [142] coupled with technique from [75] to prove the well-formedness of the obtained ciphertexts. Concretely, when signing a message, the user hides his own signature on the message among $N - 1$ pseudo-signatures that satisfy the same verification equation without being short. While elegant in its design, this results in linear overhead in the total number of group members N for group public key size and signature size. In 2012, Camenisch, Neven, and Rückert (CNR) [57] proposed anonymous attribute tokens from lattices, which can be seen as generalizations of group signatures. Their construction improves on the GKV construction [89] by achieving constant-size group public key and a stronger notion of anonymity, where the adversary is given access to the opening oracle. Nevertheless, the CNR scheme [57] inherits the linear signature size from the GKV scheme [89]. Therefore, it would be interesting to break the linear signature size barrier. In 2013, Laguillaumie, Langlois, Libert, and Stehlé (LLS) [101] put forward the first scheme with the signature size being logarithmic in N , at the cost of relatively large parameters and hence stronger hardness assumptions than the constructions in [57, 89]. In their construction [101], each group member is assigned an ℓ -bit binary string, where $\ell = \lceil \log N \rceil$, and only needs to encrypt $\ell + 1$ vectors when signing messages, thus removing the linear dependency on N . It is observed that these three constructions [57, 89, 101] are relatively complex. First, the encryption layer in these three schemes has to be initialized together with the signature layer, which in some sense restricts the choice of the encryption layer. In

addition, these schemes employed sophisticated zero-knowledge protocols to prove the honest behaviour of the signing procedure.

In 2015, simpler and more efficient schemes were independently and concurrently proposed by Nguyen, Zhang, and Zhang (NZZ) [152] and Ling, Nguyen, and Wang (LNW) [117]. Concretely, Nguyen et al. [152] designed a new non-interactive zero-knowledge (NIZK) protocol for a simple identity-encoding function and decreased the group public key size and signature size by a factor of $\mathcal{O}(\log N)$ compared to [89, 101]. However, the user secret key in [152] is still a trapdoor matrix as in [89, 101] and the scheme requires larger parameters and hence stronger hardness assumptions. On the other hand, Ling et al. [117] introduced a Stern-type [173] zero-knowledge protocol showing possession of a valid message-signature pair for the underlying Boyen signature scheme [34]. This construction still maintains the logarithmic complexity in N for group public key size and signature size as in the LLS scheme [101] while relying on much weaker assumptions than [101]. In addition, Ling et al. [117] extended their scheme to the setting of ideal lattices, thus saving a factor of λ in the size of the group public key, where λ is the security parameter. More recently, Libert, Mouhartem, and Nguyen (LMN) [110] proposed the first group signature scheme with MDO from lattices. For their purpose, Libert et al. [110] combined the LNW scheme [117] with two layers of identity-based encryption scheme [81]. Then they developed the ZK protocol in [117] to further prove that the double encryption process was conducted honestly. Though confronted with more involved relations than the LNW scheme [117], Libert et al. [110] employed the new strategy for Stern-type protocols proposed in [105] and obtained a much simpler zero-knowledge protocol than [117].

The group signature schemes we have discussed so far all relied on the use of a trapdoor that consists of a short lattice basis, which enables us to efficiently sample from discrete Gaussian distribution over the lattice. However, as stressed by Lyubashevsky [125], the employment of such trapdoor is one of the reasons for the inefficiency of lattice-based signatures. Although lattice-based signature scheme without

trapdoor has already been suggested by Lyubashevsky [125], it seems more challenging for constructing group signature scheme that does not rely on such trapdoor, due to the difficulty in constructing compatible ZK protocols. In 2016, Libert, Ling, Nguyen, and Wang [108] managed to remove the use of such trapdoor by designing a lattice-based accumulator [22] and a Stern-like [173] ZK protocol demonstrating knowledge of a hash chain without revealing it. Avoiding use of such trapdoor results in much smaller parameters and hence leads to substantial efficiency improvements as well as weaker hardness assumptions than [57, 89, 101, 117, 152]. We would like to emphasize that all the aforementioned schemes are designed only for static groups.

Lattice-Based Group Signatures in the Non-Static Setting. There exist several group signature schemes equipped with certain dynamic features from lattices. In 2014, Langlois, Ling, Nguyen, and Wang [102] proposed the first lattice-based group signature scheme with verifier-local revocation (VLR) [30], in which only the verifiers are required to download the up-to-date group information and there is no additional overhead to the non-revoked users. They achieved almost the same asymptotical efficiency as the LLS scheme [101], while depending on weaker assumptions than [101]. Their scheme [102] operated in the model proposed in [30], thus satisfying a weaker notion of anonymity (selfless anonymity), where the adversary has no access to the secret keys of honest users. As one contribution, they presented a Stern-like [173] ZK argument of knowledge of a valid message-signature pair for the Bonsai tree signature scheme [61]. In 2016, Libert, Ling, Mouhartem, Nguyen, and Wang [105] tackled the orthogonal problem of dynamic user enrollments, which was formalized by Kiayias and Yung [98] as well as Bellare et al. [21]. Towards their goal, they employed a variant of the Böhl et al.’s signature scheme [26] to enroll new users. Then when it comes to proving the honest behaviour of the user when signing messages, they provided a ZK protocol allowing to demonstrate the knowledge of a valid message-signature pair for the Böhl et al.’s signature scheme [26].

The main disadvantage of all these lattice-based group signature schemes is that

they are typically not practical, due to the large parameters, large key sizes, and inefficiency of the underlying ZK protocols. Quite recently, Boschini, Camenisch, and Neven [33] proposed a group signature scheme with dynamically growing groups, where the signature size and public key size seem small enough to be used in practice. More recently, Pino, Lyubashevsky, and Seiler [67] presented a even more efficient scheme, by designing a new zero-knowledge proof for proving that a committed value belongs to a particular set of small size.

1.4 Contributions of this Thesis

Overview of Our Motivation. Since its introduction by Chaum and van Heyst [62], numerous group signature schemes from number-theoretic assumptions have been proposed, with various efficiency and functionality. Although we believed that the currently-employed number-theoretic assumptions are secure enough for the near future, how long they will be still secure? Nowadays, when building cryptographic schemes, one should also take into account the emerging technologies that may leads to new attacks. The so-called quantum computers are such an example. Applying Shor’s algorithm [170], factoring-based instantiations are no longer secure once the quantum computers become a reality. Therefore, designing cryptographic schemes that are based on as many fundamentally different hard problems as possible seems to be the best countermeasure against newly appeared or even unexpected threats. Thus, it is desirable to have instantiations based on alternative post-quantum assumptions, such as lattice-based assumptions, code-based assumptions, hash-based assumptions, and multivariate-based assumptions. However, in contrast to other fields of cryptography, lattice-based constructions enjoy the worst-case to average-case reductions, first discovered by Ajtai [4] and further developed by numerous researchers (see e.g. [126, 130, 153–155, 160]). Specifically, the security of lattice-based constructions that are based on the average-case problems SIS and LWE is guaranteed by

the worst-case hardness of lattice problems. Such strong security guarantee makes lattice-based constructions attractive among other cryptographic schemes, almost all of which are based on *average-case* hardness. For example, in the discrete logarithm problem, given a randomly chosen group G of order n generated by a generator g and a group element $A = g^a \in G$, it should be hard to find such a . However, there is no sound way to choose a provably hard group from the family of all groups of order n [163]. In the lattice-based constructions that enjoys the worst-case hardness, such question never arises. Thus, it is quite encouraging to have lattice-based group signatures.

As we have seen, since 2010, there have been many group signature schemes from lattices [57, 89, 101, 102, 105, 108, 110, 117, 152]. However, compared to their number-theoretic-based counterpart, lattice-based group signatures are not fully developed. In particular, there are few group signatures dealing with specific needs of applications, where ordinary group signatures often fall too short. There are only three group signature scheme with advanced features from lattices. The first one is a scheme supporting verifier-local revocation, designed by Langlois et al. [102], which is driven by the need to revoke misbehaving group members. The second one is a scheme that enables dynamic admission of users, constructed by Libert et al. [105], which is motivated by the desire to enroll new users. The third one is a scheme with MDO, proposed by Libert et al. [110]. The group signature scheme with MDO is motivated by several kinds of applications, such as the need to open all signatures related to forbidden messages. The state-of-affairs is somewhat unsatisfactory. These schemes do not cover many applications of group signatures. For example, in some application, both dynamic user enrollment and revocation of members are needed, and thus fully dynamic group signatures are desirable. In another application, the group size may be too large so that it is appealing to have a group signature scheme whose parameters do not depend on the size of the group. In fact, we currently do not have group signatures from lattices that support full dynamicity, deniability, constant-size

signatures, forward-security, and accountability of the group manager, all of which have been achieved in the setting of number theory. This motivates us to design as many lattice-based group signatures with advanced features as possible.

In this thesis, we construct several lattice-based group signature schemes with advanced features. Specifically, we propose schemes that supports full dynamicity, deniability, constant-size signatures, forward-security, and accountability of the group manager from lattices. While we have seen their constructions from number-theoretic assumptions, it is non-trivial to achieve in the setting of lattices due to various reasons. The major one is a lack of suitable zero-knowledge protocols. To construct a secure group signature scheme, it often involves complicated combination of a signature scheme, an encryption scheme, and a zero-knowledge protocol. While there are some efficient zero-knowledge proofs for lattice problems [23, 24, 124, 125, 129, 132] that employ the “Fiat-Shamir with Aborts” technique, it is not clear how to apply their results to achieve such advanced features. Observing the versatility of Stern-like [173] zero-knowledge protocols proposed by Ling et al. [116], we develop their techniques and manage to apply to our cases to achieve additional features. In fact, their Stern-like zero-knowledge protocols [116] are so flexible that they were employed to construct lattice-based group signatures (e.g. [102, 105]), policy-based signatures from lattices [65], code-based group signatures [74], lattice-based group encryption [106], and so on. Now we elaborate on our contributions below.

Motivation 1. The design of group signatures with full dynamicity is motivated by many applications of group signatures. For example, suppose the network account of all students in a university is instantiated using a group signature scheme. Since new students are admitted to the university in the beginning of every academic year, the underlying group signature scheme is required to enroll new users. On the other hand, graduated students should not be able to access the internal resources provided by the university any more. Therefore, group signature schemes supporting both enrollment and revocation are highly desirable. For the time being, all group signature schemes

with full dynamicity [31, 111, 112, 143] are based on number-theoretic assumptions. However, there is no fully dynamic group signature scheme from lattices. This motivates us to work on fully dynamic group signatures from lattices. Furthermore, we observe that achieving partial dynamicity in previous works from lattices [102, 105] is not easy. Thus it is inspiring to even achieve full dynamicity with ease.

Contribution 1. In this thesis, we present the first lattice-based group signature scheme achieving full dynamicity. Our scheme fulfills the stringent security definitions suggested by Bootle et al. [31], relying on the hardness of the SIS and LWE problems. The security analysis of our scheme is in the random oracle model, as all known group signatures from lattices. Given the security parameter λ and total number of group members N , our scheme achieves signature size $\tilde{\mathcal{O}}(\lambda \cdot \log N)$, group public key size $\tilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \log N)$, and user's secret signing key size $\tilde{\mathcal{O}}(\lambda) + \log N$. At each time period when the group information is updated, the verifier is only required to download an extra $\tilde{\mathcal{O}}(\lambda)$ bits for the purpose of performing verification of signatures, while each active signer only needs to download $\tilde{\mathcal{O}}(\lambda \cdot \log N)$ bits. In Table 1.1¹, we give a detailed comparison of efficiency and functionality for our construction and known lattice-based group signatures. The cost of achieving the full dynamicity feature is very reasonable and we do not need to rely on lattice trapdoors. Somewhat surprisingly, our scheme even produces shorter signatures than the scheme from [108] - which is arguably the most efficient group signature scheme from lattices prior to ours. In addition, our scheme is obtained in a relatively simple manner, by extending the static scheme [108] to allow both dynamic enrollment and revocation through an updatable Merkle hash tree, which we describe in more detail in Section 3.1.

Motivation 2. In addition to full dynamicity, another concerning functionality of group signatures that we consider in this thesis is *deniability*, suggested by Ishida et al. [91]. It allows the group manager to provide a digital evidence that a given

¹Note that in this table, we do not include the schemes in [33, 67], in which concrete efficiency is analyzed and whose parameters seem small enough to be practical.

group user did *not* generate a signature in question. In [91], Ishida et al. discussed various situations in which such functionality helps to protect the privacy of users. As a motivated example, let us consider the following scenario. Assume that the entrance and exit control for all buildings in a city is implemented using a group signature scheme. When entering/leaving a building, the user generates a signature on a message like “I, as the signer, entered/left this building at this time”. Now suppose the police has identified a suspect of a crime and is going to check his alibi which is in the form of a signature on a message proving that he was in somewhere else at the time of the crime. However, the claimed alibi might not belong to the suspect. If the police asked the group manager to open the signature, then the privacy of the actual signer is seriously violated. It would be quite helpful if the control system is carried through a group signature scheme with deniability, where the group manager can simply open the signature if it belongs to the suspect or generate a evidence showing that it does not belong to the suspect. In the latter case, the privacy of the real signer is protected (against the police). As shown by Ishida et al. [91], the main technical challenge towards realizing the deniability functionality consists of constructing a zero-knowledge proof/argument that a given ciphertext does not decrypt to a particular message. Such a mechanism is non-trivial to realize in general, but Ishida et al. [91] managed to achieve it from pairing-based assumptions. To the best of our knowledge, this is the only group signature scheme with deniability and there is no such scheme from lattices. This drives us to work on lattice-based group signatures with deniability.

Contribution 2. In this thesis, we provide the first lattice-based group signature scheme with deniability. Concretely, we incorporate the deniability functionality suggested by Ishida et al. [91] into Bootle et al.’s model [31] and then demonstrate how to extend our FDGS scheme to enable the deniability functionality. In the process, we design a zero-knowledge argument of knowledge protocol proving that a given Regev ciphertext does not decrypt to a specific message. Furthermore, this additional fea-

ture does not incur any extra assumptions nor computational overhead, compared to our FDGS scheme.

Motivation 3. In the last quarter of century, numerous group signature schemes have been proposed, some of which produce very short signatures [15, 28]. In the setting of bilinear maps, many schemes [14, 25, 37, 68, 90, 113] achieve constant-size signatures. This property is quite attractive especially when the size of the group is very large. In the lattice setting, however, none of the existing constructions achieved this feature. Due to various reasons, one has to fix the number N in the setup phase. For the schemes from [57, 89, 101, 102, 105, 110, 117, 152], all of which employed full-fledged lattice-based signature schemes [34, 61, 81], this is because that either the syntax of the model they employed [20] takes N as input or their security reductions have to guess a target user with probability $1/N$ and thus cannot go through unless N is known in advance. For the scheme from [108] and our newly constructed FDGS scheme, where group members are assigned the leaves from a Merkle hash tree, this is because the size N of the trees has to be determined so that the setup algorithm can proceed successfully. Consequently, the parameters of all existing lattice-based group signature schemes, including the signature sizes, are inevitably dependent on N . This state-of-affairs is somewhat unsatisfactory, given that constant-size group signature schemes have been achieved in the setting of bilinear maps. This stimulates us to explore the problem of constructing constant-size group signature scheme from the lattice setting.

Contribution 3. In this thesis, we introduce the first lattice-based group signature scheme achieving constant-size signatures. This is the same as in the setting of pairing-based group signatures [37, 90]. The scheme operates in Bellare et al.’s model for partially dynamic groups [21] and is proven secure assuming the hardness of the RSIS and RLWE problems in the random oracle model. The essential difference between our scheme and previous works on lattice-based group signatures is that we do not have to fix N in the setup phase. As a consequence, the execution of the scheme is totally

independent of N . The sizes of the public key, users' signing keys, and signatures are of order $\tilde{O}(\lambda)$, where λ is the security parameter. A comparison between our scheme and the existing lattice-based group signatures, in terms of asymptotic efficiency and functionality, can be found in the Table 1.1.

Motivation 4. Numerous group signature schemes with different security models, different levels of security and functionalities, e.g., [15, 20, 28, 90, 113], were proposed since its introduction by Chaum and van Heyst [62]. Unfortunately, the exposure of group signing keys renders almost all the existing schemes unsatisfactory in practice, in the sense that the security of the scheme is no longer guaranteed when the key exposure happens. As investigated by Song [171], secret key exposure in group signatures is much more damaging than in ordinary digital signatures. If the group signing key of one member is exposed to the adversary, then he can sign arbitrary messages in the name of this innocent user. Once this happened, the whole system has to be reinitialized, which is obviously undesirable in practice. What is worse, all previously generated group signatures were rendered invalid, since we do not know how to distinguish signatures produced by the innocent user from those produced by the adversary. Moreover, some misbehaving group members may repudiate their illegally signed signatures by giving away their group signing key and proclaiming themselves as victims of the key exposure problem. The aforementioned problems induced by the exposure of group signing keys motivated Song [171] to use the notion of forward-security and constructed two forward-secure group signature (FSGS) schemes in the random oracle model. Therefore, when some group member's signing key is compromised, all the signatures generated during past periods remain valid. Furthermore, FSGS prevents dishonest group members from repudiating signatures generated during past periods by simply exposing keys.

Given that Song's work appeared at the time when no rigorous security definitions of group signatures were available, Nakanishi, Hira, and Funabiki [145] defined a formal security model of FSGS for static groups. In addition, they presented a scheme

from pairing that first achieved logarithmic complexity in total number of time periods T for all algorithms and for sizes of all parameters. Subsequently, Libert and Yung [114] extended Nakanishi et al.’s work [145] to allow for dynamic user enrollment and designed a scheme from bilinear maps. Their scheme achieved better efficiency than all previous schemes [145, 171]. More precisely, signature size, signing and verification costs do not depend on total number of time periods T and other metrics are at most log-squared complexity in T . However, all these schemes are constructions based on number-theoretic assumptions and are fragile in the presence of quantum adversaries. Considering the great threat of key exposure to group signatures and the vulnerability of group signatures from number-theoretic assumptions against quantum adversaries, it would be desirable to investigate the problem of designing lattice-based forward-secure group signatures. Furthermore, it would be inspiring to achieve it with reasonable overhead, e.g., with complexity at most poly-logarithmic in T .

Contribution 4. In this thesis, we introduce the first **FSGS** scheme from lattices. The scheme works in Nakanishi et al.’s model [145] for static groups. Under the **SIS** and the **LWE** assumptions, our scheme achieves full anonymity and a stronger notion of traceability named forward-secure traceability, which captures the traceability in the setting of key exposure problems and was formally defined in [145]. Like all previous lattice-based group signatures, our scheme is proven to be secure in the random oracle model. For a security parameter λ , maximum number of group members N , and total time periods T , our scheme features group public key size $\tilde{O}(\lambda^2(\log N + \log T))$, signature size $\tilde{O}(\lambda(\log N + \log T))$, and secret key size $\tilde{O}(\lambda^2(\log N + \log T)^2 \log T)$. In comparison to all known lattice-based group signatures, our scheme incurs a very reasonable overhead, the details are in the Table 1.1.

Motivation 5. In a group signature scheme, to prevent misuse of the signing capability of group members, the group manager is able to open signatures and identify the signer of any message using an embedded opening mechanism. While such a mechanism is necessary to ensure user accountability, it has granted too much power

to the group manager, i.e., the group manager can revoke anonymity whenever he wants and we do not have any method to verify whether this trust is well placed. One possible solution to restrict the power of the opening authority would be a group signature scheme with message-dependent opening (MDO) [164]. Recall that in a group signature scheme with MDO, the group manager can only identify the signers of messages admitted by an additional authority named *admitter*. However, we remark that group signatures with MDO may not solve this issue. Once the group manager gets admission to open a specific message, he can identify all the signers, including some innocent ones, of this specific message. Furthermore, colluding with the *admitter*, the group manager again is able to open all signatures.

To tackle this issue, Kohlweiss and Miers [99] suggested a new model, named accountable tracing signature (ATS), which provides an additional mechanism to ensure the accountability of the group manager. In an ATS scheme, there are two kinds of group users, traceable users and non-traceable users. For traceable users, the group manager can correctly identify them for any message they signed and hence can revoke their anonymity. For non-traceable users, the group manager has no ability to identify them for the messages they signed and hence they are anonymous even to the group manager. When a user joins the group, the group manager first determines whether this user is traceable and then issues corresponding (traceable/nontraceable) certificate to the user. In a later phase, the group manager reveals which user he deems traceable using an “accounting” algorithm, thus serving as a way to check the accountability of the group manager.

As an example, let us consider the surveillance control of a building, which is implemented using an accountable tracing signature scheme. On one hand, the customers in this building would like to have the privacy protected as much as possible. On the other hand, the police who are conducting security check in this building would like to know as much as they can. To balance the interests of these two parties, the police can in advance narrow down some suspects and asks the group manager

to make these suspected users traceable and the remaining non-suspected users non-traceable. To check whether the suspect entered this building, the police can ask the group manager to open all signatures that were used for authentication at the entrance. Since only the suspects are traceable, the group manager can only identify them if they indeed entered this building. However, if a standard group signature scheme (e.g., [15,20,21,28]) were used, then the privacy of innocent users are seriously violated. One might think that a traceable signature scheme suggested by Kiayias, Tsiounis, and Yung [96] would work. By requesting a user-specific trapdoor from the group manager, the police can trace all the signatures created by the suspects. However, this only achieves privacy of innocent users against the *police*, not the *group manager*. In fact, in a traceable signature scheme, the group manager has the full power to identify the signers of all signatures and hence can violate the privacy of all users without being detected. In contrast, if an **ATS** scheme is used, then the group manager must later reveal which users he chose to be traceable, thus ensuring the accountability.

To our knowledge, the **ATS** scheme proposed by Kohlweiss and Miers [99] is the only such scheme and it relies on number-theoretic assumptions which are delicate when confronted with quantum attacks. This motivates us to look into the problem of designing lattice-based accountable tracing signatures.

Contribution 5. In this thesis, we suggest the first accountable tracing signature scheme based on lattice assumptions. The scheme satisfies the security requirements suggested by Kohlweiss and Miers [99], under the hardness of the **RSIS** and the **RLWE** problems. We analyze our scheme in the random oracle model as all known lattice-based group signatures. For a security parameter λ , our **ATS** scheme features group public key size and user secret key size $\tilde{O}(\lambda)$. However, when compared with all known lattice-based group signatures, the accountability (of the group manager) comes at a price: the signature size is of order $\tilde{O}(\lambda^2)$. More details are in the Table 1.1.

Remark 1.4.1. It is worth mentioning that the analyses of our constructions are

based on the theoretical definition of the security parameter λ instead of the practical one. More precisely, we assume that the running time of the best attack against a cryptosystem with security parameter λ should be no less than $\lambda^{\omega(1)}$ instead of 2^λ . As a result, the signature sizes in our constructions are $\tilde{O}(\lambda)$ and $\tilde{O}(\lambda^2)$ according to the theoretical definition while it should be $\tilde{O}(\lambda^2)$ and $\tilde{O}(\lambda^3)$ according to the practical one, in which the latter is much less efficient than the former. Even employing the theoretical definition of λ , we have to admit that the signature sizes of our schemes are still too big to be really useful due to the large poly-logarithmic factor concealed in the notation $\tilde{O}$. Therefore, the contributions of this thesis are more of theoretical interest than practical one. However, we hope that our results will inspire more efficient schemes in the near future.

1.5 Organization of this Thesis

The rest of the thesis is organized as follows.

- In Chapter 2, we first present the basic notions that will be used throughout this thesis, then we recall some background on lattices, some lattice-based cryptographic primitives, and zero-knowledge argument systems.
- In Chapter 3, we introduce our fully dynamic group signature (FDGS) scheme from (general) lattices that operates in the model suggested by Bootle et al. [31]. Further, we embed the deniability function [91] into the Bootle et al.'s model and then show how to extend the obtained FDGS scheme to achieve deniability.
- In Chapter 4, we describe our constant-size group signature scheme from (ideal) lattices. The scheme satisfies the security model proposed by Bellare et al. [21].
- In Chapter 5, we then show forward-secure group signature (FSGS) scheme from (general) lattices in the model of Nakanishi et al. [145].

Table 1.1: Comparison of efficiency and functionality for known group signatures from lattice assumptions.

Scheme	Sig. size	Group PK size	Signer's SK size	Trap- door?	Model	Extra info per epoch	Denia- bility
GKV [89]	$\tilde{O}(\lambda^2 \cdot N)$	$\tilde{O}(\lambda^2 \cdot N)$	$\tilde{O}(\lambda^2)$	yes	static	NA	?
CNR [57]	$\tilde{O}(\lambda^2 \cdot N)$	$\tilde{O}(\lambda^2)$	$\tilde{O}(\lambda)$	yes	static	NA	?
LLLS [101]	$\tilde{O}(\lambda \cdot \ell)$	$O(\lambda^2 \cdot \ell)$	$\tilde{O}(\lambda^2)$	yes	static	NA	?
LLNW [102]	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda^2 \cdot \ell)$	$\tilde{O}(\lambda \cdot \ell)$	yes	VLR	Sign: no Ver: $\tilde{O}(\lambda) \cdot R$	?
NZZ [152]	$\tilde{O}(\lambda + \ell^2)$	$\tilde{O}(\lambda^2 \cdot \ell^2)$	$\tilde{O}(\lambda^2)$	yes	static	NA	?
LNW-I [117]	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda^2 \cdot \ell)$	$\tilde{O}(\lambda)$	yes	static	NA	★
LNW-II [117]	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda) + \ell$	yes	static	NA	★
LLNW [108]	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda^2 + \lambda \cdot \ell)$	$\tilde{O}(\lambda \cdot \ell)$	FREE	static	NA	★
LLM+ [105]	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda^2 \cdot \ell)$	$\tilde{O}(\lambda)$	yes	partially dynamic	NA	★
LMN [110]	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda^2 \cdot \ell)$	$\tilde{O}(\lambda)$	yes	MDO	NA	?
Section 3.4	$\tilde{O}(\lambda \cdot \ell)$	$\tilde{O}(\lambda^2 + \lambda \cdot \ell)$	$\tilde{O}(\lambda) + \ell$	FREE	fully dynamic	Sign: $\tilde{O}(\lambda \cdot \ell)$ Ver: $\tilde{O}(\lambda)$	yes
Section 4.4	$\tilde{O}(\lambda)$	$\tilde{O}(\lambda)$	$\tilde{O}(\lambda)$	yes	partially dynamic	NA	★
Section 5.3	$\tilde{O}(\lambda \cdot e)$	$\tilde{O}(\lambda^2 \cdot e)$	$\tilde{O}(\lambda^2 e^2 \cdot d)$	yes	forward secure	NA	?
Section 6.5	$\tilde{O}(\lambda^2)$	$\tilde{O}(\lambda)$	$\tilde{O}(\lambda)$	yes	ATS	NA	?

The comparison is done based on the following three parameters: security parameter λ , total number of group users $N = 2^\ell$, and total number of time periods $T = 2^d$. Define $e = \ell + d$. As for the scheme from [102], R is the number of revoked users at each time period. In the last column, “★” means that the respective scheme can directly achieve the deniability feature via our technique, while “?” indicates that the possibility of the respective scheme to possess such feature is currently still an open question. For the schemes listed above, only the LNW-II scheme [117], our constant-size group signature scheme described in Section 4.4, and our accountable tracing signature scheme in Section 6.5 are based on assumptions in the setting of ideal lattices while others rely on various SIS and LWE assumptions in the setting of general lattices.

- In Chapter 6, we construct our accountable tracing signature (ATS) scheme from (ideal) lattices, which is proven secure in the model suggested by Kohlweiss and Miers [99].
- In Chapter 7, we conclude this thesis and explore some open questions left behind for further considerations.

In particular, the FDGS scheme presented in Section 3.4 is from our paper [119] published in the proceeding of ACNS 2017. The extended result about achieving deniability described in Section 3.5 is based on our manuscript [122] that was submitted to the journal of Theoretic Computer Science. The constant-size group signature scheme constructed in Section 4.4 is from our paper [120] published in PKC 2018. The FSGS scheme showed in Section 5.3 is from the manuscript [121] that was submitted to the PQCrypto 2019 while the ATS in Section 6.5 is from the manuscript [118] that is to appear in CT-RSA 2019.

Chapter 2

Preliminaries

2.1 Notations

In this section, we define some general notations that are used in this thesis.

All vectors are meant for column vectors unless otherwise specified. Let $\mathbf{a} \in \mathbb{R}^{m_1}$ and $\mathbf{b} \in \mathbb{R}^{m_2}$ be two vectors for positive integers m_1, m_2 . Denote $(\mathbf{a} \parallel \mathbf{b}) \in \mathbb{R}^{m_1+m_2}$, instead of $(\mathbf{a}^\top, \mathbf{b}^\top)^\top$, as the concatenation of these two vectors. In terms of concatenation of two matrices $\mathbf{A} \in \mathbb{R}^{n \times m_1}$ and $\mathbf{B} \in \mathbb{R}^{n \times m_2}$, for $n \in \mathbb{Z}_+$, we use the notion $[\mathbf{A} \mid \mathbf{B}] \in \mathbb{R}^{n \times (m_1+m_2)}$. Let $\mathbf{a} = (a_1, a_2, \dots, a_m)^\top \in \mathbb{R}^m$, define the Euclidean norm as $\|\mathbf{a}\| = \sqrt{\sum_{j=1}^m a_j^2}$, and the infinity norm as $\|\mathbf{a}\|_\infty = \max\{|a_1|, |a_2|, \dots, |a_m|\}$. Let $\mathbf{S} = [\mathbf{s}_1 \mid \mathbf{s}_2 \mid \dots \mid \mathbf{s}_n] \in \mathbb{R}^{n \times n}$ for n -dimensional vectors $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$. Define the length of the matrix $\mathbf{S}$ to be $\|\mathbf{S}\| = \max\{\|\mathbf{s}_1\|, \|\mathbf{s}_2\|, \dots, \|\mathbf{s}_n\|\}$.

For a positive integer B , define the set $\{1, 2, \dots, B\}$ as $[B]$, the set $\{0, 1, \dots, B\}$ as $[0, B]$, and the set containing all the integers from $-B$ to B as $[-B, B]$. Let X be a finite set and χ be a distribution. The expression $s \stackrel{\$}{\leftarrow} X$ means s is chosen uniformly at random from the set X and $s \leftarrow \chi$ means that s is chosen according to the distribution χ . For $a \in \mathbb{R}$, $\log a$ denotes logarithm of a with base 2.

Let λ be a security parameter. An algorithm is said to be efficient if it runs in probabilistic polynomial time (PPT). Let ϵ be a function that maps elements from

$\mathbb{N}$ to non-negative real numbers $\mathbb{R}_{\geq 0}$. It is defined to be negligible in λ , denoted as $\text{negl}(\lambda)$, if for any constant $c > 0$, $\epsilon(\lambda) < \frac{1}{\lambda^c}$ for sufficiently large λ . If an event happens with negligible probability, then we say its complement event occurs with overwhelming probability.

For two distributions χ_1 and χ_2 over a countable set S , define the statistical distance between them as $\Delta(\chi_1, \chi_2) = \frac{1}{2} \sum_{s \in S} |\chi_1(s) - \chi_2(s)|$, where $\chi_i(s)$ means the probability of outputting s according to the distribution χ_i . If the statistical distance between χ_1 and χ_2 is negligible in λ , then we say χ_1 and χ_2 are statistically close, often denoted as $\chi_1 \approx_s \chi_2$. In other words, χ_1 is within negligibly statistical distance from χ_2 .

In this thesis, we use the notation $\mathcal{O}$, Ω , and ω to estimate the running time and object sizes. A function $f(\lambda) = \mathcal{O}(g(\lambda))$ if there exists a constant $c > 0$ such that $|f(\lambda)| \leq c \cdot g(\lambda)$ for sufficiently large λ while $f(\lambda) = \Omega(g(\lambda))$ if there exists a constant $c > 0$ such that $|f(\lambda)| \geq c \cdot g(\lambda)$ for sufficiently large λ . A function $f(\lambda) = \omega(g(\lambda))$ if for any constant $c > 0$ such that $|f(\lambda)| \geq c \cdot |g(\lambda)|$ for sufficiently large λ . We also use the soft-O notation $\tilde{\mathcal{O}}$. A function $f(\lambda) = \tilde{\mathcal{O}}(g(\lambda))$ if there exists a constant $c > 0$ such that $f(\lambda) = \mathcal{O}(g(\lambda) \cdot \log^c n)$.

2.2 Lattices

In this section, we will recall the background on lattices, including the definitions of lattices and ideal lattices, some canonical lattice problems, well-known average-case problems from lattices and ideal lattices, a specific family of integer lattices and discrete Gaussian distributions over these integer lattices.

2.2.1 Definition of Lattices

An n -dimensional lattice Λ over $\mathbb{R}^n$ is a discrete additive subgroup of $\mathbb{R}^n$ defined in the following way.

Definition 2.2.1 (Lattice). Let $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_m$ be vectors over $\mathbb{R}^n$ that are linearly independent. The lattice Λ generated by $\mathbf{S} = [\mathbf{s}_1 | \mathbf{s}_2 | \dots | \mathbf{s}_m] \in \mathbb{R}^{n \times m}$ is denoted as

$$\Lambda(\mathbf{S}) = \left\{ \sum_{k=1}^m \mathbf{s}_k \cdot a_k \mid a_k \in \mathbb{Z} \forall k \in [m] \right\} = \{ \mathbf{S} \cdot \mathbf{a} \mid \mathbf{a} = (a_1, a_2, \dots, a_m)^\top \in \mathbb{Z}^m \}$$

The vectors $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_m$ are referred as the basis of the lattice $\Lambda(\mathbf{S})$. For simplicity, we often refer to the matrix $\mathbf{S}$ as the basis and write Λ instead of $\Lambda(\mathbf{S})$ if no ambiguity occurs. n is said to be the dimension of the lattice Λ and m is said to be the rank of the lattice Λ . In this thesis, we only consider full-rank lattices where the rank and dimension are the same. In other words, $m = n$. If the rank of a lattice Λ is larger than 1, then Λ consists of many bases. Fortunately, there are some invariants of the lattice Λ no matter what basis we choose, which we define below.

Definition 2.2.2 (Determinant). Consider the lattice $\Lambda(\mathbf{S})$ of dimension n . The determinant of Λ , represented as $\det(\Lambda)$, is defined to be $\sqrt{\det(\mathbf{S}^\top \cdot \mathbf{S})} = |\det(\mathbf{S})|$. Another invariant of the lattice Λ is the length of the shortest vector that is *non-zero* in the lattice Λ , denoted as λ_1 . An alternative way to define λ_1 is that it is the smallest positive real number r such that a ball of radius r consists of lattice vector that spans a space of dimension exactly 1. This equivalent definition motivates the following generalization of λ_1 , known as successive minima.

Definition 2.2.3 (Successive minima). Let $\Lambda(\mathbf{S})$ be a lattice of dimension n . The i -th successive minima, for each $i \in [n]$, is defined as

$$\lambda_i(\Lambda) = \inf \{ r \mid \dim (\text{span} (\Lambda(\mathbf{S}) \cap \overline{\mathbf{B}}(\mathbf{0}, r))) = i \},$$

Where $\overline{\mathbf{B}}(\mathbf{0}, r) = \{ \mathbf{a} \in \mathbb{R}^n : \|\mathbf{a}\| \leq r \}$.

Equivalently, the i -th successive minima is the smallest positive real number such that a ball with radius r contains exactly i linearly independent lattice vectors.

Now we recall the Gram-Schmidt orthogonalization procedure that takes as input n linearly independent vectors and outputs n orthogonal vectors. Let $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$ be n linearly independent vectors. Roughly speaking, for the j -th new vector with $j \in [n]$, it is created by subtracting from the original j -th vector $\mathbf{s}_j$ the projection on the span of the previous $j - 1$ new vectors. Formally, it is defined as follows.

Definition 2.2.4. Let $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$ be n linearly independent vectors over $\mathbb{R}^n$. The Gram-Schmidt orthogonalization of these n vectors is a sequence of n new orthogonal vectors $\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, \dots, \tilde{\mathbf{s}}_n$ computed as

$$\tilde{\mathbf{s}}_1 = \mathbf{s}_1; \text{ for } j = 2, \dots, n, \tilde{\mathbf{s}}_j = \mathbf{s}_j - \sum_{k=1}^{j-1} a_{j,k} \tilde{\mathbf{s}}_k, \text{ where we have } a_{j,k} = (\mathbf{s}_j^\top \cdot \tilde{\mathbf{s}}_k) / (\tilde{\mathbf{s}}_k^\top \cdot \tilde{\mathbf{s}}_k).$$

Denote $\tilde{\mathbf{S}} = [\tilde{\mathbf{s}}_1 | \tilde{\mathbf{s}}_2 | \dots | \tilde{\mathbf{s}}_n] \in \mathbb{R}^{n \times n}$. Then $\tilde{\mathbf{S}}$ is named as the Gram-Schmidt orthogonalization of $\mathbf{S} = [\mathbf{s}_1 | \mathbf{s}_2 | \dots | \mathbf{s}_n]$.

2.2.2 Ideal Lattices

Apart from the general lattices defined above, we will work with some specific lattices - ideal lattices - in this thesis. Let $f(X)$ be a monic irreducible polynomial of degree n over the integers. Define a ring $R = \mathbb{Z}[X]/(f(X))$ to be the ring of integer polynomials modulo $f(X)$. Let τ be an additive isomorphism from R to some lattice $\tau(R)$ over $\mathbb{R}^n$ and $\mathcal{I}$ be an ideal of the ring R . Then an ideal lattice defined with respect to the ring R , the isomorphism τ and the ideal $\mathcal{I}$ is simply $\tau(\mathcal{I})$ over $\mathbb{Z}^n$.

For example, let the ring be $R = \mathbb{Z}[X]/(X^n + 1)$, where n is a power of 2. For the additive isomorphism, simply choose the coefficient embedding that maps an element $a = a_0 + a_1 \cdot X + a_2 \cdot X^2 + \dots + a_{n-1} \cdot X^{n-1} \in R$ to $\tau(a) = (a_0, a_1, a_2, \dots, a_{n-1})^\top \in \mathbb{Z}^n$. In terms of the ideal $\mathcal{I}$, choose it to be bR for $b = b_0 + b_1 \cdot X + b_2 \cdot X^2 + \dots + b_{n-1} \cdot X^{n-1} \in R$. Then it is verifiable that the ideal lattice corresponding to the ring R , the isomorphism

τ and the ideal $\mathcal{I} = bR$ is generated by the following matrix/basis:

$$\begin{bmatrix} b_0 & -b_{n-1} & \dots & -b_1 \\ b_1 & b_0 & \dots & -b_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & \dots & b_0 \end{bmatrix}.$$

2.2.3 Some Canonical Lattice Problems

Now we recall some standard computational lattice problems, whose hardness results serve as the foundation of the lattice-based cryptography. The most basic one is the Shortest Vector Problem (SVP) and its approximation variant SVP_γ that are defined formally as follows.

Definition 2.2.5 (SVP, SVP_γ). Let Λ be a lattice of dimension n . The goal of the SVP is to find a vector $\mathbf{a} \in \Lambda$ such that $\|\mathbf{a}\| = \lambda_1(\Lambda)$ while the goal of the approximation variant SVP_γ is to find a non-zero vector $\mathbf{b} \in \Lambda$ such that $\|\mathbf{b}\| \leq \lambda_1(\Lambda) \cdot \gamma$.

In addition to the search version of the SVP_γ problem, we also need the decision version of it, where one is asked to determine whether it is a YES or NO instance.

Definition 2.2.6 (GapSVP $_\gamma$). Let Λ be an n -dimensional lattice and r be a positive rational number. An instance of the GapSVP $_\gamma$ problem is parameterized with the pair (Λ, r) . Return YES if there exists a non-zero vector $\mathbf{a} \in \Lambda$ such that $\|\mathbf{a}\| \leq r$. In other words, we have $\lambda_1(\Lambda) \leq r$. Return NO if for any non-zero vector $\mathbf{b} \in \Lambda$ such that $\|\mathbf{b}\| > r \cdot \gamma$. In this case, we know $\lambda_1(\Lambda) > r \cdot \gamma$.

Another computational lattice problem we consider in this thesis is the Closest Vector Problem (CVP) and its approximation variant CVP_γ .

Definition 2.2.7 (CVP, CVP_γ). Given inputs a n -dimensional lattice Λ and a target vector $\mathbf{c}$, the CVP asks to output a lattice vector $\mathbf{a}$ such that $\|\mathbf{a} - \mathbf{c}\| \leq \|\mathbf{b} - \mathbf{c}\|$ for any other lattice vector $\mathbf{b}$. In other words, $\|\mathbf{a} - \mathbf{c}\| = \text{dist}(\mathbf{c}, \Lambda) \triangleq \min_{\mathbf{b} \in \Lambda} \|\mathbf{b} - \mathbf{c}\|$.

For the approximation variant CVP_γ , the purpose is to output a lattice vector $\mathbf{a}$ such that $\|\mathbf{a} - \mathbf{c}\| \leq \text{dist}(\mathbf{c}, \Lambda) \cdot \gamma$.

The third computational lattice problem we use in this thesis is the Shortest Independent Vector Problem (SIVP) and its approximation variant SIVP_γ .

Definition 2.2.8 ($\text{SIVP}, \text{SIVP}_\gamma$). Given a lattice Λ of dimension n , the target of the SIVP problem is to output n linearly independent lattice vectors $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$ such that $\max_{j \in [n]} \|\mathbf{s}_j\| = \lambda_n(\Lambda)$. In terms of the approximation variant SIVP_γ , the target is then to find n linearly independent lattice vectors $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$ such that $\max_{j \in [n]} \|\mathbf{s}_j\| \leq \lambda_n(\Lambda) \cdot \gamma$.

As conjectured by Micciancio and Regev [141], there is no efficient (possibly quantum) algorithm which can solve the above approximation problems (even when restricted on ideal lattices) to within polynomial factors in the worst case. Even though the conjectured hardness of the lattice problems, lattice-based cryptography was not a reality until the breakthroughs proposed by Ajtai [4] and Regev [160], in which they showed the connection between the worst-case lattice problems and the average-case problems: Short Integer Solution (SIS) problem and Learning With Errors (LWE) problem.

We recall the average-case problems SIS and LWE in the next section, in conjunction with their connection to the worst-case lattice problems.

2.2.4 SIS and LWE

Definition 2.2.9 ($\text{SIS}_{n,m,q,\beta}^\infty$ [4, 81]). Given a uniformly random input matrix $\mathbf{M}$ over $\mathbb{Z}_q^{n \times m}$, the $\text{SIS}_{n,m,q,\beta}^\infty$ problem asks to output a vector $\mathbf{a} \in \mathbb{Z}_q^m$ such that $\mathbf{M} \cdot \mathbf{a} = \mathbf{0} \bmod q$ and $0 < \|\mathbf{a}\|_\infty \leq \beta$.

A related problem we consider in this thesis is the Inhomogeneous Short Integer Solution (ISIS) problem, where the goal is to output a vector $\mathbf{a} \in \mathbb{Z}_q^m$ such that $\mathbf{M} \cdot \mathbf{a} = \mathbf{b} \bmod q$ and $0 < \|\mathbf{a}\|_\infty \leq \beta$ for uniformly random $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{b} \in \mathbb{Z}_q^n$.

Let $q > \beta \cdot \tilde{O}(\sqrt{n})$ be an integer and m, β be polynomials in n . Then solving the $\text{SIS}_{n,m,q,\beta}^\infty$ and $\text{ISIS}_{n,m,q,\beta}^\infty$ problems is at least as hard as solving the SIVP_γ problem in the worst case for some $\gamma = \beta \cdot \tilde{O}(\sqrt{nm})$ ([81, 139, 140]).

Definition 2.2.10 ($\text{LWE}_{n,q,\chi}$ [160]). For positive integers $n, m, q \geq 2$ and a probability distribution χ over integers $\mathbb{Z}$, define a distribution $\mathcal{A}_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ for $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ as follows: it samples a uniformly random vector $\mathbf{a}$ over $\mathbb{Z}_q^n$ and an error element e according to χ , and outputs $(\mathbf{a}, \mathbf{a}^\top \cdot \mathbf{s} + e)$. Then the goal of the $\text{LWE}_{n,q,\chi}$ problem is to distinguish m samples chosen from a uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ from m samples chosen from the distribution $\mathcal{A}_{\mathbf{s},\chi}$ for some $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.

Let $B = \tilde{O}(\sqrt{n})$ and χ be an efficiently samplable distribution over $\mathbb{Z}$ that outputs samples $e \in \mathbb{Z}$ with $|e| \leq B$ with all but negligible probability in n . If $q \geq 2$ is an arbitrary modulus, then the $\text{LWE}_{n,q,\chi}$ problem is at least as hard as the worst-case problems GapSVP_γ and SIVP_γ with $\gamma = \tilde{O}(n \cdot q/B)$ through an efficient quantum reduction (see [40, 41, 153, 154, 160]). Additionally, it is showed that the hardness of the LWE problem is maintained when the secret $\mathbf{s}$ is chosen from the error distribution χ (see [13]).

In addition to the SIS and LWE problems, some security of our constructions in this thesis also relies on the hardness of these two problems when restricted on ideal lattices, denoted as RSIS and RLWE respectively, which we recall in the following section.

2.2.5 Rings, RSIS and RLWE

For an odd positive integer $q > 1$, let $\mathbb{Z}_q = [-\frac{q-1}{2}, \frac{q-1}{2}]$. In this thesis, we are interested in the following rings: $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$ with n being a power of 2.

For $a = a_0 + a_1 \cdot X + \dots + a_{n-1} \cdot X^{n-1} \in R$, we define $\|a\|_\infty = \max_i \{|a_i|\}$. Similarly, for vector $\mathbf{b} = (b_1, \dots, b_m)^\top \in R^m$, we define $\|\mathbf{b}\|_\infty = \max_j \{\|b_j\|_\infty\}$.

We are ready to recall the RSIS and RLWE problems with respect to the rings R, R_q .

Definition 2.2.11 ($\text{RSIS}_{n,m,q,\beta}^\infty$ [126, 134, 155]). Given a uniformly random matrix $\mathbf{M} = [a_1|a_2|\dots|a_m]$ over $R_q^{1 \times m}$, the $\text{RSIS}_{n,m,q,\beta}^\infty$ problem asks to find a ring vector $\mathbf{b} = (b_1, b_2, \dots, b_m)^\top$ over R^m such that $\mathbf{M} \cdot \mathbf{b} = a_1 \cdot b_1 + a_2 \cdot b_2 + \dots + a_m \cdot b_m = 0$ over R_q and $0 < \|\mathbf{b}\|_\infty \leq \beta$.

For polynomial bounded m, β and $q \geq \beta \cdot \tilde{\mathcal{O}}(\sqrt{n})$, it was proven that the $\text{RSIS}_{n,m,q,\beta}^\infty$ problem is no easier than the SIVP_γ problem on *any* ideal lattice with respect to the ring R and an additive isomorphism τ that maps the ring R to some integer lattices over $\mathbb{Z}^n$, where $\gamma = \beta \cdot \tilde{\mathcal{O}}(\sqrt{nm})$ (see [103, 126, 155]).

Definition 2.2.12 ($\text{RLWE}_{n,m,q,\chi}$ [130, 131, 172]). For positive integers $n, m, q \geq 2$ and a probability distribution χ over the ring R , define a distribution $A_{s,\chi}$ over $R_q \times R_q$ for $s \xleftarrow{\$} R_q$ in the following way: it first samples a uniformly random element $a \in R_q$, an error element $e \leftarrow \chi$, and then outputs $(a, a \cdot s + e)$. The target of the $\text{RLWE}_{n,m,q,\chi}$ problem is to distinguish m samples chosen from a uniform distribution over $R_q \times R_q$ and m samples chosen from the distribution $A_{s,\chi}$ for $s \xleftarrow{\$} R_q$.

Let $q \geq 2$ and $B = \tilde{\mathcal{O}}(\sqrt{n})$ be positive integers. χ is a distribution over R which efficiently outputs samples $e \in R$ with $\|e\|_\infty \leq B$ with overwhelming probability in n . Then if there is a polynomial-time (possibly quantum) algorithm that solves the $\text{RLWE}_{n,m,q,\chi}$ problem, then there exists a polynomial-time quantum algorithm that solves the SIVP_γ problem and the SVP_γ problem on *any* ideal lattice with respect to the ring R and an additive isomorphism τ that maps the ring R to some integer lattices over $\mathbb{Z}^n$, where $\gamma = \tilde{\mathcal{O}}(\sqrt{n} \cdot q/B)$ (see [40, 103, 130, 154]). Furthermore, as in the case of the LWE problem, one can equivalently assume the secret s is sampled from the error distribution χ (see [40, 130]).

In this thesis, we employ the notions of coefficient embedding τ that is defined in Section 2.2.2 and a ring homomorphism rot that maps a ring element $a \in R_q$ to a matrix $\text{rot}(a) = [\tau(a) \mid \tau(a \cdot X) \mid \dots \mid \tau(a \cdot X^{n-1})]$ over $\mathbb{Z}_q^n$ (see [136, 175]). Using

these two notions, the element product $c = a \cdot b$ over R_q can be interpreted as the matrix-vector multiplication $\tau(c) = \text{rot}(a) \cdot \tau(b)$ over $\mathbb{Z}_q$.

To deal with vectors and matrices over the ring R_q , we generalize the notions τ and rot in the following manner. For a vector $\mathbf{b} = (b_1, b_2, \dots, b_m)^\top \in R_q^m$, let $\tau(\mathbf{b}) = (\tau(b_1) \parallel \tau(b_2) \parallel \dots \parallel \tau(b_m)) \in \mathbb{Z}_q^{mn}$. For a matrix $\mathbf{M} = [a_1 \mid \dots \mid a_m]$ over $R_q^{1 \times m}$, define $\text{rot}(\mathbf{M})$ to be the matrix

$$\text{rot}(\mathbf{M}) = [\text{rot}(a_1) \mid \dots \mid \text{rot}(a_m)] \in \mathbb{Z}_q^{n \times mn}.$$

Using the generalized notions, we can interpret $c = \mathbf{M} \cdot \mathbf{b}$ over the ring R_q as $\tau(c) = \text{rot}(\mathbf{M}) \cdot \tau(\mathbf{b})$ over $\mathbb{Z}_q$.

2.2.6 q -ary Lattices and Gaussian Distributions

In many lattice-based constructions, people use a specific family of the so-called q -ary lattices, which consists of $q\mathbb{Z}^m$ as a sublattice. In this section, we first recall these q -ary lattices and then review the Gaussian distribution over these lattices.

Let $q \geq 2$ and n, m be positive integers. For $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$, we define the full-rank q -ary lattices with dimension m as follows:

$$\Lambda^\perp(\mathbf{M}) = \{\mathbf{b} \in \mathbb{Z}^m : \mathbf{M} \cdot \mathbf{b} = \mathbf{0} \bmod q\},$$

$$\Lambda(\mathbf{M}) = \{\mathbf{b} \in \mathbb{Z}^m : \mathbf{b} = \mathbf{M}^\top \cdot \mathbf{a} \bmod q \text{ for some } \mathbf{a} \in \mathbb{Z}_q^n\}.$$

For any non-zero vector $\mathbf{z} \in \mathbb{Z}_q^n$ that admits an integral solution to the equation $\mathbf{M} \cdot \mathbf{b} = \mathbf{z} \bmod q$, define the coset

$$\Lambda^{\mathbf{z}}(\mathbf{M}) = \{\mathbf{b} \in \mathbb{Z}^m : \mathbf{M} \cdot \mathbf{b} = \mathbf{z} \bmod q\}.$$

Now let us look at the Gaussian distributions. For any vector $\mathbf{c} \in \mathbb{R}^n$ and any

positive real number s , define the following:

$$\rho_{s,\mathbf{c}}(\mathbf{b}) = \exp\left(-\pi \frac{\|\mathbf{b} - \mathbf{c}\|^2}{s^2}\right) \quad \text{and} \quad \rho_{s,\mathbf{c}}(\Lambda) = \sum_{\mathbf{b} \in \Lambda} \rho_{s,\mathbf{c}}(\mathbf{b}).$$

Then the discrete Gaussian distribution over the lattice Λ with parameter s and center $\mathbf{c}$, denoted as $D_{\Lambda,s,\mathbf{c}}$, is defined to be $D_{\Lambda,s,\mathbf{c}}(\mathbf{b}) = \rho_{s,\mathbf{c}}(\mathbf{b})/\rho_{s,\mathbf{c}}(\Lambda)$ for any $\mathbf{b} \in \Lambda$. We often omit $\mathbf{c}$ if it is $\mathbf{0}$.

Note that the coset $\Lambda^{\mathbf{z}}(\mathbf{M})$ is not a lattice for any non-zero $\mathbf{z} \in \mathbb{Z}_q^n$, since obviously $\mathbf{0}$ is not inside this set. However, we can still define the discrete Gaussian distribution over the coset $\Lambda^{\mathbf{z}}(\mathbf{M})$ in a similar way: $D_{\Lambda^{\mathbf{z}}(\mathbf{M}),s,\mathbf{c}}(\mathbf{b}) = \rho_{s,\mathbf{c}}(\mathbf{b})/\rho_{s,\mathbf{c}}(\Lambda^{\mathbf{z}}(\mathbf{M}))$ for any $\mathbf{b} \in \Lambda^{\mathbf{z}}(\mathbf{M})$.

Now we are going to recall some well-known facts about the discrete Gaussian distributions.

Lemma 2.2.13 ([81, 155]). Let n, q, m be some positive integers such that $q \geq 2$ and $m \geq 2n \log q$. Define a positive real number s such that $s \geq \omega(\sqrt{\log m})$.

- Then for all but a $2q^{-n}$ fraction of all matrices $\mathbf{M}$ over $\mathbb{Z}_q^{n \times m}$, the distribution of the syndrome $\mathbf{z} = \mathbf{M} \cdot \mathbf{b} \bmod q$ for $\mathbf{b} \leftarrow D_{\mathbb{Z}_q^m,s}$ is statistically close to uniform over $\mathbb{Z}_q^n$. Besides, given $\mathbf{M} \cdot \mathbf{b} = \mathbf{z} \bmod q$, the conditional distribution of $\mathbf{b} \leftarrow D_{\mathbb{Z}_q^m,s}$ is $D_{\Lambda^{\mathbf{z}}(\mathbf{M}),s}$.
- Let $a \leftarrow D_{\mathbb{Z},s}$, $t = \log n$, and $\beta = \lceil s \cdot t \rceil$. Then the probability of $|a| > \beta$ is negligible.
- The min-entropy of the distribution $D_{\mathbb{Z}_q^m,s}$ is at least $m - 1$. In other words, for any $\mathbf{b} \in D_{\mathbb{Z}_q^m,s}$, we have $D_{\mathbb{Z}_q^m,s}(\mathbf{b}) \leq 2^{1-m}$.

In this thesis, we are also interested in the ring setting. Therefore we extend the above results specialized to the ring setting defined in Section 2.2.5, following the work by Ducas and Micciancio [72].

Recall the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$, where $q > 1$ is an odd positive integer and n is a power of 2. For any matrix $\mathbf{M} = [a_1|a_2|\cdots|a_m] \in R_q^{1 \times m}$, define:

$$\Lambda^\perp(\mathbf{M}) = \{\mathbf{b} = (b_1, b_2, \dots, b_m)^\top \in R^m : \mathbf{M} \cdot \mathbf{b} = a_1 \cdot b_1 + \cdots + a_m \cdot b_m = 0 \text{ over } R_q\},$$

$$\Lambda^z(\mathbf{M}) = \{\mathbf{b} \in R^m : \mathbf{M} \cdot \mathbf{b} = z \text{ over } R_q\}.$$

Note that under the coefficient embedding τ , we can identify the ring R with n -dimensional vector space $\mathbb{Z}^n$. Therefore, the discrete Gaussian distribution over the ring R , denoted as $D_{R,s}$, can be interpreted as $D_{\mathbb{Z}^n,s}$. Similarly, $D_{R,s}^m$ is interpreted as $D_{\mathbb{Z}^n,s}^m$. We further have the following lemma from [72].

Lemma 2.2.14 ([72]). Let $n \geq 4$ be a power of 2, $q > 1$ be a power of 3, and $m \geq 2\lceil \log q \rceil + 2$ be a integer. Define the rings R and R_q as above. Let s be a real number such that $s \geq \omega(\sqrt{\log nm})$. Then with all but negligible probability over the uniformly random choice of $\mathbf{M}$ over $R_q^{1 \times m}$, the syndrome $z = \mathbf{M} \cdot \mathbf{b}$ for $\mathbf{b} \leftarrow D_{R,s}^m$ is within negligibly statistical distance from the uniform distribution over the ring R .

2.3 Lattice-Based Cryptographic Primitives

In Section 2.2, we recall the basic background on lattices. In this section, we present some lattice-based primitives that will be used throughout this thesis. We first describe some useful algorithms involving trapdoors for lattices. Then we briefly describe the Bonsai tree signature scheme [61] and a variant of Ducas-Micciancio signature scheme [72, 73].

2.3.1 Trapdoors for Lattices

We first recall some algorithms from previous works that will be used in this thesis. The **TrapGen** algorithm is used to generate a matrix $\mathbf{M}$ that is statistically close to random together with a good basis of the q -ary lattice $\Lambda^\perp(\mathbf{M})$. The **SampleD** algorithm employs some good basis of the lattice $\Lambda^\perp(\mathbf{M})$ to output a short vector in $\Lambda^\mathbf{z}(\mathbf{M})$ if $\Lambda^\mathbf{z}(\mathbf{M})$ is not empty.

Lemma 2.3.1 (TrapGen [11]). Let n, m, q be positive integers such that $q \geq 2$ and m is of order $O(n \log q)$. Then the PPT algorithm $\text{TrapGen}(n, m, q)$ outputs a tuple $(\mathbf{M}, \mathbf{T})$ satisfying the following conditions:

- $\mathbf{M}$ is within negligibly statistical distance from the uniformly random distribution over $\mathbb{Z}_q^{n \times m}$,
- $\mathbf{T}$ is a basis for $\Lambda^\perp(\mathbf{M})$, that is, $\mathbf{M} \cdot \mathbf{T} = \mathbf{0} \bmod q$, and
- $\|\tilde{\mathbf{T}}\| \leq \mathcal{O}(\sqrt{n \log q})$.

Lemma 2.3.2 (SampleD [81]). Given a basis $\mathbf{T} \in \mathbb{Z}^{m \times m}$ of the full-rank q -ary lattice $\Lambda^\perp(\mathbf{M})$ for a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$, a vector $\mathbf{z}$ over $\mathbb{Z}_q^n$, and a positive real number $s \geq \|\tilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log n})$, the PPT algorithm $\text{SampleD}(\mathbf{M}, \mathbf{T}, \mathbf{z}, s)$ outputs a vector $\mathbf{b} \in \Lambda^\mathbf{z}(\mathbf{M})$ that is statistically close to the distribution $D_{\Lambda^\mathbf{z}(\mathbf{M}), s}$.

We also need the following two algorithms to securely delegate basis.

Lemma 2.3.3 (ExtBasis [61]). Given a basis $\mathbf{T} \in \mathbb{Z}^{m \times m}$ of the full-rank q -ary lattice $\Lambda^\perp(\mathbf{M})$ for some $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$, and a matrix $\mathbf{M}' \in \mathbb{Z}_q^{n \times m'}$ containing $\mathbf{M}$ as a submatrix, the PPT algorithm $\text{ExtBasis}(\mathbf{T}, \mathbf{M}')$ outputs a basis $\mathbf{T}' \in \mathbb{Z}^{m' \times m'}$ of the q -ary lattice $\Lambda^\perp(\mathbf{M}')$ such that $\|\tilde{\mathbf{T}}'\| = \|\tilde{\mathbf{T}}\|$.

Lemma 2.3.4 (RandBasis [61]). Let Λ be an m -dimensional integer lattice generated by a basis $\mathbf{T} \in \mathbb{Z}^{m \times m}$ and s be a positive real number such that $s \geq \|\tilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log n})$. Then there exists a PPT algorithm $\text{RandBasis}(\mathbf{T}, s)$ that finds a new basis $\mathbf{T}'$ for the

lattice Λ such that $\|\mathbf{T}'\| \leq s \cdot \sqrt{m}$. In addition, for any two bases $\mathbf{T}_0, \mathbf{T}_1$ of Λ and any $s \geq \max\{\|\widetilde{\mathbf{T}}_0\|, \|\widetilde{\mathbf{T}}_1\|\} \cdot \omega(\sqrt{\log n})$, the outputs of the algorithm $\text{RandBasis}(\mathbf{T}_0, s)$ and algorithm $\text{RandBasis}(\mathbf{T}_1, s)$ are within negligibly statistical distance.

Remark 2.3.5. Note that the ExtBasis algorithm is often combined with the algorithm RandBasis to securely delegate basis. A more efficient mechanism for this combined task was described by Micciancio and Peikert [138], who also improved the results of TrapGen and SampleD algorithms. However, we present our scheme in Section 5.3 using the above algorithms due to the fact that the signature scheme employed in Section 5.3 is the Bonsai tree signature scheme [61] that originally used the above ones.

When working in the ring setting, Ducas and Micciancio [72] extended the above algorithms to the ideal lattices using the new notion of lattice trapdoors introduced in [138].

Let n, m, q, k be some positive integers such that $n \geq 4$ is a power of 2, $m \geq 2\lceil \log q \rceil + 2$, and $q = 3^k$. Define the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$. Let s be a real number such that $s \geq \omega(\sqrt{\log nm})$. Define a gadget matrix $\mathbf{G} = [1 \mid 3 \mid 3^2 \mid \dots \mid 3^{k-1}] \in R^{1 \times k}$. We first recall the $\mathbf{G}$ -trapdoor notion adapted from [72, 138].

Definition 2.3.6 ($\mathbf{G}$ -trapdoor [72, 138]). For any matrix $\mathbf{M} \in R_q^{1 \times (m+k)}$, and an invertible tag $h \in R_q$, a $\mathbf{G}$ -trapdoor for $\mathbf{M}$ with tag h is a matrix $\mathbf{R} \in R_q^{m \times k}$ such that

$$\mathbf{M} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I}_k \end{bmatrix} = h \cdot \mathbf{G},$$

where $\mathbf{I}_k$ is the k -dimensional identity matrix over R_q .

For $a = a_0 + a_1 \cdot X + \dots + a_{n-1} \cdot X^{n-1}$, define $\|a\| = \|\tau(a)\| = \sqrt{\sum_{j=0}^{n-1} a_j^2}$, where τ is the coefficient embedding. Similarly, for vector $\mathbf{b} = (b_1, b_2, \dots, b_m)^\top \in R^m$, define

$\|\mathbf{b}\| = \|\tau(b)\|$. The quality of the $\mathbf{G}$ -trapdoor matrix $\mathbf{R} \in R_q^{m \times k}$ for $\mathbf{M}$ is measured by its spectral norm, defined as

$$s_1(\mathbf{R}) = \sup_{\mathbf{b} \in R_q^k} \frac{\|\mathbf{R} \cdot \mathbf{b}\|}{\|\mathbf{b}\|}.$$

As observed by Micciancio and Peikert [138], the rows of the matrix $\begin{bmatrix} \mathbf{R} \\ \mathbf{I}_k \end{bmatrix}$ in Definition 2.3.6 can appear in any order, which only causes the permutation of the columns of $\mathbf{M}$. Therefore, if $\mathbf{R}$ is a trapdoor for $\mathbf{M}$, then it can be made into an equally good trapdoor for any matrix $[\mathbf{M} \mid \mathbf{N}]$ by simply appending suitable zero rows to $\mathbf{R}$ without worsening the quality of the trapdoor. It is easy to see that this is not a secure way to delegate basis.

We are prepared to describe the trapdoor generation algorithm **TrapGen**, the sampling discrete Gaussian algorithm **SampleD**, and the secure delegating basis algorithm **DelBasis** specialized to the ring setting, following [72, 138].

Lemma 2.3.7 (Ring **TrapGen** [72, 138]). Given the integers n, m, q, k , the real number s , and the rings R, R_q as above. For an invertible tag $h \in R_q$, there exists an efficient algorithm **TrapGen** that outputs a tuple $(\mathbf{M}, \mathbf{R}) \in R_q^{1 \times (m+k)} \times R_q^{m \times k}$ such that $\mathbf{M}$ is statistically close to uniform, $\mathbf{R}$ is a $\mathbf{G}$ -trapdoor for $\mathbf{M}$ with tag h , and $s_1(\mathbf{R}) \leq s\sqrt{n} \cdot \mathcal{O}(\sqrt{m} + \sqrt{k} + \omega(\sqrt{\log n}))$.

Lemma 2.3.8 (Ring **SampleD** [72, 138]). Let the parameters be defined as in Lemma 2.3.7. Given the inputs a matrix $\mathbf{M} \in R_q^{1 \times (m+k)}$, a $\mathbf{G}$ -trapdoor $\mathbf{R} \in R_q^{m \times k}$ for $\mathbf{M}$ with tag $h \in R_q$, a syndrome $z \in R_q$, and a parameter $s \geq s_1(\mathbf{R}) \cdot \omega(\sqrt{\log n})$, the PPT algorithm **SampleD**($\mathbf{M}, \mathbf{R}, z, s$) then produces a vector $\mathbf{b} \in \Lambda^z(\mathbf{M})$ that is within negligibly statistical distance from the distribution $D_{\Lambda^z(\mathbf{M}), s}$.

Lemma 2.3.9 (Ring **DelBasis** [72, 138]). Let the parameters be defined as in Lemma 2.3.7. $\mathbf{R} \in R_q^{m \times k}$ is a $\mathbf{G}$ -trapdoor for $\mathbf{M} \in R_q^{1 \times (m+k)}$ with tag $h \in R_q$. Let

$\mathbf{M}' \in R_q^{1 \times (m'+k)}$ be any matrix containing $\mathbf{M}$ as a submatrix such that $m' \geq m + k$ and s be a parameter with $s \geq s_1(\mathbf{R}) \cdot \omega(\sqrt{\log n})$. Then the efficient algorithm DelBasis outputs a $\mathbf{G}$ -trapdoor $\mathbf{R}' \in R_q^{m' \times k}$ for $\mathbf{M}'$ with tag $h \in R_q$. Furthermore, $s_1(\mathbf{R}') \leq s \cdot (\sqrt{m} + \sqrt{k})$.

Note that compared with Lemma 2.3.3, the above delegating basis algorithm additionally requires that $m' \geq m + k$. Fortunately, this is often the case in applications that use delegation.

2.3.2 The Bonsai Tree Signatures

Our forward-secure group signature scheme described in Section 5.3 builds on the bonsai tree signature scheme proposed by Cash, Hofheinz, Katz, and Peikert [61]. Now we describe it briefly. The scheme takes the following parameters: the security parameter λ , an integer $n = \mathcal{O}(\lambda)$, the message length ℓ , a sufficiently large integer $q = \text{poly}(n)$, an integer m of order $\mathcal{O}(n \log q)$, two real numbers $L = \mathcal{O}(\sqrt{n \log q})$ and $s = \omega(\sqrt{\log n}) \cdot L$, and an integer bound $\beta = \lceil s \cdot \log n \rceil$. The verification key is the tuple $(\mathbf{A}_0, \mathbf{A}_1^0, \mathbf{A}_1^1, \dots, \mathbf{A}_\ell^0, \mathbf{A}_\ell^1, \mathbf{u})$ while the signing key is $\mathbf{S}_0$, where $(\mathbf{A}_0, \mathbf{S}_0)$ is generated by the $\text{TrapGen}(n, m, q)$ algorithm as described in Lemma 2.3.1 and matrices $\mathbf{A}_1^0, \mathbf{A}_1^1, \dots, \mathbf{A}_\ell^0, \mathbf{A}_\ell^1$ and vector $\mathbf{u}$ are all uniformly random and independent over $\mathbb{Z}_q^{n \times m}$ and $\mathbb{Z}_q^n$, respectively.

To sign a binary message $\text{id} = (\text{id}[1], \dots, \text{id}[\ell])^\top \in \{0, 1\}^\ell$, the signer forms the matrix $\mathbf{A}_{\text{id}}$ in the following way: $\mathbf{A}_{\text{id}} = [\mathbf{A}_0 | \mathbf{A}_1^{\text{id}[1]} | \dots | \mathbf{A}_\ell^{\text{id}[\ell]}] \in \mathbb{Z}_q^{n \times (\ell+1)m}$. He then outputs a vector $\mathbf{v} \in \Lambda^{\mathbf{u}}(\mathbf{A}_{\text{id}})$ via $\text{SampleD}(\text{ExtBasis}(\mathbf{S}_0, \mathbf{A}_{\text{id}}), \mathbf{u}, s)$. To verify the validity of $\mathbf{v}$ on message id , the verifier computes $\mathbf{A}_{\text{id}}$ as above and checks if $\mathbf{A}_{\text{id}} \cdot \mathbf{v} = \mathbf{u} \bmod q$ and $\|\mathbf{v}\|_\infty \leq \beta$ hold. The authors proved that this signature scheme is existentially unforgeable under non-adaptive chosen message attacks based on the hardness of the SIS problem.

2.3.3 The Ducas-Micciancio Signatures

Our group signature schemes described in Section 4.4 and Section 6.5 employed a variant of the Ducas-Micciancio signature scheme [72, 73]. Specifically, the signature scheme is used to enroll new users to the group. For our purpose, it suffices if the signature scheme supports at most polynomially bounded number of messages since there are only polynomial number of users. Therefore, we choose the stateful version of Ducas-Micciancio signatures, which provide better parameters and tighter security proof. In addition, to achieve adaptive security, we embed the chameleon hash function presented in Appendix B.3 [73] into the above non-adaptively secure stateful version.

Let us look at this variant in details, which works with the following parameters.

- Let n, m, q, k be some positive integers such that $n \geq 4$ is a power of 2, $m \geq 2\lceil \log q \rceil + 2$, and $q = 3^k$. Define the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$. Let s, s_0 be real numbers such that $s \geq \omega(\sqrt{\log nm})$ and $s_0 \geq \sqrt{n} \cdot \omega(\log^2 n)$. Define a gadget matrix $\mathbf{G} = [1 \mid 3 \mid 3^2 \mid \dots \mid 3^{k-1}] \in R^{1 \times k}$.
- For any real constants $c > 1$ and $\alpha_0 \geq \frac{1}{c-1}$, define a series of sets $\mathcal{T}_j = \{0, 1\}^{c_j}$ of lengths $c_j = \lfloor \alpha_0 c^j \rfloor$ for $j \in [d]$, where $d \geq \log_c(\omega(\log n))$. For each tag $t = (t_0, t_1, \dots, t_{c_j})^\top \in \mathcal{T}_j$ for $j \in [d]$, associate it with a ring element $t(X) = \sum_{k=0}^{c_j} t_k \cdot X^k \in R_q$. Let $c_0 = 0$ and then define $t_{[i]}(X) = \sum_{k=c_{i-1}}^{c_i-1} t_k \cdot X^k$ and $t_{[i]} = (t_{c_{i-1}}, \dots, t_{c_i-1})^\top$ for $i \in [j]$. Then one can check $t = (t_{[1]} \| t_{[2]} \| \dots \| t_{[j]})$ and $t(X) = \sum_{i=1}^j t_{[i]}(X)$.
- Let the message dimension be $m_s = \text{poly}(n)$. Also, let $\ell = \lfloor \log \frac{q-1}{2} \rfloor + 1$, and $\bar{m} = m + k$ and $\bar{m}_s = m_s \cdot \ell$. Let $\beta = \tilde{\mathcal{O}}(n)$ be an integer.
- Let $S \in \mathbb{Z}$ be a state initialized to 0.

Given the above parameters, the key generation algorithm runs the **TrapGen** algorithm described in Lemma 2.3.7 to obtain a tuple $(\mathbf{A}, \mathbf{R}) \in R_q^{1 \times \bar{m}} \times R_q^{m \times k}$; samples

$\mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]} \xleftarrow{\$} R_q^{1 \times k}$, $\mathbf{F} \xleftarrow{\$} R_q^{1 \times \ell}$, $\mathbf{F}_0 \xleftarrow{\$} R_q^{1 \times \overline{m}}$, $\mathbf{F}_1 \xleftarrow{\$} R_q^{1 \times \overline{m}_s}$, and $u \xleftarrow{\$} R_q$. It then outputs the secret signing key $\mathbf{R}$ that is a $\mathbf{G}$ -trapdoor for matrix $\mathbf{A}$ with an invertible tag h . For simplicity, choose $h = 1$. The public verification key then consists of the following:

$$\mathbf{A}, \mathbf{F}_0 \in R_q^{1 \times \overline{m}}; \quad \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]} \in R_q^{1 \times k}; \quad \mathbf{F} \in R_q^{1 \times \ell}; \quad \mathbf{F}_1 \in R_q^{1 \times \overline{m}_s}; \quad u \in R_q.$$

To sign a ring vector $\mathbf{m} \in R_q^{m_s}$, decompose $\mathbf{m}$ to a new vector $\overline{\mathbf{m}}$, denoted as $\overline{\mathbf{m}} = \text{rdec}(\mathbf{m}) \in R^{\overline{m}_s 1}$, whose coefficients are in the set $\{-1, 0, 1\}$. The signer then performs the following steps.

- Let $t = (t_0, t_1, \dots, t_{c_d-1})^\top \in \mathcal{T}_d$, where $S = \sum_{j=0}^{c_d-1} 2^j \cdot t_j$, and compute the matrix $\mathbf{A}_t = [\mathbf{A} | \mathbf{A}_{[0]} + \sum_{i=1}^d t_{[i]} \mathbf{A}_{[i]}] \in R_q^{1 \times (\overline{m}+k)}$. Update the state to $S + 1$.
- Choose $\mathbf{r} \in R^{\overline{m}}$ such that $\|\mathbf{r}\|_\infty \leq \beta$.
- Let $y = \mathbf{F}_0 \cdot \mathbf{r} + \mathbf{F}_1 \cdot \overline{\mathbf{m}} \in R_q$ and $u_p = \mathbf{F} \cdot \text{rdec}(y) + u \in R_q$.
- Using secret signing key $\mathbf{R}$, run the algorithm $\text{SampleD}(\mathbf{A}_t, \mathbf{R}, u_p, s_0)^2$ described in Lemma 2.3.8 to obtain a vector $\mathbf{v} \in R^{\overline{m}+k}$.
- Output the tuple $(t, \mathbf{r}, \mathbf{v})$ as a signature for $\mathbf{m} \in R_q^{m_s}$.

To check the validity of a signature $(t, \mathbf{r}, \mathbf{v})$ on message $\mathbf{m} \in R_q^{m_s}$, the verifier first forms the matrix $\mathbf{A}_t$ as above and verifies the following:

$$\begin{cases} \mathbf{A}_t \cdot \mathbf{v} = \mathbf{F} \cdot \text{rdec}(\mathbf{F}_0 \cdot \mathbf{r} + \mathbf{F}_1 \cdot \text{rdec}(\mathbf{m})) + u, \\ \|\mathbf{r}\|_\infty \leq \beta, \quad \|\mathbf{v}\|_\infty \leq \beta. \end{cases}$$

He outputs 1 if all these three conditions hold and 0 otherwise.

¹The notion rdec is formally defined in Section 2.4.3.

²Note that the signer does not need to securely delegate basis, it thus suffices for him to append suitable zero rows to $\mathbf{R}$, serving as the trapdoor for $\mathbf{A}_t$.

Lemma 2.3.10 ([72, 73]). Given at most polynomially bounded number of signature queries, the above variant is existentially unforgeable under adaptive chosen message attacks assuming the hardness of the $\text{RSIS}_{n, \bar{m}, q, \tilde{O}(n^2)}$ problem.

2.4 Zero-Knowledge Protocols

In this section, we describe the Stern-like [173] zero-knowledge argument of knowledge protocols that will be used throughout this thesis. Before that, we first recall the basic definitions of zero-knowledge argument systems. Finally, we review the decomposition technique that was proposed in [116], leading to the versatility of Stern-like protocols.

2.4.1 Definition

Zero-knowledge proof systems, introduced by Goldwasser, Micali, and Rackoff [86], allow a prover to convince a verifier the validity of a statement without leaking any extra information. Before we go to the details, let us first look at the **NP** languages and interactive proof systems.

An **NP** language $\mathcal{L}_\varrho$ is closely related to an **NP** relation ϱ , which is a subset of $\{0, 1\}^* \times \{0, 1\}^*$ such that the membership $(x, w) \in \varrho$ is recognizable in polynomial time. The associated language $\mathcal{L}_\varrho$ is then defined as the set $\{x : \exists w \text{ s.t. } |w| = \text{poly}(|x|) \wedge (x, w) \in \varrho\}$. Fix an **NP** relation and the associated the language $\mathcal{L}_\varrho$. Consider a pair of polynomial-time algorithms $\langle \mathcal{P}, \mathcal{V} \rangle$ with $\mathcal{P}$ being probabilistic and $\mathcal{V}$ being deterministic. We say $\langle \mathcal{P}, \mathcal{V} \rangle$ form an interactive proof system for the language $\mathcal{L}_\varrho$ with perfect completeness and soundness error e if the following two conditions hold:

- **Completeness.** For any $(x, w) \in \varrho$: $\Pr[\text{Out}_{\mathcal{V}}\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1] = 1$.
- **Soundness.** For any $(y, w) \notin \varrho$ and for any $\hat{\mathcal{P}}$: $\Pr[\text{Out}_{\mathcal{V}}\langle \hat{\mathcal{P}}(y, w), \mathcal{V}(y) \rangle = 1] \leq e$,

where $\text{Out}_{\mathcal{V}}$ is the output of the verifier $\mathcal{V}$ that is either 1 or 0 when interacting with the prover. Note that the above soundness condition refers to all potential (possibly computationally unbounded) prover. If it is relaxed to only probabilistic polynomial-time (PPT) prover, then the interactive proof system is named as interactive argument system.

Let us now define statistical zero-knowledge requirement in an interactive argument system, where the zero-knowledge holds against any (possibly unbounded) verifier. Loosely speaking, we say the interactive argument system for a language $\mathcal{L}_{\varrho}$ is zero-knowledge if whatever can be computed after interacting with the prover $\mathcal{P}$ on any $x \in \mathcal{L}_{\varrho}$ can also be computed without interaction with $\mathcal{P}$. This notion is formulated by a simulator $\mathcal{S}$, that is a probabilistic polynomial-time algorithm. Given $x \in \mathcal{L}_{\varrho}$, when interacting with any verifier $\hat{\mathcal{V}}(x)$, the simulator $\mathcal{S}(x)$ is able to output a transcript that is within statistical distance from the real transcript generated by an honest prover $\mathcal{P}(x, w)$ who interacts with $\hat{\mathcal{V}}(x)$.

Definition 2.4.1 (Statistical Zero-Knowledge [83, 86]). Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be an interactive argument system for a language $\mathcal{L}_{\varrho}$. It is said to be statistical zero-knowledge if for any verifier $\hat{\mathcal{V}}$, there exists a PPT simulator $\mathcal{S}$ such that for any $(x, w) \in \varrho$, the following two conditions hold:

- $\Pr[\text{Out}_{\mathcal{S}}\langle \mathcal{S}(x), \hat{\mathcal{V}}(x) \rangle = \perp] \leq 1/2$,
- $\text{View}_{\hat{\mathcal{V}}}\langle \mathcal{S}(x), \hat{\mathcal{V}}(x) \rangle \approx_s \text{View}_{\hat{\mathcal{V}}}\langle \mathcal{P}(x, w), \hat{\mathcal{V}}(x) \rangle$,

where $\text{Out}_{\mathcal{S}}$ is the output of the simulator when interacting with the verifier; and $\text{View}_{\hat{\mathcal{V}}}$ is the view of the verifier, including all the random coins chosen by itself and all the messages received from the simulator or the prover.

A related notion is argument of knowledge, in which the prover not only claims the validity of a statement, but also shows the knowledge of why the statement is valid. This notion is formulated by a PPT algorithm knowledge extractor $\mathcal{K}$. Given

access to a possibly cheating PPT prover $\hat{\mathcal{P}}$ who can convince the verifier the validity of a statement $x \in \mathcal{L}_\varrho$ with some probability, the knowledge extractor $\mathcal{K}$ is able to output a witness w for the assertion that $x \in \mathcal{L}_\varrho$ with a related probability.

Definition 2.4.2 (Argument of Knowledge [19, 83]). Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be an interactive argument system for a language $\mathcal{L}_\varrho$ with knowledge error κ . For any $x \in \mathcal{L}_\varrho$ and any PPT prover $\hat{\mathcal{P}}$, let $\epsilon(x)$ be the probability that the verifier $\mathcal{V}$ accepts on input x . The system is said to be argument of knowledge if there exists a PPT knowledge extractor $\mathcal{K}$ who is given x and a rewindable black-box access to the prover $\hat{\mathcal{P}}$, and is able to output a witness w such that $(x, w) \in \varrho$ with probability at least $\text{poly}(\epsilon(x) - \kappa)$.

Note that one can think of the knowledge error κ as the probability the prover convinces the verifier without knowledge of the witness.

Let us now look at an interactive protocol for example between two PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ for an NP relation ϱ and the related language $\mathcal{L}_\varrho$. Given a common input $x \in \mathcal{L}_\varrho$, the interaction between $\mathcal{V}$ and $\mathcal{P}$ who is given an additional private input w such that $(x, w) \in \varrho$ is as follows:

- Prover $\mathcal{P}$ first sends a message a that acts as a commitment to the verifier $\mathcal{V}$;
 - $\mathcal{V}$ sends back a random t -bit string e as a challenge when seeing the commitment;
 - $\mathcal{P}$ then computes a reply z and sends it to $\mathcal{V}$ as a response to the challenge;
- Seeing all these data, $\mathcal{V}$ decides to accept or reject according to the validity of the response z with respect to the commitment a and challenge e .

Definition 2.4.3 (Σ -Protocol [66]). The above interactive protocol is said to be Σ -protocol for relation ϱ if the following three conditions hold:

- **Completeness.** For any $(x, w) \in \varrho$, $\Pr[\text{Out}_\mathcal{V}(\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle) = 1] = 1$.
- **Special Soundness.** Given any x and any two accepting transcripts on x , (a, e, z) and (a, e', z') satisfying $e \neq e'$, there exists a PPT algorithm $\mathcal{E}$ that can efficiently compute w' such that $(x, w') \in \varrho$.

- **Special Honest-Verifier Zero-Knowledge.** On inputs $x \in \mathcal{L}$ and a random t -bit string e , there exists a PPT simulator $\mathcal{S}$ that outputs (a, z) such that the transcript (a, e, z) has the same distribution as the one generated by honest prover on input x who knows the corresponding witness.

In [66], Damgård showed that the Σ -protocol is proof of knowledge with knowledge error 2^{-t} . Note that the above definition can be generalized in the sense that: (i) for any $(x, w) \in \varrho$, the verifier accepts with probability at least $1 - \alpha$ for some $\alpha \in [0, 1)$; (ii) the algorithm $\mathcal{E}$ outputs a pair (x, w') such that $(x, w') \in \varrho'$, where $\varrho \subseteq \varrho'$. More details can be found in [23, 92].

2.4.2 Stern-Like Protocols

The statistical zero-knowledge arguments of knowledge (ZKAoK) protocols presented in this thesis are Stern-like protocols [173]. In 1996, Stern [173] suggested a three-move zero-knowledge proof of knowledge for the well-known syndrome decoding (SD) problem. In particular, the proof system is a Σ -protocol in the generalized sense [23, 92] (where 3 accepting transcripts are needed for the special soundness, instead of just 2). In 2008, Kawachi, Tanaka, Xagawa [94] adapted Stern's technique to the lattice setting, obtaining a zero-knowledge proof of knowledge system for a restricted version of ISIS^∞ problem. Specifically, they designed a zero-knowledge proof system for the following relation:

$$\varrho_{\text{KTX}} = \{((\mathbf{M}, \mathbf{u}), \mathbf{w}) \in (\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n) \times \{0, 1\}^m : \text{wt}(\mathbf{w}) = w \text{ and } \mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q\},$$

where $\text{wt}(\mathbf{w})$ is the hamming weight of the vector $\mathbf{w}$. As observed by Ling, Nguyen, Stehlé, and Wang [116], this restricted relation is not enough for many applications (e.g. [57, 89, 172]) where the secret key $\mathbf{w}$ is usually a general solution to the $\text{ISIS}_{n,m,q,\beta}^\infty$ problem. Elegantly, Ling et al. [116] showed a Stern-like zero-knowledge proof of

knowledge protocol for the exact relation associated to the $\text{ISIS}_{n,m,q,\beta}^\infty$ problem.

$$\mathcal{Q}_{\text{ISIS}_{n,m,q,\beta}^\infty} = \{((\mathbf{M}, \mathbf{u}), \mathbf{w}) \in (\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n) \times \mathbb{Z}^m : \|\mathbf{w}\|_\infty \leq \beta \text{ and } \mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q.\}$$

At the heart of their construction [116] are decomposition and extension techniques that remove the restriction on the ISIS^∞ problem as in the work by Kawachi et al. [94]. If a statistically hiding and computationally binding string commitment scheme, such as the KTX scheme [94], is employed in the first commitment step, then one actually obtains a statistical ZKAoK with perfect completeness, constant soundness error $2/3$, and knowledge error $2/3$. In many applications, the protocol is repeated $\kappa = \omega(\log \lambda)$ times to make the soundness error negligibly small in λ . The Ling et al.'s zero-knowledge protocol [116] is so versatile that it finds applications in designing an improved identity-based identification scheme [116], group signature schemes [102, 117], and a lattice-based accumulator with supporting zero-knowledge argument of knowledge of membership [108], and so on. In 2016, Libert, Ling, Mouhartem, Nguyen, and Wang [105] generalized the Stern's protocol to address the flexibility of it. Together with the decomposition and extension techniques [116], one is able to obtain a statistical ZKAoK of a secret vector $\mathbf{w}$ that satisfies certain constraints (such as small norms or special arrangement of the coordinates) and a linear equation of the form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q$ for public $\mathbf{M}, \mathbf{u}$. Now let us recall the abstract form of the Stern's protocol proposed in [105].

Let q, K, L be positive integers such that $q \geq 2$, $L \geq K$, and $\text{VALID} \subseteq \{-1, 0, 1\}^L$. Given a finite set $\mathcal{S}$, associate every $\eta \in \mathcal{S}$ with a permutation Γ_η of L elements such that the following conditions hold:

$$\begin{cases} \mathbf{w} \in \text{VALID} \iff \Gamma_\eta(\mathbf{w}) \in \text{VALID} \text{ for any } \eta \in \mathcal{S}, \\ \text{If } \mathbf{w} \in \text{VALID} \text{ and } \eta \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\eta(\mathbf{w}) \text{ is uniform in } \text{VALID}. \end{cases} \quad (2.1)$$

Our target is to construct a statistical ZKAoK for the following abstract relation:

$$\mathcal{Q}_{\text{abstract}} = \left\{ (\mathbf{M}, \mathbf{u}), \mathbf{w} \in (\mathbb{Z}_q^{K \times L} \times \mathbb{Z}_q^K) \times \text{VALID} : \mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q. \right\}$$

To obtain the desired ZKAoK protocol, one has to prove that $\mathbf{w} \in \text{VALID}$ and $\mathbf{w}$ satisfies the linear equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q$. To prove $\mathbf{w} \in \text{VALID}$ in a zero-knowledge manner, the prover chooses $\eta \xleftarrow{\$} \mathcal{S}$ and allows the verifier to check $\Gamma_\eta(\mathbf{w}) \in \text{VALID}$. According to the first condition in (2.1), the verifier should be convinced that $\mathbf{w}$ is indeed from the set VALID . At the same time, the verifier cannot learn any extra information about $\mathbf{w}$ due to the second condition in (2.1). Furthermore, to prove in ZK that the linear equation holds, the prover first chooses $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_q^L$ as a masking vector and then shows the verifier that the equation $\mathbf{M} \cdot (\mathbf{w} + \mathbf{r}_w) = \mathbf{M} \cdot \mathbf{r}_w + \mathbf{u} \bmod q$ holds.

In Figure 2.1, we describe in details the interaction between two PPT algorithms prover $\mathcal{P}$ and verifier $\mathcal{V}$. The system utilizes a statistically hiding and computationally binding string commitment scheme COM , e.g., the SIS-based or RSIS-based scheme from [94], which depends on the environment (the setting of general lattices or ideal lattices) it is being used.

Theorem 2.4.4 ([105]). Let COM be a statistically hiding and computationally binding string commitment scheme. Then the interactive protocol depicted in Figure 2.1 is a statistical ZKAoK with perfect completeness, soundness error $2/3$, knowledge error $2/3$, and communication cost $\mathcal{O}(L \log q)$. Specifically:

- There exists a PPT simulator that on input $(\mathbf{M}, \mathbf{u})$, with probability $2/3$ it outputs an accepted transcript that is within statistical distance from the one produced by an honest prover who knows the witness.
- There exists a polynomial-time algorithm $\mathcal{E}$ that, takes as inputs $(\mathbf{M}, \mathbf{u})$ and three accepting transcripts on $(\mathbf{M}, \mathbf{u})$, $(\text{CMT}, 1, \text{RSP}_1)$, $(\text{CMT}, 2, \text{RSP}_2)$, and

1. **Commitment:** The prover $\mathcal{P}$ samples $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_q^L$, $\eta \xleftarrow{\$} \mathcal{S}$ and randomness ρ_1, ρ_2, ρ_3 for COM. Then he sends $\text{CMT} = (C_1, C_2, C_3)$ to $\mathcal{V}$, where

$$C_1 = \text{COM}(\eta, \mathbf{M} \cdot \mathbf{r}_w \bmod q; \rho_1), \quad C_2 = \text{COM}(\Gamma_\eta(\mathbf{r}_w); \rho_2), \\ C_3 = \text{COM}(\Gamma_\eta(\mathbf{w} + \mathbf{r}_w \bmod q); \rho_3).$$

2. **Challenge:** $\mathcal{V}$ sends back a challenge $Ch \xleftarrow{\$} \{1, 2, 3\}$ to $\mathcal{P}$.
3. **Response:** According to the choice of Ch , the prover $\mathcal{P}$ sends RSP computed in the following way:

- $Ch = 1$: Let $\mathbf{t}_w = \Gamma_\eta(\mathbf{w})$, $\mathbf{t}_r = \Gamma_\eta(\mathbf{r}_w)$, and $\text{RSP} = (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$.
- $Ch = 2$: Let $\eta_2 = \eta$, $\mathbf{w}_2 = \mathbf{w} + \mathbf{r}_w \bmod q$, and $\text{RSP} = (\eta_2, \mathbf{w}_2, \rho_1, \rho_3)$.
- $Ch = 3$: Let $\eta_3 = \eta$, $\mathbf{w}_3 = \mathbf{r}_w$, and $\text{RSP} = (\eta_3, \mathbf{w}_3, \rho_1, \rho_2)$.

Verification: When receiving RSP from the verifier $\mathcal{P}$, the prover $\mathcal{V}$ performs as follows:

- $Ch = 1$: Verify that $\mathbf{t}_w \in \text{VALID}$, $C_2 = \text{COM}(\mathbf{t}_r; \rho_2)$, $C_3 = \text{COM}(\mathbf{t}_w + \mathbf{t}_r \bmod q; \rho_3)$.
- $Ch = 2$: Verify that $C_1 = \text{COM}(\eta_2, \mathbf{M} \cdot \mathbf{w}_2 - \mathbf{u} \bmod q; \rho_1)$, $C_3 = \text{COM}(\Gamma_{\eta_2}(\mathbf{w}_2); \rho_3)$.
- $Ch = 3$: Check that $C_1 = \text{COM}(\eta_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1)$, $C_2 = \text{COM}(\Gamma_{\eta_3}(\mathbf{w}_3); \rho_2)$.

In each case, if all the conditions hold, $\mathcal{V}$ outputs 1.

Figure 2.1: Stern-type statistical ZKAoK for the NP relation $\varrho_{\text{abstract}}$.

(CMT, 3, RSP₃), outputs $\mathbf{w}' \in \text{VALID}$ such that $\mathbf{M} \cdot \mathbf{w}' = \mathbf{u} \bmod q$. (This is indeed the **Special Soundness** of the Σ -protocols defined in Definition 2.4.3.)

We recall the proof in Appendix A for completeness.

Looking ahead, all the relations (Sections 3.4.3, 4.3.2, 4.4.3, 5.3.3, 6.4.2, and 6.5.3) we consider in this thesis will be reduced to instances of the above abstract protocol.

2.4.3 Decompositions

We now recall the integer decomposition technique from [116], which is crucial in obtaining an instance of the above abstract protocol. For any positive integer B , let $\delta_B := \lfloor \log_2 B \rfloor + 1 = \lceil \log_2(B+1) \rceil$ and the sequence $B_1, \dots, B_{\delta_B}$, where $B_j = \lfloor \frac{B+2^{j-1}}{2^j} \rfloor$, for any $j \in [\delta_B]$. It is then verifiable that $\sum_{j=1}^{\delta_B} B_j = B$. In addition, for any integer $a \in [0, B]$, one can decompose a into a vector of the form $\text{idec}_B(a) = (a^{(1)}, a^{(2)}, \dots, a^{(\delta_B)})^\top \in \{0, 1\}^{\delta_B}$, satisfying that $(B_1, B_2, \dots, B_{\delta_B}) \cdot \text{idec}_B(a) = a$. The procedure of the decomposition is presented below in a deterministic manner.

1. $a' := a$.
2. For $j = 1$ to δ_B do:
 - (i) If $a' \geq B_j$ then $a^{(j)} := 1$, else $a^{(j)} := 0$;
 - (ii) $a' := a' - B_j \cdot a^{(j)}$.
3. Output $\text{idec}_B(a) = (a^{(1)}, a^{(2)}, \dots, a^{(\delta_B)})^\top$.

When dealing with vectors of dimension $m \in \mathbb{Z}_+$ and of range $[-B, B]$, we define $\text{vdec}_{m,B}$ that maps $\mathbf{a} = (a_1, a_2, \dots, a_m)^\top$ to a vector of the following form:

$$\mathbf{a}' = (\sigma(a_1) \cdot \text{idec}_B(|a_1|) \parallel \sigma(a_2) \cdot \text{idec}_B(|a_2|) \parallel \dots \parallel \sigma(a_m) \cdot \text{idec}_B(|a_m|)) \in \{-1, 0, 1\}^{m\delta_B},$$

where $\forall j \in [m]$: $\sigma(a_j) = 0$ if $a_j = 0$; $\sigma(a_j) = -1$ if $a_j < 0$; $\sigma(a_j) = 1$ if $a_j > 0$.

Define a matrix $\mathbf{G}_{m,B} \in \mathbb{Z}^{m \times m\delta_B}$ to be

$$\mathbf{G}_{m,B} = \begin{bmatrix} B_1 \dots B_{\delta_B} & & \\ & \ddots & \\ & & B_1 \dots B_{\delta_B} \end{bmatrix}$$

Then we have

$$\mathbf{a} = \mathbf{G}_{m,B} \cdot \mathbf{vdec}_{m,B}(\mathbf{a}). \quad (2.2)$$

The above decomposition procedure is also utilized to deal with polynomials in the ring R_q . Recall that in this thesis we let $\mathbb{Z}_q = [-\frac{q-1}{2}, \frac{q-1}{2}]$. For $B \in [1, \frac{q-1}{2}]$, define the injective function $\mathbf{rdec}_B : R_q \mapsto R^{\delta_B}$ that maps $a \in R_q$ with $\|a\|_\infty \leq B$ to $\mathbf{a} \in R^{\delta_B}$ with $\|\mathbf{a}\|_\infty \leq 1$. The function $\mathbf{rdec}_B$ is described in detail below.

1. Let $\tau(a) = (a_0, \dots, a_{n-1})^\top$. For each i , let $\sigma(a_i) = 0$ if $a_i = 0$; $\sigma(a_i) = -1$ if $a_i < 0$; and $\sigma(a_i) = 1$ if $a_i > 0$.
2. $\forall i$, compute $\mathbf{w}_i = \sigma(a_i) \cdot \mathbf{idec}_B(|a_i|) = (w_{i,1}, \dots, w_{i,\delta_B})^\top \in \{-1, 0, 1\}^{\delta_B}$.
3. Form the vector $\mathbf{w} = (\mathbf{w}_0 \parallel \dots \parallel \mathbf{w}_{n-1}) \in \{-1, 0, 1\}^{n\delta_B}$, and let $\mathbf{a} \in R^{\delta_B}$ be the vector such that $\tau(\mathbf{a}) = \mathbf{w}$.
4. Output $\mathbf{rdec}_B(a) = \mathbf{a}$.

To deal with ring vectors of dimension $m \in \mathbb{Z}_+$ and of infinity bound $B \in \mathbb{Z}_+$, we generalize the notion $\mathbf{rdec}_B(\mathbf{v})$ as follows: it maps a ring vector $\mathbf{v} = (v_1, \dots, v_m)^\top \in R_q^m$ such that $\|\mathbf{v}\|_\infty \leq B$ to a vector $\mathbf{rdec}_B(\mathbf{v}) = (\mathbf{rdec}_B(v_1) \parallel \dots \parallel \mathbf{rdec}_B(v_m)) \in R^{m\delta_B}$, whose coefficients are in the set $\{-1, 0, 1\}$. Similarly, define matrices $\mathbf{H}_B \in \mathbb{Z}^{n \times n\delta_B}$

and $\mathbf{H}_{m,B} \in \mathbb{Z}^{nm \times nm\delta_B}$ to be

$$\mathbf{H}_B = \begin{bmatrix} B_1 \dots B_{\delta_B} & & \\ & \ddots & \\ & & B_1 \dots B_{\delta_B} \end{bmatrix}, \quad \text{and} \quad \mathbf{H}_{m,B} = \begin{bmatrix} \mathbf{H}_B & & \\ & \ddots & \\ & & \mathbf{H}_B \end{bmatrix}.$$

Then we have

$$\tau(a) = \mathbf{H}_B \cdot \tau(\text{rdec}_B(a)) \bmod q \quad \text{and} \quad \tau(\mathbf{v}) = \mathbf{H}_{m,B} \cdot \tau(\text{rdec}_B(\mathbf{v})) \bmod q.$$

For simplicity purpose, when $B = \frac{q-1}{2}$, we will use the notion rdec instead of $\text{rdec}_{\frac{q-1}{2}}$, and matrix $\mathbf{H}$ instead of $\mathbf{H}_{\frac{q-1}{2}}$.

Chapter 3

Group Signatures from Lattices: Achieving Full Dynamicity (and Deniability) with Ease

Full dynamicity is a desirable functionality, which enables the admission of new users and at the same time providing the feature for revocation of members. However, all existing group signatures with full dynamicity [31, 111, 112, 143] are based on number-theoretic assumptions and there is no lattice-based fully dynamic group signature scheme so far. In this chapter, we present the first lattice-based group signature that offers full dynamicity. Thus it is the first FDGS scheme that is secure against quantum attacks. Moreover, we achieve this non-trivial feat in a relatively simple manner. Starting with Libert et al.’s fully static construction [108] - which is arguably the most efficient lattice-based group signature to date, we introduce simple-but-insightful tweaks that allow for direct upgrade to the fully dynamic setting. More startlingly, our scheme even produces slightly shorter signatures than the former, thanks to an adaptation of a technique proposed by Ling et al. [116], allowing to prove inequalities in zero-knowledge. The scheme satisfies the strong security requirements of Bootle et al.’s model [31], under the SIS and the LWE assumptions.

Furthermore, we demonstrate how to equip the obtained group signature scheme with the deniability functionality in a simple way. This attractive functionality, put forward by Ishida et al. [91], enables the group manager to provide an evidence that a given user is *not* the owner of a signature in question. In the process, we design a zero-knowledge protocol for proving that a given **LWE** ciphertext does not decrypt to a particular message.

Organization. In Section 3.1, we give an overview of our techniques. Section 3.2 recalls the syntax and security requirements of fully dynamic group signatures suggested by Bootle et al. [31]. In Section 3.3, we develop an updatable Merkle tree accumulator. Our main scheme is constructed and analyzed in Section 3.4. In Section 3.5, we demonstrate how to achieve deniability.

3.1 Overview of our Techniques

Our starting point is the scheme [108], which works in the static setting. Instead of relying on trapdoor-based ordinary signatures as in prior works, the LLNW scheme employs on a **SIS**-based Merkle tree accumulator. For a group of $N = 2^\ell$ users, the manager chooses uniformly random vectors $\mathbf{x}_0, \dots, \mathbf{x}_{N-1}$; hashes them to $\mathbf{p}_0, \dots, \mathbf{p}_{N-1}$, respectively; builds a tree on top of these hash values; and publishes the tree root $\mathbf{u}$. The signing key of user i consists of $\mathbf{x}_i$ and the witness for the fact that $\mathbf{p}_i$ was accumulated in $\mathbf{u}$. When issuing signatures, the user proves knowledge of a valid pair $(\mathbf{x}_i, \mathbf{p}_i)$ and of the tree path from $\mathbf{p}_i$ to $\mathbf{u}$. The user also has to encrypt the binary representation $\text{bin}(i)$ of his identity i , and prove that the ciphertext is well-formed. The encryption layer is also lattice-trapdoor-free, since it utilizes the Naor-Yung double-encryption paradigm [146] with Regev’s **LWE**-based encryption scheme. To upgrade the LLNW scheme directly into a fully dynamic group signature, we now let the user compute the pair $(\mathbf{x}_i, \mathbf{p}_i)$ on his own (for enabling non-frameability), and we employ the following three ideas/techniques.

First, we add a dynamic ingredient into the static Merkle tree accumulator from [108]. To this end, we equip it with an efficient updating algorithm with complexity $\mathcal{O}(\log N)$: to change an accumulated value, we simply update the values at the corresponding leaf and along its path to the root.

Second, we create a simple rule to handle user enrollment and revocation efficiently (i.e., without resetting the whole tree). Specifically, we use the updating algorithm to set up the system so that: (i)- If a user has not joined the group or has been revoked, the value at the leaf associated with him is set as $\mathbf{0}$; (ii)- When a user joins the group, that value is set as his public key $\mathbf{p}_i$. Our setup guarantees that only active users (i.e., who has joined and has not been revoked at the given epoch) have their *non-zero* public keys accumulated into the updated root. This rule effectively separates active users who can sign from those who cannot: when signing messages, the user proceeds as in the LLNW scheme, and is asked to additionally prove in zero-knowledge that $\mathbf{p}_i \neq \mathbf{0}$. In other words, the seemingly big gap between being fully static and being fully dynamic has been reduced to a small difference!

Third, the arising question now is how to additionally prove the inequality $\mathbf{p}_i \neq \mathbf{0}$ in the framework of the Stern-like [173] protocol from [108]. One would naturally expect that this extra job could be done without losing too much in terms of efficiency. Here, the surprising and somewhat unexpected fact is that we can actually do it while *gaining* efficiency, thanks to the following simple idea. Recall that, in [108], to prove knowledge of $\mathbf{p}_i \in \{0, 1\}^{nk}$, an extension technique from [116] is employed, in which $\mathbf{p}_i$ is extended into a vector of dimension $2nk$. We note that, the authors of [116] also suggested a slightly modified version of their technique, that allows to simultaneously prove that $\mathbf{p}_i \in \{0, 1\}^{nk}$ and $\mathbf{p}_i$ is non-zero while working only with dimension $2nk - 1$. This intriguing tweak enables us to obtain a zero-knowledge protocol with slightly lower communication cost, and hence, group signatures with slightly smaller size than in [108].

Another problem we have to overcome is that the fully dynamic setting requires a

proof of correct opening, which boils down to proving correct decryption for Regev’s encryption scheme [160]. It involves modular linear equations with bounded-norm secrets, and can be handled using Stern-like techniques from [105, 116].

Now, to equip the obtained fully dynamic group signature scheme with the deniability functionality, we first incorporate Ishida et al.’s notion of deniability [91] into Bootle et al.’s model [31]. Then we describe how to make our scheme deniable, via a zero-knowledge argument of knowledge that a given Regev ciphertext [160] does not decrypt to a given message. The latter reduces to proving knowledge of a *non-zero* vector $\mathbf{b} \in \{-1, 0, 1\}^\ell$ satisfying a modular linear equation, where $\mathbf{b}$ is the difference between the result of decryption and the given message. This in turn can be handled by developing the above method for proving inequality in the framework of Stern’s protocol.

To summarize, we solve the prominent open question of achieving full dynamicity in the field of lattice-based group signatures. The scheme can be easily extended to a deniable group signature. Moreover, our solution is simple and comparatively efficient. Our results, while not yielding truly practical schemes, would certainly help to enrich the field and bring lattice-based group signatures one step closer to practice.

3.2 Fully Dynamic Group Signatures

We recall the definition and security notions of fully dynamic group signatures (FDGS) presented by Bootle et al. [31]. A FDGS scheme involves the following entities: a group manager **GM** that determines who can join the group, a tracing manager **TM** who can open signatures, and a set of users who are potential group members. Users can join/leave the group at the discretion of **GM**. We assume **GM** will publish some information info_τ regularly associated with a distinct index τ (referred as epoch hereafter). Without loss of generality, assume there is one-to-one correspondence between information and the associated epoch. The information describes changes of the group,

e.g., current group members or members that are excluded from the group. We assume the published group information is authentic. By comparing current group information with previous one, it allows any party to identify revoked users at current epoch. For simplicity assume $\tau_1 < \tau_2$ if info_{τ_1} is published before info_{τ_2} , i.e., the epoch preserves the order in which the corresponding group information was published. In existing models, the keys of authorities are supposed to be generated honestly; while in [31], Bootle et al. consider a stronger model where the keys of authorities can be maliciously generated by the adversary.

3.2.1 Syntax of fully dynamic group signatures.

A FDGS scheme is a tuple of following polynomial-time algorithms.

$\text{GSetup}(\lambda) \rightarrow \text{pp}$. On input security parameter λ , this algorithm generates public parameters pp .

$\langle \text{GKgen}_{\text{GM}}(\text{pp}), \text{GKgen}_{\text{TM}}(\text{pp}) \rangle$. This is an interactive protocol between the group manager GM and the tracing manager TM . If it completes successfully, algorithm GKgen_{GM} outputs a manager key pair (mpk, msk) . Meanwhile, GM initializes the group information info and the registration table reg . The algorithm GKgen_{TM} outputs a tracing key pair (tpk, tsk) . Set group public key $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$\text{UKgen}(\text{pp}) \rightarrow (\text{upk}, \text{usk})$. Given public parameters pp , this algorithm generates a user key pair (upk, usk) .

$\langle \text{Join}(\text{info}_{\tau_{\text{current}}}, \text{gpk}, \text{upk}, \text{usk}); \text{Issue}(\text{info}_{\tau_{\text{current}}}, \text{msk}, \text{upk}) \rangle$. This is an interactive algorithm run by a user and GM . If it completes successfully, this user becomes a group member with an identifier uid and the Join algorithm stores secret group signing key $\text{gsk}[\text{uid}]$ while Issue algorithm stores registration information in the table reg with index uid .

$\text{GUpdate}(\text{gpk}, \text{msk}, \text{info}_{\tau_{\text{current}}}, \mathcal{S}, \mathbf{reg}) \rightarrow \text{info}_{\tau_{\text{new}}}$. This is an algorithm run by GM to update group information while advancing the epoch. Given $\text{gpk}, \text{msk}, \text{info}_{\tau_{\text{current}}}$, registration table $\mathbf{reg}$, a set $\mathcal{S}$ of active users to be removed from the group, GM computes new group information $\text{info}_{\tau_{\text{new}}}$ and may update the table $\mathbf{reg}$. If there is no change to the group, GM outputs $\perp$.

$\text{Sign}(\text{gpk}, \text{gsk}[\text{uid}], \text{info}_{\tau}, M) \rightarrow \Sigma$. This algorithm outputs a group signature Σ on message M by user uid . It outputs $\perp$ if this user is inactive at epoch τ .

$\text{Verify}(\text{gpk}, \text{info}_{\tau}, M, \Sigma) \rightarrow 0/1$. This algorithm checks the validity of the signature Σ on message M at epoch τ .

$\text{Trace}(\text{gpk}, \text{tsk}, \text{info}_{\tau}, \mathbf{reg}, M, \Sigma) \rightarrow (\text{uid}, \Pi_{\text{trace}})$. This is an algorithm run by TM. Given the inputs, TM returns an identity uid of a group member who signed the message and a proof indicating this tracing result or $\perp$ if it fails to trace to a group member.

$\text{Judge}(\text{gpk}, \text{uid}, \text{info}_{\tau}, \Pi_{\text{trace}}, M, \Sigma) \rightarrow 0/1$. This algorithm checks the validity of Π_{trace} outputted by the Trace algorithm.

3.2.2 Notions of Correctness and Security requirements

As put forward by Bootle et al. [31], a FDGS scheme must satisfy *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*.

We will define these requirements using a set of experiments, in which the adversary has access to a set of oracles. The oracles use the following global variables: **HUL** is a list of users whose keys are generated honestly. **BUL** is a list of users whose secret signing keys are revealed to the adversary; **CUL** is a list of users whose public keys are chosen by the adversary; **SL** is a set of signatures that are generated by the oracle **Sign**; **CL** is a set of signatures that are generated by the oracle **Chal_b**.

AddU(uid). This oracle adds an honest user uid to the group at current epoch. It returns **upk**[uid] and adds uid to the list **HUL**.

CrptU(uid, pk). This oracle allows the adversary to create a new user by choosing the key **upk**[uid] as pk. It then adds uid to the list **CUL**. It returns $\perp$ if $\text{uid} \in \text{HUL} \cup \text{CUL}$. This actually prepares for calling the oracle **SndToGM**.

SndToGM(uid, M_{in}). This oracle engages in the $\langle \text{Join}, \text{Issue} \rangle$ protocol between the adversary, who corrupts the user uid in the list **CUL**, and the honest **Issue**-executing group manager.

SndToU(uid, M_{in}). This oracle engages in the $\langle \text{Join}, \text{Issue} \rangle$ protocol between the adversary, who corrupts the group manager, and the honest **Join**-executing user uid in the list **HUL**.

RReg(uid). This oracle returns registration information **reg**[uid] of user uid.

MReg(uid, ρ). This oracle modifies **reg**[uid] to any ρ chosen by the adversary.

RevealU(uid). This oracle returns secret signing key **gsk**[uid] to the adversary, and adds this user to the list **BUL**.

Sign(uid, M, τ). This oracle returns a signature on the message M by the user uid at epoch τ , and then add the signature to the list **SL**.

Chal_b(info _{τ} , uid₀, uid₁, M). This oracle receives inputs from the adversary, returns a signature on M by the user uid _{b} at epoch τ , and then adds the signature to the list **CL**. It is required that the two challenged users are active at epoch τ and this oracle can only be called once.

Trace($M, \Sigma, \text{Info}_\tau$). This oracle returns the signer of the signature together with a proof, with respect to the epoch τ . We require that the signature is not in the list **CL**.

GUpdate($\mathcal{S}$). This oracle allows the adversary to update the group at current epoch τ_{current} . It is required that $\mathcal{S}$ is a set of active users at current epoch.

We also need the following polynomial-time algorithm in security experiments to ease composition.

IsActive($\text{info}_\tau, \text{reg}, \text{uid}$) $\rightarrow$ 0/1: It outputs 1 if this user is active at epoch τ and 0 otherwise.

Correctness demands that a signature generated by an honest and active user is always accepted by algorithm **Verify**, and that algorithm **Trace** can always identify that user, as well as produces a proof accepted by algorithm **Judge**.

Definition 3.2.1 (Correctness). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{corr}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{corr}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL} := \emptyset.$

$\langle (\text{info}, \text{mpk}, \text{msk}); (\text{tpk}, \text{tsk}) \rangle \leftarrow \langle \text{GKgen}_{\text{GM}}(\text{pp}), \text{GKgen}_{\text{TM}}(\text{pp}) \rangle.$

Set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$. $(\text{uid}, M, \tau) \leftarrow \mathcal{A}^{\text{AddU, RReg, GUpdate}}(\text{gpk}, \text{info}).$

If $\text{uid} \notin \text{HUL}$ or $\text{gsk}[\text{uid}] = \perp$ or $\text{info}_\tau = \perp$ or $\text{IsActive}(\text{info}_\tau, \text{reg}, \text{uid}) = 0$, return 0.

$\Sigma \leftarrow \text{Sign}(\text{gpk}, \text{gsk}[\text{uid}], \text{info}_\tau, M), (\text{uid}^*, \Pi_{\text{trace}}) \leftarrow \text{Trace}(\text{gpk}, \text{tsk}, \text{info}_\tau, \text{reg}, M, \Sigma).$

Return 1 if $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 0$ or $\text{uid} \neq \text{uid}^*$ or

$\text{Judge}(\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{\text{trace}}, m, \Sigma) = 0.$

Else, return 0.

A fully dynamic group signature scheme is said to be correct if the advantage of the adversary against correctness, define as $\mathbf{Adv}_{\text{FDGS}, \mathcal{A}}^{\text{corr}}(\lambda) = \Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{corr}}(\lambda) = 1]$, is negligible in λ .

Anonymity requires that it is infeasible for any PPT adversary to distinguish signatures generated by two active users of its choice at the chosen epoch, even if it can corrupt any user, can choose the key of **GM**, and is given access to the **Trace** oracle.

Definition 3.2.2 (Anonymity). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiments $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-b}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-b}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL}, \text{CUL}, \text{BUL}, \text{CL}, \text{SL} := \emptyset.$

$(\text{st}_{\text{init}}, \text{info}, \text{mpk}, \text{msk}) \leftarrow \mathcal{A}^{(\cdot, \text{GKgen}_{\text{TM}})}(\text{pp})(\text{init} : \text{pp}).$

If GKgen_{TM} did not accept or $\mathcal{A}$'s output is not well-formed, return $\perp$.

Denote the output of GKgen_{TM} as (tpk, tsk) , and set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$b^* \leftarrow \mathcal{A}^{\text{AddU}, \text{CrptU}, \text{RevealU}, \text{SndToU}, \text{Trace}, \text{MReg}, \text{Chal}_b}(\text{play} : \text{st}_{\text{init}}, \text{gpk}).$

Return b^* .

A fully dynamic group signature scheme is said to be anonymous if the advantage of $\mathcal{A}$, define as $\mathbf{Adv}_{\text{FDGS}, \mathcal{A}}^{\text{anon}}(\lambda) = |\Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-0}(\lambda) = 1]|$, is negligible in λ .

Non-Frameability makes sure that the adversary cannot generate a valid signature that traces to an honest user even if it can corrupt all other users, and can choose keys of both managers.

Definition 3.2.3 (Non-frameability). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{non-frame}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{non-frame}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL}, \text{CUL}, \text{BUL}, \text{SL} = \emptyset.$

$(\text{st}_{\text{init}}, \text{info}, \text{mpk}, \text{msk}, \text{tpk}, \text{tsk}) \leftarrow \mathcal{A}(\text{init} : \text{pp}).$

Return 0 if $\mathcal{A}$'s output is not well-formed, otherwise set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$(M, \Sigma, \text{uid}, \Pi_{\text{trace}}, \text{info}_\tau) \leftarrow \mathcal{A}^{\text{CrptU}, \text{RevealU}, \text{SndToU}, \text{MReg}, \text{Sign}}(\text{play} : \text{st}_{\text{init}}, \text{gpk}).$

If $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 0$ or $\text{uid} \notin \text{HUL} \setminus \text{BUL}$ or $(\text{uid}, M, \Sigma, \tau) \in \text{SL}$, return 0.

Return 1 if $\text{Judge}(\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{\text{Trace}}, M, \Sigma) = 1$.

Else return 0.

A fully dynamic group signature scheme is said to be non-frameable if the advantage of $\mathcal{A}$, define as $\mathbf{Adv}_{\text{FDGS}, \mathcal{A}}^{\text{non-frame}}(\lambda) = \Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{non-frame}}(\lambda) = 1]$, is negligible in λ .

Traceability ensures that the adversary cannot produce a valid signature that cannot be traced to an active user at the chosen epoch, even if it can corrupt any user and can choose the key of TM.

Definition 3.2.4 (Traceability). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{trace}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{trace}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL}, \text{CUL}, \text{BUL}, \text{SL} = \emptyset.$

$(\text{st}_{\text{init}}, \text{tpk}, \text{tsk}) \leftarrow \mathcal{A}^{\langle \text{GKgen}_{\text{GM}}(\text{pp}), \cdot \rangle}(\text{init} : \text{pp}).$

Return 0 if GKgen_{GM} did not accept or $\mathcal{A}$'s output is not well-formed.

Denote the output of GKgen_{GM} as $(\text{mpk}, \text{msk}, \text{info})$, and set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$(M, \Sigma, \tau) \leftarrow \mathcal{A}^{\text{AddU}, \text{CrptU}, \text{SndToGM}, \text{RevealU}, \text{MReg}, \text{Sign}, \text{GUpdate}}(\text{play} : \text{st}_{\text{init}}, \text{gpk}, \text{info}).$

If $\text{Verify}(\text{gpk}, \text{info}_{\tau}, M, \Sigma) = 0$, return 0.

$(\text{uid}, \Pi_{\text{trace}}) \leftarrow \text{Trace}(\text{gpk}, \text{tsk}, \text{info}_{\tau}, \text{reg}, M, \Sigma).$

Return 1 if $\text{IsActive}(\text{info}_{\tau}, \text{reg}, \text{uid}) = \perp$ or $\text{uid} = 0$ or

$\text{Judge}(\text{gpk}, \text{uid}, \text{info}_{\tau}, \Pi_{\text{trace}}, M, \Sigma) = 0.$

Else return 0.

A fully dynamic group signature scheme is said to be traceable if the advantage of the adversary $\mathcal{A}$ against traceability, define as $\mathbf{Adv}_{\text{FDGS},\mathcal{A}}^{\text{trace}}(\lambda) = \Pr[\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{trace}}(\lambda) = 1]$, is negligible in λ .

Tracing Soundness requires that it is infeasible to produce a valid signature that traces to two different users, even if all group users and both managers are fully controlled by the adversary.

Definition 3.2.5 (Tracing Soundness). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{trace-sound}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{trace-sound}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{CUL} = \emptyset. (\text{st}_{\text{init}}, \text{info}, \text{mpk}, \text{msk}, \text{tpk}, \text{tsk}) \leftarrow \mathcal{A}(\text{init} : \text{pp}).$

Return 0 if $\mathcal{A}$'s output is not well-formed, otherwise set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$(M, \Sigma, \text{uid}_0, \Pi_{\text{trace},0}, \text{uid}_1, \Pi_{\text{trace},1}, \text{info}_\tau) \leftarrow \mathcal{A}^{\text{CrptU}, \text{MReg}}(\text{play} : \text{st}_{\text{init}}, \text{gpk}).$
 If $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 0$ or $\text{Judge}(\text{gpk}, \text{uid}_0, \text{info}_\tau, \Pi_{\text{Trace},0}, M, \Sigma) = 0$,
 or $\text{uid}_0 = \perp$ or $\text{uid}_1 = \perp$, return 0.
 Return 1 if $\text{uid}_0 \neq \text{uid}_1 \wedge \text{Judge}(\text{gpk}, \text{uid}_1, \text{info}_\tau, \Pi_{\text{Trace},1}, M, \Sigma) = 1$
 Else return 0.

A fully dynamic group signature scheme is said to be tracing sound if the advantage of the adversary $\mathcal{A}$, define as $\text{Adv}_{\text{FDGS}, \mathcal{A}}^{\text{trace-sound}}(\lambda) = \Pr[\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{trace-sound}}(\lambda) = 1]$, is negligible in λ .

3.3 Updatable Lattice-Based Merkle Hash Trees

We first recall the lattice-based Merkle-tree accumulator from [108], and then, we equip it with a simple updating algorithm which allows to change an accumulated value in time logarithmic in the size of the accumulated set. This updatable hash tree will serve as the building block of our construction in Section 3.4.1.

3.3.1 Cryptographic Accumulators

An *accumulator scheme* is a tuple of polynomial-time algorithms defined below.

TSetup $_{\lambda}$ On input security parameter λ , output the public parameter pp .

TAcc $_{\text{pp}}$ On input a set $R = \{\mathbf{d}_0, \dots, \mathbf{d}_{N-1}\}$ of N data values, output an accumulator value $\mathbf{u}$.

TWitness $_{\text{pp}}$ On input a data set R and a value $\mathbf{d}$, output $\perp$ if $\mathbf{d} \notin R$; otherwise output a witness w for the fact that $\mathbf{d}$ is accumulated in $\text{TAcc}(R)$. (Typically, the size of w should be short (*e.g.*, constant or logarithmic in N) to be useful.)

TVerify $_{\text{pp}}$ On input accumulator value $\mathbf{u}$ and a value-witness pair $(\mathbf{d}, w)$, output 1 (which indicates that $(\mathbf{d}, w)$ is valid for the accumulator $\mathbf{u}$) or 0.

An accumulator scheme is called correct if for all public parameters $\text{pp} \leftarrow \text{TSetup}(\lambda)$, we have $\text{TVerify}_{\text{pp}}(\text{TAcc}_{\text{pp}}(R), \mathbf{d}, \text{TWitness}_{\text{pp}}(R, \mathbf{d})) = 1$ for all $\mathbf{d} \in R$.

A natural security requirement for accumulators, as considered in [17, 55, 108], says that it is infeasible to prove that a value $\mathbf{d}^*$ was accumulated in a value $\mathbf{u}$ if it was not. This property is formalized as follows.

Definition 3.3.1 ([108]). An accumulator scheme $(\text{TSetup}, \text{TAcc}, \text{TWitness}, \text{TVerify})$ is called secure if for all PPT adversaries $\mathcal{A}$:

$$\Pr[\text{pp} \leftarrow \text{TSetup}(\lambda); (R, \mathbf{d}^*, w^*) \leftarrow \mathcal{A}(\text{pp}) : \mathbf{d}^* \notin R \wedge \text{TVerify}_{\text{pp}}(\text{TAcc}_{\text{pp}}(R), \mathbf{d}^*, w^*) = 1] = \text{negl}(\lambda).$$

3.3.2 The LLNW Merkle-tree Accumulator

NOTATIONS. Hereunder we will use the notation $x \xleftarrow{\$} S$ to indicate that x is chosen uniformly at random from finite set S . For bit $b \in \{0, 1\}$, we let $\bar{b} = 1 - b$.

The Merkle-tree accumulator scheme from [108] works with parameters $n = \mathcal{O}(\lambda)$, $q = \tilde{\mathcal{O}}(n^{1.5})$, $k = \lceil \log_2 q \rceil$, and $m = 2nk$. The set $\mathbb{Z}_q$ is identified by $\{0, \dots, q-1\}$. Define the “powers-of-2” matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & 4 & \dots & 2^{k-1} & & \\ & & & & \dots & & \\ & & & & & 1 & 2 & 4 & \dots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}.$$

Note that for every $\mathbf{v} \in \mathbb{Z}_q^n$, we have $\mathbf{v} = \mathbf{G} \cdot \text{bin}(\mathbf{v})$, where $\text{bin}(\mathbf{v}) \in \{0, 1\}^{nk}$ denotes the binary representation of $\mathbf{v}$. The scheme is built upon the following family of SIS-based collision-resistant hash functions.

Definition 3.3.2. The function family $\mathcal{H}$ mapping $\{0, 1\}^{nk} \times \{0, 1\}^{nk}$ to $\{0, 1\}^{nk}$ is defined as $\mathcal{H} = \{h_{\mathbf{A}} \mid \mathbf{A} \in \mathbb{Z}_q^{n \times m}\}$, where for $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ with $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times nk}$, and

for any $(\mathbf{u}_0, \mathbf{u}_1) \in \{0, 1\}^{nk} \times \{0, 1\}^{nk}$, we have:

$$h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1) = \text{bin}(\mathbf{A}_0 \cdot \mathbf{u}_0 + \mathbf{A}_1 \cdot \mathbf{u}_1 \bmod q) \in \{0, 1\}^{nk}.$$

Note that $h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{u} \Leftrightarrow \mathbf{A}_0 \cdot \mathbf{u}_0 + \mathbf{A}_1 \cdot \mathbf{u}_1 = \mathbf{G} \cdot \mathbf{u} \bmod q$.

A Merkle tree with $N = 2^\ell$ leaves, where ℓ is a positive integer, then can be constructed based on the function family $\mathcal{H}$ as follows.

TSetup(λ). Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, and output $\text{pp} = \mathbf{A}$.

TAcc_A($R = \{\mathbf{d}_0 \in \{0, 1\}^{nk}, \dots, \mathbf{d}_{N-1} \in \{0, 1\}^{nk}\}$). For every $j \in [0, N-1]$, let $\text{bin}(j) = (j_1, \dots, j_\ell) \in \{0, 1\}^\ell$ be the binary representation of j , and let $\mathbf{d}_j = \mathbf{u}_{j_1, \dots, j_\ell}$. Form the tree of depth $\ell = \log N$ based on the N leaves $\mathbf{u}_{0,0,\dots,0}, \dots, \mathbf{u}_{1,1,\dots,1}$ as follows:

1. At depth $i \in [\ell]$, the node $\mathbf{u}_{b_1, \dots, b_i} \in \{0, 1\}^{nk}$, for all $(b_1, \dots, b_i) \in \{0, 1\}^i$, is defined as $h_{\mathbf{A}}(\mathbf{u}_{b_1, \dots, b_i, 0}, \mathbf{u}_{b_1, \dots, b_i, 1})$.
2. At depth 0: The root $\mathbf{u} \in \{0, 1\}^{nk}$ is defined as $h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1)$.

The algorithm outputs the accumulator value $\mathbf{u}$.

TWitness_A($R, \mathbf{d}$). If $\mathbf{d} \notin R$, return $\perp$. Otherwise, $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N-1]$ with binary representation $(j_1, \dots, j_\ell)$. Output the witness w defined as:

$$w = ((j_1, \dots, j_\ell), (\mathbf{u}_{j_1, \dots, j_{\ell-1}, \bar{j}_\ell}, \dots, \mathbf{u}_{j_1, \bar{j}_2}, \mathbf{u}_{\bar{j}_1})) \in \{0, 1\}^\ell \times (\{0, 1\}^{nk})^\ell,$$

for $\mathbf{u}_{j_1, \dots, j_{\ell-1}, \bar{j}_\ell}, \dots, \mathbf{u}_{j_1, \bar{j}_2}, \mathbf{u}_{\bar{j}_1}$ computed by algorithm **TAcc_A**(R).

TVerify_A($\mathbf{u}, \mathbf{d}, w$). Let the given witness w be of the form:

$$w = ((j_1, \dots, j_\ell), (\mathbf{w}_\ell, \dots, \mathbf{w}_1)) \in \{0, 1\}^\ell \times (\{0, 1\}^{nk})^\ell.$$

The algorithm recursively computes the path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1, \mathbf{v}_0 \in \{0, 1\}^{nk}$ as follows: $\mathbf{v}_\ell = \mathbf{d}$ and

$$\forall i \in \{\ell - 1, \dots, 1, 0\} : \mathbf{v}_i = \begin{cases} h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), & \text{if } j_{i+1} = 0; \\ h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), & \text{if } j_{i+1} = 1. \end{cases} \quad (3.1)$$

Then it returns 1 if $\mathbf{v}_0 = \mathbf{u}$. Otherwise, it returns 0.

The following lemma states the correctness and security of the above Merkle tree accumulator.

Lemma 3.3.3 ([108]). The given accumulator scheme is correct and is secure in the sense of Definition 3.3.1, assuming the hardness of the $\text{SIS}_{n,m,q,1}^\infty$ problem.

3.3.3 An Efficient Updating Algorithm

Unlike the static group signature scheme from [108], our fully dynamic construction of Section 3.4 requires to regularly edit the accumulated values without having to reconstruct the whole tree. To this end, we equip the Merkle tree accumulator from [108] with a simple, yet efficient, updating algorithm: to change the value at a given leaf, we simply modify all values in the path from that leaf to the root. The algorithm, which takes as input a bit string $\text{bin}(j) = (j_1, j_2, \dots, j_\ell)$ and a value $\mathbf{d}^* \in \{0, 1\}^{nk}$, is formally described below.

Given the tree in Section 3.3.2, algorithm $\text{TUpdate}_{\mathbf{A}}((j_1, j_2, \dots, j_\ell), \mathbf{d}^*)$ performs the following steps:

1. Let $\mathbf{d}_j$ be the current value at the leaf of position determined by $\text{bin}(j)$, and let $((j_1, \dots, j_\ell), (\mathbf{w}_{j,\ell}, \dots, \mathbf{w}_{j,1}))$ be its associated witness.
2. Set $\mathbf{v}_\ell := \mathbf{d}^*$ and recursively compute the path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1, \mathbf{v}_0 \in \{0, 1\}^{nk}$ as in (3.1).

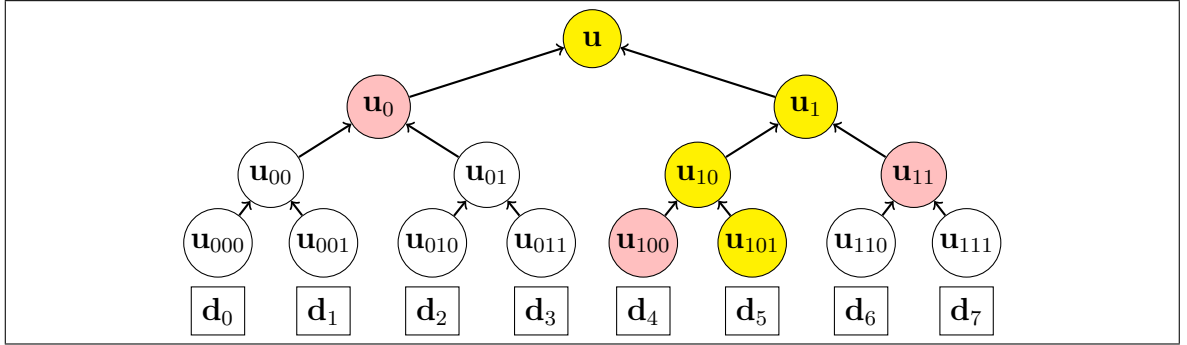


Figure 3.1: An illustration of Merkle tree with $2^3 = 8$ leaves.

3. Set $\mathbf{u} := \mathbf{v}_0$; $\mathbf{u}_{j_1} := \mathbf{v}_1$; $\dots$; $\mathbf{u}_{j_1, j_2, \dots, j_{\ell-1}} := \mathbf{v}_{\ell-1}$; $\mathbf{u}_{j_1, j_2, \dots, j_{\ell}} := \mathbf{v}_{\ell} = \mathbf{d}^*$.

It can be seen that the provided algorithm runs in time $\mathcal{O}(\ell) = \mathcal{O}(\log N)$. In Fig. 3.1, we give an illustrative example of a tree with $2^3 = 8$ leaves. It accumulates the data blocks $\mathbf{d}_0, \dots, \mathbf{d}_7$ into the value $\mathbf{u}$ at the root. The bit string (101) and the pink nodes form a witness to the fact that $\mathbf{d}_5$ is accumulated in $\mathbf{u}$. If we replace $\mathbf{d}_5$ by a new value $\mathbf{d}^*$, we only need to update the yellow nodes.

3.4 Fully Dynamic Group Signatures from Lattices

In this section, we construct our lattice-based fully dynamic group signature and prove its security in Bootle et al.'s model [31]. We start with the LLNW scheme [108], which works in the static setting.

While other constructions of lattice-based group signatures employ trapdoor-based ordinary signature schemes (e.g., [34, 61]) to certify users, the LLNW scheme relies on a SIS-based Merkle tree accumulator which we recalled in Section 3.3.2. The GM, who manages a group of $N = 2^\ell$ users, chooses uniformly random vectors $\mathbf{x}_0, \dots, \mathbf{x}_{N-1} \in \{0, 1\}^m$; hashes them to $\mathbf{p}_0, \dots, \mathbf{p}_{N-1} \in \{0, 1\}^{nk}$, respectively; builds a tree on top of these hash values; and lets the tree root $\mathbf{u} \in \{0, 1\}^{nk}$ be part of the group public key. The signing key of user i consists of $\mathbf{x}_i$ and the witness for the fact that $\mathbf{p}_i$ was accumulated in $\mathbf{u}$. When generating group signatures, the user proves knowledge of a valid pair $(\mathbf{x}_i, \mathbf{p}_i)$ and of the tree path from $\mathbf{p}_i$ to $\mathbf{u}$. The user

also has to encrypt the binary representation $\text{bin}(i)$ of his identity i , and prove that the ciphertext is well-formed. The encryption layer utilizes the Naor-Yung double-encryption paradigm [146] with Regev’s LWE-based cryptosystem, and thus, it is also lattice-trapdoor-free.

To upgrade the LLNW scheme directly into a fully dynamic group signature, some tweaks and new ideas are needed. First, to enable the non-frameability feature, we let the user compute the pair $(\mathbf{x}_i, \mathbf{p}_i)$ on his own. The second problem we have to consider is that Merkle hash trees seem to be a static primitive. To this end, we equip the accumulator with an efficient updating algorithm (see Section 3.3.3). Now, the challenging question is how to handle user enrollment and revocation in a simple manner (i.e., without having to reset the whole tree). To tackle these issues, we associate each of the N potential users with a leaf in the tree, and then use the updating algorithm to set up the system so that:

1. If a user has not joined the group or has been revoked, the value at the leaf associated with him is set as $\mathbf{0}$;
2. When a user joins the group, that value is set as his public key $\mathbf{p}_i$.

Our setup guarantees that only active users (i.e., who has joined and has not been revoked at the given epoch) have their *non-zero* public keys accumulated into the updated root. This effectively gives us a method to separate active users who can sign from those who cannot: when signing messages, the user proceeds as in the LLNW scheme, and is asked to additionally prove in ZK that $\mathbf{p}_i \neq \mathbf{0}$.

At this point, the arising question is how to prove the inequality $\mathbf{p}_i \neq \mathbf{0}$ in the framework of the Stern-like [173] protocol from [108]. One would naturally hope that this extra job could be done without losing too much in terms of efficiency. Here, the surprising and somewhat unexpected fact is that we can actually do it while *gaining* efficiency, thanks to a technique originally proposed in [116].

To begin with, let $\mathbf{B}_t^L$ denote the set of all vectors in $\{0, 1\}^L$ having Hamming

weight exactly t . In Stern-like protocols (see Section 2.4.2), a common technique for proving in ZK the possession of $\mathbf{p} \in \{0, 1\}^{nk}$ consists of appending nk “dummy” entries to it to obtain $\mathbf{p}^* \in \mathcal{B}_{nk}^{2nk}$, and demonstrating to the verifier that a random permutation of $\mathbf{p}^*$ belongs to the “target set” $\mathcal{B}_{nk}^{2nk}$. This suffices to convince the verifier that the original vector $\mathbf{p}$ belongs to $\{0, 1\}^{nk}$, while the latter cannot learn any additional information about $\mathbf{p}$, thanks to the randomness of the permutation. This extending-then-permuting technique was first proposed in [116], and was extensively used in the underlying protocol of the LLNW scheme. Now, to address our question, we will employ a modified version of this technique, which was also initially suggested in [116]. Let us think of another “target set”, so that it is possible to extend $\mathbf{p} \in \{0, 1\}^{nk}$ to an element of that set if and only if $\mathbf{p}$ is non-zero. That set is $\mathcal{B}_{nk}^{2nk-1}$. Indeed, the extended vector $\mathbf{p}^*$ belongs to $\mathcal{B}_{nk}^{2nk-1}$ if and only if the original vector has Hamming weight at least $nk - (nk - 1) = 1$, which means that it cannot be a zero-vector. When combining with the permuting step, this modification allows us to additionally prove the given inequality while working with smaller dimension. As a result, our fully dynamic scheme produces slightly shorter signatures than the original static scheme.

Finally, we remark that the fully dynamic setting requires a proof of correct opening, which boils down to proving correct decryption for Regev’s encryption scheme. It involves modular linear equations with bounded-norm secrets, and can be easily handled using Stern-like techniques from [105, 116].

3.4.1 Description of Our FDGS Scheme

Details of our scheme are described as follows.

GSetup(λ). On input security parameter λ , this algorithm specifies the following:

- An expected number of potential users $N = 2^\ell = \text{poly}(\lambda)$.
- Dimension $n = \mathcal{O}(\lambda)$, prime modulus $q = \tilde{\mathcal{O}}(n^{1.5})$, and $k = \lceil \log_2 q \rceil$. These parameters implicitly determine the “powers-of-2” matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$, as

defined in Section 3.3.2.

- Matrix dimensions $m = 2nk$ for the hashing layer, and $m_E = 2(n + \ell)k$ for the encryption layer.
- An integer $\beta = \sqrt{n} \cdot \omega(\log n)$, and a β -bounded noise distribution χ .
- A hash function $\mathcal{H}_{\text{FS}} : \{0, 1\}^* \rightarrow \{1, 2, 3\}^\kappa$, where $\kappa = \omega(\log \lambda)$, to be modelled as a random oracle in the Fiat-Shamir transformations [75].
- Let $\text{COM} : \{0, 1\}^* \times \{0, 1\}^m \rightarrow \mathbb{Z}_q^n$ be the string commitment scheme from [94], to be used in our zero-knowledge argument systems.
- Uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

The algorithm outputs public parameters

$$\text{pp} = \{\lambda, N, n, q, k, m, m_E, \ell, \beta, \chi, \kappa, \mathcal{H}_{\text{FS}}, \text{COM}, \mathbf{A}\}.$$

$\langle \text{GKgen}_{\text{GM}}(\text{pp}), \text{GKgen}_{\text{TM}}(\text{pp}) \rangle$. The group manager **GM** and the tracing manager **TM** initialize their keys and the public group information as follows.

- $\text{GKgen}_{\text{GM}}(\text{pp})$. This algorithm samples $\text{msk} \xleftarrow{\$} \{0, 1\}^m$ and computes $\text{mpk} = \mathbf{A} \cdot \text{msk} \bmod q$, and outputs (mpk, msk) . Here, we consider mpk as an identifier of the group managed by **GM** who has mpk as his public key. Furthermore, as in [31, Sec. 3.3, full version], we assume that the group information board is visible to everyone, but can only be edited by a party knowing msk .
- $\text{GKgen}_{\text{TM}}(\text{pp})$. This algorithm initializes the Naor-Yung double-encryption mechanism with the ℓ -bit version Regev encryption scheme. It first chooses $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m_E}$. For each $i \in \{1, 2\}$, it samples $\mathbf{S}_i \xleftarrow{\$} \chi^{n \times \ell}$, $\mathbf{E}_i \leftarrow \chi^{\ell \times m_E}$, and computes $\mathbf{P}_i = \mathbf{S}_i^\top \cdot \mathbf{B} + \mathbf{E}_i \in \mathbb{Z}_q^{\ell \times m_E}$. Then, **TM** sets $\text{tsk} = (\mathbf{S}_1, \mathbf{E}_1)$, and $\text{tpk} = (\mathbf{B}, \mathbf{P}_1, \mathbf{P}_2)$.

- TM sends **tpk** to GM who initializes the following:
 - Table **reg** := (**reg**[0][1], **reg**[0][2], ..., **reg**[N−1][1], **reg**[N−1][2]), where for each $i \in [0, N-1]$: **reg**[i][1] = $\mathbf{0}^{nk}$ and **reg**[i][2] = 0. Looking ahead, **reg**[i][1] will be used to record the public key of a registered user, while **reg**[i][2] stores the epoch at which the user joins.
 - The Merkle tree $\mathcal{T}$ built on top of **reg**[0][1], ..., **reg**[N−1][1]. (Note that $\mathcal{T}$ is an all-zero tree at this stage, but it will be modified when a new user joins the group, or when GM computes the updated group information.)
 - Counter of registered users $c := 0$.

Then, GM outputs **gpk** = (**pp**, **mpk**, **tpk**) and announces the initial group information **info** = $\emptyset$. He keeps $\mathcal{T}$ and c for himself.

UKgen(pp). Each potential group user samples $\mathbf{usk} = \mathbf{x} \xleftarrow{\$} \{0, 1\}^m$, and computes $\mathbf{upk} = \mathbf{p} = \text{bin}(\mathbf{A} \cdot \mathbf{x}) \bmod q \in \{0, 1\}^{nk}$.

Without loss of generality, we assume that every honestly generated **upk** is a non-zero vector. (Otherwise, the user would either pick $\mathbf{x} = \mathbf{0}$ or accidentally find a solution to the $\text{SIS}_{n,m,q,1}^\infty$ problem associated with matrix $\mathbf{A}$ - both happen with negligible probability.)

Join, Issue. If the user with key pair $(\mathbf{upk}, \mathbf{usk}) = (\mathbf{p}, \mathbf{x})$ requests to join the group at epoch τ , he sends **p** to GM. If the latter accepts the request, then the two parties proceed as follows.

1. GM issues a member identifier for the user as $\mathbf{uid} = \text{bin}(c) \in \{0, 1\}^\ell$. The user then sets his long-term signing key as $\mathbf{gsk}[c] = (\text{bin}(c), \mathbf{p}, \mathbf{x})$.
2. GM performs the following updates:
 - Update $\mathcal{T}$ by running algorithm $\text{TUpdate}_{\mathbf{A}}(\text{bin}(c), \mathbf{p})$.

- Register the user to table **reg** as $\mathbf{reg}[c][1] := \mathbf{p}$; $\mathbf{reg}[c][2] := \tau$.
- Increase the counter $c := c + 1$.

GUpdate(gpk, msk, $\mathbf{info}_{\tau_{\text{current}}}$, $\mathcal{S}$, **reg**). This algorithm is run by **GM** to update the group information while also advancing the epoch. It works as follows.

1. Let the set $\mathcal{S}$ contain the public keys of registered users to be revoked. If $\mathcal{S} = \emptyset$, then go to Step 2.

Otherwise, $\mathcal{S} = \{\mathbf{reg}[i_1][1], \dots, \mathbf{reg}[i_r][1]\}$, for some $r \in [1, N]$ and some $i_1, \dots, i_r \in [0, N-1]$. Then, for all $t \in [r]$, **GM** runs **TUpdate_A**($\text{bin}(i_t), \mathbf{0}^{nk}$) to update the tree $\mathcal{T}$.

2. At this point, by construction, each of the zero leaves in the tree $\mathcal{T}$ corresponds to either a revoked user or a potential user who has not yet registered. In other words, only active users who are allowed to sign in the new epoch τ_{new} have their non-zero public keys, denoted by $\{\mathbf{p}_j\}_j$, accumulated in the root $\mathbf{u}_{\tau_{\text{new}}}$ of the updated tree.

For each j , let $w_j \in \{0, 1\}^\ell \times (\{0, 1\}^{nk})^\ell$ be the witness for the fact that $\mathbf{p}_j$ is accumulated in $\mathbf{u}_{\tau_{\text{new}}}$. Then **GM** publishes the group information of the new epoch as:

$$\mathbf{info}_{\tau_{\text{new}}} = (\mathbf{u}_{\tau_{\text{new}}}, \{w_j\}_j).$$

We remark that the $\mathbf{info}_\tau$ outputted at each epoch by **GM** is technically not part of the verification key. Indeed, as we will describe below, in order to verify signatures bound to epoch τ , the verifiers only need to download the first component $\mathbf{u}_\tau$ of size $\tilde{\mathcal{O}}(\lambda)$ bits. Meanwhile, each active signer only has to download the respective witness of size $\tilde{\mathcal{O}}(\lambda) \cdot \ell$.

Sign(gpk, gsk[j], $\mathbf{info}_\tau$, M). To sign message M using the group information $\mathbf{info}_\tau$ at epoch τ , the user possessing $\mathbf{gsk}[j] = (\text{bin}(j), \mathbf{p}, \mathbf{x})$ first checks if $\mathbf{info}_\tau$ includes a

witness containing $\text{bin}(j)$. If this is not the case, return $\perp$. Otherwise, the user downloads $\mathbf{u}_\tau$ and the witness of the form $(\text{bin}(j), (\mathbf{w}_\ell, \dots, \mathbf{w}_1))$ from info_τ , and proceeds as follows.

1. Encrypt vector $\text{bin}(j) \in \{0, 1\}^\ell$ twice using Regev's encryption scheme. Namely, for each $i \in \{1, 2\}$, sample $\mathbf{r}_i \xleftarrow{\$} \{0, 1\}^{m_E}$ and compute

$$\begin{aligned} \mathbf{c}_i &= (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}) \\ &= \left(\mathbf{B} \cdot \mathbf{r}_i \bmod q, \mathbf{P}_i \cdot \mathbf{r}_i + \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{bin}(j) \bmod q \right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell. \end{aligned}$$

2. Generate a NIZKAoK Π_{gs} to demonstrate the possession of a valid tuple

$$\zeta = (\mathbf{x}, \mathbf{p}, \text{bin}(j), \mathbf{w}_\ell, \dots, \mathbf{w}_1, \mathbf{r}_1, \mathbf{r}_2) \quad (3.2)$$

such that:

- (i) $\text{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}, (\text{bin}(j), (\mathbf{w}_\ell, \dots, \mathbf{w}_1))) = 1$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{G} \cdot \mathbf{p} \bmod q$;
- (ii) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\text{bin}(j)$ with randomness $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively;
- (iii) $\mathbf{p} \neq \mathbf{0}^{nk}$.

Note that statements (i) and (ii) were covered by the LLNW protocol [108]. Meanwhile, statement (iii) is handled using the technique described at the beginning of this Section. We thus obtain a Stern-like interactive zero-knowledge argument system which is a slight modification of the one from [108]. The detailed description of the protocol is presented in Section 3.4.3. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\text{gs}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}\}_{i=1}^\kappa)$, where

$$\text{CH} = \mathcal{H}_{\text{FS}}(M, (\{\text{CMT}_i\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1, \mathbf{c}_2)) \in \{1, 2, 3\}^\kappa.$$

3. Output the group signature

$$\Sigma = (\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2). \quad (3.3)$$

Verify(gpk, info_τ, M, Σ). This algorithm proceeds as follows:

1. Download $\mathbf{u}_\tau \in \{0, 1\}^{nk}$ from info_τ.
2. Parse Σ as $\Sigma = (\{\text{CMT}_i\}_{i=1}^\kappa, (Ch_1, \dots, Ch_\kappa), \{\text{RSP}\}_{i=1}^\kappa, \mathbf{c}_1, \mathbf{c}_2)$.
If $(Ch_1, \dots, Ch_\kappa) \neq \mathcal{H}_{\text{FS}}(M, (\{\text{CMT}_i\}_{i=1}^\kappa, \mathbf{A}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1, \mathbf{c}_2))$, then return 0.
3. For each $i \in [\kappa]$, run the verification phase of the protocol in Section 3.4.3 to check the validity of RSP_i with respect to CMT_i and Ch_i. If any of the conditions does not hold, then return 0.
4. Return 1.

Trace(gpk, ts_k, info_τ, reg, M, Σ). This algorithm parses ts_k as (S₁, E₁), parses Σ as in (3.3), and performs the following steps.

1. Use S₁ to decrypt $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ to obtain a string $\mathbf{b}' \in \{0, 1\}^\ell$ (i.e., by computing $\left\lfloor \frac{(\mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1})}{[q/2]} \right\rfloor$).
2. If info_τ does not include a witness containing $\mathbf{b}'$, then return ⊥.
3. Let $j' \in [0, N - 1]$ be the integer having binary representation $\mathbf{b}'$. If the record reg[j'][1] in table reg is $\mathbf{0}^{nk}$, then return ⊥.
4. Generate a NIZKAoK Π_{trace} to demonstrate the possession of $\mathbf{S}_1 \in \mathbb{Z}^{n \times \ell}$, $\mathbf{E}_1 \in \mathbb{Z}^{\ell \times m_E}$, and $\mathbf{y} \in \mathbb{Z}^\ell$, such that:

$$\begin{cases} \|\mathbf{S}_1\|_\infty \leq \beta; \|\mathbf{E}_1\|_\infty \leq \beta; \|\mathbf{y}\|_\infty \leq \lceil q/5 \rceil; \\ \mathbf{S}_1^\top \cdot \mathbf{B} + \mathbf{E}_1 = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} = \mathbf{y} + [q/2] \cdot \mathbf{b}' \bmod q. \end{cases} \quad (3.4)$$

As the statement involves modular linear equations with bounded-norm secrets, we can obtain a statistical zero-knowledge argument by employing the Stern-like interactive protocol from [105]. For completeness, the detailed description of the protocol is presented in Section 3.4.4. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\text{trace}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}\}_{i=1}^\kappa)$, where

$$\text{CH} = \mathcal{H}_{\text{FS}}\left(\left(\{\text{CMT}_i\}_{i=1}^\kappa, \text{gpk}, \text{info}_\tau, M, \Sigma, \mathbf{b}'\right) \in \{1, 2, 3\}^\kappa. \quad (3.5)$$

5. Set $\text{uid} = \mathbf{b}'$ and output $(\text{uid}, \Pi_{\text{trace}})$.

Judge($\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{\text{trace}}, M, \Sigma$). This algorithm consists of verifying the argument Π_{trace} w.r.t. common input $(\text{gpk}, \text{info}_\tau, M, \Sigma, \text{uid})$, in a similar manner as in algorithm **Verify**.

If Π_{trace} does not verify, return 0. Otherwise, return 1.

3.4.2 Analysis of the Scheme

EFFICIENCY. We first analyze the efficiency of the scheme described in Section 3.4.1, with respect to security parameter λ and parameter $\ell = \log N$.

- The public key gpk contains several matrices, and has bit-size $\tilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$.
- For each $j \in [0, N - 1]$, the signing key $\text{gsk}[j]$ has bit-size $\ell + nk + m = \tilde{\mathcal{O}}(\lambda) + \ell$.
- At each epoch, the signature verifiers downloads $nk = \tilde{\mathcal{O}}(\lambda)$ bits, while each active signer downloads $\tilde{\mathcal{O}}(\lambda \cdot \ell)$ bits.
- The size of signature Σ is dominated by that of the Stern-like NIZKAoK Π_{gs} , which is $\mathcal{O}(|\zeta| \cdot \log q) \cdot \omega(\log \lambda)$, where $|\zeta|$ denotes the bit-size of the witness-tuple ζ in (3.2). Overall, Σ has bit-size $\tilde{\mathcal{O}}(\lambda \cdot \ell)$.

- The Stern-like NIZKAoK Π_{trace} has bit-size $\tilde{O}(\ell^2 + \lambda \cdot \ell)$.

CORRECTNESS. We now demonstrate that the scheme is correct with overwhelming probability, based on the perfect completeness of Stern-like protocols, and the correctness of Regev's encryption scheme.

First, note that a signature $\Sigma = (\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ generated by an active and honest user j is always accepted by algorithm **Verify**. Indeed, such a user can always compute a tuple $\zeta = (\mathbf{x}, \mathbf{p}, \text{bin}(j), \mathbf{w}_\ell, \dots, \mathbf{w}_1, \mathbf{r}_1, \mathbf{r}_2)$ satisfying conditions (i),(ii) and (iii) in the **Sign** algorithm. The completeness of the underlying argument system then guarantees that Σ is always accepted by algorithm **Verify**.

Next, we show that algorithm **Trace** outputs $\text{bin}(j)$ with overwhelming probability, and produces a proof Π_{trace} accepted by algorithm **Judge**. Observe that, the decryption algorithm essentially computes

$$\mathbf{e} = \mathbf{c}_{1,2} - \mathbf{S}_1^T \mathbf{c}_{1,1} = \mathbf{E}_1 \cdot \mathbf{r}_1 + \lfloor q/2 \rfloor \cdot \text{bin}(j) \pmod{q},$$

and sets the j -th bit of $\mathbf{b}'$ to be 0 if j -th entry of $\mathbf{e}$ is closer to 0 than to $\lfloor q/2 \rfloor$ and 1 otherwise. Note that our parameters are set so that $\|\mathbf{E}_1 \cdot \mathbf{r}_1\|_\infty < q/5$, for $\mathbf{E}_1 \leftarrow \chi^{\ell \times m_E}$ and $\mathbf{r}_1 \xleftarrow{\$} \{0, 1\}^{m_E}$. This ensures that $\mathbf{b}' = \text{bin}(j)$ with overwhelming probability.

Further, as the user is active, info_τ must contain $w = (\text{bin}(j), \mathbf{w}_\ell, \dots, \mathbf{w}_1)$ and $\text{reg}[j][1]$ in table **reg** is not $\mathbf{0}^{nk}$. Therefore, algorithm **Trace** will move to the 4-th step, where it can always obtain the tuple $(\mathbf{S}_1, \mathbf{E}_1, \mathbf{y})$ satisfying the conditions in (3.4). By the completeness of the argument system, Π_{trace} will be accepted by the algorithm **Judge**.

SECURITY. In Theorem 3.4.1, we prove that our scheme satisfies the security requirements of the Bootle et al.'s model [31].

Theorem 3.4.1. Assume that the Stern-like argument systems used in Section 3.4.1 are simulation-sound. Then, in the random oracle model, the given fully dynamic

group signature satisfies the anonymity, traceability, non-frameability and tracing soundness requirements under the $\text{LWE}_{n,q,\chi}$ and $\text{SIS}_{n,m,q,1}^\infty$ assumptions.

In the random oracle model, the proof of Theorem 3.4.1 relies on the following facts:

1. The Stern-like zero-knowledge argument systems being used are simulation-sound;
2. The underlying encryption scheme, which is obtained from Regev cryptosystem [160] via the Naor-Yung transformation [146], is IND-CCA secure;
3. The Merkle tree accumulator we employ is secure in the sense of Definition 3.3.1;
4. For a properly generated key-pair $(\mathbf{x}, \mathbf{p})$, it is infeasible to find $\mathbf{x}' \in \{0, 1\}^m$ such that $\mathbf{x}' \neq \mathbf{x}$ and $\text{bin}(\mathbf{A} \cdot \mathbf{x}' \bmod q) = \mathbf{p}$.

The proof of Theorem 3.4.1 is established by Lemmas 3.4.2-3.4.5 given below.

Lemma 3.4.2. Assume that the $\text{LWE}_{n,q,\chi}$ problem is hard. Then the given FDGS scheme provides anonymity in the random oracle model.

Proof. Let $\mathcal{A}, \mathcal{B}$ be PPT algorithms, act as the adversary and challenger in the games respectively. We construct a sequence of indistinguishable games, in which the first game is the experiment $\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-0}(\lambda)$ and the last one is the experiment $\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-1}(\lambda)$. We will prove the lemma by demonstrating that any two consecutive games are indistinguishable. For each i , we denote by W_i the output of the adversary in game i .

Specifically, we consider the following games.

Game 0: This is exactly the experiment $\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-0}(\lambda)$.

Game 1: This game is the same as Game 0 with only one modification: add $(\mathbf{S}_2, \mathbf{E}_2)$ to tsk . This will make no difference to the view of the adversary. So we have $\Pr[W_1 = 1] = \Pr[\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-0}(\lambda) = 1]$.

Game 2: This game is identical to Game 1 except that it generates a simulated proof for the oracle **Trace** by programming the random oracle $\mathcal{H}_{\text{FS}}$ even though the challenger $\mathcal{B}$ has correct witnesses to generate a real proof. The view of the adversary, however, is statistically indistinguishable between Game 1 and Game 2 by statistical zero-knowledge property of our argument system generating Π_{trace} . Therefore we have $|\Pr[W_1 = 1] - \Pr[W_2 = 1]| = \text{negl}(\lambda)$.

Game 3: This game uses $\mathbf{S}_2$ instead of $\mathbf{S}_1$ to simulate the oracle **Trace**. The view of $\mathcal{A}$ is the same as in Game 2 until event F_1 , where $\mathcal{A}$ queries the **trace** oracle a valid signature $(M, \Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2, \text{info}_\tau)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting distinct bit strings, happens. Since F_1 breaks the soundness of our argument generating Π_{gs} , we have $|\Pr[W_2 = 1] - \Pr[W_3 = 1]| \leq \Pr[F_1] \leq \text{Adv}_{\Pi_{\text{gs}}}^{\text{sound}}(\lambda) = \text{negl}(\lambda)$.

Game 4: This game generates a simulated proof for the oracle **Chal**. The view is indistinguishable to $\mathcal{A}$ by statistical zero-knowledge property of our argument system generating Π_{gs} . Therefore we have $|\Pr[W_3 = 1] - \Pr[W_4 = 1]| = \text{negl}(\lambda)$.

Game 5: This is the same as Game 4 except that $\mathbf{c}_1$ is now encryption of $\text{bin}(i_1)$ while $\mathbf{c}_2$ is still encryption of $\text{bin}(i_0)$ for the **Chal** query. By the semantic security of our encryption scheme for public key $(\mathbf{B}, \mathbf{P}_1)$ (which relies on **LWE** assumption), this change is negligible to the adversary. Recall that we use $\mathbf{S}_2$ for the **Trace** queries. So the change of $\mathbf{c}_1$ makes no difference to the view of the adversary. Therefore we have $|\Pr[W_4 = 1] - \Pr[W_5 = 1]| = \text{negl}(\lambda)$.

Game 6: This game follows Game 5 with one change: it switches back to use $\mathbf{S}_1$ for the **Trace** queries and discards $(\mathbf{S}_2, \mathbf{E}_2)$. This modification is indistinguishable to the adversary until event F_2 , where $\mathcal{A}$ queries the **trace** oracle a valid signature $(M, \Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2, \text{info}_\tau)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting different bit strings, happens. However event F_2 breaks the simulation soundness of the argument system generating Π_{gs} . Therefore we have $|\Pr[W_5 = 1] - \Pr[W_6 = 1]| \leq \text{Adv}_{\Pi_{\text{gs}}}^{\text{ss}}(\lambda) = \text{negl}(\lambda)$.

Game 7: In this game, it changes $\mathbf{c}_2$ to be encryption of $\text{bin}(i_1)$ for the **Chal** query while the remaining parts are the same as in Game 6. By the semantic security of our encryption scheme for public key $(\mathbf{B}, \mathbf{P}_2)$, this change is negligible to the adversary. Note that we now use $\mathbf{S}_1$ for the **Trace** queries. So changing $\mathbf{c}_2$ makes no difference to the view of the adversary. Therefore we obtain the following: $|\Pr[W_6 = 1] - \Pr[W_7 = 1]| = \text{negl}(\lambda)$.

Game 8: We now generate real proof for the **Chal** query instead of simulated proof. By the statistical zero-knowledge property of our argument system generating Π_{gs} , the view of the adversary in Game 7 and Game 8 are statistically indistinguishable, i.e., we have $|\Pr[W_7 = 1] - \Pr[W_8 = 1]| = \text{negl}(\lambda)$.

Game 9: This game produces real proof, for the **Trace** queries. By the statistical zero-knowledge property of our argument system generating Π_{trace} , we have Game 8 and Game 9 are statistically indistinguishable to the adversary, i.e., we have $\Pr[W_8 = 1] \approx \Pr[W_9 = 1]$. This is actually the experiment $\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-1}(\lambda)$. Hence, we have $\Pr[W_9 = 1] = \Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-1}(\lambda) = 1]$.

As a result, we have that

$$|\Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{anon}-0}(\lambda) = 1]| = \text{negl}(\lambda),$$

and hence our scheme is anonymous. $\square$

Lemma 3.4.3. Assume that the $\text{SIS}_{n,m,q,1}^\infty$ problem is hard. Then the given FDGS scheme provides non-frameability in the random oracle model.

Proof. We prove non-frameability by contradiction. Suppose that $\mathcal{A}$ succeeds with non-negligible advantage ϵ . Then we can build a PPT algorithm $\mathcal{B}$ that either breaks the security of our accumulator or solves $\text{SIS}_{n,q,m,1}^\infty$ problem, which are featured by $\mathbf{A}$, with also non-negligible probability.

Given a matrix $\mathbf{A}$ from the environment that $\mathcal{B}$ is in, it first generates all parameters $\mathbf{pp}$ as we do in $\mathbf{GSetup}$, then invokes $\mathcal{A}$ with $\mathbf{pp}$, and then proceeds as described in the experiment. Here $\mathcal{B}$ can consistently answer all the oracle queries made by $\mathcal{A}$. When $\mathcal{A}$ outputs $(M^*, \Sigma^*, \text{bin}(j^*), \Pi_{\text{trace}}^*, \text{info}_\tau)$ and wins, $\mathcal{B}$ does following.

Parse Σ^* as $(\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$, where $\Pi_{\text{gs}}^* = (\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}_i^*\}_{i=1}^\kappa)$ and RSP_i^* is a valid response w.r.t CMT_i^* and CH_i^* for $i \in [\kappa]$ since $\mathcal{A}$ wins. We claim that $\mathcal{A}$ queried the tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^k, \mathbf{A}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1^*, \mathbf{c}_2^*)$, denoted as ξ^* , to the hash oracle $\mathcal{H}_{\text{FS}}$ with overwhelming probability. Otherwise guessing correctly this value occurs only with probability $3^{-\kappa}$, which is negligible. Therefore, with probability $\epsilon' = \epsilon - 3^{-\kappa}$, the tuple ξ^* has been an input of one hash query, denoted as $t^* \in \{1, 2, \dots, Q_H\}$, where Q_H is the total number of hash queries made by $\mathcal{A}$. To employ the Forking lemma of Brickell et al. [45], $\mathcal{B}$ runs polynomial-number executions of $\mathcal{A}$ exactly the same as in the original run until the t^* hash query, that is, from t^* hash query on, $\mathcal{B}$ will answer $\mathcal{A}$ with fresh and independent values for hash queries on each new execution. Note that the input of t^* hash query must be ξ^* as in the original run. By the Forking Lemma [45], with probability $\geq \frac{1}{2}$, $\mathcal{B}$ obtains 3-fork involving the same tuple ξ^* with pairwise distinct hash value $\text{CH}_{t^*}^{(1)}, \text{CH}_{t^*}^{(2)}, \text{CH}_{t^*}^{(3)} \in \{1, 2, 3\}^\kappa$. We have $(\text{CH}_{t^*,j}^{(1)}, \text{CH}_{t^*,j}^{(2)}, \text{CH}_{t^*,j}^{(3)}) = (1, 2, 3)$ with probability $1 - (\frac{7}{9})^\kappa$ for some $j \in \{1, 2, \dots, \kappa\}$. Then by the argument of knowledge of the system generating Π_{gs} , from the valid responses $(\text{RSP}_{t^*,j}^{(1)}, \text{RSP}_{t^*,j}^{(2)}, \text{RSP}_{t^*,j}^{(3)})$, we can extract the witnesses $\zeta' = (\mathbf{x}', \mathbf{p}', w'_\tau, \mathbf{r}'_1, \mathbf{r}'_2)$, where

$$w'_\tau = (\text{bin}(j'), \mathbf{w}'_{\ell, \tau}, \dots, \mathbf{w}'_{1, \tau}) \in \{0, 1\}^\ell \times (\{0, 1\}^{nk})^\ell,$$

such that we have:

$$(i) \quad \text{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}', w'_\tau) = 1, \mathbf{p}' \neq 0;$$

$$(ii) \quad \mathbf{A} \cdot \mathbf{x}' = \mathbf{G} \cdot \mathbf{p}' \pmod{q};$$

(iii) $\mathbf{c}_1^*, \mathbf{c}_2^*$ are encryptions of $\text{bin}(j')$ with randomness $\mathbf{r}'_1$ and $\mathbf{r}'_2$.

By the correctness of our encryption scheme, $\mathbf{c}_1^*$ is decrypted to $\text{bin}(j')$. The fact $\mathcal{A}$ wins implies algorithm **Judge** outputs 1. By the soundness of our system generating Π_{trace} , $\text{bin}(j^*)$ is decrypted from $\mathbf{c}_1^*$. Hence we have $\text{bin}(j') = \text{bin}(j^*)$ with overwhelming probability.

Now we consider the following two cases:

Case 1: $\text{bin}(j')$ is an inactive user at epoch τ , i.e., the value at leaf $\text{bin}(j')$ is $\mathbf{0}^{nk}$. This violates the accumulator security since we have $\text{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}', w'_\tau) = 1$ with $\mathbf{p}' \neq \mathbf{0}$ at leaf $\text{bin}(j')$.

Case 2: $\text{bin}(j^*)$ is an active user at epoch τ . The fact $\text{bin}(j^*) \in \text{HUL} \setminus \text{BUL}$ indicates $\mathcal{A}$ does not know $\text{gsk}[j^*] = (\text{bin}(j^*), \mathbf{p}', \mathbf{x}_{j^*})$, where $\mathbf{x}_{j^*}$ was initially chosen by $\mathcal{B}$ and satisfies $\mathbf{A} \cdot \mathbf{x}_{j^*} = \mathbf{G} \cdot \mathbf{p}' \pmod{q}$. Following the same argument of [108], we claim that $\mathbf{x}_{j^*} \neq \mathbf{x}'$ with probability at least $1/2$. Then we find a nonzero vector $\mathbf{z} = \mathbf{x}_{j^*} - \mathbf{x}'$ satisfying $\mathbf{A}\mathbf{z} = \mathbf{0} \pmod{q}$. Recall $\mathbf{x}_{j^*}, \mathbf{x}' \in \{0, 1\}^m$. Thus $\mathcal{B}$ solves a $\text{SIS}_{n,q,m,1}^\infty$ problem with non-negligible probability if $\mathcal{A}$ breaks the non-frameability of our construction with non-negligible probability.

□

Lemma 3.4.4. Assume that the $\text{SIS}_{n,m,q,1}^\infty$ problem is hard. Then the given FDGS scheme satisfies traceability in the random oracle model.

Proof. Recall that adversary $\mathcal{A}$ wins traceability experiment if it generates a valid signature that either: (i) traces to an inactive user; or (ii) traces to an active user but we cannot generate a proof accepted by the algorithm **Judge**. We will prove that both cases occur with negligible probability.

Let $(\text{info}_\tau, M, \Sigma)$ be the output of $\mathcal{A}$ in the experiment $\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{trace}}(\lambda)$ and we compute $(\text{bin}(j), \Pi_{\text{trace}})$ by running the algorithm **Trace**. Parse Σ as $(\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$, where $\Pi_{\text{gs}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}_i\}_{i=1}^\kappa)$, and RSP_i is a valid response w.r.t. CMT_i and CH_i for $i \in [\kappa]$ since $(\text{info}_\tau, M, \Sigma)$ is a valid signature outputted by $\mathcal{A}$. Now

we are in the same situation as in Lemma 3.4.3. By the same technique, we can extract witnesses $\zeta = (\mathbf{x}, \mathbf{p}, w_\tau, \mathbf{r}_1, \mathbf{r}_2)$, where $w_\tau = (\text{bin}(j'), \mathbf{w}_{\ell, \tau}, \dots, \mathbf{w}_{1, \tau}) \in \{0, 1\}^\ell \times (\{0, 1\}^{nk})^\ell$ satisfying $\text{TVerify}_\mathbf{A}(\mathbf{u}_\tau, \mathbf{p}, w_\tau) = 1$, $\mathbf{p} \neq \mathbf{0}^{nk}$, and $\mathbf{c}_1, \mathbf{c}_2$ are encryptions of $\text{bin}(j')$ using randomness $\mathbf{r}_1$ and $\mathbf{r}_2$. Correctness of our encryption scheme implies $\mathbf{c}_1$ is decrypted to $\text{bin}(j')$. Correct decryption of the **Trace** algorithm, which is run by the challenger, indicates that $\text{bin}(j)$ is decrypted from $\mathbf{c}_1$. Hence we have $\text{bin}(j') = \text{bin}(j)$ with overwhelming probability.

Now, we note that case (i) considered above only happens with negligible probability. In fact, if $\text{bin}(j)$ is not active at epoch τ , and $\text{TVerify}_\mathbf{A}(\mathbf{u}_\tau, \mathbf{p}, w_\tau) = 1$ with $\mathbf{p} \neq \mathbf{0}$ at leaf $\text{bin}(j)$, then the security of our accumulator is violated. Furthermore, case (ii) also occurs with negligible probability since the challenger possesses valid witnesses to generate the proof Π_{trace} , which will be accepted by the **Judge** algorithm, thanks to the completeness of the underlying argument system. It then follows that our scheme satisfies the traceability requirement. $\square$

Lemma 3.4.5. The given FDGS scheme satisfies tracing soundness in the random oracle model.

Proof. Let $(M, \Sigma, \text{bin}(j_0), \Pi_{\text{trace}, 0}, \text{bin}(j_1), \Pi_{\text{trace}, 1}, \text{info}_\tau)$ be the output of the adversary in the experiment $\text{Exp}_{\text{FDGS}, \mathcal{A}}^{\text{trace-sound}}(\lambda)$. Recall that adversary wins if the following conditions hold:

- (i) $\text{bin}(j_0) \neq \text{bin}(j_1)$;
- (ii) $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 1$;
- (iii) $\text{Judge}(\text{gpk}, \text{bin}(j_b), \text{info}_\tau, \Pi_{\text{trace}, b}, M, \Sigma) = 1$ for $b \in \{0, 1\}$;
- (iv) $\text{bin}(j_b) \neq \perp$ for $b \in \{0, 1\}$.

Let us parse $\Pi_{\text{trace}, b}$ as $(\{\text{CMT}_{i,b}\}_{i=1}^\kappa, \text{CH}_b, \{\text{RSP}_{i,b}\}_{i=1}^\kappa)$. The fact that the **Judge** algorithm outputs 1 implies that $\text{RSP}_{i,b}$ is a valid response w.r.t. $\text{CMT}_{i,b}$ and $\text{CH}_{i,b}$ for

$i \in [\kappa]$ and $b \in \{0, 1\}$. By using the extraction technique for argument system generating Π_{trace} as in Lemma 3.4.3, we can extract the witnesses $\mathbf{S}_{1,b}, \mathbf{E}_{1,b}, \mathbf{y}_b$ for $b \in \{0, 1\}$ satisfying:

$$\begin{cases} \|\mathbf{S}_{1,b}\|_\infty \leq \beta; \|\mathbf{E}_{1,b}\|_\infty \leq \beta; \|\mathbf{y}_b\|_\infty \leq \lceil q/5 \rceil; \\ \mathbf{S}_{1,b}^\top \cdot \mathbf{B} + \mathbf{E}_{1,b} = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_{1,b}^\top \cdot \mathbf{c}_{1,1} = \mathbf{y}_b + \lfloor q/2 \rfloor \cdot \text{bin}(j_b) \bmod q. \end{cases} \quad (3.6)$$

Note that, we have $(\mathbf{S}_{1,0}^\top - \mathbf{S}_{1,1}^\top) \cdot \mathbf{c}_{1,1} = \mathbf{y}_1 - \mathbf{y}_0 + \lfloor q/2 \rfloor \cdot (\text{bin}(j_1) - \text{bin}(j_0)) \bmod q$. Suppose that $\text{bin}(j_0) \neq \text{bin}(j_1)$, then $\|\lfloor q/2 \rfloor \cdot (\text{bin}(j_1) - \text{bin}(j_0))\|_\infty = \lfloor q/2 \rfloor$. On the other hand, we have $\|\mathbf{y}_1 - \mathbf{y}_0\|_\infty \leq 2 \cdot \lceil q/5 \rceil$. It then follows that

$$\|\mathbf{y}_1 - \mathbf{y}_0 + \lfloor q/2 \rfloor \cdot (\text{bin}(j_1) - \text{bin}(j_0))\|_\infty > 0,$$

which implies that $\mathbf{S}_{1,0}^\top \neq \mathbf{S}_{1,1}^\top$. In other words, we can obtain two solutions for the equation $\mathbf{S}^\top \cdot \mathbf{B} + \mathbf{E} = \mathbf{P}_1 \bmod q$, which contradicts the fact that there exists at most one solution for LWE samples $(\mathbf{B}, \mathbf{P}_1)$.

By contradiction, we have $\text{bin}(j_0) = \text{bin}(j_1)$ with overwhelming probability. This implies that case (i) does not hold with overwhelming probability once cases (ii),(iii) and (iv) hold. As a result, the advantage of the adversary attacking the tracing soundness of our scheme is negligible (in λ). In other words, our construction satisfies tracing soundness. $\square$

3.4.3 The Main Zero-Knowledge Argument System

In this section, let us present the statistical ZKAoK that will be invoked by the signer when generating group signatures. Our protocol is a modification of the one from [108], in which we additionally prove that $\mathbf{p} \neq \mathbf{0}^{nk}$ using the technique discussed at the beginning of Section 3.4. We will employ a new strategy for Stern-like protocols,

suggested in [105], which consists of unifying all the equations to be proved into just one equation of the form $\mathbf{M} \cdot \mathbf{z} = \mathbf{u} \bmod q$, for some public matrix $\mathbf{M}$ and vector $\mathbf{u}$ over $\mathbb{Z}_q$. This strategy will help us to deliver a simpler presentation than in [108].

Let us first examine the equations associated with the execution of algorithm $\text{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}, ((j_1, \dots, j_\ell), (\mathbf{w}_\ell, \dots, \mathbf{w}_1)))$. This algorithm computes the tree path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1, \mathbf{v}_0 \in \{0, 1\}^{nk}$ as follows: $\mathbf{v}_\ell = \mathbf{p}$ and

$$\forall i \in \{\ell - 1, \dots, 1, 0\} : \mathbf{v}_i = \begin{cases} h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), & \text{if } j_{i+1} = 0; \\ h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), & \text{if } j_{i+1} = 1. \end{cases} \quad (3.7)$$

The algorithm outputs 1 if $\mathbf{v}_0 = \mathbf{u}_\tau$. Observe that relation (3.7) can be equivalently rewritten as: $\forall i \in \{\ell - 1, \dots, 1, 0\}$,

$$\begin{aligned} & \bar{j}_{i+1} \cdot h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}) + j_{i+1} \cdot h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}) = \mathbf{v}_i \\ \Leftrightarrow & \bar{j}_{i+1} \cdot (\mathbf{A}_0 \cdot \mathbf{v}_{i+1} + \mathbf{A}_1 \cdot \mathbf{w}_{i+1}) + j_{i+1} \cdot (\mathbf{A}_0 \cdot \mathbf{w}_{i+1} + \mathbf{A}_1 \cdot \mathbf{v}_{i+1}) = \mathbf{G} \cdot \mathbf{v}_i \bmod q \\ \Leftrightarrow & \mathbf{A} \cdot \begin{pmatrix} \bar{j}_{i+1} \cdot \mathbf{v}_{i+1} \\ j_{i+1} \cdot \mathbf{v}_{i+1} \end{pmatrix} + \mathbf{A} \cdot \begin{pmatrix} j_{i+1} \cdot \mathbf{w}_{i+1} \\ \bar{j}_{i+1} \cdot \mathbf{w}_{i+1} \end{pmatrix} = \mathbf{G} \cdot \mathbf{v}_i \bmod q \\ \Leftrightarrow & \mathbf{A} \cdot \text{ext}(j_{i+1}, \mathbf{v}_{i+1}) + \mathbf{A} \cdot \text{ext}(\bar{j}_{i+1}, \mathbf{w}_{i+1}) = \mathbf{G} \cdot \mathbf{v}_i \bmod q. \end{aligned}$$

In the above, we use the notation $\text{ext}(b, \mathbf{v})$, for bit b and vector $\mathbf{v}$, to denote the vector $\begin{pmatrix} \bar{b} \cdot \mathbf{v} \\ b \cdot \mathbf{v} \end{pmatrix}$. Later on, we will also use the notation $\text{ext}_2(b)$, for bit b , to denote the 2-dimensional vector $(\bar{b}, b)^\top$.

Given the above discussion, our protocol can be summarized as follows.

Public input: $\mathbf{A}, \mathbf{G}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2}), \mathbf{c}_2 = (\mathbf{c}_{2,1}, \mathbf{c}_{2,2})$.

Prover's goal: Proving knowledge of secret $\mathbf{x} \in \{0, 1\}^m$, $\mathbf{0} \neq \mathbf{p} \in \{0, 1\}^{nk}$, $j_1, \dots, j_\ell \in$

$\{0, 1\}$, $\mathbf{v}_1, \dots, \mathbf{v}_{\ell-1}, \mathbf{w}_1, \dots, \mathbf{w}_\ell \in \{0, 1\}^{nk}$, $\mathbf{r}_1, \mathbf{r}_2 \in \{0, 1\}^{m_E}$, such that the following equations hold:

$$\left\{ \begin{array}{l} \mathbf{A} \cdot \text{ext}(j_1, \mathbf{v}_1) + \mathbf{A} \cdot \text{ext}(\bar{j}_1, \mathbf{w}_1) = \mathbf{G} \cdot \mathbf{u}_\tau \bmod q; \\ \mathbf{A} \cdot \text{ext}(j_2, \mathbf{v}_2) + \mathbf{A} \cdot \text{ext}(\bar{j}_2, \mathbf{w}_2) - \mathbf{G} \cdot \mathbf{v}_1 = \mathbf{0} \bmod q; \\ \dots \dots \dots \\ \mathbf{A} \cdot \text{ext}(j_\ell, \mathbf{p}) + \mathbf{A} \cdot \text{ext}(\bar{j}_\ell, \mathbf{w}_\ell) - \mathbf{G} \cdot \mathbf{v}_{\ell-1} = \mathbf{0} \bmod q; \\ \mathbf{A} \cdot \mathbf{x} - \mathbf{G} \cdot \mathbf{p} = \mathbf{0} \bmod q; \\ \mathbf{B} \cdot \mathbf{r}_b = \mathbf{c}_{b,1} \bmod q, \text{ for } b \in \{1, 2\}; \\ \mathbf{P}_b \cdot \mathbf{r}_b + \left\lfloor \frac{q}{2} \right\rfloor \cdot (j_1, \dots, j_\ell)^\top = \mathbf{c}_{b,2} \bmod q, \text{ for } b \in \{1, 2\}. \end{array} \right. \quad (3.8)$$

Now, we employ extending-then-permuting techniques for Stern-like protocols [105, 108, 116, 173] to handle the secret objects, as follows. For any positive integer i , we let $\mathcal{S}_i$ denote the set of all permutations of i elements.

- To prove that $\mathbf{p} \in \{0, 1\}^{nk}$ and $\mathbf{p} \neq \mathbf{0}^{nk}$, we append $nk - 1$ “dummy” entries to $\mathbf{p}$ to get $\mathbf{p}^* \in \mathbb{B}_{nk}^{2nk-1}$. Note that, for any $\pi_p \in \mathcal{S}_{2nk-1}$, we have:

$$\mathbf{p}^* \in \mathbb{B}_{nk}^{2nk-1} \Leftrightarrow \pi_p(\mathbf{p}^*) \in \mathbb{B}_{nk}^{2nk-1}.$$

- To prove that $\mathbf{x} \in \{0, 1\}^m$, we append m “dummy” entries to $\mathbf{x}$ to get $\mathbf{x}^* \in \mathbb{B}_m^{2m}$. Note that, for any $\pi_x \in \mathcal{S}_{2m}$, we have: $\mathbf{x}^* \in \mathbb{B}_m^{2m} \Leftrightarrow \pi_x(\mathbf{x}^*) \in \mathbb{B}_m^{2m}$.
- Similarly, we extend $\mathbf{v}_1, \dots, \mathbf{v}_{\ell-1}, \mathbf{w}_1, \dots, \mathbf{w}_\ell$ to $\mathbf{v}_1^*, \dots, \mathbf{v}_{\ell-1}^*, \mathbf{w}_1^*, \dots, \mathbf{w}_\ell^* \in \mathbb{B}_{nk}^{2nk}$, respectively. We also extend $\mathbf{r}_1, \mathbf{r}_2$ to $\mathbf{r}_1^*, \mathbf{r}_2^* \in \mathbb{B}_{m_E}^{2m_E}$, respectively.
- For each $i = 1, \dots, \ell$, to prove that the bit $j_i \in \{0, 1\}$ is involved in both encryption layer and Merkle tree layer, we extend it to $\mathbf{j}_i = \text{ext}_2(j_i)$. Then we use the following permuting technique.

For bit $b \in \{0, 1\}$, we let T_b be the permutation that transforms vector $\mathbf{t} = (t_0, t_1)^\top \in \mathbb{Z}^2$ into vector $T_b(\mathbf{t}) = (t_b, t_{\bar{b}})$. Note that, for any $b_i \in \{0, 1\}$, we

have: $\mathbf{j}_i = \text{ext}_2(j_i) \Leftrightarrow T_{b_i}(\mathbf{j}_i) = \text{ext}_2(j_i \oplus b_i)$, where $\oplus$ denotes the addition modulo 2. Then, to prove knowledge of $j_i \in \{0, 1\}$, we instead prove knowledge of $\mathbf{j}_i = \text{ext}_2(j_i)$, by sampling $b_i \xleftarrow{\$} \{0, 1\}$ and showing the verifier that the right-hand side of the equivalence holds. Here, the crucial point is that b_i acts as a “one-time pad” that perfectly hides j_i . Furthermore, to prove that j_i appears in a different layer of the system, we will set up a similar equivalence in that layer, and use the same b_i at both places.

- Now, we have to handle the extended vectors appearing in the context after the above extensions. They are $\widehat{\mathbf{v}}_1 = \text{ext}(j_1, \mathbf{v}_1^*), \dots, \widehat{\mathbf{v}}_{\ell-1} = \text{ext}(j_{\ell-1}, \mathbf{v}_{\ell-1}^*) \in \{0, 1\}^{4nk}$ and $\widehat{\mathbf{w}}_1 = \text{ext}(\bar{j}_1, \mathbf{w}_1^*), \dots, \widehat{\mathbf{w}}_\ell = \text{ext}(\bar{j}_\ell, \mathbf{w}_\ell^*) \in \{0, 1\}^{4nk}$, as well as $\widehat{\mathbf{p}} = \text{ext}(j_\ell, \mathbf{p}^*) \in \{0, 1\}^{4nk-2}$. To prove that these vector are well-formed, we will use the following permuting techniques.

Let $r \in \{2nk, 2nk - 1\}$. For $b \in \{0, 1\}$ and $\pi \in \mathcal{S}_r$, we define the permutation $F_{b,\pi}$ that transforms $\mathbf{t} = \begin{pmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \end{pmatrix} \in \mathbb{Z}^{2r}$ consisting of 2 blocks of size r into $F_{b,\pi}(\mathbf{t}) = \begin{pmatrix} \pi(\mathbf{t}_b) \\ \pi(\mathbf{t}_{\bar{b}}) \end{pmatrix}$. Namely, $F_{b,\pi}$ first rearranges the blocks of $\mathbf{t}$ according to b (it keeps the arrangement of blocks if $b = 0$, or swaps them if $b = 1$), then it permutes each block according to π . Note that, we have the following, for all $b_1, \dots, b_\ell \in \{0, 1\}$, $\phi_{v,1}, \dots, \phi_{v,\ell-1}, \phi_{w,1}, \dots, \phi_{w,\ell} \in \mathcal{S}_{2nk}$, and $\pi_p \in \mathcal{S}_{2nk-1}$:

$$\begin{cases} \forall i \in [\ell - 1] : \widehat{\mathbf{v}}_i = \text{ext}(j_i, \mathbf{v}_i^*) \iff F_{b_i, \phi_{v,i}}(\widehat{\mathbf{v}}_i) = \text{ext}(j_i \oplus b_i, \phi_{v,i}(\mathbf{v}_i^*)); \\ \forall i \in [\ell] : \widehat{\mathbf{w}}_i = \text{ext}(\bar{j}_i, \mathbf{w}_i^*) \iff F_{b_i, \phi_{w,i}}(\widehat{\mathbf{w}}_i) = \text{ext}(\bar{j}_i \oplus b_i, \phi_{w,i}(\mathbf{w}_i^*)); \\ \widehat{\mathbf{p}} = \text{ext}(j_\ell, \mathbf{p}^*) \iff F_{b_\ell, \pi_p}(\widehat{\mathbf{p}}) = \text{ext}(j_\ell \oplus b_\ell, \pi_p(\mathbf{p}^*)). \end{cases}$$

Combining these permuting techniques with the ones described earlier, we can prove that the considered extended vectors are well-formed, with respect to the secret $j_1, \dots, j_\ell, \mathbf{v}_1^*, \dots, \mathbf{v}_{\ell-1}^*, \mathbf{w}_1^*, \dots, \mathbf{w}_\ell^*, \mathbf{p}^*$.

Given the above transformations, we now unify all the secret objects into vector $\mathbf{z} \in \{0, 1\}^L$, where $L = 10nk\ell + 2m + 4m_E + 2\ell - 3$, of the following form (we abuse the notation of transposition):

$$\left(\mathbf{v}_1^* \parallel \widehat{\mathbf{v}}_1 \parallel \widehat{\mathbf{w}}_1 \parallel \dots \parallel \mathbf{v}_{\ell-1}^* \parallel \widehat{\mathbf{v}}_{\ell-1} \parallel \widehat{\mathbf{w}}_{\ell-1} \parallel \mathbf{p}^* \parallel \widehat{\mathbf{p}} \parallel \widehat{\mathbf{w}}_\ell \parallel \mathbf{x}^* \parallel \mathbf{r}_1^* \parallel \mathbf{r}_2^* \parallel \mathbf{j}_1 \parallel \dots \parallel \mathbf{j}_\ell \right). \quad (3.9)$$

Then, we observe that, the equations in (3.8) can be equivalently combined into one equation $\mathbf{M} \cdot \mathbf{z} = \mathbf{u} \bmod q$, where matrix $\mathbf{M}$ and vector $\mathbf{u}$ are built from the public input.

Next, to apply the permuting techniques we have discussed above, let us define **VALID** as the set of all vectors in $\{0, 1\}^L$, that have the form (3.9), where $\mathbf{x}^* \in \mathcal{B}_m^{2m}$, $\mathbf{r}_1^*, \mathbf{r}_2^* \in \mathcal{B}_{m_E}^{2m_E}$, $\mathbf{p}^* \in \mathcal{B}_{nk}^{2nk-1}$, $\mathbf{v}_1^*, \dots, \mathbf{v}_{\ell-1}^* \in \mathcal{B}_{nk}^{2nk}$, and there exist $j_1, \dots, j_\ell \in \{0, 1\}$, $\mathbf{w}_1^*, \dots, \mathbf{w}_\ell^* \in \mathcal{B}_{nk}^{2nk}$, such that:

$$\begin{cases} \widehat{\mathbf{p}} = \text{ext}(j_\ell, \mathbf{p}^*), \quad \forall i \in [\ell] : \mathbf{j}_i = \text{ext}_2(j_i), \\ \forall i \in [\ell - 1] : \widehat{\mathbf{v}}_i = \text{ext}(j_i, \mathbf{v}_i^*), \quad \forall i \in [\ell] : \widehat{\mathbf{w}}_i = \text{ext}(\bar{j}_i, \mathbf{w}_i^*). \end{cases}$$

It can be seen that our vector $\mathbf{z}$ belongs to this tailored set **VALID**. To prove that $\mathbf{z} \in \mathbf{VALID}$ using random permutations, let us determine how to permute the coordinates of $\mathbf{z}$. To this end, we first define the set:

$$\overline{\mathcal{S}} = \{0, 1\}^\ell \times \mathcal{S}_{2m} \times \mathcal{S}_{2nk-1} \times (\mathcal{S}_{2m_E})^2 \times (\mathcal{S}_{2nk})^{2\ell-1}.$$

Then, $\forall \eta = \left((b_1, \dots, b_\ell), \pi_x, \pi_p, (\pi_{r,1}, \pi_{r,2}), (\phi_{v,1}, \dots, \phi_{v,\ell-1}, \phi_{w,1}, \dots, \phi_{w,\ell}) \right) \in \overline{\mathcal{S}}$, we let Γ_η be the permutation that, when applying to vector $\mathbf{t} \in \mathbb{Z}^L$ whose blocks are as in (3.9), it transforms those blocks as follows:

- For all $i \in [\ell - 1]$: $\mathbf{v}_i^* \mapsto \phi_{v,i}(\mathbf{v}_i^*)$; $\widehat{\mathbf{v}}_i \mapsto F_{b_i, \phi_{v,i}}(\widehat{\mathbf{v}}_i)$.
- For all $i \in [\ell]$: $\widehat{\mathbf{w}}_i \mapsto F_{b_i, \phi_{w,i}}(\widehat{\mathbf{w}}_i)$; $\mathbf{j}_i \mapsto T_{b_i}(\mathbf{j}_i)$.

- $\mathbf{p}^* \mapsto \pi_p(\mathbf{p}^*); \quad \hat{\mathbf{b}} \mapsto F_{b_\ell, \pi_p}(\hat{\mathbf{p}}).$
- $\mathbf{x}^* \mapsto \pi_x(\mathbf{x}^*); \quad \mathbf{r}_1^* \mapsto \pi_{r,1}(\mathbf{r}_1^*); \quad \mathbf{r}_2^* \mapsto \pi_{r,2}(\mathbf{r}_2^*).$

It now can be checked that we have the desired equivalence: For all $\eta \in \overline{\mathcal{S}}$,

$$\mathbf{z} \in \text{VALID} \iff \Gamma_\eta(\mathbf{z}) \in \text{VALID}.$$

Furthermore, $\Gamma_\eta(\mathbf{z})$ is uniform in **VALID** if η is uniform in $\overline{\mathcal{S}}$ and $\mathbf{z} \in \text{VALID}$. In other words, **VALID**, $\overline{\mathcal{S}}$ and Γ_η satisfy the conditions specified in (2.1) and we have reduced the considered statement to an instance of the abstract protocol from 2.4.2. Therefore, both parties run the protocol of Figure 2.1 and we obtain a statistical ZKAoK for the considered statement, with perfect completeness, soundness error $2/3$ and communication cost $\mathcal{O}(L \cdot \log q)$, which is of order $\mathcal{O}(n \cdot \log^2 n \cdot \ell) = \tilde{\mathcal{O}}(\lambda \cdot \ell)$ for the setting of parameters we use in our scheme from Section 3.4.1.

3.4.4 Zero-Knowledge Argument of Correct Decryption

We now present the underlying ZKAoK used by the tracing manager to generate Π_{trace} . The protocol allows the prover to prove knowledge of decryption keys for the multi-bit version of Regev's encryption scheme, as well as prove the correctness of decryption. While the argument system in Section 3.4.3 deals only with $\{0, 1\}$ witness vectors, here, we have to handle secret vectors/matrices of infinity norm larger than 1. To this end, we will employ the decomposition technique recalled in Section 2.4.3 to transform them to vectors with infinity norm 1.

Once we have obtained a transformed witness vector in the set $\{-1, 0, 1\}$, we then employ the usual extending-permuting technique for Stern-like protocols as in [105, 108, 116, 173]. For any positive integer m , let $\mathbf{R}_m^3$ be the set of all vectors in $\{-1, 0, 1\}^{3m}$ that have exactly m coordinates equal to j , for each $j \in \{-1, 0, 1\}$. Note that this set is “closed” under permutations, i.e., for any $\pi \in \mathcal{S}_{3m}$ and any vector $\mathbf{x} \in \mathbf{R}_m^3$, the

following equivalence holds:

$$\mathbf{x} \in \mathbb{R}_m^3 \Leftrightarrow \pi(\mathbf{x}) \in \mathbb{R}_m^3. \quad (3.10)$$

Our protocol is summarized as follows.

Public input: $\mathbf{B} \in \mathbb{Z}_q^{n \times m_E}$, $\mathbf{P}_1 \in \mathbb{Z}_q^{\ell \times m_E}$, $\mathbf{c}_{1,1} \in \mathbb{Z}_q^n$, $\mathbf{c}_{1,2} \in \mathbb{Z}_q^\ell$, $\mathbf{b}' \in \{0, 1\}^\ell$.

Prover's goal: Proving knowledge of $\mathbf{S}_1 \in [-\beta, \beta]^{n \times \ell}$, $\mathbf{E}_1 \in [-\beta, \beta]^{\ell \times m_E}$, and $\mathbf{y} \in [-\lceil q/5 \rceil, \lceil q/5 \rceil]^\ell$, such that:

$$\begin{cases} \mathbf{S}_1^\top \cdot \mathbf{B} + \mathbf{E}_1 = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} = \mathbf{y} + \lfloor q/2 \rfloor \cdot \mathbf{b}' \bmod q. \end{cases} \quad (3.11)$$

For each $j \in [\ell]$, let $\mathbf{s}_j, \mathbf{e}_j, \mathbf{p}_j$ be the j -th column of matrix $\mathbf{S}_1, \mathbf{E}_1^\top, \mathbf{P}_1^\top$, respectively; and let y_j, c_j, b_j be the j -th entry of vector $\mathbf{y}, \mathbf{c}_{1,2}, \mathbf{b}'$, respectively. Then observe that (3.11) can be re-written as:

$$\forall j \in [\ell] : \begin{cases} \mathbf{B}^T \cdot \mathbf{s}_j + \mathbf{I}_{m_E} \cdot \mathbf{e}_j = \mathbf{p}_j \bmod q \\ \mathbf{c}_{1,1}^T \cdot \mathbf{s}_j + 1 \cdot y_j = c_j - b_j \cdot \lfloor q/2 \rfloor \bmod q. \end{cases} \quad (3.12)$$

Using basic algebra, we can manipulate the equations in (3.12) by rearranging the secret vectors and forming public matrices accordingly. As a result, we obtain a unifying equation of the form

$$\mathbf{M}_0 \cdot \mathbf{z}_0 = \mathbf{u} \bmod q,$$

where $\mathbf{M}_0$, $\mathbf{u}$ are built from the public input while $\mathbf{z}_0 = (\mathbf{z}_1 \parallel \mathbf{z}_2)$, with

$$\begin{cases} \mathbf{z}_1 = (\mathbf{s}_1 \parallel \cdots \parallel \mathbf{s}_\ell \parallel \mathbf{e}_1 \parallel \cdots \parallel \mathbf{e}_\ell) \in [-\beta, \beta]^{(n+m_E)\ell}, \\ \mathbf{z}_2 = (y_1, \dots, y_\ell)^\top \in \lceil -q/5 \rceil, \lceil q/5 \rceil^\ell. \end{cases}$$

Next, we can use the decomposition and extension techniques described above to handle our secret vectors. First, decompose $\mathbf{z}_1$ into $\bar{\mathbf{z}}_1 \in \{-1, 0, 1\}^{L_1}$ with $L_1 = (n + m_E)\ell\delta_\beta$ and $\mathbf{z}_2$ into $\bar{\mathbf{z}}_2 \in \{-1, 0, 1\}^{L_2}$ with $L_2 = \ell\delta_{\lceil q/5 \rceil}$, and form the secret vector $\bar{\mathbf{z}}_0 = (\bar{\mathbf{z}}_1 \parallel \bar{\mathbf{z}}_2) \in \{-1, 0, 1\}^{L'}$ with $L' = L_1 + L_2$. Then extend $\bar{\mathbf{z}}_0$ to $\mathbf{z} \in \mathbb{R}_{L'}^3$. Using some basic algebra, we can form public matrix $\mathbf{M}$ such that

$$\mathbf{M} \cdot \mathbf{z} = \mathbf{M}_0 \cdot \mathbf{z}_0 = \mathbf{u} \bmod q.$$

Now, define the tailored set $\text{VALID} = \mathbb{R}_{L'}^3$, and let $\bar{\mathcal{S}} = \mathcal{S}_L$ with $L = 3L'$. For every $\eta \in \bar{\mathcal{S}}$, define $\Gamma_\eta = \eta$. It is checked that our secret vector $\mathbf{z} \in \text{VALID}$. Observe the equivalence in (3.10), for any $\eta \in \bar{\mathcal{S}}$, the following equivalence also holds:

$$\mathbf{z} \in \text{VALID} \iff \Gamma_\eta(\mathbf{z}) \in \text{VALID}.$$

At this point, as in Section 3.4.3, we can see that the desired statistical ZKAoK protocol can be obtained by running the interactive protocol described in Fig. 2.1 and we obtain a statistical ZKAoK for the considered statement, with perfect completeness, soundness error $2/3$ and communication cost $\mathcal{O}(L \cdot \log q)$, which is of order $\tilde{\mathcal{O}}(\lambda \cdot \ell + \ell^2)$ for the setting of parameters we use in our scheme from Section 3.4.1.

3.5 Achieving Deniability with Ease

This section presents how to make our fully dynamic group signature scheme deniable, in the sense of [91] and in a relatively simple manner. In Section 3.5.1, we first

incorporate Ishida et al.’s notion of deniability into Bootle et al.’s model [31]. Then, we show in Section 3.5.2 that a scheme satisfying this extended model can be easily constructed based on the one from Section 3.4.1 and a zero-knowledge argument of “denial opening”, demonstrating that a given Regev ciphertext does not decrypt to a particular message. The zero-knowledge argument is described in detail in Section 3.5.3.

3.5.1 Fully Dynamic Group Signatures with Deniability

We now extend Bootle et al.’s model of FDGS (in Section 3.2) to capture the notion of deniability. Our extension follows Ishida et al.’s approach [91] when they incorporate the deniability functionality into Sakai et al.’s model [165].

A fully dynamic group signature scheme with deniability (FDGSwD) consists of all algorithms of an FDGS, as specified in Section 3.2, and two additional algorithms: DTrace and DJudge.

$\text{DTrace}(\text{gpk}, \text{tsk}, \text{info}_\tau, \text{reg}, \text{uid}', M, \Sigma) \rightarrow \Pi_{D(\text{uid}')}.$ The denial opening algorithm, run by the TM, returns a proof $\Pi_{D(\text{uid}')}$ showing that uid' did not compute the signature Σ on the message M at time τ .

$\text{DJudge}(\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{D(\text{uid}')} , M, \Sigma) \rightarrow 0/1.$ The denial judgement algorithm checks the validity of $\Pi_{D(\text{uid}')}$ outputted by the DTrace algorithm.

An FDGSwD scheme must satisfy *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*. To formalize these requirements, we first specify an additional oracle, called DTrace.

$\text{DTrace}(\text{uid}', M, \Sigma, \text{info}_\tau).$ This oracle returns a denial opening proof $\Pi_{D(\text{uid}')}.$ It is required that Σ is not generated by the challenged oracle or the user uid' is not in the challenged user set to eliminate the cases where the adversary trivially breaks anonymity.

The correctness and security requirements of an FDGSwD scheme are extensions of those in Bootle et al.'s model (See section 3.2.2). We highlight the difference below. *Correctness* additionally requires that a (denial) proof generated by algorithm **DTrace**, with respect to a message-signature pair (M, Σ) and an arbitrary user uid' who is not the real signer, is always accepted by algorithm **DJudge**. A formal definition is given below.

Definition 3.5.1 (Correctness). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{corr}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{corr}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL} := \emptyset.$

$\langle (\text{info}, \text{mpk}, \text{msk}); (\text{tpk}, \text{tsk}) \rangle \leftarrow \langle \text{GKgen}_{\text{GM}}(\text{pp}), \text{GKgen}_{\text{TM}}(\text{pp}) \rangle.$

Set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$. $(\text{uid}, M, \tau) \leftarrow \mathcal{A}^{\text{AddU, RReg, GUpdate}}(\text{gpk}, \text{info}).$

If $\text{uid} \notin \text{HUL}$ or $\text{gsk}[\text{uid}] = \perp$ or $\text{info}_\tau = \perp$ or $\text{IsActive}(\text{info}_\tau, \text{reg}, \text{uid}) = 0$, return 0.

$\Sigma \leftarrow \text{Sign}(\text{gpk}, \text{gsk}[\text{uid}], \text{info}_\tau, M)$, $(\text{uid}^*, \Pi_{\text{Trace}}) \leftarrow \text{Trace}(\text{gpk}, \text{tsk}, \text{info}_\tau, \text{reg}, M, \Sigma).$

$\text{uid}' \leftarrow \mathcal{A}^{\text{AddU, RReg, GUpdate}}(\Sigma, \text{uid}^*, \Pi_{\text{Trace}}).$

If $\text{uid}' = \text{uid} \vee \text{uid}' \notin \text{HUL}$, then return 0.

$\Pi_{D(\text{uid}')} \leftarrow \text{DTrace}(\text{gpk}, \text{tsk}, \text{info}_\tau, \text{reg}, \text{uid}', M, \Sigma).$

Return 1 if $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 0$ or $\text{uid} \neq \text{uid}^*$ or

$\text{Judge}(\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{\text{Trace}}, M, \Sigma) = 0$ or $\text{DJudge}(\text{gpk}, \text{uid}', \text{info}_\tau, \Pi_{D(\text{uid}')} M, \Sigma) = 0.$

Else, return 0.

Define $\mathbf{Adv}_{\text{FDGSwD}, \mathcal{A}}^{\text{corr}}(\lambda)$ of adversary $\mathcal{A}$ against correctness of an FDGSwD scheme as $\Pr[\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{corr}}(\lambda) = 1]$. An FDGSwD scheme is correct if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Anonymity is defined with respect to an adversary who also has access to the additional oracle **DTrace** specified above. Details of the formal definition are given as follows.

Definition 3.5.2 (Anonymity). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiments $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{anon-b}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{anon-b}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL}, \text{CUL}, \text{BUL}, \text{CL}, \text{SL} := \emptyset.$

$(\text{st}_{\text{init}}, \text{info}, \text{mpk}, \text{msk}) \leftarrow \mathcal{A}^{(\cdot, \text{GKgen}_{\text{TM}})}(\text{pp})(\text{init} : \text{pp}).$

If GKgen_{TM} did not accept or $\mathcal{A}$'s output is not well-formed, return $\perp$.

Denote the output of GKgen_{TM} as (tpk, tsk) , and set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$b^* \leftarrow \mathcal{A}^{\text{AddU}, \text{CrptU}, \text{RevealU}, \text{SndToU}, \text{Trace}, \text{MReg}, \text{DTrace}, \text{Chal}_b}(\text{play} : \text{st}_{\text{init}}, \text{gpk}).$

Return b^* .

Define $\mathbf{Adv}_{\text{FDGSwD}, \mathcal{A}}^{\text{anon}}(\lambda)$ of adversary $\mathcal{A}$ against anonymity of an FDGSwD scheme as $|\Pr[\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{anon-1}}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{anon-0}}(\lambda) = 1]|$. An FDGSwD scheme is anonymous if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Non-frameability additionally requires that it is infeasible for the adversary to generate a valid signature whose denial opening proof with respect to an honest user is not accepted by the DJudge algorithm. Details are given below.

Definition 3.5.3 (Non-frameability). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{non-frame}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{non-frame}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL}, \text{CUL}, \text{BUL}, \text{SL} = \emptyset.$

$(\text{st}_{\text{init}}, \text{info}, \text{mpk}, \text{msk}, \text{tpk}, \text{tsk}) \leftarrow \mathcal{A}(\text{init} : \text{pp}).$

Return 0 if $\mathcal{A}$'s output is not well-formed, otherwise set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$(M, \Sigma, \text{uid}, \Pi_{\text{Trace}}, \text{info}_\tau) \leftarrow \mathcal{A}^{\text{CrptU}, \text{RevealU}, \text{SndToU}, \text{MReg}, \text{Sign}}(\text{play} : \text{st}_{\text{init}}, \text{gpk}).$

$\Pi_{D(\text{uid})} \leftarrow \text{DTrace}(\text{gpk}, \text{tsk}, \text{info}_\tau, \text{reg}, \text{uid}, M, \Sigma).$

If $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 0$ or $\text{uid} \notin \text{HUL} \setminus \text{BUL}$ or $(\text{uid}, M, \Sigma, \tau) \in \text{SL}$, return 0.

Return 1 if $\text{Judge}(\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{\text{Trace}}, M, \Sigma) = 1$ or

$\text{DJudge}(\text{gpk}, \text{uid}, \text{info}_\tau, \Pi_{D(\text{uid})}, M, \Sigma) = 0.$

Else return 0.

Define $\mathbf{Adv}_{\text{FDGSwD}, \mathcal{A}}^{\text{non-frame}}(\lambda)$ of adversary $\mathcal{A}$ against non-frameability of an FDGSwD scheme as $\Pr[\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{non-frame}}(\lambda) = 1]$. An FDGSwD scheme is non-frameable if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Traceability additionally demands that the adversary is not able to produce a valid signature such that opening proof is accepted by the **Judge** algorithm, but the denial opening proof with respect to the same user is also accepted by the **DJudge** algorithm. The formal definition is given in the following.

Definition 3.5.4 (Traceability). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda), \text{HUL}, \text{CUL}, \text{BUL}, \text{SL} = \emptyset$.

$(\text{st}_{\text{init}}, \text{tpk}, \text{tsk}) \leftarrow \mathcal{A}^{\langle \text{GKgen}_{\text{GM}}(\text{pp}), \cdot \rangle}(\text{init} : \text{pp})$.

If GKgen_{GM} did not accept or $\mathcal{A}$'s output is not well-formed, return 0

Denote the output of GKgen_{GM} as $(\text{mpk}, \text{msk}, \text{info})$, and set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$(M, \Sigma, \tau) \leftarrow \mathcal{A}^{\text{AddU, CrptU, SndToGM, RevealU, MReg, Sign, GUpdate}}(\text{play} : \text{st}_{\text{init}}, \text{gpk}, \text{info})$.

If $\text{Verify}(\text{gpk}, \text{info}_{\tau}, M, \Sigma) = 0$, return 0.

$(\text{uid}, \Pi_{\text{Trace}}) \leftarrow \text{Trace}(\text{gpk}, \text{tsk}, \text{info}_{\tau}, \text{reg}, M, \Sigma)$.

$\Pi_{D(\text{uid})} \leftarrow \mathcal{A}^{\text{AddU, CrptU, SndToGM, RevealU, MReg, Sign, GUpdate}}(\text{play} : \text{st}_{\text{init}}, \text{uid}, \Pi_{\text{Trace}})$.

Return 1 if $\text{IsActive}(\text{info}_{\tau}, \text{reg}, \text{uid}) = 0$ or $\text{uid} = \perp$ or

$\text{Judge}(\text{gpk}, \text{uid}, \text{info}_{\tau}, \Pi_{\text{Trace}}, M, \Sigma) = 0$ or $\text{DJudge}(\text{gpk}, \text{uid}, \text{info}_{\tau}, \Pi_{D(\text{uid})}, M, \Sigma) = 1$.

Else, return 0.

Define $\mathbf{Adv}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace}}(\lambda)$ of adversary $\mathcal{A}$ against traceability of an FDGSwD scheme as $\Pr[\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace}}(\lambda) = 1]$. An FDGSwD scheme is traceable if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Tracing Soundness also requires that the adversary should not be able to generate a valid signature such that the denial opening proof for the actual signer is accepted by the **DJudge** algorithm. We give the formal definition below.

Definition 3.5.5 (Tracing Soundness). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace-sound}}(\lambda)$ as follows.

Experiment: $\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace-sound}}(\lambda)$

$\text{pp} \leftarrow \text{GSetup}(\lambda)$, $\text{CUL} = \emptyset$. $(\text{st}_{\text{init}}, \text{info}, \text{mpk}, \text{msk}, \text{tpk}, \text{tsk}) \leftarrow \mathcal{A}(\text{init} : \text{pp})$.

Return 0 if $\mathcal{A}$'s output is not well-formed, otherwise set $\text{gpk} = (\text{pp}, \text{mpk}, \text{tpk})$.

$(M, \Sigma, \text{uid}_0, \Pi_{\text{Trace}, 0}, \Pi_{D(\text{uid}_0)}, \text{uid}_1, \Pi_{\text{Trace}, 1}, \text{info}_\tau) \leftarrow \mathcal{A}^{\text{CrptU}, \text{MReg}}(\text{play} : \text{st}_{\text{init}}, \text{gpk})$.

If $\text{Verify}(\text{gpk}, \text{info}_\tau, M, \Sigma) = 0$ or $\text{Judge}(\text{gpk}, \text{uid}_0, \text{info}_\tau, \Pi_{\text{Trace}, 0}, M, \Sigma) = 0$,
or $\text{uid}_0 = \perp$ or $\text{uid}_1 = \perp$, return 0.

Return 1 if $\text{uid}_0 \neq \text{uid}_1 \wedge \text{Judge}(\text{gpk}, \text{uid}_1, \text{info}_\tau, \Pi_{\text{Trace}, 1}, M, \Sigma) = 1$ or

$\text{DJudge}(\text{gpk}, \text{uid}_0, \text{info}_\tau, \Pi_{D(\text{uid}_0)}, M, \Sigma) = 1$.

Else, return 0.

Define $\mathbf{Adv}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace-sound}}(\lambda)$ of adversary $\mathcal{A}$ against tracing soundness of an FDGSwD scheme as $\Pr[\mathbf{Exp}_{\text{FDGSwD}, \mathcal{A}}^{\text{trace}}(\lambda) = 1]$. An FDGSwD scheme is tracing sound if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

3.5.2 Our Construction

As shown by Ishida et al. [91], the main difficulty towards realizing the deniability functionality is to prove in zero-knowledge that a given ciphertext does not decrypt to a particular message. Such a mechanism is non-trivial to realize in general, but it can be done quite easily for our scheme by using the Stern-like technique for proving inequality. Let us look at the equation

$$\mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} = \mathbf{y} + \lfloor q/2 \rfloor \cdot \mathbf{b}' \bmod q,$$

that appears in the argument of correct decryption from Section 3.4.4. To realize deniability, the plaintext $\mathbf{b}' \in \{0, 1\}^\ell$ should be hidden, and we have to prove that $\mathbf{b}' \neq \text{uid}'$, for some given $\text{uid}' \in \{0, 1\}^\ell$. Subtracting both sides of the equation by

$\lfloor q/2 \rfloor \cdot \text{uid}'$, we get the following:

$$\mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} - \lfloor q/2 \rfloor \cdot \text{uid}' = \mathbf{y} + \lfloor q/2 \rfloor \cdot \mathbf{b} \bmod q,$$

where $\mathbf{b} = \mathbf{b}' - \text{uid}' \in \{-1, 0, 1\}^\ell$ is non-zero. Therefore, the problem towards realizing the deniability is reduced to proving knowledge of a non-zero vector $\mathbf{b} \in \{-1, 0, 1\}^\ell$. This can be done by just appending $2\ell - 1$ “dummy” entries to $\mathbf{b}$ and randomly permuting the extended vector. We defer the detailed description to Section 3.5.3.

Now, let us describe and analyze our fully dynamic group signature scheme with deniability. Besides all the algorithms specified in Section 3.4.1, the scheme contains two additional algorithms, **DTrace** and **DJudge**, as follows.

DTrace(gpk, tsk, info_τ, reg, uid', M, Σ). Let $\mathbf{b}'$ be the decryption result of the **Trace** algorithm, denote $\mathbf{b} = \mathbf{b}' - \text{uid}'$. This algorithm generates a **NIZKAoK** to demonstrate the possession of $\mathbf{S}_1 \in \mathbb{Z}^{n \times \ell}$, $\mathbf{E}_1 \in \mathbb{Z}^{\ell \times m_E}$, $\mathbf{y} \in \mathbb{Z}^\ell$, and non-zero vector $\mathbf{b} \in \{-1, 0, 1\}^\ell$ such that

$$\begin{cases} \|\mathbf{S}_1\|_\infty \leq \beta; \|\mathbf{E}_1\|_\infty \leq \beta; \|\mathbf{y}\|_\infty \leq \lceil q/5 \rceil; \\ \mathbf{S}_1^\top \cdot \mathbf{B} + \mathbf{E}_1 = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} - \lfloor q/2 \rfloor \cdot \text{uid}' = \mathbf{y} + \lfloor q/2 \rfloor \cdot \mathbf{b} \bmod q. \end{cases} \quad (3.13)$$

The statistical **ZKAoK** protocol is described in Section 3.5.3 in detail. Since the protocol has soundness error $2/3$, we thus run it $\kappa = \omega(\log \lambda)$ times. This results in soundness error $(2/3)^\kappa$, which is negligible. In addition, the protocol is an interactive version, we thus employ the Fiat-Shamir heuristic [75], achieving a non-interactive one. Finally, the generated proof is a triple $\Pi_{D(\text{uid}')} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}\}_{i=1}^\kappa)$, with

$$\text{CH} = \mathcal{H}_{\text{FS}}(\{\text{CMT}_i\}_{i=1}^\kappa, \text{gpk}, \text{info}_\tau, M, \Sigma, \text{uid}') \in \{1, 2, 3\}^\kappa. \quad (3.14)$$

DJudge($\text{gpk}, \text{uid}', \text{info}_\tau, \Pi_{D(\text{uid}')} , M, \Sigma$). This algorithm outputs $\perp$ if the verification procedure (described in Section 3.4.1) fails. Otherwise, it checks the validity of the proof $\Pi_{D(\text{uid}')}$ with respect to the public input $(\text{gpk}, \text{info}_\tau, M, \Sigma, \text{uid}')$, as in the verification process.

Return 1 if the proof $\Pi_{D(\text{uid}')}$ is valid and 0 otherwise.

EFFICIENCY. In comparison with the **FDGS** scheme analyzed in Section 3.4.2, the extended scheme with deniability additionally features the **NIZKAoK** $\Pi_{D(\text{uid}')}$, which is of bit-size $\tilde{O}(\ell^2 + \lambda \cdot \ell)$. (Recall that λ is the security parameter while $\ell = \log N$.)

CORRECTNESS. The correctness of the scheme is analyzed as in Section 3.4.2, but here we additionally need to prove that algorithm **DJudge** will accept the proof $\Pi_{D(\text{uid}')}$ for uid' that is not the real signer. Since the tracing manager can correctly identify the signer, he possesses a valid tuple $(\mathbf{S}_1, \mathbf{E}_1, \mathbf{y}, \mathbf{b})$ such that the conditions in (3.13) hold. Correctness then follows from the perfect completeness of our argument system generating $\Pi_{D(\text{uid}')}$.

ANONYMITY. It is analyzed as in Lemma 3.4.2, except that the challenger additionally needs to answer the **DTrace** oracle queries. However, this can be done in the same way it answers the **Trace** oracle queries. In other words, when the challenger replies with real proof Π_{trace} , it also replies with real proof $\Pi_{D(\cdot)}$; when the challenger replies with simulated proof Π_{trace} , it also replies with simulated proof $\Pi_{D(\cdot)}$. This minor change does not affect anonymity, since the argument system generating $\Pi_{D(\cdot)}$ is statistically zero-knowledge.

NON-FRAMEABILITY. This is proved by contradiction. Let $\mathcal{A}$ be a PPT algorithm that breaks the non-frameability game with non-negligible probability ϵ . Then we are able to construct another PPT algorithm $\mathcal{B}$ that violates the security of our accumulator or solves an instance of the $\text{SIS}_{n,q,m,1}^\infty$ problem featured by a matrix $\mathbf{A}$, with some non-negligible probability related to ϵ .

The algorithm $\mathcal{B}$ simulates the view of the algorithm $\mathcal{A}$ as follows. It first produces

all public parameters $\mathbf{pp}$ as in the $\mathbf{GSetup}$ algorithm except that now the matrix $\mathbf{A}$ is from its environment instead of generating it uniformly. It then invokes $\mathcal{A}$ by sending $\mathcal{A}$ the public parameter $\mathbf{pp}$. Now $\mathcal{B}$ interacts with $\mathcal{A}$ as described in the game $\mathbf{Exp}_{\mathbf{FDGSwD}, \mathcal{A}}^{\text{non-frame}}(\lambda)$. Note that $\mathcal{B}$ is able to reply all the queries from $\mathcal{A}$ since it knows all the secrets in the game. Finally, consider the case where $\mathcal{A}$ outputs the tuple $(M^*, \Sigma^*, \mathbf{bin}(j^*), \Pi_{\text{trace}}^*, \mathbf{info}_\tau)$ and wins the game. The algorithm $\mathcal{B}$ then proceeds as in Lemma 3.4.3 until $\mathcal{B}$ extracts the witness $\zeta' = (\dots, \mathbf{bin}(j'), \dots)$. Since our encryption scheme is correct, $\mathbf{c}_1^*$ will be decrypted to $\mathbf{bin}(j')$, meaning $\mathbf{bin}(j')$ is the originator of the signature Σ^* . The fact that $\mathcal{A}$ wins implies either case (i) $\mathbf{Judge}(\mathbf{gpk}, \mathbf{bin}(j^*), \mathbf{info}_\tau, \Pi_{\text{trace}}^*, M^*, \Sigma^*) = 1$ or case (ii)

$$\mathbf{DJudge}(\mathbf{gpk}, \mathbf{bin}(j^*), \mathbf{info}_\tau, \Pi_{D(\mathbf{bin}(j^*))}, M^*, \Sigma^*) = 0.$$

Recall that $\mathbf{bin}(j^*)$ is the framed user outputted by the adversary. We claim that the equality $\mathbf{bin}(j') = \mathbf{bin}(j^*)$ holds in both case (i) and case (ii). Lemma 3.4.3 proves that the equality holds in case (i). Now we prove it also holds in case (ii). Otherwise, $\mathbf{bin}(j') \neq \mathbf{bin}(j^*)$, i.e., $\Pi_{D(\mathbf{bin}(j^*))}$ is a denial proof for user $\mathbf{bin}(j^*)$, who is not the actual signer $\mathbf{bin}(j')$ of Σ^* . Since the underlying argument protocol has perfect completeness, the $\mathbf{DJudge}$ algorithm will accept the validity of the proof $\Pi_{D(\mathbf{bin}(j^*))}$, a contradiction of case (ii). Hence, the equality holds in case (ii). Now $\mathcal{B}$ proceeds again as described in Lemma 3.4.3. In the end, the algorithm $\mathcal{B}$ either breaks the security of our accumulator or solves a $\mathbf{SIS}_{n,m,q,1}^\infty$ instance featured by matrix A . Therefore, our $\mathbf{FDGSwD}$ scheme satisfies non-frameability.

TRACEABILITY. Except for the analysis of Lemma 3.4.4, we need to consider an additional case (iii) in which $\mathcal{A}$ wins, and prove that case (iii) occurs with negligible probability. Case (iii) is the event that $\mathcal{A}$ outputs the tuple $(\mathbf{info}_\tau, M, \Sigma)$ with $\Sigma = (\Pi_{\mathbf{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ being a valid signature; Let the (honestly generated) opening result be $(\mathbf{bin}(j), \Pi_{\text{trace}})$, where $\mathbf{bin}(j)$ is an active user and Π_{trace} is accepted by the $\mathbf{Judge}$

algorithm; and then $\mathcal{A}$ outputs a denial proof $\Pi_{D(\text{bin}(j))}$, which is accepted by the **DJudge** algorithm, after seeing the opening result.

To prove this case happens with negligible probability, we proceed as in Lemma 3.4.4 and obtain that $\text{bin}(j') = \text{bin}(j)$ with overwhelming probability, where $\text{bin}(j')$ is part of the extracted witness, meaning $\mathbf{c}_1, \mathbf{c}_2$ are encryptions of $\text{bin}(j')$. The assumption that $\Pi_{D(\text{bin}(j))}$ will be accepted by the **DJudge** algorithm, however, implies that with overwhelming probability $\text{bin}(j') - \text{bin}(j) \neq \mathbf{0}$ according to the soundness of the underlying argument system generating $\Pi_{D(\cdot)}$. This results in a contradiction. Therefore, case (iii) only happens with negligible probability, which implies that our **FDGSwD** scheme is traceable.

TRACING SOUNDNESS. Except for the analysis of Lemma 3.4.5, we need to additionally prove that it is computationally hard for the adversary to generate a valid signature together with a denial proof (for the actual signer) that is accepted by the **DJudge** algorithm. In other words, we need to prove the event, when $\mathcal{A}$ halts and outputs the tuple $(M, \Sigma, \text{uid}_0, \Pi_{\text{trace},0}, \Pi_{D(\text{uid}_0)}, \text{uid}_1, \Pi_{\text{trace},1}, \text{info}_\tau)$, both $\Pi_{\text{trace},0}$ and $\Pi_{D(\text{uid}_0)}$ are accepted, happens with negligible probability. Parse Σ as $(\Pi_{\text{gs}}, (\mathbf{c}_{1,1}, \mathbf{c}_{1,2}), \mathbf{c}_2)$. Since $\Pi_{\text{trace},0}$ is a valid opening proof, as in Lemma 3.4.5, we can extract witness $\mathbf{S}_1, \mathbf{E}_1, \mathbf{y}$ such that $\mathbf{S}_1^\top \cdot \mathbf{B} + \mathbf{E}_1 = \mathbf{P}_1 \bmod q$ and

$$\mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} = \mathbf{y} + \lfloor q/2 \rfloor \cdot \text{uid}_0 \bmod q. \quad (3.15)$$

The fact that $\Pi_{D(\text{uid}_0)}$ is accepted by the **DJudge** algorithm implies decryption of $(\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ is not equal to uid_0 , due to the soundness of the underlying argument system. However, from equation (3.15), we know decryption of $(\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ is uid_0 , by correctness of our encryption scheme. This results in a contradiction. Therefore, the considered event occurs only with negligible probability, and the **FDGSwD** scheme is tracing sound.

3.5.3 Zero-Knowledge Argument of “Denial Opening”

We now present the underlying ZKAoK used by the tracing manager to generate $\Pi_{D(\cdot)}$. The protocol enables the prover to show that plaintext underling a given ciphertext is not equal to a particular message. The only difference of this protocol with the one described in Section 3.4.4 is that we need to additionally prove knowledge of a *non-zero* vector $\mathbf{b} \in \{-1, 0, 1\}^\ell$. Toward this goal, we employ the following extension and permutation techniques for Stern-like protocols, which is developed from Ling et al.’s work [116].

For any integer $\mathbf{m} \in \mathbb{Z}_+$, denote $\tilde{\mathbf{R}}_{\mathbf{m}}^3$ as the set that contains all trinary vectors of dimension $3\mathbf{m} - 1$ such that $\mathbf{m} - 1$ coordinates equal to 0, $\mathbf{m}$ coordinates equal to -1 , and $\mathbf{m}$ coordinates equal to 1. To prove that $\mathbf{b} \in \{-1, 0, 1\}^\ell$ with $\mathbf{b} \neq \mathbf{0}^\ell$, we attach $2\ell - 1$ suitable entries to $\mathbf{b}$ to obtain a new vector $\mathbf{b}^* \in \tilde{\mathbf{R}}_\ell^3$. Observe that $\mathbf{b} \in \{-1, 0, 1\}^\ell$ is not a zero vector if and only if the extended vector $\mathbf{b}^* \in \tilde{\mathbf{R}}_\ell^3$. Furthermore, for any $\pi_b \in \mathcal{S}_{3\ell-1}$, the following equivalence holds:

$$\mathbf{b}^* \in \tilde{\mathbf{R}}_\ell^3 \Leftrightarrow \pi_b(\mathbf{b}^*) \in \tilde{\mathbf{R}}_\ell^3.$$

Now, by using the same technique described in Section 3.4.4, we can unify equations in (3.13) into

$$\tilde{\mathbf{M}} \cdot \tilde{\mathbf{z}} = \tilde{\mathbf{u}} \bmod q,$$

where $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{u}}$ are formed only from public information while $\tilde{\mathbf{z}} = (\mathbf{z} \parallel \mathbf{b}^*) \in \{-1, 0, 1\}^{\tilde{L}}$, where $\mathbf{z} \in \mathbf{R}_{L'}^3$, as defined in Section 3.4.4, $\mathbf{b}^* \in \tilde{\mathbf{R}}_\ell^3$, and $\tilde{L} = L + 3\ell - 1$.

Define $\widetilde{\text{VALID}} = \mathbf{R}_{L'}^3 \times \tilde{\mathbf{R}}_\ell^3$ and $\tilde{\mathcal{S}} = \mathcal{S}_L \times \mathcal{S}_{3\ell-1}$. For every $\tilde{\eta} = (\eta, \pi_b) \in \tilde{\mathcal{S}}$, define $\Gamma_{\tilde{\eta}} = \tilde{\eta}$. It is easy to see that our secret vector $\tilde{\mathbf{z}}$ is included in this tailed set $\widetilde{\text{VALID}}$ and the following equivalence holds:

$$\tilde{\mathbf{z}} \in \widetilde{\text{VALID}} \iff \Gamma_{\tilde{\eta}}(\tilde{\mathbf{z}}) \in \widetilde{\text{VALID}},$$

for any $\tilde{\eta} \in \tilde{\mathcal{S}}$. In addition, the randomness of the permutation $\Gamma_{\tilde{\eta}}$ perfectly hides our secret vector $\tilde{\mathbf{z}}$. In other words, the conditions in (2.1) are satisfied and we have reduced the considered statement to an instance of the abstract relation $\varrho_{\text{abstract}}$ described in Section 2.4.2. At this point, we can see that the desired statistical ZKAoK protocol can be obtained from the one in Fig. 2.1.

We remark that the above technique for obtaining zero-knowledge argument of “denial opening” is also applicable for several other existing lattice-based group signatures [105, 108, 117] and our constant-size group signature scheme described in Section 4.4, which employ similar tracing mechanisms.

Chapter 4

Constant-size Group Signatures from Lattices

In the previous chapter, we show how to achieve full dynamicity and deniability for lattice-based group signatures in a simple manner. In this chapter, we will present the first constant-size group signature (CGS) scheme from lattices, which means that the size of signatures produced by the scheme is independent of N and only relies on the security parameter λ . The scheme supports dynamic enrollment of users and is proven secure in the random oracle model under the RSIS and RLWE assumptions. At the heart of our design is a zero-knowledge argument of knowledge for the possession of a valid message-signature pair for the Ducas-Micciancio signature scheme [72, 73], that may be of independent interest.

Organization. We first give an overview of our techniques in Section 4.1. Then we recall the (partially) dynamic model suggested by Bellare et al. [21] in Section 4.2. Our zero-knowledge argument of knowledge for the Ducas-Micciancio signature scheme is described in Section 4.3 and our main scheme is constructed and analyzed in Section 4.4.

4.1 Overview of Our Techniques

Our scheme relies on the RSIS-based signature scheme by Ducas and Micciancio [72], which exploits the “confined guessing” technique [26] in the ring setting to achieve short public key. We employ the stateful and adaptively secure version of the scheme, described in [73], which suffices for building group signatures and which allows to work with even shorter key.

The scheme follows the usual sign-then-encrypt-then-prove approach for constructing group signatures. Each user generates a secret-public key pair $(\mathbf{x}, p)$ and becomes a certified group member once receiving a Ducas-Micciancio signature on his public key p . When generating group signatures, the user first encrypts his public key p to ciphertext $\mathbf{c}$ via a CCA-secure encryption scheme obtained by applying the Naor-Yung transformation [146] to a variant of the RLWE-based scheme by Lyubashevsky, Peikert and Regev [131]. Then he proves in zero-knowledge that: (i) he has a valid secret key $\mathbf{x}$ corresponding to p ; (ii) he possesses a Ducas-Micciancio signature on p ; and (iii) $\mathbf{c}$ is a correct ciphertext of p . The protocol is then transformed into a signature via the Fiat-Shamir heuristic [75].

To instantiate the above approach, we design a zero-knowledge argument of knowledge of a valid message-signature pair for the Ducas-Micciancio signature, which is based on Stern’s framework [173]. We observe that a similar protocol for the Boyen signature [34] was proposed by Ling et al. [117], but their method is sub-optimal in terms of efficiency. We thus propose a refined technique that allows to achieve better communication cost, and hence, shorter signature size. We believe that our protocol is of independent interest. Indeed, apart from group signatures, zero-knowledge protocols for valid message-signature pairs are essential ingredients for designing various privacy-enhancing constructions, such as anonymous credentials [54], compact e-cash [53, 109], policy-based signatures [18, 65], and much more.

On the practical front, as all known lattice-based group signatures, our scheme

is not truly practical. Even though the scheme produces signatures of constant size $\tilde{O}(\lambda)$, due to a large poly-logarithmic factor contained in the $\tilde{O}$ notation, the signature size is too big to be really useful in practice. We, however, hope that our result will inspire more efficient constructions in the near future.

4.2 Dynamic Group Signatures

In this section, we recall the definition and security notions of the (partially) dynamic group signatures, as put forward by Bellare, Shi and Zhang [21]. A dynamic group signature scheme involves a trusted party who generates the initial keys, an authority named issuer who can enroll new users, an authority named opener who can open signatures, and a set of potential group members.

4.2.1 Syntax of Dynamic Group Signatures

The scheme consists of the following polynomial-time algorithms.

GKg(λ): Given the security parameter λ , the trusted party runs this algorithm to generate a triple $(\mathbf{gpk}, \mathbf{ik}, \mathbf{ok})$. The issue key $\mathbf{ik}$ is given to the issuer, the opening key $\mathbf{ok}$ is given to the opener and the group public key $\mathbf{gpk}$ is made public.

UKg(λ): A user who intends to be a group member runs this algorithm to obtain a personal key pair $(\mathbf{upk}, \mathbf{usk})$. It is assumed that $\mathbf{upk}$ is public.

$\langle \text{Join}, \text{lss} \rangle$: This is an interactive protocol run by the issuer and a user. If it completes successfully, the issuer registers this user to the group and this user becomes a group member. The final state of the **Join** is the secret signing key $\mathbf{gsk}_i$ while the final state of the **lss** is the registration information $\mathbf{reg}[i]$ stored in the registration table **reg**.

Sign($\mathbf{gpk}, \mathbf{gsk}_i, M$): A group member, using his group signing key $\mathbf{gsk}_i$, runs this algorithm to obtain a signature Σ on message M .

Verify(gpk, M , Σ): This algorithm outputs 1/0 indicating whether or not Σ is a valid signature on message M , with respect to the group public key **gpk**.

Open(gpk, ok, reg, M , Σ): Given **gpk**, a message-signature pair (M, Σ) and **ok**, the opener, who has read-access to the registration table **reg**, runs this algorithm to obtain a pair (i, Π_{open}) , where $i \in \mathbb{N} \cup \{\perp\}$. In case $i = \perp$, $\Pi_{\text{open}} = \perp$.

Judge(gpk, M , Σ , i , **upk _{i} , Π_{open}):** This algorithm outputs 1/0 to check whether or not Π_{open} is a proof that i produced Σ , with respect to the group public key **gpk** and message M .

4.2.2 Notions of Correctness and Security Definitions

Now we recall the correctness and security definitions of dynamic group signatures below. Since these requirements will be modelled via experiments, where the adversary has access to some oracles. We therefore recall the oracles as specified by [21].

AddU(i): This *add user* oracle adds an honest user to the group and returns **upk**.

CruptU(i , **upk):** This *corrupt user* oracle allows the adversary to corrupt this user i and set this user public key to be **upk** chosen by the adversary.

SndTol(i , M_{in}): After corrupting user i , the adversary runs this *send to issuer* oracle to interact with the honest and **lss**-executing issuer while the adversary plays the role of user i . If it completes successful, the oracle returns M_{out} to the adversary and stores **reg**[i] to the registration table as the final state of the **lss** algorithm.

SndToU(i , M_{in}): This *send to user* oracle is run, by the adversary who corrupts the issuer, to interact with an honest and **Join**-executing user i while the adversary plays the role of the issuer. If it completes successful, the oracle returns M_{out} to the adversary and stores **gsk** _{i} as the final state of the **Join** algorithm.

RevealU(i): This *reveal user* oracle reveals all the secret keys $(\text{usk}_i, \text{gsk}_i)$ of user i to the adversary.

RReg(i): This *read registration table* oracle allows the adversary to read $\mathbf{reg}[i]$.

WReg(i, ρ): This *write registration table* oracle allows the adversary to change $\mathbf{reg}[i]$ to arbitrarily string ρ chosen by the adversary.

Sign(i, M): This *signing* oracle allows the adversary to query a signature of the message M under the group signing key gsk_i , as long as user i is an honest user and gsk_i is defined.

Ch_b(i_0, i_1, M): Given the inputs (i_0, i_1, M) from the adversary attacking anonymity, this *challenge* oracle returns a signature Σ of the message M under the group signing key gsk_{i_b} of user i_b to the adversary and stores (M, Σ) in a set G .

Open(M, Σ): This *opening* oracle allows the adversary to query the opening result of the pair (M, Σ) as long as (M, Σ) was not obtained from the *challenge* oracle.

Correctness requires that for any signature generated by honest group members, the following should hold: the signature should be valid; the opening algorithm, given the message and signature, should correctly identify the signer; the proof returned by the opening algorithm should be accepted by the judge.

Definition 4.2.1 (Correctness). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{DGS}, \mathcal{A}}^{\text{corr}}(\lambda)$ as follows.

$\mathbf{Exp}_{\text{GS}, \mathcal{A}}^{\text{corr}}(\lambda)$

$(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKg}(\lambda); \text{CU} \leftarrow \emptyset; \text{HU} \leftarrow \emptyset; (i, M) \leftarrow \mathcal{A}(\text{gpk} : \text{AddU}, \text{RReg}).$

If $i \notin \text{HU}$, then return 0; If $\text{gsk}_i = \epsilon$, then return 0.

$\Sigma \leftarrow \text{Sign}(\text{gpk}, \text{gsk}_i, M)$; If $\text{Verify}(\text{gpk}, M, \Sigma) = 0$, then return 1.

$(j, \Pi_{\text{open}}) \leftarrow \text{Open}(\text{gpk}, \text{ok}, \mathbf{reg}, M, \Sigma)$; If $j \neq i$, then return 1.

If $\text{Judge}(\text{gpk}, M, \Sigma, i, \text{upk}_i, \Pi_{\text{open}}) = 0$ then return 1.

Else return 0.

A dynamic group signature scheme is said to be correct if the advantage of the adversary, defined as $\mathbf{Adv}_{\text{DGS}, \mathcal{A}}^{\text{corr}}(\lambda) = \Pr[\mathbf{Exp}_{\text{DGS}, \mathcal{A}}^{\text{corr}}(\lambda) = 1]$, is negligible in λ .

Full Anonymity requires that it is infeasible to recover the identity of a signer from a signature, even if the adversary is given access to the opening oracle. As pointed out by [20, 21], it is sufficient that the adversary is unable to distinguish which of two signers of its choice signed a targeted message of its choice.

Definition 4.2.2 (Full Anonymity). For any security parameter λ and any PPT adversary $\mathcal{A}$, the experiments $\mathbf{Exp}_{\text{DGS}, \mathcal{A}}^{\text{anon}-b}(\lambda)$ are defined as follows.

$\mathbf{Exp}_{\text{GS}, \mathcal{A}}^{\text{anon}-b}(\lambda)$ // $b \in \{0, 1\}$

$(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKg}(\lambda); \text{CU} \leftarrow \emptyset; \text{HU} \leftarrow \emptyset; G \leftarrow \emptyset.$

$b' \leftarrow \mathcal{A}(\text{gpk}, \text{ik} : \text{Ch}_b, \text{Open}, \text{SndToU}, \text{WReg}, \text{RevealU}, \text{CruptU}).$

Return b' .

A dynamic group signature scheme is fully anonymous if the advantage of the adversary, defined as $\mathbf{Adv}_{\text{DGS}, \mathcal{A}}^{\text{anon}}(\lambda) = |\Pr[\mathbf{Exp}_{\text{DGS}, \mathcal{A}}^{\text{anon}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{DGS}, \mathcal{A}}^{\text{anon}-0}(\lambda) = 1]|$, is negligible in λ .

Traceability requires that every valid signature should be traced to some group member and the opener is able to generate a proof accepted by the judge.

Definition 4.2.3 (Traceability). For any security parameter λ and any PPT adversary $\mathcal{A}$, the experiment $\mathbf{Exp}_{\text{DGS}, \mathcal{A}}^{\text{trace}}(\lambda)$ is described as follows.

$\mathbf{Exp}_{\text{GS}, \mathcal{A}}^{\text{trace}}(\lambda)$

$(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKg}(\lambda); \text{CU} \leftarrow \emptyset; \text{HU} \leftarrow \emptyset.$

$(M, \Sigma) \leftarrow \mathcal{A}(\text{gpk}, \text{ok} : \text{SndToU}, \text{AddU}, \text{RReg}, \text{RevealU}, \text{CruptU}).$

If $\text{Verify}(\text{gpk}, M, \Sigma) = 0$, then return 0.

$(i, \Pi_{\text{open}}) \leftarrow \text{Open}(\text{gpk}, \text{ok}, \text{reg}, M, \Sigma).$

If $i = \perp$ or $\text{Judge}(\text{gpk}, M, \Sigma, i, \text{upk}_i, \Pi_{\text{open}}) = 0$, then return 1.

Else return 0.

A dynamic group signature scheme is said to be traceable if the advantage of the adversary against traceability, defined as $\mathbf{Adv}_{\text{DGS},\mathcal{A}}^{\text{trace}}(\lambda) = \Pr[\mathbf{Exp}_{\text{DGS},\mathcal{A}}^{\text{trace}}(\lambda) = 1]$, is negligible in λ .

Non-frameability requires that the adversary is unable to generate a proof, which is accepted by the judge, that an honest user generated a valid signature unless this user really did generate this signature.

Definition 4.2.4 (Non-frameability). For any security parameter λ and any PPT adversary $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{DGS},\mathcal{A}}^{\text{nf}}(\lambda)$ below.

$\mathbf{Exp}_{\text{DGS},\mathcal{A}}^{\text{nf}}(\lambda)$

$(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKg}(\lambda); \text{CU} \leftarrow \emptyset; \text{HU} \leftarrow \emptyset.$

$(M, \Sigma, i, \Pi_{\text{open}}) \leftarrow \mathcal{A}(\text{gpk}, \text{ok}, \text{ik} : \text{SndToU}, \text{WReg}, \text{Sign}, \text{RevealU}, \text{CruptU}).$

If $\text{Verify}(\text{gpk}, M, \Sigma) = 0$, then return 0.

If the following are all true then return 1, else return 0:

- $i \in \text{HU}$ and $\text{gsk}_i \neq \epsilon$ and $\text{Judge}(\text{gpk}, M, \Sigma, i, \text{upk}_i, \Pi_{\text{open}}) = 1$
- $\mathcal{A}$ did not query $\text{RevealU}(i)$ and $\text{Sign}(i, M)$

A dynamic group signature scheme is said to be non-frameable if the advantage of the adversary against non-frameability, defined as $\mathbf{Adv}_{\text{DGS},\mathcal{A}}^{\text{nf}}(\lambda) = \Pr[\mathbf{Exp}_{\text{DGS},\mathcal{A}}^{\text{nf}}(\lambda) = 1]$, is negligible in λ .

4.3 ZKAoK for the Ducas-Micciancio Signature

This section presents our statistical zero-knowledge argument of knowledge for a valid message-signature pair for the Ducas-Micciancio signature scheme [72, 73]. In the process, we will need to prove knowledge of a witness vector of the “mixing” form

$$(\mathbf{z} \parallel t_0 \cdot \mathbf{z} \parallel \dots \parallel t_{c_d-1} \cdot \mathbf{z}), \quad (4.1)$$

where $\mathbf{z} \in \{-1, 0, 1\}^{\mathbf{m}}$ and $\mathbf{t} = (t_0, \dots, t_{c_d-1})^\top \in \{0, 1\}^{c_d}$ for some positive integers $\mathbf{m}$ and c_d .

We note that, in their ZK protocol for the Boyen signature [34], Ling et al. [117] also derived a vector of similar form. To handle such a vector in the Stern's framework [173], Ling et al. used a permutation in the symmetric group $\mathcal{S}_{3\mathbf{m}}$ to hide the value of $\mathbf{z}$ and a permutation in the symmetric group $\mathcal{S}_{2c_d}$ to hide the value of $\mathbf{t}$. As a consequence, the cost of communicating the permutations from the prover to the verifier is $3\mathbf{m} \log \mathbf{m} + 2c_d \log c_d$ bits. This is sub-optimal, because the cost is much larger than the number of secret bits.

In Section 4.3.1, we put forward a refined permuting technique in which the total cost for the permutations is exactly the total bit-size of $\mathbf{z}$ and $\mathbf{t}$. We then employ this technique as a building block for our ZK protocol in Section 4.3.2.

4.3.1 A Refined Permuting Technique

We first observe that the coefficients of the vector described in (4.1) are highly correlated: most of them are products of t_i and z_j , where both t_i and z_j do appear at other positions. Thus, to prove the well-formedness of such a vector, we have to solve two sub-problems: (i) proving that a secret integer z is an element of the set $\{-1, 0, 1\}$; (ii) proving that a secret integer y is the product of secret integers $t \in \{0, 1\}$ and $z \in \{-1, 0, 1\}$. Furthermore, these sub-protocols must be compatible and extendable, so that we can additionally prove that the *same* t and z satisfy other relations.

Technique for proving that $z \in \{-1, 0, 1\}$. For any integer a , let us denote by $[a]_3$ the integer $a' \in \{-1, 0, 1\}$ such that $a' = a \bmod 3$. For integer $z \in \{-1, 0, 1\}$, we define the 3-dimensional vector $\text{enc}_3(z)$ as follows:

$$\text{enc}_3(z) = ([z + 1]_3, [z]_3, [z - 1]_3)^\top \in \{-1, 0, 1\}^3.$$

Namely, $\text{enc}_3(-1) = (0, -1, 1)^\top$, $\text{enc}_3(0) = (1, 0, -1)^\top$ and $\text{enc}_3(1) = (-1, 1, 0)^\top$.

Now, for any integer $e \in \{-1, 0, 1\}$, define the permutation π_e that transforms vector $\mathbf{v} = (v^{(-1)}, v^{(0)}, v^{(1)})^\top \in \mathbb{Z}^3$ into vector

$$\pi_e(\mathbf{v}) = (v^{([-e-1]_3)}, v^{([-e]_3)}, v^{([-e+1]_3)})^\top.$$

We then observe that, for any $z, b \in \{-1, 0, 1\}$, the following equivalence holds.

$$\mathbf{v} = \text{enc}_3(z) \iff \pi_e(\mathbf{v}) = \text{enc}_3([z + e]_3). \quad (4.2)$$

In the framework of Stern's protocol, the above technique in fact allows us to prove knowledge of $z \in \{-1, 0, 1\}$, where z may satisfy other relations. To do this, we first extend z to $\mathbf{v} = \text{enc}_3(z)$. Then, to show that $\mathbf{v}$ is a well-formed extension, we pick a uniformly random $e \xleftarrow{\$} \{-1, 0, 1\}$, and send $\pi_e(\mathbf{v})$ to the verifier. Thanks to the equivalence observed in (4.2), when seeing that $\pi_e(\mathbf{v}) = \text{enc}_3([z + e]_3)$, the verifier should be convinced that $\mathbf{v} = \text{enc}_3(z)$, which implies that $z \in \{-1, 0, 1\}$. Meanwhile, since e acts as a “one-time pad”, the value of z is completely hidden from the verifier. Furthermore, to prove that z satisfies other relations, we can use the same “one-time pad” e at other appearances of z . An example of that is to prove that z is involved in a product $t \cdot z$, which we now present.

Technique for proving that $y = t \cdot z$. For any $b \in \{0, 1\}$, we denote by $\bar{b}$ the bit $1 - b$. The addition operation modulo 2 is denoted by $\oplus$.

For any $t \in \{0, 1\}$ and $z \in \{-1, 0, 1\}$, we construct the 6-dimensional integer vector $\text{ext}(t, z) \in \{-1, 0, 1\}^6$ as follows:

$$\text{ext}(t, z) = \left(\bar{t} \cdot [z+1]_3, t \cdot [z+1]_3, \bar{t} \cdot [z]_3, t \cdot [z]_3, \bar{t} \cdot [z-1]_3, t \cdot [z-1]_3 \right)^\top.$$

Now, for any $b \in \{0, 1\}$ and $e \in \{-1, 0, 1\}$, we define the permutation $\psi_{b,e}(\cdot)$ that

transforms vector

$$\mathbf{v} = \left(v^{(0,-1)}, v^{(1,-1)}, v^{(0,0)}, v^{(1,0)}, v^{(0,1)}, v^{(1,1)} \right)^\top \in \mathbb{Z}^6$$

into vector

$$\psi_{b,e}(\mathbf{v}) = \left(v^{(b,[-e-1]_3)}, v^{(\bar{b},[-e-1]_3)}, v^{(b,[-e]_3)}, v^{(\bar{b},[-e]_3)}, v^{(b,[-e+1]_3)}, v^{(\bar{b},[-e+1]_3)} \right)^\top.$$

We then observe that the following equivalence holds for any $t, b \in \{0, 1\}$ and any $z, e \in \{-1, 0, 1\}$.

$$\mathbf{v} = \mathbf{ext}(t, z) \iff \psi_{b,e}(\mathbf{v}) = \mathbf{ext}(t \oplus b, [z + e]_3). \quad (4.3)$$

Example 4.3.1. Let $t = 1$ and $z = -1$. Then we have

$$\mathbf{v} = \mathbf{ext}(t, z) = (0, 0, 0, -1, 0, 1)^\top = (v^{(0,-1)}, v^{(1,-1)}, v^{(0,0)}, v^{(1,0)}, v^{(0,1)}, v^{(1,1)})^\top.$$

Suppose that $b = 0$ and $e = 1$, then

$$\psi_{b,e}(\mathbf{v}) = (v^{(0,1)}, v^{(1,1)}, v^{(0,-1)}, v^{(1,-1)}, v^{(0,0)}, v^{(1,0)})^\top = (0, 1, 0, 0, 0, -1)^\top,$$

which is equal to $\mathbf{ext}(1, 0) = \mathbf{ext}(1 \oplus 0, [-1 + 1]_3)$.

In the framework of Stern's protocol, the above technique will be used to prove that $y = t \cdot z$, as follows. We first extend y to $\mathbf{v} = \mathbf{ext}(t, z)$. Then, to prove that $\mathbf{v}$ is well-formed, we sample $b \xleftarrow{\$} \{0, 1\}$, $e \xleftarrow{\$} \{-1, 0, 1\}$, and demonstrate to the verifier that $\psi_{b,e}(\mathbf{v}) = \mathbf{ext}(t \oplus b, [z + e]_3)$. Thanks to the equivalence observed in (4.3), the verifier should be convinced that $\mathbf{v}$ is well-formed, implying that the original integer y does have the form $t \cdot z$. Meanwhile, the random integers b, e essentially act as “one-time pads” that perfectly hide the values of t and z , respectively. Moreover, if

we want to prove that the same t, z appear elsewhere, we can use the same b, e at those places.

Next, we will describe the somewhat straightforward generalizations of the above two core techniques, which enable us to prove knowledge of vector $\mathbf{z} \in \{-1, 0, 1\}^m$ as well as vector of the form (4.1). Based on the above discussions, one can see that the target is to obtain equivalences similar to (4.2) and (4.3), which are useful in Stern's framework.

Proving that $\mathbf{z} \in \{-1, 0, 1\}^m$. For any vector $\mathbf{a} \in \mathbb{Z}^m$, we will also use the notation $[\mathbf{a}]_3$ to denote the vector $\mathbf{a}' \in \{-1, 0, 1\}^m$ such that $\mathbf{a}' = \mathbf{a} \bmod 3$.

For $\mathbf{z} = (z_1, \dots, z_m)^\top \in \{-1, 0, 1\}^m$, we define the following extension:

$$\text{enc}(\mathbf{z}) = \left(\text{enc}_3(z_1) \parallel \dots \parallel \text{enc}_3(z_m) \right) \in \{-1, 0, 1\}^{3m}.$$

For any vector $\mathbf{e} = (e_1, \dots, e_m)^\top \in \{-1, 0, 1\}^m$, we define the permutation $\Pi_{\mathbf{e}}$ that acts as follows. When applied to vector $\mathbf{v} = (\mathbf{v}_1 \parallel \dots \parallel \mathbf{v}_m) \in \mathbb{Z}^{3m}$ consisting of m blocks of size 3, it transforms $\mathbf{v}$ into vector:

$$\Pi_{\mathbf{e}}(\mathbf{v}) = \left(\pi_{e_1}(\mathbf{v}_1) \parallel \dots \parallel \pi_{e_m}(\mathbf{v}_m) \right).$$

It then follows from (4.2) that the following holds for any $\mathbf{z}, \mathbf{e} \in \{-1, 0, 1\}^m$.

$$\mathbf{v} = \text{enc}(\mathbf{z}) \iff \Pi_{\mathbf{e}}(\mathbf{v}) = \text{enc}([\mathbf{z} + \mathbf{e}]_3). \quad (4.4)$$

Handling a “mixing” vector. We now tackle the “mixing” vector discussed earlier, i.e.,

$$\mathbf{y} = \left(\mathbf{z} \parallel t_0 \cdot \mathbf{z} \parallel \dots \parallel t_{c_d-1} \cdot \mathbf{z} \right).$$

For any $\mathbf{z} = (z_1, \dots, z_m)^\top \in \{-1, 0, 1\}^m$ and $\mathbf{t} = (t_0, \dots, t_{c_d-1})^\top \in \{0, 1\}^{c_d}$, we

define vector $\text{mix}(\mathbf{t}, \mathbf{z}) \in \{-1, 0, 1\}^{3\mathbf{m}+6\mathbf{m}c_d}$ of the form:

$$\left(\text{enc}(\mathbf{z}) \parallel \text{ext}(t_0, z_1) \parallel \dots \parallel \text{ext}(t_0, z_{\mathbf{m}}) \parallel \dots \parallel \text{ext}(t_{c_d-1}, z_1) \parallel \dots \parallel \text{ext}(t_{c_d-1}, z_{\mathbf{m}}) \right),$$

which is an extension of vector $\mathbf{y}$. Next, for $\mathbf{b} = (b_0, \dots, b_{c_d-1})^\top \in \{0, 1\}^{c_d}$ and $\mathbf{e} = (e_1, \dots, e_{\mathbf{m}})^\top \in \{-1, 0, 1\}^{\mathbf{m}}$, we define the permutation $\Psi_{\mathbf{b}, \mathbf{e}}$ that acts as follows.

When applied to vector

$$\mathbf{v} = (\mathbf{v}_{-1} \parallel \mathbf{v}_{0,1} \parallel \dots \parallel \mathbf{v}_{0,\mathbf{m}} \parallel \dots \parallel \mathbf{v}_{c_d-1,1} \parallel \dots \parallel \mathbf{v}_{c_d-1,\mathbf{m}}) \in \mathbb{Z}^{3\mathbf{m}+6\mathbf{m}c_d},$$

where block $\mathbf{v}_{-1}$ has length $3\mathbf{m}$ and each block $\mathbf{v}_{i,j}$ has length 6, it transforms $\mathbf{v}$ into vector

$$\begin{aligned} \Psi_{\mathbf{b}, \mathbf{e}}(\mathbf{v}) = & \left(\Pi_{\mathbf{e}}(\mathbf{v}_{-1}) \parallel \psi_{b_0, e_1}(\mathbf{v}_{0,1}) \parallel \dots \parallel \psi_{b_0, e_{\mathbf{m}}}(\mathbf{v}_{0,\mathbf{m}}) \parallel \dots \parallel \right. \\ & \left. \psi_{b_{c_d-1}, e_1}(\mathbf{v}_{c_d-1,1}) \parallel \dots \parallel \psi_{b_{c_d-1}, e_{\mathbf{m}}}(\mathbf{v}_{c_d-1,\mathbf{m}}) \right). \end{aligned}$$

Then, observe that the following desirable equivalence holds for all $\mathbf{t}, \mathbf{b} \in \{0, 1\}^{c_d}$ and $\mathbf{z}, \mathbf{e} \in \{-1, 0, 1\}^{\mathbf{m}}$.

$$\mathbf{v} = \text{mix}(\mathbf{t}, \mathbf{z}) \iff \Psi_{\mathbf{b}, \mathbf{e}}(\mathbf{v}) = \text{mix}(\mathbf{t} \oplus \mathbf{b}, [\mathbf{z} + \mathbf{e}]_3). \quad (4.5)$$

4.3.2 Zero-Knowledge Protocol for the Ducas-Micciancio Signature

We now present our statistical ZKAoK of a valid message-signature pair for the Ducas-Micciancio signature scheme. Let $n, q, m, k, \overline{m}, \ell, \beta, d, c_0, \dots, c_d$ as specified in Section 2.3.3. The protocol can be summarized as follows.

- The public input consists of

$$\mathbf{A}, \mathbf{F}_0 \in R_q^{1 \times \overline{m}}; \quad \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]} \in R_q^{1 \times k}; \quad \mathbf{F}, \mathbf{F}_1 \in R_q^{1 \times \ell}; \quad u \in R_q.$$

- The prover's secret input consists of message $p \in R_q$ and signature $(t, \mathbf{r}, \mathbf{v})$, where

$$\begin{cases} t = (t_0, \dots, t_{c_1-1}, \dots, t_{c_d-1}, \dots, t_{c_d-1})^\top \in \{0, 1\}^{cd}; \\ \mathbf{r} \in R^{\overline{m}}; \quad \mathbf{v} = (\mathbf{s} \parallel \mathbf{z}) \in R^{\overline{m}+k}; \quad \mathbf{s} \in R^{\overline{m}}; \quad \mathbf{z} \in R^k; \end{cases}$$

- The prover's goal is to prove in zero-knowledge that $\|\mathbf{r}\|_\infty \leq \beta$, $\|\mathbf{v}\|_\infty \leq \beta$, and that the equation

$$\mathbf{A} \cdot \mathbf{s} + \mathbf{A}_{[0]} \cdot \mathbf{z} + \sum_{i=1}^d \mathbf{A}_{[i]} \cdot t_{[i]} \cdot \mathbf{z} = \mathbf{F} \cdot \mathbf{y} + u \quad (4.6)$$

holds for $\{t_{[i]} = \sum_{j=c_{i-1}}^{c_i-1} t_j \cdot X^j\}_{i=1}^d$ and

$$\mathbf{y} = \text{rdec}(\mathbf{F}_0 \cdot \mathbf{r} + \mathbf{F}_1 \cdot \text{rdec}(p)) \in R^\ell. \quad (4.7)$$

Our strategy is to reduce the considered statement to an instance of the abstract protocol from Section 2.4.2. The reduction consists of 2 steps.

DECOMPOSING-UNIFYING. In the first step, we will employ the decomposition techniques from Section 2.4.3 together with the notations **rot** and τ from Section 2.2.5 to transform equations (4.6) and (4.7) into one equation of the form $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u} \bmod q$, where $\mathbf{M}_0, \mathbf{u}$ are public, and the coefficients of vector $\mathbf{w}_0$ are in the set $\{-1, 0, 1\}$.

Let $\mathbf{s}^* = \tau(\text{rdec}_\beta(\mathbf{s})) \in \{-1, 0, 1\}^{n\overline{m}\delta_\beta}$, $\mathbf{z}^* = \tau(\text{rdec}_\beta(\mathbf{z})) \in \{-1, 0, 1\}^{nk\delta_\beta}$ and $\mathbf{r}^* = \tau(\text{rdec}_\beta(\mathbf{r})) \in \{-1, 0, 1\}^{n\overline{m}\delta_\beta}$. Then, we observe that, equation (4.6) is equivalent

to,

$$\begin{aligned} & [\text{rot}(\mathbf{A}_{[0]}) \cdot \mathbf{H}_{k,\beta}] \cdot \mathbf{z}^\star + \sum_{i=1}^d \sum_{j=c_{i-1}}^{c_i-1} [\text{rot}(\mathbf{A}_{[i]} \cdot X^j) \cdot \mathbf{H}_{k,\beta}] \cdot t_j \cdot \mathbf{z}^\star + \\ & [\text{rot}(\mathbf{A}) \cdot \mathbf{H}_{\overline{m},\beta}] \cdot \mathbf{s}^\star - [\text{rot}(\mathbf{F})] \cdot \tau(\mathbf{y}) = \tau(u) \bmod q, \end{aligned}$$

and equation (4.7) is equivalent to

$$[\text{rot}(\mathbf{F}_0) \cdot \mathbf{H}_{\overline{m},\beta}] \cdot \mathbf{r}^\star + [\text{rot}(\mathbf{F}_1)] \cdot \tau(\text{rdec}(p)) - [\mathbf{H}] \cdot \tau(\mathbf{y}) = \mathbf{0} \bmod q.$$

Now, using basic algebra, we can manipulate the two derived equations: rearranging the secret vectors and combining them, as well as concatenating the public matrices (namely, those written inside $[\cdot]$) accordingly. As a result, we obtain an unifying equation of the form:

$$\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u} \bmod q,$$

where $\mathbf{u} = (\tau(u) \parallel \mathbf{0}) \in \mathbb{Z}_q^{2n}$ and $\mathbf{M}_0$ are public, and $\mathbf{w}_0 = (\mathbf{w}_1 \parallel \mathbf{w}_2)$, with

$$\begin{cases} \mathbf{w}_1 = (\mathbf{z}^\star \parallel t_0 \cdot \mathbf{z}^\star \parallel \dots \parallel t_{c_d-1} \cdot \mathbf{z}^\star) \in \{-1, 0, 1\}^{(k\delta_\beta + c_d k\delta_\beta)n}; \\ \mathbf{w}_2 = (\mathbf{s}^\star \parallel \mathbf{r}^\star \parallel \tau(\mathbf{y}) \parallel \tau(\text{rdec}(p))) \in \{-1, 0, 1\}^{(\overline{m}\delta_\beta + \ell)2n}. \end{cases}$$

EXTENDING-PERMUTING. In this second step, we will transform the equation $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u} \bmod q$ obtained in the first step into an equation of the form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q$, where the secret vector $\mathbf{w}$ satisfies the conditions required by the abstract protocol. In the process, we will employ the techniques introduced in Section 4.3.1.

Specifically, we extend the blocks of vector $\mathbf{w}_0 = (\mathbf{w}_1 \parallel \mathbf{w}_2)$ as follows.

$$\begin{aligned} \mathbf{w}_1 &\mapsto \mathbf{w}'_1 = \text{mix}(t, \mathbf{z}^\star) \in \{-1, 0, 1\}^{L_1}; \\ \mathbf{w}_2 &\mapsto \mathbf{w}'_2 = \text{enc}(\mathbf{w}_2) \in \{-1, 0, 1\}^{L_2}. \end{aligned} \tag{4.8}$$

Then we form vector $\mathbf{w} = (\mathbf{w}'_1 \| \mathbf{w}'_2) \in \{-1, 0, 1\}^L$, where

$$L = L_1 + L_2; \quad L_1 = (k\delta_\beta + 2c_d k\delta_\beta)3n; \quad L_2 = (\overline{m}\delta_\beta + \ell)6n.$$

At the same time, we insert suitable zero-columns to matrix $\mathbf{M}_0$ to obtain matrix $\mathbf{M} \in \mathbb{Z}_q^{2n \times L}$ such that $\mathbf{M} \cdot \mathbf{w} = \mathbf{M}_0 \cdot \mathbf{w}_0$.

Up to this point, we have transformed the considered relations into one equation of the desired form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q$. We now specify the set **VALID** that contains the obtained vector $\mathbf{w}$, the set $\mathcal{S}$ and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}\}$, such that the conditions in (2.1) hold.

Define **VALID** as the set of all vectors $\mathbf{v}' = (\mathbf{v}'_1 \| \mathbf{v}'_2) \in \{-1, 0, 1\}^L$, satisfying the following:

- There exist $t \in \{0, 1\}^{c_d}$ and $\mathbf{z}^* \in \{-1, 0, 1\}^{nk\delta_\beta}$ such that $\mathbf{v}'_1 = \text{mix}(t, \mathbf{z}^*)$.
- There exists $\mathbf{w}_2 \in \{-1, 0, 1\}^{(\overline{m}\delta_\beta + \ell)2n}$ such that $\mathbf{v}'_2 = \text{enc}(\mathbf{w}_2)$.

Clearly, our vector $\mathbf{w}$ belongs to this tailored set **VALID**.

Now, let $\mathcal{S} = \{0, 1\}^{c_d} \times \{-1, 0, 1\}^{nk\delta_\beta} \times \{-1, 0, 1\}^{(\overline{m}\delta_\beta + \ell)2n}$, and associate every element $\phi = (\mathbf{b}, \mathbf{e}, \mathbf{f}) \in \mathcal{S}$ with permutation Γ_ϕ that acts as follows. When applied to vector $\mathbf{v}^* = (\mathbf{v}_1^* \| \mathbf{v}_2^*) \in \mathbb{Z}^L$, where $\mathbf{v}_1^* \in \mathbb{Z}^{L_1}$ and $\mathbf{v}_2^* \in \mathbb{Z}^{L_2}$, it transforms $\mathbf{v}^*$ into vector

$$\Gamma_\phi(\mathbf{v}^*) = \left(\Psi_{\mathbf{b}, \mathbf{e}}(\mathbf{v}_1^*) \| \Pi_{\mathbf{f}}(\mathbf{v}_2^*) \right).$$

Based on the equivalences observed in (4.4) and (4.5), it can be checked that **VALID**, $\mathcal{S}$ and Γ_ϕ satisfy the conditions specified in (2.1). In other words, we have reduced the considered statement to an instance of the abstract protocol from Section 2.4.2.

The interactive protocol. Given the above preparations, our interactive protocol works as follows.

- The public input consists of matrix $\mathbf{M}$ and vector $\mathbf{u}$, which are built from $\mathbf{A}$, $(\mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]}, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u)$, as discussed above.
- The prover's witness is vector $\mathbf{w} \in \text{VALID}$, which is obtained from the original witnesses $(p, t, \mathbf{r}, \mathbf{v})$, as described above.

Both parties then run the protocol of Figure 2.1. The protocol uses the KTX string commitment scheme COM , which is statistically hiding and computationally binding under the (R)SIS assumption. We therefore obtain the following result, as a corollary of Theorem 2.4.4.

Theorem 4.3.2. Assume that COM is a statistically hiding and computationally binding string commitment scheme. Then the protocol described above is a statistical ZKAoK of a valid message-signature pair for the Ducas-Micciancio signature scheme, with perfect completeness, soundness error $2/3$ and communication cost $\tilde{\mathcal{O}}(\lambda)$.

Proof. For simulation, we simply run the simulator of Theorem 2.4.4. As for extraction, we invoke the knowledge extractor of Theorem 2.4.4 to obtain a vector $\mathbf{w}' \in \text{VALID}$ such that $\mathbf{M} \cdot \mathbf{w}' = \mathbf{u} \bmod q$. Then, by “backtracking” the transformations being done, we can extract from $\mathbf{w}'$ a satisfying witness $(p', t', \mathbf{r}', \mathbf{v}')$ for the considered statement.

The perfect completeness, soundness error and communication cost of the protocol directly follow from those of the abstract protocol in Section 2.4.2. In particular, the communication cost is:

$$\mathcal{O}(L \cdot \log q) = \mathcal{O}\left((k\delta_\beta + 2c_d k\delta_\beta)3n \cdot \log q + (\overline{m}\delta_\beta + \ell)6n \cdot \log q\right) = \mathcal{O}(n \cdot \log^4 n) = \tilde{\mathcal{O}}(\lambda),$$

for the setting of parameters for the Ducas-Micciancio signature in Section 2.3.3. $\square$

4.4 Constant-Size Group Signatures from Lattices

In the description below, the Ducas-Micciancio signature scheme [72, 73] as described in Section 2.3.3 is used to design a group signature scheme for (partially) dynamic groups. Group public key consists of three parts: (i) a verification key from the Ducas-Micciancio signature scheme, (ii) *two* public keys of an extended version of LPR encryption scheme [131] and (iii) a public matrix $\mathbf{B}$ for users to generate their short secret vectors together with public syndromes as user key pairs. The issue key is the corresponding signing key of the verification key while the opening key is *any one* of the corresponding secret keys of the two public keys.

When a user joins the group, it first generates a short vector together with a public syndrome using matrix $\mathbf{B}$. It then interacts with the issuer. The issuer signs the public syndrome of this user using the issue key. If the interaction completes successfully, the user obtains a signature on his syndrome from the issuer while the issuer registers this user to the group.

Once registered as a group member, the user can sign messages on behalf of the group. When signing a message, it first encrypts the public syndrome twice using the two public keys. The user then generates a ZKAoK of his syndrome, of the signature on the syndrome obtained from the issuer, of the short vector corresponding to his syndrome and of randomness used in the encryptions of the syndrome. This ZKAoK protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via Fiat-Shamir transform [75]. The signature then consists of the NIZKAoK Π_{gs} and the two ciphertexts of the syndrome. Note that the ZK argument together with double encryption enables CCA-security of the underlying encryption scheme, which is known as the Naor-Yung transformation [146]. This enables full anonymity of our group signature scheme.

When one needs to know the validity of a signature, one simply verifies Π_{gs} . In case of dispute, the opener can decrypt the syndrome using his opening key. To prevent

corrupted opening, the opener is required to generate a NIZKAoK of correct opening Π_{open} . Only when Π_{open} is a valid proof, will the judger accept the opening result.

4.4.1 Description of Our Scheme

Details of the scheme are described below.

GKg(λ): Given the security parameter λ , the trusted party proceeds as follows.

- Choose parameter $n = \mathcal{O}(\lambda)$ being a power of 2, and modulus $q = \tilde{\mathcal{O}}(n^{2.5})$, where $q = 3^k$ for some positive integer k . Let $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$.
Also, let $\ell = \lfloor \log \frac{q-1}{2} \rfloor + 1$, $m \geq 2\lceil \log q \rceil + 2$, and $\overline{m} = m + k$.
- Choose integer d and sequence $c_0, \dots, c_d$ as described in Section 2.3.3.
- Choose integer bounds $\beta = \tilde{\mathcal{O}}(n)$, $B = \tilde{\mathcal{O}}(\sqrt{n})$, and let χ be a B -bounded distribution over R .
- Let $\mathcal{H}_{\text{FS}} : \{0, 1\}^* \rightarrow \{1, 2, 3\}^\kappa$, where $\kappa = \omega(\log \lambda)$, be a collision-resistant hash function, to be modelled as a random oracle in the Fiat-Shamir transformations [75].
- Let COM be the statistically hiding and computationally binding commitment scheme from [94], to be used in our zero-knowledge argument systems.
- Draw a uniformly random matrix $\mathbf{B} \in R_q^{1 \times m}$.
- Generate verification key

$$\mathbf{A}, \mathbf{F}_0 \in R_q^{1 \times \overline{m}}, \quad \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]} \in R_q^{1 \times k}; \quad \mathbf{F}, \mathbf{F}_1 \in R_q^{1 \times \ell}; \quad u \in R_q$$

and signing key $\mathbf{R} \in R_q^{m \times k}$ for the Ducas-Micciancio signature scheme, as described in Section 2.3.3.

- Initialize the Naor-Yung double-encryption mechanism [146] with an extended version of the LPR encryption scheme [131] that allows to encrypt $\{-1, 0, 1\}$ ring vectors of length ℓ . Specifically, sample $s_1, s_2 \leftarrow \chi$, $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^\ell$, $\mathbf{a} \xleftarrow{\$} R_q^\ell$, and compute

$$\mathbf{b}_1 = \mathbf{a} \cdot s_1 + \mathbf{e}_1 \in R_q^\ell; \quad \mathbf{b}_2 = \mathbf{a} \cdot s_2 + \mathbf{e}_2 \in R_q^\ell.$$

Set the public parameter $\mathbf{pp}$, the group public key $\mathbf{gpk}$, the issue key $\mathbf{ik}$ and the opening key $\mathbf{ok}$ as follows:

$$\mathbf{pp} = \{n, q, k, R, R_q, \ell, m, \overline{m}, \chi, d, c_0, \dots, c_d, B, \beta, \kappa, \mathcal{H}_{\text{FS}}, \text{COM}, \mathbf{B}\},$$

$$\mathbf{gpk} = \{\mathbf{pp}, \mathbf{A}, \{\mathbf{A}_{[j]}\}_{j=0}^d, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u, \mathbf{a}, \mathbf{b}_1, \mathbf{b}_2\},$$

$$\mathbf{ik} = \mathbf{R}, \quad \mathbf{ok} = (s_1, \mathbf{e}_1).$$

The trusted party then makes $\mathbf{gpk}$ public and sends $\mathbf{ik}$ to the issuer and $\mathbf{ok}$ to the opener.

Assume that after receiving $\mathbf{ik}$ from the trusted party, the issuer initializes his internal state $S = 0$ and the registration table $\mathbf{reg}$.

UKg(gpk): The user samples $\mathbf{x} \in R^m$, whose coefficients are uniformly random in the set $\{-1, 0, 1\}$. Then he computes $p = \mathbf{B} \cdot \mathbf{x} \in R_q$. Set $\mathbf{upk} = p$ and $\mathbf{usk} = \mathbf{x}$.

(Join, lss): When receiving the joining request from a user with public key $\mathbf{upk} = p$, the issuer verifies that $\mathbf{upk}$ was not previously used by a registered user, and aborts if this is not the case. Otherwise, he proceeds as follows.

- Set the tag $t = (t_0, t_1, \dots, t_{c_d-1})^\top \in \mathcal{T}_d$, where $S = \sum_{j=0}^{c_d-1} 2^j \cdot t_j$, and compute $\mathbf{A}_t = [\mathbf{A} | \mathbf{A}_{[0]} + \sum_{i=1}^d t_{[i]} \mathbf{A}_{[i]}] \in R_q^{1 \times (\overline{m}+k)}$.
- Using the signing key $\mathbf{R}$, generate a Ducas-Micciancio signature $(t, \mathbf{r}, \mathbf{v})$ on

message $\mathbf{rdec}(p) \in R^\ell$ - whose coefficients are in $\{-1, 0, 1\}$. As described in Section 2.3.3, one has $\mathbf{r} \in R^{\overline{m}}$, $\mathbf{v} \in R^{\overline{m}+k}$ and

$$\begin{cases} \mathbf{A}_t \cdot \mathbf{v} = \mathbf{F} \cdot \mathbf{rdec}(\mathbf{F}_0 \cdot \mathbf{r} + \mathbf{F}_1 \cdot \mathbf{rdec}(p)) + u, \\ \|\mathbf{r}\|_\infty \leq \beta, \quad \|\mathbf{v}\|_\infty \leq \beta. \end{cases} \quad (4.9)$$

The issuer then sends the triple $(t, \mathbf{r}, \mathbf{v})$ to the user. The latter sets his group signing key as $\mathbf{gsk} = (t, \mathbf{r}, \mathbf{v}, \mathbf{x})$ while the former stores $\mathbf{reg}[S] = p$ and updates S to $S + 1$.

Sign(gpk, gsk_i, M): To sign a message $M \in \{0, 1\}^*$ using $\mathbf{gsk} = (t, \mathbf{r}, \mathbf{v}, \mathbf{x})$, the group member who has public key $p \in R_q$ proceeds as follows.

- Encrypt the ring vector $\mathbf{rdec}(p) \in R_q^\ell$ with coefficients in $\{-1, 0, 1\}$ twice. Namely, for each $i \in \{1, 2\}$, sample $g_i \leftarrow \chi$, $\mathbf{e}_{i,1} \leftarrow \chi^\ell$, and $\mathbf{e}_{i,2} \leftarrow \chi^\ell$ and compute

$$\begin{aligned} \mathbf{c}_i &= (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}) \\ &= (\mathbf{a} \cdot g_i + \mathbf{e}_{i,1}, \mathbf{b}_i \cdot g_i + \mathbf{e}_{i,2} + \lfloor q/4 \rfloor \cdot \mathbf{rdec}(p)) \in R_q^\ell \times R_q^\ell. \end{aligned}$$

- Generate a NIZKAoK $\Pi_{\mathbf{gs}}$ to demonstrate the possession of a valid tuple

$$\zeta = (t, \mathbf{r}, \mathbf{v}, \mathbf{x}, p, g_1, g_2, \mathbf{e}_{1,1}, \mathbf{e}_{2,1}, \mathbf{e}_{1,2}, \mathbf{e}_{2,2}) \quad (4.10)$$

such that

- (i) The conditions from (4.9) hold.
- (ii) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\mathbf{rdec}(p)$ with B -bounded randomness $g_1, \mathbf{e}_{1,1}, \mathbf{e}_{1,2}$ and $g_2, \mathbf{e}_{2,1}, \mathbf{e}_{2,2}$, respectively.
- (iii) $\|\mathbf{x}\|_\infty \leq 1$ and $\mathbf{B} \cdot \mathbf{x} = p$.

This is done by running the argument system described in Section 4.4.3. The protocol is an extension of the one for the Ducas-Micciancio signature from Section 4.3.2, in which the prover additionally proves statements (ii) and (iii). The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via Fiat-Shamir heuristic [75] as $\Pi_{\text{gs}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}_i\}_{i=1}^\kappa)$ where $\text{CH} = \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^\kappa, \xi)$ with

$$\xi = (\mathbf{A}, \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]}, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u, \mathbf{B}, \mathbf{a}, \mathbf{b}_1, \mathbf{b}_2, \mathbf{c}_1, \mathbf{c}_2) \quad (4.11)$$

- Output the group signature $\Pi = (\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$.

Verify(gpk, M, Σ): Given the inputs, this algorithm proceeds as follows.

1. Parse Σ as $\Sigma = (\{\text{CMT}_i\}_{i=1}^\kappa, (Ch_1, \dots, Ch_\kappa), \{\text{RSP}_i\}_{i=1}^\kappa, \mathbf{c}_1, \mathbf{c}_2)$.
If $(Ch_1, \dots, Ch_\kappa) \neq \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^\kappa, \xi)$, then return 0, where ξ is as in (4.11).
2. For each $i \in [\kappa]$, run the verification phase of the protocol in Section 4.4.3 to check the validity of RSP_i with respect to CMT_i and Ch_i . If any of the conditions does not hold, then return 0.
3. Return 1.

Open(gpk, ok, reg, M, Σ): Let $\text{ok} = (s_1, \mathbf{e}_1)$ and $\Sigma = (\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$. This algorithm then does the following.

1. Use s_1 to decrypt $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ as follows.
 - (a) It computes

$$\mathbf{p}'' = \frac{\mathbf{c}_{1,2} - \mathbf{c}_{1,1} \cdot s_1}{\lfloor q/4 \rfloor}.$$

- (b) For each coefficient of $\mathbf{p}''$,

- if it is closer to 0 than to -1 and 1 , then round it to 0;
 - if it is closer to -1 than to 0 and 1 , then round it to -1 ;
 - if it is closer to 1 than to 0 and -1 , then round it to 1 .
- (c) Denote the rounded $\mathbf{p}''$ as $\mathbf{p}' \in R_q^\ell$ with coefficients in $\{-1, 0, 1\}$.
- (d) Let $p' \in R_q$ such that $\tau(p') = \mathbf{H} \cdot \tau(\mathbf{p}')$. Recall that $\mathbf{H} \in \mathbb{Z}_q^{n \times n\ell}$ is the decomposition matrix for elements of R_q (see Section 2.4.3).
2. If **reg** does not include an entry p' , then return $(\perp, \perp)$.
3. Otherwise, generate a **NIZKAoK** Π_{open} to demonstrate the possession of a tuple $(s_1, \mathbf{e}_1, \mathbf{y}) \in R_q \times R_q^\ell \times R_q^\ell$

$$\begin{cases} \|s_1\|_\infty \leq B; \|\mathbf{e}_1\|_\infty \leq B; \|\mathbf{y}\|_\infty \leq \lceil q/10 \rceil; \\ \mathbf{a} \cdot s_1 + \mathbf{e}_1 = \mathbf{b}_1; \\ \mathbf{c}_{1,2} - \mathbf{c}_{1,1} \cdot s_1 = \mathbf{y} + \lfloor q/4 \rfloor \cdot \text{rdec}(p'). \end{cases} \quad (4.12)$$

We remark that conditions in (4.12) involve only linear secret objects with bounded norms, and can be handled using the Stern-like techniques from Sections 4.3.2 and 4.4.3. As a result, we can obtain a statistical **ZKAoK** for the considered statement. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\text{open}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}\}_{i=1}^\kappa)$, where

$$\text{CH} = \mathcal{H}_{\text{FS}}(\{\text{CMT}_i\}_{i=1}^\kappa, \mathbf{a}, \mathbf{b}_1, M, \Sigma, p') \in \{1, 2, 3\}^\kappa. \quad (4.13)$$

4. Output (p', Π_{open}) .

Judge(gpk, $M, \Sigma, p', \Pi_{\text{open}}$): If **Verify** algorithm outputs 0, then this algorithm returns

0. Otherwise, this algorithm then verifies the argument Π_{open} w.r.t. common input $(\mathbf{a}, \mathbf{b}_1, M, \Sigma, p')$, in a similar manner as in algorithm **Verify**. If Π_{open} does

not verify, then return 0; otherwise, return 1.

4.4.2 Analysis of the Scheme

EFFICIENCY. We first analyze the efficiency of the scheme described in Section 4.4.1, with respect to security parameter λ .

- The public key $\mathbf{gpk}$ has bit-size $\mathcal{O}(\lambda \cdot \log^2 \lambda) = \tilde{\mathcal{O}}(\lambda)$.
- The signing key $\mathbf{gsk}_i$ has bit-size $\mathcal{O}(\lambda \cdot \log^2 \lambda) = \tilde{\mathcal{O}}(\lambda)$.
- The size of a signature Σ is dominated by that of the Stern-like NIZKAoK Π_{gs} , which is $\mathcal{O}(\tilde{L} \cdot \log q) \cdot \omega(\log \lambda)$, where $\tilde{L}$ denotes the bit-size of a vector $\tilde{\mathbf{w}} \in \widetilde{\text{VALID}}$ as described in Section 4.4.3. Recall $\mathcal{O}(\tilde{L} \cdot \log q) = \mathcal{O}(\lambda \cdot \log^4 \lambda)$. As a result, Σ has bit-size $\mathcal{O}(\lambda \cdot \log^4 \lambda) \cdot \omega(\log \lambda) = \tilde{\mathcal{O}}(\lambda)$.
- The Stern-like NIZKAoK Π_{open} has bit-size $\mathcal{O}(\lambda \cdot \log^3 \lambda) \cdot \omega(\log \lambda) = \tilde{\mathcal{O}}(\lambda)$.

CORRECTNESS. The correctness of the above group signature scheme relies on the following facts: (i) the underlying argument systems to generate Π_{gs} and Π_{open} are perfectly complete; (ii) the underlying encryption scheme, which is an extended version of LPR encryption scheme [131] is correct.

Specifically, for an honest user, when he signs a message on behalf of the group, he is able to demonstrate the possession of a valid tuple ζ of the form (4.10). With probability 1, Π_{gs} is accepted by the **Verify** algorithm, which is implied by the perfect completeness of the argument system to generate Π_{gs} . As for the correctness of the **Open** algorithm, note that

$$\begin{aligned}
\mathbf{c}_{1,2} - \mathbf{c}_{1,1} \cdot s_1 &= \mathbf{b}_1 \cdot g_1 + \mathbf{e}_{1,2} + \lfloor q/4 \rfloor \cdot \text{rdec}(p) - (\mathbf{a} \cdot g_1 + \mathbf{e}_{1,1}) \cdot s_1 \\
&= (\mathbf{a} \cdot s_1 + \mathbf{e}_1) \cdot g_1 + \mathbf{e}_{1,2} + \lfloor q/4 \rfloor \cdot \text{rdec}(p) - (\mathbf{a} \cdot g_1 + \mathbf{e}_{1,1}) \cdot s_1 \\
&= \mathbf{e}_1 \cdot g_1 + \mathbf{e}_{1,2} - \mathbf{e}_{1,1} \cdot s_1 + \lfloor q/4 \rfloor \cdot \text{rdec}(p)
\end{aligned}$$

where $\|\mathbf{e}_1\|_\infty \leq B$, $\|s_1\|_\infty \leq B$, $\|g_1\|_\infty \leq B$, $\|\mathbf{e}_{1,1}\|_\infty \leq B$, $\|\mathbf{e}_{1,2}\|_\infty \leq B$. Recall $B = \tilde{\mathcal{O}}(\sqrt{n})$ and $q = \tilde{\mathcal{O}}(n^{2.5})$. Hence we have:

$$\|\mathbf{e}_1 \cdot g_1 + \mathbf{e}_{1,2} - \mathbf{e}_{1,1} \cdot s_1\|_\infty \leq 2n \cdot B^2 + B = \tilde{\mathcal{O}}(n^2) \leq \left\lceil \frac{q}{10} \right\rceil = \tilde{\mathcal{O}}(n^{2.5}).$$

With probability 1, the rounding procedure described in the **Open** algorithm recovers $\mathbf{rdec}(p)$ and hence outputs p , which is the actual signer. Thus the opener is able to identify the signer of a signature and hence correctness of the **Open** algorithm holds.

As the opener correctly recovers $\mathbf{rdec}(p)$ and p , it possesses a valid tuple $(s_1, \mathbf{e}_1, \mathbf{y})$ satisfying conditions in (4.12). It then follows from the perfect completeness of the argument system to generate Π_{open} , the judge will accept the opening result outputted by the opener and hence correctness of the **Judge** algorithm holds.

SECURITY. In Theorem 4.4.2, we prove that our scheme satisfies the security requirements of the Bellare, Shi and Zhang's model [21]. For the proof of non-frameability, we will use the following simple lemma.

Lemma 4.4.1. Let $\mathbf{B} \in R_q^{1 \times m}$, where $m \geq 2\lceil \log q \rceil + 2$. If $\mathbf{x}$ is a uniformly random element of R^m such that $\|\mathbf{x}\|_\infty \leq 1$, then with probability at least $1 - 2^{-n}$, there exists another $\mathbf{x}' \in R^m$ such that $\|\mathbf{x}'\|_\infty \leq 1$ and $\mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{x}' \in R_q$.

Proof. Note that there are in total 3^{nm} elements $\mathbf{x} \in R^m$ such that $\|\mathbf{x}\|_\infty \leq 1$. Among them, there exist at most $q^n - 1$ elements that do not have $\mathbf{x}'$ such that $\mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{x}'$. Hence, the probability that a uniformly random $\mathbf{x}$ has a corresponding $\mathbf{x}'$ for which $\mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{x}'$ is at least

$$\frac{3^{nm} - q^n + 1}{3^{nm}} = 1 - \frac{q^n - 1}{3^{nm}} > 1 - \frac{q^n}{2^n q^n} = 1 - 2^{-n}.$$

□

□

Theorem 4.4.2. Assume that the Stern-like argument systems used in Section 4.4.1 are simulation-sound. Then, in the random oracle model, the given group signature scheme satisfies full anonymity, traceability and non-frameability under the RLWE and RSIS assumptions.

In the random oracle model, the proof of Theorem 4.4.2 relies on the following facts:

1. The Stern-like zero-knowledge argument systems being used are simulation-sound;
2. The underlying encryption scheme, which is an extended version of the LPR encryption scheme [131], via the Naor-Yung transformation [146], is IND-CCA secure;
3. The variant of Ducas-Micciancio signature scheme described in Section 2.3.3 with at most polynomial number of signature queries is existentially unforgeable against adaptive chosen message attacks [72, 73];
4. For a properly generated user key pair $(\mathbf{x}, p)$, it is infeasible to find $\mathbf{x}' \in R_q^m$ such that $\|\mathbf{x}'\|_\infty \leq 1$, $\mathbf{x}' \neq \mathbf{x}$ and $\mathbf{B} \cdot \mathbf{x}' = p$.

The proof of Theorem 4.4.2 is established by Lemmas 4.4.3-4.4.5 given below.

Lemma 4.4.3. Assume that the $\text{RLWE}_{n,\ell,q,\chi}$ problem is hard. Then the given group signature scheme is fully anonymous in the random oracle model.

Proof. We prove this lemma using a series of games. In the first game, the challenger runs the experiment $\mathbf{Exp}_{\text{GS},\mathcal{A}}^{\text{anon}-0}(\lambda)$ while in the last game, the challenger runs the experiment $\mathbf{Exp}_{\text{GS},\mathcal{A}}^{\text{anon}-1}(\lambda)$. Denote W_i be the event that the adversary outputs 1 in Game i .

Game 0: This is exactly the experiment $\mathbf{Exp}_{\text{GS},\mathcal{A}}^{\text{anon}-0}(\lambda)$, where the adversary receives a challenged signature $\Sigma^* \leftarrow \text{Sign}(\text{gpk}, \text{gsk}_{i_0}, M)$ in the challenge phase. So $\Pr[W_0] = \Pr[\mathbf{Exp}_{\text{GS},\mathcal{A}}^{\text{anon}-0}(\lambda) = 1]$.

Game 1: This game is the same as Game 0 with one exception: the challenger will keep both decryption keys of the ring-based variant of Regev's encryption scheme instead of erasing the second decryption key $(s_2, \mathbf{e}_2)$. However, the view of the adversary $\mathcal{A}$ is still the same as in Game 0. Therefore, $\Pr[W_0] = \Pr[W_1]$.

Game 2: This game is identical to Game 1 except that it generates simulated proofs for the *opening* oracle queries by programming the random oracle $\mathcal{H}_{\text{FS}}$. Note that the challenger still follows the original game (that is, it uses s_1 to decrypt $\mathbf{c}_1$) to identify the real signer. The view of the adversary, however, is statistically indistinguishable between Game 1 and Game 2 by statistical zero-knowledge property of our argument system generating Π_{Open} . Therefore we have the following: $|\Pr[W_1] - \Pr[W_2]| = \text{negl}(\lambda)$.

Game 3: This game modifies Game 2 as follows. It uses s_2 instead of s_1 to decrypt the ciphertext $\mathbf{c}_2$ for the *opening* oracle queries. The view of $\mathcal{A}$ is the same as in Game 2 until event F_1 , where $\mathcal{A}$ queries the *opening* oracle a valid signature $(\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting distinct messages, happens. Since F_1 breaks the soundness of our argument generating Π_{gs} , we have $|\Pr[W_2] - \Pr[W_3]| \leq \Pr[F_1] \leq \text{Adv}_{\Pi_{\text{gs}}}^{\text{sound}}(\lambda) = \text{negl}(\lambda)$.

Game 4: This game changes Game 3 in the following way. It generates a simulated proof Π_{gs}^* in the *challenge* phase by programming the random oracle $\mathcal{H}_{\text{FS}}$ even though the challenger has the correct witness to generate a real proof. The view is indistinguishable to $\mathcal{A}$ by statistical zero-knowledge property of our argument system generating Π_{gs}^* . Therefore we have $|\Pr[W_3] - \Pr[W_4]| = \text{negl}(\lambda)$.

Game 5: In this game, we modify Game 4 by modifying the distribution of the challenged signature $\Sigma^* = (\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ as follows. Recall that in Game 4, both $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ encrypt the same message, i.e., $\text{rdec}(p_{i_0})$. Here we change $\mathbf{c}_1^*$ to be encryption of $\text{rdec}(p_{i_1})$ and keep $\mathbf{c}_2^*$ unchanged. By the semantic security of

the underlying encryption scheme for public key $(\mathbf{a}, \mathbf{b}_1)$ (which is implied by the $\text{RLWE}_{n,\ell,q,\chi}$ assumption since we no longer use s_1 to open signatures), the change made in this game is negligible to the adversary. Therefore we have $|\Pr[W_4] - \Pr[W_5]| = \text{negl}(\lambda)$.

Game 6: This game follows Game 5 with one change: it switches back to use s_1 for the *opening* oracle queries and erases $(s_2, \mathbf{e}_2)$ again. This modification is indistinguishable to the adversary until event F_2 , where $\mathcal{A}$ queries the *opening* oracle a valid signature $(\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting different messages, happens. However event F_2 breaks the simulation soundness of the argument system generating Π_{gs} . Therefore we have $|\Pr[W_5] - \Pr[W_6]| \leq \text{Adv}_{\Pi_{\text{gs}}}^{\text{ss}}(\lambda) = \text{negl}(\lambda)$.

Game 7: In this game, we modify Game 6 by modifying the distribution of the challenged signature $\Sigma^* = (\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ again. It changes $\mathbf{c}_2^*$ to be encryption of $\text{rdec}(p_{i_1})$ in the *challenge* phase. By the semantic security of our encryption scheme for public key $(\mathbf{a}, \mathbf{b}_2)$ (which is implied by the $\text{RLWE}_{n,\ell,q,\chi}$ assumption as we no longer use s_2 to open signatures), the change is negligible to the adversary. Hence we have $|\Pr[W_6] - \Pr[W_7]| = \text{negl}(\lambda)$. Note that now both $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ encrypt the same message, i.e., $\text{rdec}(p_{i_1})$, so Π_{gs}^* is a simulated proof even though the challenger has correct witness.

Game 8: In the game, we modify Game 7 in the following way. We switch to generate a real proof Π_{gs}^* in the *challenge* phase. By the statistical zero-knowledge property of our argument system generating Π_{gs}^* , the view of $\mathcal{A}$ is indistinguishable between Game 7 and Game 8. Therefore we have $|\Pr[W_7] - \Pr[W_8]| = \text{negl}(\lambda)$.

Game 9: This game changes Game 8 in one aspect. It produces real proofs for the *opening* oracle queries. By the statistical zero-knowledge property of our argument system generating Π_{Open} , we have Game 8 and Game 9 are statistically

indistinguishable to the adversary, i.e., we have $|\Pr[W_8] - \Pr[W_9]| = \text{negl}(\lambda)$.

This is actually the experiment $\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{anon}-1}(\lambda)$. Hence, we obtain the equation:

$$\Pr[W_9] = \Pr[\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{anon}-1}(\lambda) = 1].$$

As a result, we have that

$$|\Pr[\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{anon}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{FDGS},\mathcal{A}}^{\text{anon}-0}(\lambda) = 1]| = \text{negl}(\lambda),$$

and hence our scheme is fully anonymous. □

Lemma 4.4.4. Assume that the $\text{RSIS}_{n,\bar{m},q,\tilde{\mathcal{O}}(n^2)}^\infty$ problem is hard. Then the given group signature scheme is traceable in the random oracle model.

Proof. We prove traceability by contradiction. Suppose that $\mathcal{A}$ succeeds with non-negligible advantage ϵ . Then we build a PPT algorithm $\mathcal{B}$ that, with non-negligible probability, breaks the unforgeability of the Ducas-Micciancio signature scheme from Section 2.3.3, which is based on the hardness of the $\text{RSIS}_{n,\bar{m},q,\tilde{\mathcal{O}}(n^2)}^\infty$ problem. It then follows that our construction is traceable.

When given the verification key of the Ducas-Micciancio signature scheme, the simulator $\mathcal{B}$ runs the experiment $\mathbf{Exp}_{\text{GS},\mathcal{A}}^{\text{trace}}(\lambda)$ faithfully. $\mathcal{B}$ can answer all oracle queries made by $\mathcal{A}$ except when $\mathcal{A}$ queries the *send to issuer* SndTol oracle or *add user* AddU oracle. However, $\mathcal{B}$ can resort to his oracle queries of the signature scheme. In these two cases, $\mathcal{B}$ enrolls the corresponding user to the group. When $\mathcal{A}$ halts, it outputs $(M^*, \Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$. With non-negligible probability ϵ , $\mathcal{A}$ wins the experiment. Parse $\Pi_{\text{gs}}^* = (\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}_i^*\}_{i=1}^\kappa)$. Let

$$\xi^* = (\mathbf{A}, \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]}, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u, \mathbf{B}, \mathbf{a}, \mathbf{b}_1, \mathbf{b}_2, \mathbf{c}_1^*, \mathbf{c}_2^*).$$

Then $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ and RSP_i^* is a valid response w.r.t. CMT_i^*

and CH_i^* for $i \in [\kappa]$ by the fact that $\mathcal{A}$ wins and hence $(\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ is a valid signature on message M^* .

We claim that $\mathcal{A}$ had queried $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ to the hash oracle $\mathcal{H}_{\text{FS}}$ with overwhelming probability. Otherwise, the probability of guessing correctly the value of $\mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ is at most $3^{-\kappa}$, which is negligible. Therefore, with probability $\epsilon' = \epsilon - 3^{-\kappa}$, $\mathcal{A}$ had queried the hash oracle $\mathcal{H}_{\text{FS}}$. Denote by $\theta^* \in \{1, 2, \dots, Q_H\}$ the index of this specific query, where Q_H is the total number of hash queries made by $\mathcal{A}$.

Algorithm $\mathcal{B}$ then runs at most $32 \cdot Q_H / \epsilon'$ executions of $\mathcal{A}$. For each new run, it is exactly the same as the original run until the point of θ^* -th query to the hash oracle $\mathcal{H}_{\text{FS}}$. From this point on, $\mathcal{B}$ replies $\mathcal{A}$'s hash queries with uniformly random and independent values for each new run. This guarantees that the input of θ^* -th query $\mathcal{A}$ made to $\mathcal{H}_{\text{FS}}$ is the tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ for each new run while the output of this hash query is uniformly random and independent for each new run. To this point, by the forking lemma of Brickell et al. [47], with probability $\geq 1/2$, $\mathcal{B}$ obtains 3-fork involving the same tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ with pairwise distinct hash values $\text{CH}_{\theta^*}^{(1)}, \text{CH}_{\theta^*}^{(2)}, \text{CH}_{\theta^*}^{(3)} \in \{1, 2, 3\}^\kappa$ and corresponding valid responses $\text{RSP}_{\theta^*}^{(1)}, \text{RSP}_{\theta^*}^{(2)}, \text{RSP}_{\theta^*}^{(3)}$. A simple calculation shows that with probability $1 - (\frac{7}{9})^\kappa$, we have $\{\text{CH}_{\theta^*,j}^{(1)}, \text{CH}_{\theta^*,j}^{(2)}, \text{CH}_{\theta^*,j}^{(3)}\} = \{1, 2, 3\}$ for some $j \in \{1, 2, \dots, \kappa\}$.

Therefore, $\text{RSP}_{\theta^*,j}^{(1)}, \text{RSP}_{\theta^*,j}^{(2)}, \text{RSP}_{\theta^*,j}^{(3)}$ are 3 valid responses for all the challenges 1, 2, 3 w.r.t. the same commitment CMT_j^* . Since COM is computationally binding, $\mathcal{B}$ is able to extract the witness

$$t^* \in \mathcal{T}_d; \mathbf{r}^* \in R_q^{\overline{m}}; \mathbf{v}^* \in R_q^{\overline{m}+k}; \mathbf{p}^* \in R_q^\ell,$$

such that $\|\mathbf{r}^*\|_\infty \leq \beta$, $\|\mathbf{v}^*\|_\infty \leq \beta$, $\|\mathbf{p}^*\|_\infty \leq 1$ and

$$\mathbf{A}_{t^*} \cdot \mathbf{v}^* = \mathbf{F} \cdot \text{rdec}(\mathbf{F}_0 \cdot \mathbf{r}^* + \mathbf{F}_1 \cdot \mathbf{p}^*) + u,$$

and $\mathbf{c}_1^*, \mathbf{c}_2^*$ are correct encryptions of $\mathbf{p}^*$.

Since $\mathcal{A}$ wins the game, either we have (i) the **Open** algorithm outputs $(\perp, \perp)$ or (ii) the **Open** algorithm output $(p', \Pi_{\text{open}}^*)$ with $p' \neq \perp$ but the **Judge** algorithm rejects the opening result.

Case (i) implies that, if $\mathbf{c}_1^*$ is decrypted to $\mathbf{p}'$ and $p' \in R_q$ such that $\tau(p') = \mathbf{H} \cdot \tau(\mathbf{p}') \in \mathbb{Z}_q^n$, then p' is not in the registration table. From the extraction, we know that $\mathbf{c}_1^*$ will be decrypted to $\mathbf{p}^*$ by the correctness of our encryption scheme. Therefore, the *intermediate* opening result $\mathbf{p}'$ is equal to $\mathbf{p}^*$. On the other hand, the fact that p' is not in the registration table implies that $\mathcal{B}$ did not enroll p' to the group, that is, $\mathcal{B}$ did not query p' to his challenger when $\mathcal{A}$ made the **AddU** oracle queries or **SndTol** oracle queries. To summarize, $\mathcal{B}$ did not query signature on p' and $\mathcal{B}$ extracts a signature $(t^*, \mathbf{r}^*, \mathbf{v}^*)$ on $\mathbf{p}^* = \mathbf{p}'$ such that $\tau(p') = \mathbf{H} \cdot \tau(\mathbf{p}')$. Therefore $(\mathbf{p}^*, t^*, \mathbf{r}^*, \mathbf{v}^*)$ is a valid forgery of the **Ducas-Micciancio** signature scheme.

Case (ii) implies that, if $\mathbf{c}_1^*$ is decrypted to $\mathbf{p}'$ and $p' \in R_q$ such that $\tau(p') = \mathbf{H} \cdot \tau(\mathbf{p}') \in \mathbb{Z}_q^n$, then p' is in the registration table and Π_{open}^* generated by $\mathcal{B}$ is not accepted by the **Judge** algorithm. From the extraction, we know that $\mathbf{c}_1^*$ will be decrypted to $\mathbf{p}^*$ by the correctness of our encryption scheme. Therefore, the *intermediate* opening result $\mathbf{p}'$ is equal to $\mathbf{p}^*$. On the other hand, we claim that $\text{rdec}(p') \neq \mathbf{p}' = \mathbf{p}^*$. Otherwise, $\text{rdec}(p') = \mathbf{p}' = \mathbf{p}^*$, then $\mathcal{B}$ possesses valid witness to generate the proof Π_{open}^* . By the perfect completeness of the underlying argument system generating Π_{open}^* , it will be accepted by the **judge** algorithm with probability 1. This is a contradiction and hence we obtain $\text{rdec}(p') \neq \mathbf{p}' = \mathbf{p}^*$. Recall that in the $\langle \text{Join}, \text{Iss} \rangle$ algorithm, the issuer only generates signature on $\text{rdec}(p')$. So $\mathcal{B}$ only queries the signature on $\text{rdec}(p')$ and hence $(\mathbf{p}^*, t^*, \mathbf{r}^*, \mathbf{v}^*)$ is a valid forgery of the **Ducas-Micciancio** signature scheme.

Therefore, with probability at least $\frac{1}{2} \cdot (\epsilon - 3^{-\kappa})(1 - (\frac{7}{9})^\kappa)$, which is non-negligible, $\mathcal{B}$ breaks the unforgeability of the **Ducas-Micciancio** signature scheme. This concludes the proof. $\square$

Lemma 4.4.5. Assume that the $\text{RSIS}_{n,m,q,1}^\infty$ problem is hard. Then the given group signature scheme is non-frameable in the random oracle model.

Proof. We prove non-frameability by contradiction. Suppose that $\mathcal{A}$ succeeds with non-negligible advantage ϵ . Then we build a PPT algorithm $\mathcal{B}$ that solves a $\text{RSIS}_{n,m,q,1}$ instance $\mathbf{B} \in R_q^{1 \times m}$ with non-negligible probability.

After $\mathcal{B}$ is given a RSIS instance matrix $\mathbf{B}$, it runs the experiment $\text{Exp}_{\text{GS},\mathcal{A}}^{\text{nf}}$ faithfully. $\mathcal{B}$ can answer all the oracle queries made by $\mathcal{A}$ since $\mathcal{B}$ knows all the keys. When $\mathcal{A}$ halts, it outputs $(M^*, \Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*, p^*, \Pi_{\text{open}}^*)$. With non-negligible probability ϵ , $\mathcal{A}$ wins the experiment.

The fact that $\mathcal{A}$ wins the game implies $(M^*, \Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ is a valid message-signature pair that was not queried before. By the same extraction technique as in Lemma 4.4.4, we can extract witness $\mathbf{x}' \in R_q^m$ and $\mathbf{p}' \in R_q^\ell$ such that $\mathbf{x}', \mathbf{p}'$ have coefficients in $\{-1, 0, 1\}$, $\mathbf{B} \cdot \mathbf{x}' = p'$ with $\tau(p') = \mathbf{H} \cdot \tau(\mathbf{p}')$ and $\mathbf{c}_1^*, \mathbf{c}_2^*$ are correct encryptions of $\mathbf{p}'$. By the correctness of the encryption scheme being used, $\mathbf{c}_1^*$ will be decrypted to $\mathbf{p}'$.

The fact that $\mathcal{A}$ wins the game also implies $(p^*, \Pi_{\text{open}}^*)$ is accepted by the **Judge** algorithm. It follows from the soundness of the argument system used to generate Π_{open}^* that $\mathbf{c}_1^*$ will be decrypted to $\text{rdec}(p^*)$. Therefore, we have $\mathbf{p}' = \text{rdec}(p^*)$ and hence $p' = p^*$. Note that $\mathcal{A}$ wins the game also implies that p^* is an honest user with $\text{gsk} \neq \perp$ and $\mathcal{A}$ did not query the user secret key $\mathbf{x}^*$ that corresponds to p^* . Thus we obtain: $\mathbf{B} \cdot \mathbf{x}' = p' = p^* = \mathbf{B} \cdot \mathbf{x}^*$, where $\mathbf{x}^*$ has coefficients in $\{-1, 0, 1\}$. By Lemma 4.4.1, $\mathbf{x}' \neq \mathbf{x}^*$ with probability at least $1/2$. In the case they are not equal, we obtain a non-zero vector $\mathbf{y} = \mathbf{x}' - \mathbf{x}^*$ such that $\mathbf{B} \cdot \mathbf{y} = 0$ and $\|\mathbf{y}\|_\infty = 1$.

Therefore, with probability at least $\frac{1}{2} \cdot (\epsilon - 3^{-\kappa})(1 - (\frac{7}{9})^\kappa) \cdot \frac{1}{2}$, which is non-negligible, $\mathcal{B}$ solves a $\text{RSIS}_{n,m,q,1}$ instance $\mathbf{B} \in R_q^{1 \times m}$. This concludes the proof. $\square$

4.4.3 The Underlying Zero-Knowledge Argument System

Let us now present the statistical ZKAoK that will be invoked by the signer when generating group signatures. The protocol is an extension of the one for the Ducas-Micciancio signature from Section 4.3.2, for which the prover additionally convinces the verifier of the following two facts.

1. He knows a secret key $\mathbf{x} \in R^m$ corresponding to the public key $p \in R_q$, which satisfies $\|\mathbf{x}\|_\infty \leq 1$ and $\mathbf{B} \cdot \mathbf{x} = p$. Here, $\mathbf{B} \in R_q^{1 \times m}$ is a public matrix.
2. He has correctly encrypted the vector $\text{rdec}(p) \in R^\ell$ to a given ciphertext in $(R_q^\ell)^4$ of the form $(\mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{2,2})$, under public key $(\mathbf{a}, \mathbf{b}_1, \mathbf{b}_2) \in (R_q^\ell)^3$. To this end, he proves that equations

$$\mathbf{c}_{i,1} = \mathbf{a} \cdot g_i + \mathbf{e}_{i,1}, \quad \mathbf{c}_{i,2} = \mathbf{b}_i \cdot g_i + \mathbf{e}_{i,2} + \lfloor q/4 \rfloor \cdot \text{rdec}(p), \quad (4.14)$$

hold for B -bounded randomness $g_1, g_2 \in R$, and $\mathbf{e}_{1,1}, \mathbf{e}_{2,1}, \mathbf{e}_{1,2}, \mathbf{e}_{2,2} \in R^\ell$.

As the transformations for the “Ducas-Micciancio layer” have been established in Section 4.3.2, in the following, we only specify the transformations with respect to the newly appeared relations.

We will first apply the decomposition techniques in Section 2.4.3 to the secret objects.

- Let $\mathbf{x}^* = \tau(\mathbf{x}) \in \{-1, 0, 1\}^{nm}$.
- For $i \in \{1, 2\}$, compute $\mathbf{g}_i^* = \tau(\text{rdec}_B(g_i)) \in \{-1, 0, 1\}^{n\delta_B}$.
- For $i \in \{1, 2\}$, compute $\mathbf{e}_{i,1}^* = \tau(\text{rdec}_B(\mathbf{e}_{i,1}))$ and $\mathbf{e}_{i,2}^* = \tau(\text{rdec}_B(\mathbf{e}_{i,2}))$. Note that they are vectors in $\{-1, 0, 1\}^{n\delta_B}$.

Then the equation $\mathbf{B} \cdot \mathbf{x} = p$ can be translated as

$$[\text{rot}(\mathbf{B})] \cdot \mathbf{x}^* - [\mathbf{H}] \cdot \tau(\text{rdec}(p)) = \mathbf{0}^n \bmod q. \quad (4.15)$$

Meanwhile, let $\mathbf{a} = (a_1, \dots, a_\ell)^\top$, $\{\mathbf{b}_i = (b_{i,1}, \dots, b_{i,\ell})^\top\}_{i=1,2}$, then equations in (4.14) can be rewritten as, for $i = 1, 2$,

$$\begin{bmatrix} \text{rot}(a_1) \cdot \mathbf{H}_B \\ \vdots \\ \text{rot}(a_\ell) \cdot \mathbf{H}_B \end{bmatrix} \cdot \mathbf{g}_i^* + [\mathbf{H}_{\ell,B}] \cdot \mathbf{e}_{i,1}^* = \tau(\mathbf{c}_{i,1}) \bmod q; \quad (4.16)$$

$$\begin{bmatrix} \text{rot}(b_{i,1}) \cdot \mathbf{H}_B \\ \vdots \\ \text{rot}(b_{i,\ell}) \cdot \mathbf{H}_B \end{bmatrix} \cdot \mathbf{g}_i^* + [\mathbf{H}_{\ell,B}] \cdot \mathbf{e}_{i,2}^* + \lfloor q/4 \rfloor \cdot \tau(\text{rdec}(p)) = \tau(\mathbf{c}_{i,2}) \bmod q. \quad (4.17)$$

Before proceeding further, let us recall that, in the protocol for the Ducas-Micciancio signature from Section 4.3.2, at the end of the Decomposing-Unifying step, we did combine the secret objects into vectors $\mathbf{w}_1, \mathbf{w}_2$ of the form:

$$\begin{cases} \mathbf{w}_1 = (\mathbf{z}^* \parallel t_0 \cdot \mathbf{z}^* \parallel \dots \parallel t_{c_d-1} \cdot \mathbf{z}^*) \in \{-1, 0, 1\}^{(k\delta_\beta + c_d k \delta_\beta)n}; \\ \mathbf{w}_2 = (\mathbf{s}^* \parallel \mathbf{r}^* \parallel \tau(\mathbf{y}) \parallel \tau(\text{rdec}(p))) \in \{-1, 0, 1\}^{(\overline{m}\delta_\beta + \ell)2n}. \end{cases}$$

Since vector $\tau(\text{rdec}(p))$ has been counted as a block of vector $\mathbf{w}_2$, we now combine the newly appeared secret vectors in equations (4.15), (4.16), (4.17) into vector

$$\mathbf{w}_3 = (\mathbf{x}^* \parallel \mathbf{g}_1^* \parallel \mathbf{g}_2^* \parallel \mathbf{e}_{1,1}^* \parallel \mathbf{e}_{1,2}^* \parallel \mathbf{e}_{2,1}^* \parallel \mathbf{e}_{2,2}^*) \in \{-1, 0, 1\}^{nm+2n\delta_B+4n\ell\delta_B},$$

and let $\mathbf{w}_4 = (\mathbf{w}_2 \parallel \mathbf{w}_3) \in \{-1, 0, 1\}^{L'_4}$, for $L'_4 = (\overline{m}\delta_\beta + \ell)2n + nm + 2n\delta_B + 4n\ell\delta_B$.

Next, we extend $\mathbf{w}_4$ to vector $\mathbf{w}'_4 = \text{enc}(\mathbf{w}_4) \in \{-1, 0, 1\}^{L_4}$, where $L_4 = 3L'_4$, and form the vector

$$\tilde{\mathbf{w}} = (\mathbf{w}'_1 \parallel \mathbf{w}'_4) \in \{-1, 0, 1\}^{\tilde{L}},$$

where $\mathbf{w}'_1 = \text{mix}(t, \mathbf{z}^*) \in \{-1, 0, 1\}^{L_1}$ is the “mixing vector” obtained in (4.8), and $\tilde{L} = L_1 + L_4$.

We remark that, by suitably concatenating/extending the matrices and vectors derived from the public input, we can obtain public matrix $\widetilde{\mathbf{M}}$ and public vector $\widetilde{\mathbf{u}}$ such that $\widetilde{\mathbf{M}} \cdot \widetilde{\mathbf{w}} = \widetilde{\mathbf{u}} \bmod q$. Having obtained this desired equation, we now proceed as in Section 4.3.2.

Define $\widetilde{\text{VALID}}$ as the set of all vectors $\mathbf{v}' = (\mathbf{v}'_1 \| \mathbf{v}'_4) \in \{-1, 0, 1\}^{\widetilde{L}}$, satisfying the following:

- There exist $t \in \{0, 1\}^{c_d}$ and $\mathbf{z}^* \in \{-1, 0, 1\}^{nk\delta_\beta}$ such that $\mathbf{v}'_1 = \text{mix}(t, \mathbf{z}^*)$.
- There exists $\mathbf{w}_4 \in \{-1, 0, 1\}^{L'_4}$ such that $\mathbf{v}'_4 = \text{enc}(\mathbf{w}_4)$.

It can be seen that vector $\widetilde{\mathbf{w}}$ belongs to $\widetilde{\text{VALID}}$.

Now, let $\widetilde{\mathcal{S}} = \{0, 1\}^{c_d} \times \{-1, 0, 1\}^{nk\delta_\beta} \times \{-1, 0, 1\}^{L'_4}$, and associate every element $\phi = (\mathbf{b}, \mathbf{e}, \mathbf{f}) \in \widetilde{\mathcal{S}}$ with permutation $\widetilde{\Gamma}_\phi$ that acts as follows. When applied to vector $\mathbf{v}^* = (\mathbf{v}^*_1 \| \mathbf{v}^*_4) \in \mathbb{Z}^{\widetilde{L}}$, where $\mathbf{v}^*_1 \in \mathbb{Z}^{L_1}$ and $\mathbf{v}^*_4 \in \mathbb{Z}^{L_4}$, it transforms $\mathbf{v}^*$ into vector

$$\widetilde{\Gamma}_\phi(\mathbf{v}^*) = \left(\Psi_{\mathbf{b}, \mathbf{e}}(\mathbf{v}^*_1) \parallel \Pi_{\mathbf{f}}(\mathbf{v}^*_4) \right).$$

Based on the equivalences observed in (4.4) and (4.5), it can be checked that $\widetilde{\text{VALID}}$, $\widetilde{\mathcal{S}}$ and $\widetilde{\Gamma}_\phi$ satisfy the conditions specified in (2.1). In other words, we have reduced the considered statement to an instance of the abstract protocol from Section 2.4.2.

The interactive protocol. Given the above preparations, our interactive protocol works as follows.

- The public input consists of matrix $\widetilde{\mathbf{M}}$ and vector $\widetilde{\mathbf{u}}$, which are built from $\mathbf{A}$, $(\mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]}, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u)$, and $\mathbf{B}$, $\mathbf{c}_{1,1}$, $\mathbf{c}_{1,2}$, $\mathbf{c}_{2,1}$, $\mathbf{c}_{2,2}$, $\mathbf{a}$, $\mathbf{b}_1$, $\mathbf{b}_2$, as discussed in Section 4.3.2 and above.
- The prover's witness is vector $\widetilde{\mathbf{w}} \in \widetilde{\text{VALID}}$, which is obtained from the original witnesses $(p, t, \mathbf{r}, \mathbf{v}, \mathbf{x}, g_1, g_2, \mathbf{e}_{1,1}, \mathbf{e}_{2,1}, \mathbf{e}_{1,2}, \mathbf{e}_{2,2})$, as described in Section 4.3.2 and above.

Both parties then run the protocol of Figure 2.1. The protocol uses the KTX string commitment scheme **COM**, which is statistically hiding and computationally binding under the (R)SIS assumption. We therefore obtain the following result, as a corollary of Theorem 2.4.4.

Theorem 4.4.6. Assume that **COM** is a statistically hiding and computationally binding string commitment scheme. Then the protocol described above is a statistical ZKAoK for the considered statement, with perfect completeness, soundness error $2/3$ and communication cost $\tilde{\mathcal{O}}(\lambda)$.

Proof. For simulation, we simply run the simulator of Theorem 2.4.4. As for extraction, we invoke the knowledge extractor of Theorem 2.4.4 to obtain a vector $\tilde{\mathbf{w}}' \in \widetilde{\text{VALID}}$ such that $\tilde{\mathbf{M}} \cdot \tilde{\mathbf{w}}' = \tilde{\mathbf{u}} \bmod q$. Then, by “backtracking” all the transformations being done, we can extract a satisfying witness $(p', t', \mathbf{r}', \mathbf{v}', \mathbf{x}', g'_1, g'_2, \mathbf{e}'_{1,1}, \mathbf{e}'_{2,1}, \mathbf{e}'_{1,2}, \mathbf{e}'_{2,2})$ from vector $\tilde{\mathbf{w}}'$ for the considered statement.

The perfect completeness, soundness error and communication cost of the protocol directly follow from those of the abstract protocol in Section 2.4.2. In particular, the communication cost is:

$$\mathcal{O}(\tilde{L} \cdot \log q) = \mathcal{O}\left((k\delta_\beta + c_d k\delta_\beta)n \cdot \log q + ((\bar{m}\delta_\beta + \ell)n + nm + n\delta_B + n\ell\delta_B) \cdot \log q\right),$$

which is of order $\mathcal{O}(n \cdot \log^4 n) = \tilde{\mathcal{O}}(\lambda)$, for the setting of parameters we use in the group signature scheme of Section 4.4.1. $\square$

Chapter 5

Forward-Secure Group Signatures from Lattices

In Chapter 3 and Chapter 4, we described our fully dynamic group signature scheme and constant-size group signature scheme from lattices. However, these schemes will become unsatisfactory if the group signing keys are exposed. On one hand, all previously generated signatures are rendered invalid since there is no way to distinguish signatures produced by the adversary from those produced by the innocent users. On the other hand, some malicious user may disavow their unlawfully generated signatures by disclosing his group signing key and claiming himself to be the victim of the key exposure problem. What is worse, the damage escalates when the group size increases. In this chapter, we construct the first lattice-based forward-secure group signature (FSGS) scheme, in which the security of the scheme is preserved even if some keys are compromised to the adversary.

Our scheme satisfies forward-secure traceability and full anonymity under the SIS and LWE problems in Nakanishi et al.'s model [145]. Towards our goal, we design a scalable lattice-based key-evolving mechanism, which allows users to periodically update their secret keys and to efficiently prove in zero-knowledge that the key-evolution process is done correctly. To realize this essential building block, the Bonsai-tree struc-

ture by Cash et al. [61] is utilized to handle the key-evolution process, and the Langlois et al.’s technique [102] is developed to design its supporting zero-knowledge protocol. When compared with all known group signatures from lattice assumptions (that are *not* forward-secure), our scheme suffers from a very fair overhead: the signature size and key size are at most $\mathcal{O}(\log N)$ with N being the total number of group users; and at most $\mathcal{O}(\log^3 T)$ with T being the total number of time periods.

Organization. In Section 5.1, we sketch out the techniques in constructing our FSGS scheme. Then we recall the model of forward-secure group signatures formalized by Nakanishi et al. [145] in Section 5.2. Finally, our scheme is described and analyzed in Section 5.3.

5.1 Overview of Our Techniques

It is known that designing secure group signatures typically requires a combination of digital signatures, encryption schemes and zero-knowledge protocols. Let us first consider an ordinary group signature scheme similar to the template proposed by Bellare et al. [20]. In the scheme, each user is assigned an ℓ -bit string id as identity, where $\ell = \lceil \log N \rceil$. The user’s signing key is a signature on his identifier id . Specifically, we let the signature be a short vector $\mathbf{v}_{\text{id}}$ satisfying $\mathbf{A}_{\text{id}} \cdot \mathbf{v}_{\text{id}} = \mathbf{u} \bmod q$ for some public vector $\mathbf{u}$. When signing a message, the user first encrypts his identity id to a ciphertext $\mathbf{c}$ and proves that he possesses a valid signature on his identity that is also correctly encrypted to $\mathbf{c}$. To realize forward-secure group signatures, we further need a mechanism to update the group signing key periodically and a zero-knowledge protocol to prove that the key-update procedure is done honestly.

Inspired by the HIBE-like key-update technique from Nakanishi et al. [145] and Libert and Yung [114], which in turn follows from [27, 35, 59], we exploit the hierarchical structure of the Bonsai tree [61] to enable periodical key-update. To our best knowledge, this is the only HIBE scheme from lattices in the standard model with

supporting zero-knowledge (ZK) proofs. Let $T = 2^d$ be the total number of time periods. To enable key-update, each user id is associated with a subtree of depth d , where the leaves of the tree correspond to successive time periods in the obvious way. Let the subtree be identified by matrices $\mathbf{A}_{\text{id}}, \mathbf{A}_{\ell+1}^0, \mathbf{A}_{\ell+1}^1, \dots, \mathbf{A}_{\ell+d}^0, \mathbf{A}_{\ell+d}^1$ and $z = \text{Bin}(t)$ be the binary representation of t . In order to show the key-evolution is done correctly, we observe that it suffices to show possession of a Bonsai signature $\mathbf{v}_{\text{id}\|z}$ satisfying $[\mathbf{A}_{\text{id}} \mid \mathbf{A}_{\ell+1}^{z[1]} \mid \dots \mid \mathbf{A}_{\ell+d}^{z[d]}] \cdot \mathbf{v}_{\text{id}\|z} = \mathbf{u} \bmod q$. However, proving knowledge of the Bonsai signature departs from the protocol presented by Langlois et al. [102]. First, the matrix $\mathbf{A}_{\text{id}}$ should be kept secret while in [102] it is a public matrix. Second, the binary string z is public in our case while it is secret in [102]. However, by analyzing the above equation, it actually reduces to prove knowledge of short vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ and a binary string id such that

$$\mathbf{A}_{\text{id}} \cdot \mathbf{w}_1 + \mathbf{A}'' \cdot \mathbf{w}_2 = \mathbf{u} \bmod q,$$

with $\mathbf{v}_{\text{id}\|z} = (\mathbf{w}_1 \parallel \mathbf{w}_2)$ for some public matrix $\mathbf{A}''$. To prove knowledge of $\mathbf{w}_2$, we can employ the standard decomposition, extension, and permutation techniques by Ling et al. [116]¹. To prove knowledge of $\mathbf{w}_1$ and id , it indeed depends on how we instantiate our ordinary group signature scheme. For simplicity, we use the Bonsai tree signature scheme [61] as well. Therefore, by developing the Langlois et al.'s technique [102], we manage to obtain the desired ZK proof. However, we remark that Boyen signature scheme [34] is another possible candidate. In fact, Ling et al. [117] proposed a ZK protocol for proving a valid message-signature pair of the Boyen signature scheme, which is also compatible with [116].

As discussed so far, we have described the ZK protocol showing possession of correctly updated signing key $\mathbf{v}_{\text{id}\|\text{Bin}(t)}$, the main difficulty in achieving our forward-secure group signature scheme. The next question is then how should the user derive

¹However, in this thesis, we employ the refined permuting technique described in Section 4.3.1.

$\mathbf{v}_{\text{id} \parallel \text{Bin}(t)}$ for all possible t using his group signing key $\mathbf{v}_{\text{id}}$. To this end, we make a minor but significant change to the group signing key. Observe that in the Bonsai tree structure, once a trapdoor matrix $\mathbf{S}_{\text{id}}$ satisfying $\mathbf{A}_{\text{id}} \cdot \mathbf{S}_{\text{id}} = \mathbf{0} \bmod q$ is known, the user id is able to generate $\mathbf{v}_{\text{id} \parallel \text{Bin}(t)}$ for all possible t . Therefore, we modify the user's signing key to be $\mathbf{S}_{\text{id}}$ instead of $\mathbf{v}_{\text{id}}$.

Obviously, user id should not hold $\mathbf{S}_{\text{id}}$ for the whole period. Since otherwise, the adversary can easily generate signatures on previous time periods once $\mathbf{S}_{\text{id}}$ is exposed to him, thus violating the forward-security. One possible way is to generate $\mathbf{v}_{\text{id} \parallel \text{Bin}(t)}$ for all possible t and then delete all previous ones when entering a new period. However, this will incur linear dependency on T , which is undesirable for efficiency purpose. To achieve logarithmic overhead, we should employ the structure of the Bonsai tree. Let $\text{Nodes}_{(t \rightarrow T-1)}$ be the set of nodes such that it contains exactly one ancestor of each leaf (or the leaf itself) between t and $T - 1$ and make sure this set contains at most $\log T$ nodes². Now the signing key of user id at time t contains trapdoor matrices $\mathbf{S}_{\text{id} \parallel z}$ for all $z \in \text{Nodes}_{(t \rightarrow T-1)}$. In this way, the user can still generate a Bonsai signature $\mathbf{v}_{\text{id} \parallel \text{Bin}(t)}$ by employing the trapdoor $\mathbf{S}_{\text{id} \parallel z}$ where z is an ancestor of $\text{Bin}(t)$. What is more, it is verifiable that for each $z' \in \text{Nodes}_{(t+1 \rightarrow T-1)}$, there exists a unique ancestor $z \in \text{Nodes}_{(t \rightarrow T-1)}$. This property enables update of the signing key from time period t to $t + 1$, thanks to the basis delegation algorithm of the Bonsai tree structure.

The last thing to sort out is to identify the underlying encryption scheme. To achieve full anonymity, we should have a CCA-secure public key encryption scheme. To this end, we apply the CHK transform [60] to the identity-based encryption scheme by Gentry et al. [81]. For the obtained encryption scheme, there exists zero-knowledge protocol, e.g., [117], for proving knowledge of the plaintext that is also compatible to our setting.

To summarize, we obtain our forward-secure group signature scheme from lattices by combining several blocks from different works in a nontrivial way so that they

²This set can be determined by the Nodeselect algorithm presented by Libert and Yung [114].

perform well with the existing NIZK proofs. Our scheme satisfies full anonymity due to the CCA-security of the underlying encryption scheme and the statistical zero-knowledge of the underlying zero-knowledge protocol, and achieves forward-secure traceability due to the security of the Bonsai tree structure [61]. We believe that, our construction - while not being truly novel - would certainly help to enrich the field of lattice-based group signatures.

5.2 Forward-Secure Group Signatures

In this section, we review the model and security requirements of forward-secure group signature (FSGS), as suggested by Nakanishi et al. [145]. An FSGS scheme contains two authorities: a group manager (GM) that manages the group and a tracing manager (TM) that is able to discover the signer of any signature. The lifetime of an FSGS scheme is divided into T discrete periods $0, 1, \dots, T - 1$. At each time period t , the signing key of group member i is denoted as $\mathbf{usk}_t[i]$. To enable forward-security, after each update, the old key is assumed to be discarded promptly.

5.2.1 Syntax of Forward-Secure Group Signatures

An FSGS scheme contains the following polynomial-time algorithms.

KeyGen: This randomized key generation algorithm takes as input a tuple (λ, T, N) , where λ is the security parameter, T is the total number of time periods, and N is the maximum number of group members. It then returns $\mathbf{gpk}$, $\mathbf{msk}$, $\mathbf{mosk}$ and $\mathbf{usk}_0$, where $\mathbf{gpk}$ is the group public key, $\mathbf{msk}$ is the secret key of GM, $\mathbf{mosk}$ is the secret key of TM, and $\mathbf{usk}_0$ is an array of the initial N secret signing keys $\{\mathbf{usk}_0[0], \mathbf{usk}_0[1], \dots, \mathbf{usk}_0[N - 1]\}$, with $\mathbf{usk}_0[i]$ being the initial secret signing key of user i .

KeyUpdate: This randomized algorithm, on inputs $\mathbf{gpk}$, $\mathbf{usk}_t[i]$, i , and $t + 1$, with

$\mathbf{usk}_t[i]$ being the secret key of user i at time t , outputs the secret key $\mathbf{usk}_{t+1}[i]$ of user i at time $t + 1$.

Sign: On inputs $\mathbf{gpk}$, $\mathbf{usk}_t[i]$, user i , time period t , and message M , this randomized algorithm returns a signature Σ on message M .

Verify: This deterministic algorithm takes as inputs $\mathbf{gpk}$, time period t , message M and signature Σ , and outputs 1/0 indicating whether the signature is valid.

Open: On inputs $\mathbf{gpk}$, $\mathbf{mosk}$, t , M and Σ , this deterministic algorithm returns an index i or $\perp$.

5.2.2 Notions of Correctness and Security Requirements

As put forward by Nakanishi et al. [145], a forward-secure group signature scheme should satisfy *correctness*, *forward-secure traceability*, and *full anonymity*.

Correctness requires that the FSGS scheme must satisfy the following requirement: For all λ, T, N , $(\mathbf{gpk}, \mathbf{msk}, \mathbf{mosk}, \mathbf{usk}_0) \leftarrow \text{KeyGen}(\lambda, T, N)$, all $i \in \{0, 1, \dots, N - 1\}$, all $M \in \{0, 1\}^*$, all $\mathbf{usk}_t[i] \leftarrow \text{KeyUpdate}(\mathbf{gpk}, \mathbf{usk}_{t-1}[i], t)$ for all $t \in \{0, 1, \dots, T - 1\}$, the following conditions hold:

$$\begin{cases} \text{Verify}(\mathbf{gpk}, t, M, \text{Sign}(\mathbf{gpk}, \mathbf{usk}_t[i], t, M)) = 1, \\ \text{Open}(\mathbf{gpk}, \mathbf{mosk}, t, M, \text{Sign}(\mathbf{gpk}, \mathbf{usk}_t[i], t, M)) = i. \end{cases}$$

Forward-Secure Traceability demands that any PPT adversary, even if it can corrupt the tracing manager and some (or all) group members, is not able to create a valid signature (i) that is traced to a non-corrupted user or (ii) that is traced to a corrupted user, but the signature is signed at time period preceding the secret key exposure of this corrupted user. Note that (i) captures the standard traceability requirement as in [20] while (ii) deals with the new requirement in the context of forward-security. Details are modelled in the experiment $\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Trace}}(\lambda, T, N)$ described below. In the

experiment, the adversary can adaptively choose which user to corrupt, when to corrupt, when to halt, and when to output its forgery. Furthermore, it is allowed to query signature on any message of a member i through the signing oracle $\text{Sign}(\mathbf{usk}_t[\cdot], \cdot)$ if $i \notin \text{CU}$ at time period t .

Definition 5.2.1 (Forward-Secure Traceability). For any security parameter λ , the maximum number of group members N , the maximum number of time periods T , and for any PPT algorithm $\mathcal{A}$, define the experiment game $\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Trace}}(\lambda, T, N)$ as follows.

Experiment $\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Trace}}(\lambda, T, N)$

$(\text{gpk}, \text{msk}, \text{mosk}, \mathbf{usk}_0) \leftarrow \text{KeyGen}(\lambda, T, N)$.

$\text{st} \leftarrow (\text{gpk}, \text{mosk})$ $\text{CU} \leftarrow \emptyset$, $t := 0$, $\text{Cont} \leftarrow \text{True}$, $\text{stop} = 0$, $K \leftarrow \epsilon$.

For $(t = 0, t \leq T - 1, t++)$

While($\text{Cont} = \text{True}$) do

$(\text{Cont}, \text{st}, j) \leftarrow \mathcal{A}^{\text{Sign}(\mathbf{usk}_t[\cdot], \cdot)}(\text{choose1}, \text{st}, K)$,

if $\text{Cont} = \text{True}$, then $\text{CU} \leftarrow \text{CU} \cup \{j\}$, $K \leftarrow K \cup \{\mathbf{usk}_t[j], t\}$, endif

Endwhile

$\text{stop} \leftarrow \mathcal{A}(\text{choose2}, \text{st}, t, K)$,

if $\text{stop} = 1$, break; else $\text{Cont} = \text{True}$.

Endfor

$(t^*, M^*, \Sigma^*) \leftarrow \mathcal{A}(\text{guess}, \text{st}, t, K)$,

Output 1 if the three requirements below hold:

- $\text{Verify}(\text{gpk}, t^*, M^*, \Sigma^*) = 1$,
- $\mathcal{A}$ did not obtain Σ^* by querying its signing oracle, and
- For $i^* \leftarrow \text{Open}(\text{gpk}, \text{mosk}, t^*, M^*, \Sigma^*)$
 - $i^* \notin \text{CU}$, or
 - $i^* \in \text{CU}$, but $\mathcal{A}$ only queried $\mathbf{usk}_t[i^*]$ for $t > t^*$.

Define the advantage $\mathbf{Adv}_{\text{FSGS}, \mathcal{A}}^{\text{Trace}}(\lambda, T, N)$ of the adversary $\mathcal{A}$ against forward-secure traceability of an FSGS scheme as $\Pr[\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Trace}}(\lambda, T, N) = 1]$. An FSGS scheme is said to be forward-secure traceable if the advantage of any PPT algorithm $\mathcal{A}$ is negligible in λ .

Full Anonymity demands that it is infeasible for any PPT algorithm to determine which of two signers of its choice signed the challenged message of its choice at time period t of its choice. Details of this requirement is modelled in the experiment $\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Anon}-b}(\lambda, T, N)$ presented below. In the experiment, the algorithm $\mathcal{A}$ is accessible to all group members' and group manager's secret keys, and can query the opening of any signature except for the challenged one.

Definition 5.2.2 (Full Anonymity). For any security parameter λ , the number of group members N , the number of time periods T , and for any PPT algorithm $\mathcal{A}$, define the experiment game $\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Anon}}(\lambda, T, N)$ for below.

Experiment $\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Anon}}(\lambda, T, N)$

$(\text{gpk}, \text{msk}, \text{mosk}, \text{usk}_0) \leftarrow \text{KeyGen}(\lambda, T, N).$

$(\text{st}, t, i_0, i_1, M) \leftarrow \mathcal{A}^{\text{Open}(\text{mosk}, \cdot)}(\text{choose}, \text{gpk}, \text{msk}, \text{usk}_0).$

$b \xleftarrow{\$} \{0, 1\}$, compute $\Sigma \leftarrow \text{Sign}(\text{gpk}, \text{usk}_t[i_b], t, M).$

$b' \leftarrow \mathcal{A}^{\text{Open}(\text{mosk}, \cdot)}(\text{guess}, \text{st}, \Sigma).$

Output 1 if $b = b'$ and 0 otherwise.

Define $\mathbf{Adv}_{\text{FSGS}, \mathcal{A}}^{\text{Anon}}(\lambda, T, N)$ of algorithm $\mathcal{A}$ against full anonymity of an FSGS scheme as $|\Pr[\mathbf{Exp}_{\text{FSGS}, \mathcal{A}}^{\text{Anon}}(\lambda, T, N) = 1] - 1/2|$. An FSGS scheme is said to satisfy full anonymity if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.

5.3 Forward-Secure Group Signatures from Lattices

In the description below, for a binary tree of depth k , we identify each node at depth j with a binary vector z of length j such that $z[1]$ to $z[j]$ are ordered from the top to the bottom and a 0 and a 1 indicate the left and right branch respectively in the order of traversal. Let B be a positive integer. For a non-negative integer b smaller than B , denote $\text{Bin}(b)$ as the binary representation of b with length $\lceil \log B \rceil$.

In our forward-secure group signature scheme, the lifetime of the scheme is divided into T discrete periods $0, 1, \dots, T-1$. For simplicity, let $T = 2^d$ for some positive integer d . Following previous works [35, 114], time is arranged as the leaves of a binary tree of depth d in chronological order. More precisely, each time period t is associated with leave $\text{Bin}(t)$.

Following [35], for $j = 1, \dots, d+1$, $t \in \{0, 1, \dots, T-1\}$, a time period's "right sibling at depth j " is defined to be

$$\text{sibling}(j, t) = \begin{cases} (\text{Bin}(t)[1], \dots, \text{Bin}(t)[j-1], 1)^\top & \text{if } j \leq d \text{ and } \text{Bin}(t)[j] = 0, \\ \perp & \text{if } j \leq d \text{ and } \text{Bin}(t)[j] = 1, \\ \text{Bin}(t) & \text{if } j = d+1. \end{cases}$$

Let the node set $\text{Nodes}_{(t \rightarrow T-1)}$ be $\{\text{sibling}(1, t), \dots, \text{sibling}(d+1, t)\}$. For any $t' > t$, one can check that for any non- $\perp$ $z' \in \text{Nodes}_{(t' \rightarrow T-1)}$, there is a unique $z \in \text{Nodes}_{(t \rightarrow T-1)}$ so that z is an ancestor of z' .

For example, let $T = 8$, then time 0 is corresponding to leave $(0, 0, 0)^\top$, and time 4 is corresponding to leave $(1, 0, 0)^\top$. $\text{Nodes}_{(0 \rightarrow 7)} = \{(1)^\top, (0, 1)^\top, (0, 0, 1)^\top, (0, 0, 0)^\top\}$ and $\text{Nodes}_{(4 \rightarrow 7)} = \{\perp, (1, 1)^\top, (1, 0, 1)^\top, (1, 0, 0)^\top\}$. For any non- $\perp$ $z' \in \text{Nodes}_{(4 \rightarrow 7)}$, $(1)^\top$ is an ancestor of z' .

5.3.1 Our Lattice-Based FSGS Scheme

Our scheme works in the Nakanishi et al.'s (static) model [145]. Let the total number of time periods be $T = 2^d$ and the total number of group members be $N = 2^\ell$. The group public key consists of (i) a Bonsai tree of depth $\ell + d$ specified by a matrix $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1^0 | \mathbf{A}_1^1 \cdots | \mathbf{A}_{\ell+d}^0 | \mathbf{A}_{\ell+d}^1] \in \mathbb{Z}_q^{n \times (2\ell+2d+1)m}$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, which is used to generate group signing keys; (ii) A public matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ of the IBE scheme by Gentry et al. [81], which is used to encrypt user identities when signing messages. The secret key of the group manager is a basis of the lattice $\Lambda^\perp(\mathbf{A}_0)$ while the secret key of the tracing manager is a basis of the lattice $\Lambda^\perp(\mathbf{B})$.

Each user $\text{id} \in \{0, 1\}^\ell$ is assigned a node id. To enable periodical key-update, each user id is associated with a subtree of depth d . In our scheme, all users are assumed to be valid group members from time 0 to $T - 1$. Let z be a binary string of length d_z , where $d_z \leq d$. Define $\mathbf{A}_{\text{id}||z} = [\mathbf{A}_0 | \mathbf{A}_1^{\text{id}[1]} | \cdots | \mathbf{A}_\ell^{\text{id}[\ell]} | \mathbf{A}_{\ell+1}^{z[1]} | \cdots | \mathbf{A}_{\ell+d_z}^{z[d_z]}] \in \mathbb{Z}_q^{n \times (\ell+d_z+1)m}$. Specifically, the initial group signing key of user id at time t is the set $\{\mathbf{S}_{\text{id}||z}, z \in \text{Nodes}_{(t \rightarrow T-1)}\}$, which satisfies $\mathbf{A}_{\text{id}||z} \cdot \mathbf{S}_{\text{id}||z} = \mathbf{0} \bmod q$. Thanks to the basis delegation technique [61], users are able to compute the trapdoor matrices for all the descendent of nodes that is in the set $\text{Nodes}_{(t \rightarrow T-1)}$ and hence are able to derive all subsequent signing keys. We remark that since we do not need to delegate basis for leave nodes, we generate short vectors $\mathbf{v}_{\text{id}||\text{Bin}(t)}$ instead of $\mathbf{S}_{\text{id}||\text{Bin}(t)}$ in the group signing key for all t .

Once receiving the group signing key, every group member can issue signatures in the name of the group. When signing a message at time t , user id first produces a one-time signature key pair (ovk, osk) , and then encrypts his id to a ciphertext $\mathbf{c}$ using the IBE scheme with respect to “identity” ovk . Next, he proves in a zero-knowledge manner that: (i) he is a valid group user; (ii) he has done key-evolution honestly; (iii) the ciphertext $\mathbf{c}$ correctly encrypts id. To prove that facts (i) and (ii) hold, it is sufficient to prove possession of a short vector $\mathbf{v}_{\text{id}||\text{Bin}(t)}$ so that $\mathbf{A}_{\text{id}||\text{Bin}(t)} \cdot \mathbf{v}_{\text{id}||\text{Bin}(t)} =$

$\mathbf{u} \bmod q$. The protocol is developed from Langlois et al.'s technique [102] and Ling et al.'s technique [116]. It is run $\kappa = \omega(\log n)$ times to obtain negligible soundness error, and is made non-interactive through the Fiat-Shamir transform [75] as a tuple Π . Finally, the user generates a one-time signature sig on the pair $(\mathbf{c}, \Pi)$, and outputs the group signature consisting of $(\text{ovk}, \mathbf{c}, \Pi, \text{sig})$.

To verify a group signature, we simply checks the validity of sig and Π . In case of need, the tracing manager decrypts the ciphertext with respect to the “identity” ovk using his secret key, thus identifying the signer. The scheme is described in detail below.

KeyGen(λ, T, N): On inputs security parameter λ , total number of time periods $T = 2^d$ for some $d \in \mathbb{Z}_+$ and maximum number of group members $N = 2^\ell$ for some $\ell \in \mathbb{Z}_+$, this algorithm performs as follows:

1. Choose $n = \mathcal{O}(\lambda)$, $q = \tilde{\mathcal{O}}(n^{2.5})$, $m = \mathcal{O}(n \log q)$. Let $k = \ell + d$ and $\kappa = \omega(\log n)$.
2. Run **TrapGen**(n, m, q) as described in Lemma 2.3.1 to obtain $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S}_0 \in \mathbb{Z}^{m \times m}$.
3. Sample $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$, and $\mathbf{A}_i^b \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for all $i \in [k]$ and $b \in \{0, 1\}$.
4. Choose a one-time signature scheme $\mathcal{OTS} = (\text{OGen}, \text{OSign}, \text{Over})$, and a commitment scheme **COM** from [94] with statistical hiding and computational binding that will be employed in our argument system.
5. Let $\mathcal{H}_0 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times \ell}$ and $\mathcal{H}_1 : \{0, 1\}^* \rightarrow \{1, 2, 3\}^\kappa$ be collision-resistant hash functions.
6. Choose Gaussian parameter s_i to be $\mathcal{O}(\sqrt{nk \log q})^{i-\ell+1} \cdot \omega(\sqrt{\log n})^{i-\ell+1}$ that is used to generate short bases or sample short vectors at level i for $i \in \{\ell, \ell + 1, \dots, k\}$.

7. Let $\beta = \lceil s_k \cdot \log n \rceil$. Choose integer $B = \tilde{\mathcal{O}}(\sqrt{n})$ and χ to be a distribution over $\mathbb{Z}$ that is B -bounded.
8. Generate a master key pair $(\mathbf{B}, \mathbf{S}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}^{m \times m}$ for the IBE scheme suggested by Gentry et al. [81] via the $\text{TrapGen}(n, m, q)$ algorithm.
9. For user $i \in \{0, 1, \dots, N-1\}$, let $\text{id} = \text{Bin}(i) \in \{0, 1\}^\ell$. Let node id be the identifier of user i . Determine the node set $\text{Nodes}_{(0 \rightarrow T-1)}$.

For $z \in \text{Nodes}_{(0 \rightarrow T-1)}$, if $z = \perp$, set $\mathbf{usk}_0[i][z] = \perp$. Otherwise denote d_z as the length of z with $d_z \leq d$, compute the matrix

$$\mathbf{A}_{\text{id}\|z} = [\mathbf{A}_0 | \mathbf{A}_1^{\text{id}[1]} | \dots | \mathbf{A}_\ell^{\text{id}[\ell]} | \mathbf{A}_{\ell+1}^{z[1]} | \dots | \mathbf{A}_{\ell+d_z}^{z[d_z]}] \in \mathbb{Z}_q^{n \times (\ell+d_z+1)m}.$$

and proceed as follows:

- If z is of length d , i.e., $d_z = d$, it computes a vector $\mathbf{v}_{\text{id}\|z} \in \Lambda^{\mathbf{u}}(\mathbf{A}_{\text{id}\|z})$ via

$$\mathbf{v}_{\text{id}\|z} \leftarrow \text{SampleD}(\text{ExtBasis}(\mathbf{S}_0, \mathbf{A}_{\text{id}\|z}), \mathbf{u}, s_k).$$

Set $\mathbf{usk}_0[i][z] = \mathbf{v}_{\text{id}\|z}$.

- If z is of length less than d , i.e., $1 \leq d_z < d$, it computes a matrix $\mathbf{S}_{\text{id}\|z} \in \mathbb{Z}^{(\ell+d_z+1)m \times (\ell+d_z+1)m}$ via

$$\mathbf{S}_{\text{id}\|z} \leftarrow \text{RandBasis}(\text{ExtBasis}(\mathbf{S}_0, \mathbf{A}_{\text{id}\|z}), s_{\ell+d_z}).$$

Set $\mathbf{usk}_0[i][z] = \mathbf{S}_{\text{id}\|z}$.

Let $\mathbf{usk}_0[i] = \{\mathbf{usk}_0[i][z], z \in \text{Nodes}_{(0 \rightarrow T-1)}\}$ be the initial secret key of user i .

Let public parameter be $\mathbf{pp}$, group public key be $\mathbf{gpk}$, secret key of GM be $\mathbf{msk}$, secret key of TM be $\mathbf{mosk}$ and initial secret key be $\mathbf{usk}_0$, which are defined as

follows:

$$\mathbf{pp} = \{n, q, m, \ell, d, k, \kappa, \mathcal{OTS}, \text{COM}, \mathcal{H}_0, \mathcal{H}_1, s_\ell, \dots, s_k, \beta, B\},$$

$$\mathbf{gpk} = \{\mathbf{pp}, \mathbf{A}_0, \mathbf{A}_1^0, \mathbf{A}_1^1, \dots, \mathbf{A}_k^0, \mathbf{A}_k^1, \mathbf{u}, \mathbf{B}\},$$

$$\mathbf{msk} = \mathbf{S}_0, \quad \mathbf{mosk} = \mathbf{S},$$

$$\mathbf{usk}_0 = \{\mathbf{usk}_0[0], \mathbf{usk}_0[1], \dots, \mathbf{usk}_0[N-1]\}.$$

KeyUpdate($\mathbf{gpk}, \mathbf{usk}_t[i], i, t+1$): Compute the identifier of user i as $\text{id} = \text{Bin}(i)$, parse $\mathbf{usk}_t[i] = \{\mathbf{usk}_t[i][z], z \in \text{Nodes}_{(t \rightarrow T-1)}\}$, and determine the node set $\text{Nodes}_{(t+1 \rightarrow T-1)}$.

For $z' \in \text{Nodes}_{(t+1 \rightarrow T-1)}$, if $z' = \perp$, set $\mathbf{usk}_{t+1}[i][z'] = \perp$. Otherwise, there exists exactly one $z \in \text{Nodes}_{(t \rightarrow T-1)}$ as its prefix, i.e., $z' = z||y$ for some suffix y . Consider the following two cases.

1. If $z' = z$, let $\mathbf{usk}_{t+1}[i][z'] = \mathbf{usk}_t[i][z]$.
2. If $z' = z||y$ for some non-empty string y , then $\mathbf{usk}_t[i][z] = \mathbf{S}_{\text{id}||z}$. Consider the following two subcases.
 - If z' is of length d , run

$$\mathbf{v}_{\text{id}||z'} \leftarrow \text{SampleD}(\text{ExtBasis}(\mathbf{S}_{\text{id}||z}, \mathbf{A}_{\text{id}||z'}), \mathbf{u}, s_k),$$

and set $\mathbf{usk}_{t+1}[i][z'] = \mathbf{v}_{\text{id}||z'}$.

- If z' is of length less than d , run

$$\mathbf{S}_{\text{id}||z'} \leftarrow \text{RandBasis}(\text{ExtBasis}(\mathbf{S}_{\text{id}||z}, \mathbf{A}_{\text{id}||z'}), s_{\ell+d_{z'}}),$$

and set $\mathbf{usk}_{t+1}[i][z'] = \mathbf{S}_{\text{id}||z'}$.

Output the updated key $\mathbf{usk}_{t+1}[i] = \{\mathbf{usk}_{t+1}[i][z'], z' \in \text{Nodes}_{(t+1 \rightarrow T-1)}\}$.

$\text{Sign}(\text{gpk}, \mathbf{usk}_t[i], i, t, M)$: Compute the identifier $\text{id} = \text{Bin}(i)$. By the structure of the node set $\text{Nodes}_{(t \rightarrow T-1)}$, there exists some $z \in \text{Nodes}_{(t \rightarrow T-1)}$ such that $z = \text{Bin}(t)$ is of length d and $\mathbf{usk}_t[i][z] = \mathbf{v}_{\text{id}\|z}$.

To sign a message $M \in \{0, 1\}^*$, the signer then does the following.

1. First, generate a key pair $(\text{ovk}, \text{osk}) \leftarrow \text{OGen}(n)$ for the one-time signature scheme, and then encrypt id with respect to “identity” ovk . Concretely, let $\mathbf{G} = \mathcal{H}_0(\text{ovk}) \in \mathbb{Z}_q^{n \times \ell}$. Sample $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e}_1 \leftarrow \chi^m$, $\mathbf{e}_2 \leftarrow \chi^\ell$, and compute ciphertext $(\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$ as

$$(\mathbf{c}_1 = \mathbf{B}^\top \cdot \mathbf{s} + \mathbf{e}_1, \quad \mathbf{c}_2 = \mathbf{G}^\top \cdot \mathbf{s} + \mathbf{e}_2 + \lfloor \frac{q}{2} \rfloor \cdot \text{id}). \quad (5.1)$$

2. Second, compute the matrix $\mathbf{A}_{\text{id}\|z}$ and generate a NIZKAoK Π to demonstrate the knowledge of a tuple

$$\xi = (\text{id}, \mathbf{s}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{v}_{\text{id}\|z}) \quad (5.2)$$

satisfying

$$(a) \quad \mathbf{A}_{\text{id}\|z} \cdot \mathbf{v}_{\text{id}\|z} = \mathbf{u} \pmod{q}, \text{ and } \|\mathbf{v}_{\text{id}\|z}\|_\infty \leq \beta.$$

$$(b) \quad \text{Equations in (5.1) hold with } \|\mathbf{s}\|_\infty \leq B, \|\mathbf{e}_1\|_\infty \leq B \text{ and } \|\mathbf{e}_2\|_\infty \leq B.$$

Specifically, we run the argument system described in Section 5.3.3 with public input

$$\gamma = (\mathbf{A}_0, \mathbf{A}_1^0, \mathbf{A}_1^1, \dots, \mathbf{A}_k^0, \mathbf{A}_k^1, \mathbf{u}, \mathbf{B}, \mathbf{G}, \mathbf{c}_1, \mathbf{c}_2, t)$$

and witness tuple ξ as above. Since the protocol has soundness error $2/3$, it is run $\kappa = \omega(\log n)$ times, thus obtaining soundness error $(2/3)^\kappa$, which

is negligible. Furthermore, the protocol is an interactive one. We thus employ the Fiat-Shamir heuristic [75] to transform it to be non-interactive. Finally, the result is a triple $\Pi = ((\text{CMT}_i)_{i=1}^\kappa, \text{CH}, (\text{RSP}_i)_{i=1}^\kappa)$ with $\text{CH} = \mathcal{H}_1(M, (\text{CMT}_i)_{i=1}^\kappa, \mathbf{c}_1, \mathbf{c}_2, t)$.

3. Third, run the algorithm $\text{OSign}(\text{osk}; \mathbf{c}_1, \mathbf{c}_2, \Pi)$ to obtain a one time signature sig on the tuple $(\mathbf{c}_1, \mathbf{c}_2, \Pi)$. Finally, the signature on the message M is defined to be $\Sigma = (\text{ovk}, \mathbf{c}_1, \mathbf{c}_2, \Pi, \text{sig})$.

$\text{Verify}(\text{gpk}, t, M, \Sigma)$: This algorithm proceeds as described below:

1. Parse Σ as $\Sigma = (\text{ovk}, \mathbf{c}_1, \mathbf{c}_2, \Pi, \text{sig})$. If $\text{Over}(\text{ovk}; \text{sig}; \mathbf{c}_1, \mathbf{c}_2, \Pi) = 0$, then output 0.
2. Parse Π as $\Pi = ((\text{CMT}_i)_{i=1}^\kappa, (\text{Ch}_1, \dots, \text{Ch}_\kappa), (\text{RSP}_i)_{i=1}^\kappa)$.
If $(\text{Ch}_1, \dots, \text{Ch}_\kappa) \neq \mathcal{H}_1(M, (\text{CMT}_i)_{i=1}^\kappa, \mathbf{c}_1, \mathbf{c}_2, t)$, then output 0.
3. For $i \in [\kappa]$, run the verification step of the protocol from Section 5.3.3 to check the validity of RSP_i with respect to CMT_i and Ch_i . If any of them does not hold, then output 0.
4. Output 1.

$\text{Open}(\text{gpk}, \text{mosk}, t, M, \Sigma)$: If $\text{Verify}(\text{gpk}, t, M, \Sigma) = 0$, abort. Otherwise, let mosk be $\mathbf{S} \in \mathbb{Z}^{m \times m}$ and parse Σ as $\Sigma = (\text{ovk}, \mathbf{c}_1, \mathbf{c}_2, \Pi, \text{sig})$. Then it decrypts $(\mathbf{c}_1, \mathbf{c}_2)$ as follows:

1. Compute $\mathbf{G} = \mathcal{H}_0(\text{ovk}) = [\mathbf{g}_1 | \dots | \mathbf{g}_\ell] \in \mathbb{Z}^{n \times \ell}$ and use $\mathbf{S}$ to compute a small norm matrix $\mathbf{F}_{\text{ovk}} \in \mathbb{Z}^{m \times \ell}$ such that $\mathbf{B} \cdot \mathbf{F}_{\text{ovk}} = \mathbf{G} \bmod q$. This is done by computing $\mathbf{f}_i \leftarrow \text{SampleD}(\mathbf{B}, \mathbf{S}, \mathbf{g}_i, s_\ell)$ for all $i \in [\ell]$ and let $\mathbf{F}_{\text{ovk}} = [\mathbf{f}_1 | \dots | \mathbf{f}_\ell]$.

2. Use $\mathbf{F}_{\text{ovk}}$ to decrypt $(\mathbf{c}_1, \mathbf{c}_2)$ by computing

$$\text{id}' = \left\lfloor \frac{\mathbf{c}_2 - \mathbf{F}_{\text{ovk}}^\top \cdot \mathbf{c}_1}{\lfloor q/2 \rfloor} \right\rfloor \in \{0, 1\}^\ell.$$

3. Return $\text{id}' \in \{0, 1\}^\ell$.

Remark 5.3.1. Note that Song [171] and Libert and Yung [114] consider a mechanism that deals with time-limited group membership, where each user i is only a valid group member from time $t_{i,1}$ to $t_{i,2}$. In our construction, for simplicity, we only consider fixed group membership from time 0 to $T - 1$ as in the work by Nakanishi et al. [145]. However, our construction can directly cater to time-limited group membership as in [114, 171] without any extra overhead. This can be done by generating secret key $\{\mathbf{usk}_{(t_{i,1} \rightarrow t_{i,2})}[i][z], z \in \text{Nodes}_{(t_{i,1} \rightarrow t_{i,2})}\}$ for user i in the **KeyGen** algorithm, whose group membership is from $t_{i,1}$ to $t_{i,2}$.

5.3.2 Analysis of the Scheme

EFFICIENCY. Firstly, we examine the complexity of the scheme depicted in Section 5.3.1, with respect to parameters λ , $\ell = \log N$, and $d = \log T$. Recall $k = \ell + d$.

- The group public key $\mathbf{gpk}$ has bit-size $\tilde{O}(\lambda^2 \cdot k)$.
- The user secret key $\mathbf{usk}_t[i]$ has at most $d + 1$ trapdoor matrices, and has bit-size $\tilde{O}(\lambda^2 \cdot k^2 d)$.
- The size of signature Σ is mainly determined by that of Stern-type proof Π : $\tilde{O}(|\xi| \cdot \log q) \cdot \omega(\log \lambda)$, where $|\xi|$ is the bit-size of the witness-tuple ξ . Overall, the signature Σ has bit-size $\tilde{O}(\lambda \cdot k)$.

CORRECTNESS. We now analyze the correctness of our scheme, which relies on two facts: (i) the argument system employed is perfectly complete; (ii) the encryption

scheme obtained by transforming the IBE scheme in [81] via CHK transformation [60] is correct.

Concretely, when an honest user signs a message at time period t , he is able to show the knowledge of a valid tuple ξ of the form (5.2). Therefore, by the perfect completeness of our argument system, with probability 1, the resulting signature Π passes the verification procedure. Regarding the correctness of the **Open** algorithm, we have

$$\begin{aligned} \mathbf{c}_2 - \mathbf{F}_{\text{ovk}}^\top \cdot \mathbf{c}_1 &= \mathbf{G}^\top \cdot \mathbf{s} + \mathbf{e}_2 + \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{id} - \mathbf{F}_{\text{ovk}}^\top \cdot (\mathbf{B}^\top \cdot \mathbf{s} + \mathbf{e}_1) \\ &= \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{id} + \mathbf{e}_2 - \mathbf{F}_{\text{ovk}}^\top \cdot \mathbf{e}_1 \end{aligned}$$

where $\|\mathbf{e}_1\| \leq B$, $\|\mathbf{e}_2\|_\infty \leq B$, and $\|\mathbf{f}_i\|_\infty \leq \lceil s_\ell \cdot \log m \rceil = \tilde{\mathcal{O}}(\sqrt{n \cdot k})$. Recall that $q = \tilde{\mathcal{O}}(n^{2.5})$, $m = \mathcal{O}(n \log q)$ and $B = \tilde{\mathcal{O}}(\sqrt{n})$. Hence

$$\|\mathbf{e}_2 - \mathbf{F}_{\text{ovk}}^\top \cdot \mathbf{e}_1\|_\infty \leq B + m \cdot B \cdot \tilde{\mathcal{O}}(\sqrt{n \cdot k}) = \tilde{\mathcal{O}}(n^2) \leq \left\lceil \frac{q}{5} \right\rceil = \tilde{\mathcal{O}}(n^{2.5}).$$

With overwhelming probability, the **Open** algorithm will recover id and correctness of the **Open** algorithm holds.

SECURITY. Finally, we show that our scheme fulfills the security requirements defined in Section 5.2.2 in the following theorem.

Theorem 5.3.2. The forward-secure group signature described in Section 5.3.1 satisfies full anonymity and forward-secure traceability requirements in the random oracle model under the **LWE** and **SIS** assumptions.

The proof of Theorem 5.3.2 is established by Lemma 5.3.3 and Lemma 5.3.4.

Lemma 5.3.3. Suppose that one-time signature scheme $\mathcal{OTS}$ is strongly unforgeable. Then our forward-secure group signature scheme described in Section 5.3.1 is fully anonymous in the random oracle model assuming the hardness of the $\text{LWE}_{n,q,\chi}$ problem.

Proof. Denote $\mathcal{C}$ as the challenger and $\mathcal{A}$ as the adversary. Following [117], we prove this lemma using a series of computationally indistinguishable games. The initial game is the real game $\mathbf{Exp}_{\text{FSGS},\mathcal{A}}^{\text{Anon}}(\lambda, T, N)$ while the last game is such that the advantage of the adversary is 0.

Game 0: In this game, $\mathcal{C}$ runs the experiment $\mathbf{Exp}_{\text{FSGS},\mathcal{A}}^{\text{Anon}}(\lambda, T, N)$ faithfully. In the challenge phase, $\mathcal{A}$ outputs a message M^* together with two users $0 \leq i_0, i_1 \leq N-1$ for the targeted time t^* . $\mathcal{C}$ responds by sending back a challenged signature $\Sigma^* = (\text{ovk}^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \Pi^*, \text{sig}^*) \leftarrow \text{Sign}(\text{gpk}, \text{usk}_{t^*}[i_b], t^*, M^*)$ for a random bit $b \in \{0, 1\}$. The adversary then outputs a bit $b' \in \{0, 1\}$ and this game outputs 1 if $b' = b$ and 0 otherwise. In this experiment, $\mathcal{C}$ replies with all random strings for oracles queries of $\mathcal{H}_0, \mathcal{H}_1$.

Game 1: In this game, we modify Game 0 in two aspects: (i) In the very beginning of the experiment, we generate the pair $(\text{ovk}^*, \text{osk}^*)$ that is used in generating the challenged signature; (ii) For the signature opening queries, if $\mathcal{A}$ asks for a valid signature of the form $\Sigma = (\text{ovk}, \mathbf{c}_1, \mathbf{c}_2, \Pi, \text{sig})$ such that $\text{ovk} = \text{ovk}^*$, then $\mathcal{C}$ returns a random bit and aborts the experiment. Now we argue that $\mathcal{C}$ aborts with negligible probability and hence Game 0 and Game 1 are computationally close. Actually, before the challenged signature is sent to $\mathcal{A}$, ovk^* is independent of $\mathcal{A}$'s view, hence it is negligible that $\mathcal{A}$ queries a signature containing ovk^* . Furthermore, after the challenged signature is sent to $\mathcal{A}$, if $\mathcal{A}$ queries a *new* valid signature of the form $(\text{ovk}^*, \mathbf{c}_1, \mathbf{c}_2, \Pi, \text{sig})$, then $((\mathbf{c}_1, \mathbf{c}_2, \Pi), \text{sig})$ is a successful forgery of the $\mathcal{OTS}$ scheme, which violates the strong unforgeability of the $\mathcal{OTS}$ scheme. This proves that $\mathcal{C}$ aborts with negligible probability. From now on, we assume that $\mathcal{A}$ will not query any valid signature containing ovk^* .

Game 2: In this game, we change Game 1 in the following ways. First, instead of generating $\mathbf{B}$ using the TrapGen algorithm, we generate a uniformly random matrix $\mathbf{B}^*$ over $\mathbb{Z}_q^{n \times m}$. This change is indistinguishable to $\mathcal{A}$ since the matrix $\mathbf{B}$

is statistically close to uniform by Lemma 2.3.1. Second, we program the random oracle $\mathcal{H}_0$ as follows. For query of ovk^* , it generates a uniformly random matrix $\mathbf{G}^* \in \mathbb{Z}_q^{n \times \ell}$, let $\mathcal{H}_0(\text{ovk}^*) = \mathbf{G}^*$, and return $\mathbf{G}^*$ to $\mathcal{A}$. This change also makes no difference to $\mathcal{A}$ since the output of $\mathcal{H}_0$ is uniformly random. For query of $\text{ovk} \neq \text{ovk}^*$, we first sample $\mathbf{F}_{\text{ovk}} \leftarrow D_{\mathbb{Z}, s_\ell}^{m \times \ell}$, let $\mathcal{H}_0(\text{ovk}) = \mathbf{G} = \mathbf{B}^* \cdot \mathbf{F}_{\text{ovk}} \pmod q$, and return $\mathbf{G}$ to $\mathcal{A}$. We then keep a record of $(\text{ovk}, \mathbf{F}_{\text{ovk}}, \mathbf{G})$. $\mathbf{G}$ generated in this way is statistically close to uniform by Lemma 2.2.13. Hence, this change does not affect $\mathcal{A}$'s view significantly. When $\mathcal{A}$ queries the opening oracle a signature $(\text{ovk}, \mathbf{c}_1, \mathbf{c}_2, \Pi, \text{sig})$ with $\text{ovk} \neq \text{ovk}^*$, $\mathcal{C}$ can use the recorded $\mathbf{F}_{\text{ovk}}$ to decrypt the ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$. Hence $\mathcal{C}$ can answer all signature opening queries. It then follows that Game 2 and Game 1 are statistically indistinguishable.

Game 3: In this game, we modify Game 2 as follows. Instead of generating a real proof Π^* for the challenged signature, we generate a simulated one without using the witness by programming the random oracle $\mathcal{H}_1$. Since our argument system is statistical zero-knowledge, the view of adversary $\mathcal{A}$ is statistically close between Game 3 and Game 2.

Game 4: In this game, we change Game 3 as follows. Instead of computing

$$(\mathbf{c}_1^*, \mathbf{c}_2^*) = (\mathbf{B}^{*\top} \cdot \mathbf{s} + \mathbf{e}_1, \mathbf{G}^{*\top} \cdot \mathbf{s} + \mathbf{e}_2 + \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{Bin}(i_b)) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$$

in the challenged phase, where $\mathbf{s} \in \chi^n, \mathbf{e}_1 \in \chi^m, \mathbf{e}_2 \in \chi^\ell$, we let

$$(\mathbf{c}_1^*, \mathbf{c}_2^*) = (\mathbf{z}_1^*, \mathbf{z}_2^* + \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{Bin}(i_b)),$$

where $\mathbf{z}_1^*, \mathbf{z}_2^*$ are uniformly random vectors over $\mathbb{Z}_q^m$ and $\mathbb{Z}_q^\ell$. We claim that this modification is computationally indistinguishable to the adversary $\mathcal{A}$ assuming the hardness of the $\text{LWE}_{n,q,\chi}$ problem. Indeed, if we let $\mathbf{D} = [\mathbf{B}^* | \mathbf{G}^*] \in \mathbb{Z}_q^{n \times (m+\ell)}$, $\mathbf{e} = (\mathbf{e}_1 || \mathbf{e}_2) \in \chi^{m+\ell}$, and $\mathbf{z} = (\mathbf{z}_1^* || \mathbf{z}_2^*) \in \mathbb{Z}_q^{m+\ell}$, then to distinguish Game 3 and

Game 4 is to distinguish $(\mathbf{D}, \mathbf{D}^T \cdot \mathbf{s} + \mathbf{e})$ and $(\mathbf{D}, \mathbf{z})$. Recall that $\mathbf{B}^*$ and $\mathbf{G}^*$ are uniformly random matrices, so is $\mathbf{D}$.

Game 5: In this game, we slightly change Game 4 by substituting $(\mathbf{c}_1^*, \mathbf{c}_2^*)$ with a new independent and uniform tuple $(\mathbf{z}'_1, \mathbf{z}'_2)$. It is straightforward that Game 5 and Game 4 are statistically indistinguishable. Furthermore, the challenged signature in this game does not depend on the challenged bit b any more, and hence the advantage of $\mathcal{A}$ in this game is 0.

It then follows that $\mathbf{Adv}_{\text{FSGS}, \mathcal{A}}^{\text{Anon}}(\lambda, T, N)$ is negligible in λ because of the indistinguishability of the above games. This concludes the proof. $\square$

Lemma 5.3.4. Our forward-secure group signature scheme described in Section 5.3.1 is forward-secure traceable in the random oracle model assuming the hardness of the $\text{SIS}_{n, \overline{m}, q, 2\beta}^\infty$ problem, where $\overline{m} = (k + 1)m$.

Proof. Assume there is a PPT algorithm $\mathcal{A}$ who breaks the forward-secure traceability of our forward-secure group signature scheme with non-negligible probability, then we are able to build a new PPT algorithm $\mathcal{B}$ who solves a $\text{SIS}_{n, \overline{m}, q, 2\beta}^\infty$ instance with non-negligible probability.

Given a SIS instance $\mathbf{C} \in \mathbb{Z}_q^{n \times \overline{m}}$, the goal of $\mathcal{B}$ is to find a vector $\mathbf{v} \in \mathbb{Z}_q^{\overline{m}}$ so that $\mathbf{C} \cdot \mathbf{v} = \mathbf{0} \pmod q$ and $0 < \|\mathbf{v}\|_\infty \leq 2\beta$. Initially, $\mathcal{B}$ lets $t = 0$ and the set CU be empty. It then proceeds as follows.

- Parse $\mathbf{C}$ as $\mathbf{C} = [\mathbf{C}_0 | \mathbf{C}_1 | \cdots | \mathbf{C}_k]$ for $\mathbf{C}_j \in \mathbb{Z}_q^{n \times m}$, $j \in \{0, 1, \dots, k\}$. It then generates the remaining public parameters as in Section 5.3.1.
- Sample $\mathbf{z} = (\mathbf{z}_0 \| \mathbf{z}_1 \| \cdots \| \mathbf{z}_k) \in \mathbb{Z}^{\overline{m}}$, where each $\mathbf{z}_i$ is sampled from $D_{\mathbb{Z}^m, s_k}$. If $\|\mathbf{z}\|_\infty > \beta$, then $\mathcal{B}$ just repeats this step. Compute $\mathbf{u} = \mathbf{C} \cdot \mathbf{z} \pmod q$.
- Guess the targeted user $i^* \in \{0, 1, \dots, N - 1\}$ and targeted forgery time period $t^* \in \{0, 1, \dots, T - 1\}$ uniformly.

- Let $\text{id}^* = \text{Bin}(i^*)$ and $z^* = \text{Bin}(t^*)$. Define $\mathbf{A}_j^{\text{id}^*[j]}$ to be $\mathbf{C}_j$ for $j \in [\ell]$ and $\mathbf{A}_{\ell+j}^{z^*[j]}$ to be $\mathbf{C}_{\ell+j}$ for $j \in [d]$.
- Generate $\mathbf{A}_j^{1-\text{id}^*[j]}$ for $j \in [\ell]$ via $(\mathbf{A}_j^{1-\text{id}^*[j]}, \mathbf{S}_j) \leftarrow \text{TrapGen}(n, m, q)$ and $\mathbf{A}_{\ell+j}^{1-z^*[j]}$ for $j \in [d]$ via $(\mathbf{A}_{\ell+j}^{1-z^*[j]}, \mathbf{S}_{\ell+j}) \leftarrow \text{TrapGen}(n, m, q)$.
- Generate a master key pair $(\mathbf{B}, \mathbf{S})$ via $\text{TrapGen}(n, m, q)$.

$\mathcal{B}$ sets gpk, mosk honestly and sends them to $\mathcal{A}$. At the start of each time period $t \in \{0, 1, \dots, T-1\}$, $\mathcal{B}$ announces the beginning of t to $\mathcal{A}$. At current time period t , $\mathcal{B}$ responds to $\mathcal{A}$'s queries as follows.

- When $\mathcal{A}$ queries the random oracles $\mathcal{H}_0, \mathcal{H}_1$, $\mathcal{B}$ replies with uniformly random strings and keeps a record of the queries.
- When $\mathcal{A}$ queries the secret key of member i^* , if $i \in \text{CU}$ or $t \leq t^*$, $\mathcal{B}$ then aborts. Otherwise, for each node $z \in \text{Nodes}_{(t \rightarrow T-1)}$, it first computes the smallest index $d_{z,t}$ such that $1 \leq d_{z,t} \leq d$ and $z^*[d_{z,t}] \neq z[d_{z,t}]$. Then $\mathcal{B}$ computes $\text{usk}_t[i][z]$ via $\text{SampleD}(\text{ExtBasis}(\mathbf{S}_{\ell+d_{z,t}}, \mathbf{A}_{\text{id}^*[z]}), s_k)$ if z is of length d or via $\text{RandBasis}(\text{ExtBasis}(\mathbf{S}_{\ell+d_{z,t}}, \mathbf{A}_{\text{id}^*[z]}), s_{\ell+d_z})$ if z is of length $d_z < d$. Finally, $\mathcal{B}$ sets $\text{usk}_t[i]$ as in our construction and sends it to $\mathcal{A}$. Add i^* to the set CU.
- When $\mathcal{A}$ queries the secret key of member $i \neq i^*$, if $i \in \text{CU}$, $\mathcal{B}$ aborts. Otherwise, let $\text{id} = \text{Bin}(i)$ and ℓ_i be the smallest index such that $1 \leq \ell_i \leq \ell$ and $\text{id}[\ell_i] \neq \text{id}^*[\ell_i]$. Compute $\text{usk}_t[i][z]$ via $\text{SampleD}(\text{ExtBasis}(\mathbf{S}_{\ell_i}, \mathbf{A}_{\text{id}[z]}), \mathbf{u}, s_k)$ if z is of length d or via $\text{RandBasis}(\text{ExtBasis}(\mathbf{S}_{\ell_i}, \mathbf{A}_{\text{id}[z]}), s_{\ell+d_z})$ if z is of length $d_z < d$ for $z \in \text{Nodes}_{(t \rightarrow T-1)}$. Set $\text{usk}_t[i]$ as in our construction and send it to $\mathcal{A}$. Finally, add i to the set CU.
- When $\mathcal{A}$ queries a signature on a message for user i with $i \neq i^*$ and $i \notin \text{CU}$ at current time t , $\mathcal{B}$ responds as in our algorithm **Sign** using the corresponding witness. For user i with $i = i^*$ and $i \notin \text{CU}$ at current time t , $\mathcal{B}$ performs the

same as in our algorithm **Sign** except that it generates a simulated proof Π' by programming the hash oracle $\mathcal{H}_1$ and returns $\Sigma = (\text{ovk}, \mathbf{c}_1, \mathbf{c}_2, \Pi', \text{sig})$ to $\mathcal{A}$.

We claimed that $\mathcal{A}$ cannot distinguish whether it interacts with the real challenger or with $\mathcal{B}$. First, group public key **gpk** given to $\mathcal{A}$ is indistinguishable from the real one. This is because the output matrix $\mathbf{A}$ of the **TrapGen** algorithm is statistically close to uniform by Lemma 2.3.1 and $\mathbf{u}$ is statistically close to a uniform vector over $\mathbb{Z}_q^n$ by Lemma 2.2.13. Second, the secret signing key given to $\mathcal{A}$ is indistinguishable from the real one due to the fact that the outputs of **RandBasis** using two different bases are within statistical distance by Lemma 2.3.4. Third, the signature queries make no difference to the view of $\mathcal{A}$. This can be implied by the statistical zero-knowledge property of the underlying argument system.

When $\mathcal{A}$ halts and outputs a message M^* and a signature Σ^* at the targeted time period t' such that $\text{Verify}(\text{gpk}, t', M^*, \Sigma^*) = 1$ and Π^* is not obtained by making a signing query at M^* , check $t' = t^*$ holds or not. If not, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{A}$ outputs (t^*, M^*, Σ^*) . Parse Σ^* as

$$(\text{ovk}^*, \mathbf{c}_1^*, \mathbf{c}_2^*, (\text{CMT}_i^*)_{i=1}^\kappa, (\text{Ch}_i^*)_{i=1}^\kappa, (\text{RSP}_i^*)_{i=1}^\kappa, \text{sig}^*).$$

Run $\text{id}' \leftarrow \text{Open}(\text{gpk}, \text{mosk}, t^*, M^*, \Sigma^*)$. If $\text{id}' \neq \text{id}^*$, indicating that the guess of i^* fails, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ makes use of the forgery to solve the **SIS** problem as follows.

First, $\mathcal{A}$ must have queried $\mathcal{H}_1$ for the tuple $(M^*, (\text{CMT}_i^*)_{i=1}^\kappa, \mathbf{c}_1^*, \mathbf{c}_2^*, t^*)$, since the probability of guessing this value is at most $3^{-\kappa}$, which is negligible by our choice of κ . Let $(M^*, (\text{CMT}_i^*)_{i=1}^\kappa, \mathbf{c}_1^*, \mathbf{c}_2^*, t^*)$ be the h -th oracle query and $Q_{\mathcal{H}_1}$ be the total oracle queries $\mathcal{A}$ has made to $\mathcal{H}_1$. Next, $\mathcal{B}$ lets h be the targeted forking point and replays $\mathcal{A}$ polynomial-number times. For each new run, $\mathcal{B}$ starts with the same random tape and random input as in the original run. Further, for the first $h - 1$ queries of $\mathcal{H}_1$, $\mathcal{B}$ replies with the same random value as in the original run, but from h -th query on, $\mathcal{B}$

replies with fresh and independent value. Besides, for queries of $\mathcal{H}_0$, $\mathcal{B}$ always replies as in the original run.

Constructed in this way, $(M^*, (\text{CMT}_i^*)_{i=1}^\kappa, \mathbf{c}_1^*, \mathbf{c}_2^*, t^*)$ is always the h -th oracle query $\mathcal{A}$ made to $\mathcal{H}_1$. The improved forking lemma [47] implies that with probability $\geq \frac{1}{2}$, $\mathcal{B}$ obtains 3-fork containing the same tuple $(M^*, (\text{CMT}_i^*)_{i=1}^\kappa, \mathbf{c}_1^*, \mathbf{c}_2^*, t^*)$ with pairwise distinct hash values $\text{CH}_h^{(1)}, \text{CH}_h^{(2)}, \text{CH}_h^{(3)}$ and corresponding valid responses $\text{RSP}_h^{(1)}, \text{RSP}_h^{(2)}, \text{RSP}_h^{(3)}$. Through some basic probability calculation, we have $\{\text{CH}_{h,j}^{(1)}, \text{CH}_{h,j}^{(2)}, \text{CH}_{h,j}^{(3)}\} = \{1, 2, 3\}$ for some $j \in [\kappa]$ with probability $1 - (\frac{7}{9})^\kappa$. Hence, $\text{RSP}_{h,j}^{(1)}, \text{RSP}_{h,j}^{(2)}, \text{RSP}_{h,j}^{(3)}$ are 3 valid responses for all the challenges 1, 2, 3 with respect to the same commitment CMT_j^* . Due to the computational binding property of the COM scheme, $\mathcal{B}$ is able to extract the witness tuple

$$\xi^* = (\text{id}, \mathbf{s}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{v}_{\text{id}\|z})$$

such that $\|\mathbf{v}_{\text{id}\|z}\|_\infty \leq \beta$, $\|\mathbf{s}\|_\infty \leq B$, $\|\mathbf{e}_1\|_\infty \leq B$, $\|\mathbf{e}_2\|_\infty \leq B$ and

$$\begin{aligned} \mathbf{A}_{\text{id}\|z} \cdot \mathbf{v}_{\text{id}\|z} &= \mathbf{u} \pmod{q}, \\ (\mathbf{c}_1^* &= \mathbf{B}^T \cdot \mathbf{s} + \mathbf{e}_1, \mathbf{c}_2^* = (\mathbf{G}^*)^T \cdot \mathbf{s} + \mathbf{e}_2 + \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{id}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell, \end{aligned}$$

where $\mathbf{G}^* = \mathcal{H}_0(\text{ovk}^*)$. Conditioned on guessing correctly i^*, t^* , we have $\text{id} = \text{id}^*$ and $z = z^*$. Therefore, $\mathbf{A}_{\text{id}\|z} = \mathbf{C}$. Now we have $\mathbf{C} \cdot \mathbf{v}_{\text{id}\|z} = \mathbf{u} = \mathbf{C} \cdot \mathbf{z} \pmod{q}$. We claim that $\mathbf{v}_{\text{id}\|z} \neq \mathbf{z}$ except with negligible probability. This is because $\mathcal{A}$ either queried the secret key of user id at time after t^* or never queried the secret key at all (by successfully attacking forward-secure traceability), then $\mathbf{z}$ is not known to $\mathcal{A}$. Further, from the view of the adversary, $\mathbf{z}$ is from the distribution $D_{\Lambda^u(\mathbf{C}), s_k}$ and hence has large min-entropy, which are implied by Lemma 2.2.13. Therefore, $\mathbf{v}_{\text{id}\|z} \neq \mathbf{z}$ except with negligible probability. Hence $\mathbf{x} = \mathbf{v}_{\text{id}\|z} - \mathbf{z} \neq \mathbf{0}$ and $\|\mathbf{x}\|_\infty \leq 2\beta$. This implies that we solve an instance of the $\text{SIS}_{n, \overline{m}, q, 2\beta}^\infty$ problem featured by matrix $\mathbf{C}$ with non-negligible

probability and hence our scheme is forward-secure traceable. $\square$

5.3.3 The Main Zero-Knowledge Argument System

We now describe a statistical ZKAoK that will be used by the signer when generating group signatures. This protocol is developed from Stern-like techniques proposed by Ling et al. [116] and Langlois et al. [102].

Public input γ : $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$, $\mathbf{A}_j^b \in \mathbb{Z}_q^{n \times m}$ for $(b, j) \in \{0, 1\} \times [k]$, $\mathbf{u} \in \mathbb{Z}_q^n$, $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{G} \in \mathbb{Z}_q^{n \times \ell}$, $(\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$, $t \in \{0, 1, \dots, T-1\}$.

Secret input ξ : $\text{id} \in \{0, 1\}^\ell$, $\mathbf{s} \in \chi^n$, $\mathbf{e}_1 \in \chi^m$, $\mathbf{e}_2 \in \chi^\ell$, $\mathbf{v}_{\text{id}\|z} \in \mathbb{Z}^{(\ell+d+1)m}$ with $z = \text{Bin}(t)$.

Prover's goal:

$$\begin{cases} \mathbf{A}_{\text{id}\|z} \cdot \mathbf{v}_{\text{id}\|z} = \mathbf{u} \bmod q, \quad \|\mathbf{v}_{\text{id}\|z}\|_\infty \leq \beta; \\ \mathbf{c}_1 = \mathbf{B}^\top \cdot \mathbf{s} + \mathbf{e}_1 \bmod q, \quad \mathbf{c}_2 = \mathbf{G}^\top \cdot \mathbf{s} + \mathbf{e}_2 + \left\lfloor \frac{q}{2} \right\rfloor \cdot \text{id} \bmod q; \\ \|\mathbf{s}\|_\infty \leq B, \quad \|\mathbf{e}_1\|_\infty \leq B, \quad \|\mathbf{e}_2\|_\infty \leq B. \end{cases} \quad (5.3)$$

DECOMPOSING-UNIFYING. We first rearrange the above requirements. Let $\mathbf{A}' = [\mathbf{A} \mid \mathbf{A}_1^0 \mid \mathbf{A}_1^1 \mid \dots \mid \mathbf{A}_\ell^0 \mid \mathbf{A}_\ell^1] \in \mathbb{Z}_q^{(2\ell+1)m}$, $\mathbf{A}_{\text{id}} = [\mathbf{A}_0 \mid \mathbf{A}_1^{\text{id}[1]} \mid \dots \mid \mathbf{A}_\ell^{\text{id}[\ell]}] \in \mathbb{Z}_q^{(\ell+1)m}$, $\mathbf{A}'' = [\mathbf{A}_{\ell+1}^{z[1]} \mid \dots \mid \mathbf{A}_{\ell+d}^{z[d]}] \in \mathbb{Z}_q^{dm}$. Then $\mathbf{A}_{\text{id}\|z} = [\mathbf{A}_{\text{id}} \mid \mathbf{A}''] \in \mathbb{Z}_q^{(\ell+d+1)m}$. Let $\mathbf{v}_{\text{id}} = (\mathbf{v}_0 \parallel \mathbf{v}_1 \parallel \dots \parallel \mathbf{v}_\ell)$, $\mathbf{w}_2 = (\mathbf{v}_{\ell+1} \parallel \dots \parallel \mathbf{v}_{\ell+d})$ with each $\mathbf{v}_i \in \mathbb{Z}^m$. Then $\mathbf{v}_{\text{id}\|z} = (\mathbf{v}_{\text{id}} \parallel \mathbf{w}_2)$. Therefore $\mathbf{A}_{\text{id}\|z} \cdot \mathbf{v}_{\text{id}\|z} = \mathbf{u} \bmod q$ is equivalent to

$$\mathbf{A}_{\text{id}} \cdot \mathbf{v}_{\text{id}} + \mathbf{A}'' \cdot \mathbf{w}_2 = \mathbf{u} \bmod q. \quad (5.4)$$

Since id is part of the secret input, $\mathbf{A}_{\text{id}}$ should not be explicitly given. We note that Langlois et al. [102] already addressed this problem. The idea is as follows: they first added ℓ suitable zero-blocks of size m to vector $\mathbf{v}_{\text{id}}$ and then obtained the

extended vector $\mathbf{w}_1 = (\mathbf{v}_0 \| \mathbf{v}_1^0 \| \mathbf{v}_1^1 \| \cdots \| \mathbf{v}_\ell^0 \| \mathbf{v}_\ell^1) \in \mathbb{Z}^{(2\ell+1)m}$, where the added zero-blocks are $\mathbf{v}_1^{1-\text{id}[1]}, \dots, \mathbf{v}_\ell^{1-\text{id}[\ell]}$ and $\mathbf{v}_i^{\text{id}[i]} = \mathbf{v}_i, \forall i \in [\ell]$. Now one can check that equation (5.4) is equivalent to

$$\mathbf{A}' \cdot \mathbf{w}_1 + \mathbf{A}'' \cdot \mathbf{w}_2 = \mathbf{u} \bmod q. \quad (5.5)$$

Let $\mathbf{B}' = \begin{bmatrix} \mathbf{B}^\top & \mathbf{I}_m & \mathbf{0}^{m \times \ell} \\ \mathbf{G}^\top & \mathbf{0}^{\ell \times m} & \mathbf{I}_\ell \end{bmatrix}$, $\mathbf{B}'' = \begin{bmatrix} \mathbf{0}^{m \times \ell} \\ \lfloor q/2 \rfloor \mathbf{I}_\ell \end{bmatrix}$, and $\mathbf{w}_3 = (\mathbf{s} \| \mathbf{e}_1 \| \mathbf{e}_2) \in \mathbb{Z}^{n+m+\ell}$. Then one can check that $\mathbf{c}_1 = \mathbf{B}^\top \cdot \mathbf{s} + \mathbf{e}_1 \bmod q$, $\mathbf{c}_2 = \mathbf{G}^\top \cdot \mathbf{s} + \mathbf{e}_2 + \lfloor \frac{q}{2} \rfloor \cdot \text{id} \bmod q$ is equivalent to

$$\mathbf{B}' \cdot \mathbf{w}_3 + \mathbf{B}'' \cdot \text{id} = (\mathbf{c}_1 \| \mathbf{c}_2) \bmod q. \quad (5.6)$$

Now we can use the decomposition techniques described in Section 2.4.3 to handle our secret vectors $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ as follows.

- $\hat{\mathbf{w}}_1 = (\hat{\mathbf{v}}_0 \| \hat{\mathbf{v}}_1^0 \| \hat{\mathbf{v}}_1^1 \| \cdots \| \hat{\mathbf{v}}_\ell^0 \| \hat{\mathbf{v}}_\ell^1) \in \{-1, 0, 1\}^{(2\ell+1)m\delta_B}$ with $\hat{\mathbf{v}}_0 = \text{vdec}_{m,\beta}(\mathbf{v}_0) \in \{-1, 0, 1\}^{m\delta_\beta}$, $\forall i \in [\ell]$, $\hat{\mathbf{v}}_i^{1-\text{id}[i]} = \mathbf{0}^{m\delta_\beta}$ and $\hat{\mathbf{v}}_i^{\text{id}[i]} = \text{vdec}_{m,\beta}(\mathbf{v}_i^{\text{id}[i]}) \in \{-1, 0, 1\}^{m\delta_\beta}$;
- $\hat{\mathbf{w}}_2 = \text{vdec}_{dm,\beta}(\mathbf{w}_2) \in \{-1, 0, 1\}^{dm\delta_\beta}$;
- $\hat{\mathbf{w}}_3 = \text{vdec}_{n+m+\ell,B}(\mathbf{w}_3) \in \{-1, 0, 1\}^{(n+m+\ell)\delta_B}$;

From equation (2.2), we then observe that equation (5.5) is equivalent to

$$\mathbf{A}' \cdot \mathbf{G}_{(2\ell+1)m,\beta} \cdot \hat{\mathbf{w}}_1 + \mathbf{A}'' \cdot \mathbf{G}_{dm,\beta} \cdot \hat{\mathbf{w}}_2 = \mathbf{u} \bmod q, \quad (5.7)$$

while equation (5.6) is equivalent to

$$\mathbf{B}' \cdot \mathbf{G}_{n+m+\ell,B} \cdot \hat{\mathbf{w}}_3 + \mathbf{B}'' \cdot \text{id} = (\mathbf{c}_1 \| \mathbf{c}_2) \bmod q. \quad (5.8)$$

Using basic algebra, we can transform equations (5.7) and (5.8) into one equation of the following form:

$$\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u}_0 \pmod{q},$$

where $\mathbf{u}_0 = (\mathbf{u} \parallel \mathbf{c}_1 \parallel \mathbf{c}_2) \in \mathbb{Z}_q^{n+m+\ell}$ and $\mathbf{w}_0 = (\widehat{\mathbf{w}}_1 \parallel \widehat{\mathbf{w}}_2 \parallel \widehat{\mathbf{w}}_3 \parallel \text{id})$ while $\mathbf{M}_0$ is built from public inputs.

EXTENDING-PERMUTING. Now we will manage to transform the equation $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u}_0 \pmod{q}$ above into an equation of the form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u}_0 \pmod{q}$ such that the secret vector $\mathbf{w}$ fulfills the conditions specified in (2.1). Toward this goal, we use the technique specified in Section 4.3.1 to prove knowledge of vectors in the set $\{-1, 0, 1\}$ instead of following the usual extending-permuting technique as in Langlois et al. [102]. Furthermore, to prove knowledge of vectors in the set $\{0, 1\}$, we need the following extending-permuting technique, which is specified in [102].

Proving that $\mathbf{z} \in \{0, 1\}^m$. For $\mathbf{z} = (z_1, \dots, z_m) \in \{0, 1\}^m$, define an extension of it below:

$$\text{Enc}_2(\mathbf{z}) = (\bar{z}_1, z_1, \dots, \bar{z}_m, z_m) \in \{0, 1\}^{2m}.$$

Now for any vector $\mathbf{b} = (b_1, \dots, b_m) \in \{0, 1\}^m$, associate a permutation $\Theta_{\mathbf{b}}$ that works as follows. When applying to vector $\mathbf{v} = (v_1^0, v_1^1, \dots, v_m^0, v_m^1) \in \mathbb{Z}^{2m}$, it permutes $\mathbf{v}$ into the following vector

$$(v_1^{b_1}, v_1^{\bar{b}_1}, \dots, v_m^{b_m}, v_m^{\bar{b}_m}).$$

For any $\mathbf{z}, \mathbf{b} \in \{0, 1\}^m$, it is verifiable that the following equivalence holds.

$$\mathbf{v} = \text{Enc}_2(\mathbf{z}) \iff \Theta_{\mathbf{b}}(\mathbf{v}) = \text{Enc}_2(\mathbf{z} \oplus \mathbf{b}). \quad (5.9)$$

Let $L_1 = 3(2\ell+1)m\delta_\beta$, $L_2 = 3dm\delta_\beta$, $L_3 = 3(n+m+\ell)\delta_B$, and $L = L_1 + L_2 + L_3 + 2\ell$.

Now we define the extension of $\mathbf{w}_0 = (\widehat{\mathbf{w}}_1 \parallel \widehat{\mathbf{w}}_2 \parallel \widehat{\mathbf{w}}_3 \parallel \text{id})$ as follows.

- $\bar{\mathbf{w}}_1 = (\bar{\mathbf{v}}_0 \parallel \bar{\mathbf{v}}_1^0 \parallel \bar{\mathbf{v}}_1^1 \parallel \dots \parallel \bar{\mathbf{v}}_\ell^0 \parallel \bar{\mathbf{v}}_\ell^1) \in \{-1, 0, 1\}^{L_1}$ with $\bar{\mathbf{v}}_0 = \text{enc}(\widehat{\mathbf{v}}_0) \in \{-1, 0, 1\}^{3m\delta_\beta}$,
 $\forall i \in [\ell], \bar{\mathbf{v}}_i^{1-\text{id}[i]} = \mathbf{0}^{3m\delta_\beta}$ and $\bar{\mathbf{v}}_i^{\text{id}[i]} = \text{enc}(\widehat{\mathbf{v}}_i^{\text{id}[i]}) \in \{-1, 0, 1\}^{3m\delta_\beta}$;

- $\bar{\mathbf{w}}_2 = \text{enc}(\hat{\mathbf{w}}_2) \in \{-1, 0, 1\}^{L_2};$
- $\bar{\mathbf{w}}_3 = \text{enc}(\hat{\mathbf{w}}_3) \in \{-1, 0, 1\}^{L_3};$
- $\bar{\text{id}} = \text{Enc}_2(\text{id}) \in \{0, 1\}^{2\ell}.$

Finally, the extended vector is formed as $\mathbf{w} = (\bar{\mathbf{w}}_1 \| \bar{\mathbf{w}}_2 \| \bar{\mathbf{w}}_3 \| \bar{\text{id}}) \in \{-1, 0, 1\}^L$. In the meanwhile, we add suitable zero-columns to matrix $\mathbf{M}_0$ to obtain matrix $\mathbf{M} \in \mathbb{Z}_q^{(n+m+\ell) \times L}$ so that $\mathbf{M} \cdot \mathbf{w} = \mathbf{u}_0 \bmod q$.

Now we identify the set **VALID** that consists of the secret vector $\mathbf{w}$, the set $\mathcal{S}$ and the associated permutation set $\{\Gamma_\eta : \eta \in \mathcal{S}\}$ so that the requirements in (2.1) are satisfied.

Define **VALID** to be the set of vectors of the form $\mathbf{z} = (\mathbf{z}_1 \| \mathbf{z}_2 \| \mathbf{z}_3 \| \mathbf{z}_4) \in \{-1, 0, 1\}^L$ such that there exists $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_\ell \in \{-1, 0, 1\}^{m\delta_\beta}$, $\mathbf{x}_{\ell+1} \in \{-1, 0, 1\}^{dm\delta_\beta}$, $\mathbf{x}_{\ell+2} \in \{-1, 0, 1\}^{(n+m+\ell)\delta_\beta}$, $\mathbf{x} \in \{0, 1\}^\ell$,

- $\mathbf{z}_1 = (\mathbf{y}_0 \| \mathbf{y}_1^0 \| \mathbf{y}_1^1 \| \dots \| \mathbf{y}_\ell^0 \| \mathbf{y}_\ell^1) \in \{-1, 0, 1\}^{3(2\ell+1)m\delta_\beta}$ with $\mathbf{y}_0 = \text{enc}(\mathbf{x}_0)$ and for each $i \in [\ell]$, $\mathbf{y}_i^{1-\mathbf{x}[i]} = \mathbf{0}^{3m\delta_\beta}$, $\mathbf{y}_i^{\mathbf{x}[i]} = \text{enc}(\mathbf{x}_i)$;
- $\mathbf{z}_2 = \text{enc}(\mathbf{x}_{\ell+1})$ and $\mathbf{z}_3 = \text{enc}(\mathbf{x}_{\ell+2})$;
- $\mathbf{z}_4 = \text{Enc}_2(\mathbf{x}) \in \{0, 1\}^{2\ell}.$

It is clear that our vector $\mathbf{w}$ is included in the set **VALID**.

Now, let $\mathcal{S} = (\{-1, 0, 1\}^{m\delta_\beta})^{2\ell+1} \times \{-1, 0, 1\}^{dm\delta_\beta} \times \{-1, 0, 1\}^{(n+m+\ell)\delta_B} \times \{0, 1\}^\ell$.

For any

$$\eta = (\mathbf{b}_0, \mathbf{b}_1^0, \mathbf{b}_1^1, \dots, \mathbf{b}_\ell^0, \mathbf{b}_\ell^1, \mathbf{c}, \mathbf{d}, \mathbf{e}) \in \mathcal{S}, \quad \mathbf{e} = (e_1, \dots, e_\ell)^\top,$$

define the permutation $\Gamma_\eta : \mathbb{Z}^L \rightarrow \mathbb{Z}^L$ as follows. When applied to a vector

$$\mathbf{z} = (\mathbf{y}_0 \| \mathbf{y}_1^0 \| \mathbf{y}_1^1 \| \dots \| \mathbf{y}_\ell^0 \| \mathbf{y}_\ell^1 \| \mathbf{z}_2 \| \mathbf{z}_3 \| \mathbf{z}_4) \in \mathbb{Z}^L$$

where the first $2\ell + 1$ blocks are of size $3m\delta_\beta$ and the last three blocks are of size $3dm\delta_\beta$, $3(n + m + \ell)\delta_B$ and 2ℓ , respectively; it transforms $\mathbf{z}$ to vector $\Gamma_\eta(\mathbf{z})$ of the

following form:

$$\left(\Pi_{\mathbf{b}_0}(\mathbf{y}_0) \parallel \Pi_{\mathbf{b}_1^{e_1}}(\mathbf{y}_1^{e_1}) \parallel \Pi_{\mathbf{b}_1^{1-e_1}}(\mathbf{y}_1^{1-e_1}) \parallel \cdots \parallel \Pi_{\mathbf{b}_\ell^{e_\ell}}(\mathbf{y}_\ell^{e_\ell}) \parallel \Pi_{\mathbf{b}_\ell^{1-e_\ell}}(\mathbf{y}_\ell^{1-e_\ell}) \parallel \right. \\ \left. \Pi_{\mathbf{c}}(\mathbf{z}_2) \parallel \Pi_{\mathbf{d}}(\mathbf{z}_3) \parallel \Theta_{\mathbf{e}}(\mathbf{z}_4) \right).$$

Observing the equivalences in (4.4) and (5.9) , it is verifiable that **VALID**, $\mathcal{S}$, Γ_η fulfill the conditions in (2.1). Therefore, we have successfully reduced the considered statement into an instance of the abstracted relation described in Section 2.4.2. At this point, we can run the interactive protocol described in Fig. 2.1 and obtain the desired statistical ZKAoK protocol.

Chapter 6

Accountable Tracing Signatures from Lattices

In Chapters 3-5, we presented several lattice-based group signature schemes that achieve different kinds of functionality. In all these schemes, the group manager (or the opening authority) has the ability to open any signature and there is no way to keep him accountable. Accountable tracing signature (ATS) suggested by Kohlweiss and Miers [99] is a good solution to make the group manager responsible to his opening behaviour. To our knowledge, the ATS scheme presented by Kohlweiss and Miers [99] is the only such scheme and is built from bilinear maps.

In this thesis, we provide the first lattice-based accountable tracing signature scheme. The scheme is proven secure under the model suggested by Kohlweiss and Miers, assuming the hardness of the RSIS and the RLWE problems. At the heart of our construction are a lattice-based key-oblivious encryption (KOE) scheme and a zero-knowledge argument system allowing to prove that a given ciphertext is a valid RLWE encryption under some hidden but certified key. Specifically, the argument system allows to prove quadratic relation where some term $\mathbf{a} \cdot g$ is a product of two witness secrets $\mathbf{a}$ and g that may also involve in other equations. We believe that our KOE scheme and our argument system are useful in other protocols in the lattice-based

cryptography, which may be of independent interest.

Organization. In Section 6.1, an overview of our techniques is given. Section 6.2 recalls the model of the accountable tracing signatures while Section 6.3 describes our key-oblivious encryption scheme from lattice assumptions. Our zero-knowledge protocol for proving the RLWE relation with hidden vector is presented in Section 6.4 and our main scheme is constructed and analyzed in Section 6.5.

6.1 Overview of Our Techniques

Recall that in an ordinary group signature scheme [20, 21], to ensure traceability, the user needs to encrypt his identifier information and prove the well-formedness of the ciphertext. In an ATS scheme, however, not all users are being traceable by group manager. Therefore we should have a mechanism to distinguish them. A possible method is to let traceable user encrypt his identity under a public key ($\mathbf{pk}$) such that the group manager knows the underlying secret key ($\mathbf{sk}$), while for non-traceable users, group manager does not know the secret key. However, there is no incentive for users to be traceable. We should have a way to choose traceable users obliviously. Thus, it would be desirable if we can randomize $\mathbf{pk}$ to a new public key $\mathbf{epk}$ so that it is infeasible to decide how these keys are related without the knowledge of the secret key and the randomness for randomization. Specifically, when a user joins the group, group manager first randomizes $\mathbf{pk}$ to $\mathbf{epk}$ and sends the latter to the user together with his certificate. The difference between traceable users and non-traceable ones is that whether the group manager knows the underlying secret key. Since the obliviousness property of the randomization, the users are unaware of whether they are traceable. Now when signing messages, the user encrypts his identity using his own/secret randomized public key $\mathbf{epk}$ and prove well-formedness of the ciphertext. Several questions in terms of this method arise. Does this encryption scheme exist? How should we randomize the public key to get the desirable obliviousness? More

importantly, how should the user prove the honest behaviour of encryption if the underlying encryption key is secret?

To address the first two questions, Kohlweiss and Miers [99] proposed a key-oblivious encryption (KOE) scheme such that we are able to randomize the public key. In addition, the randomization is oblivious. As shown by Kohlweiss and Miers [99], a KOE scheme can be obtained from a key-private homomorphic public key encryption scheme and they gave an explicit construction based on ElGamal cryptosystem [77]. However, one has to be careful when constructing such KOE scheme from lattices. In fact, an implicit condition for the key-private public key encryption scheme is that the public key and the ciphertext should have the same dimension, which is often not the case for the encryption schemes in the lattice setting, e.g., [81, 160]. Furthermore, the encryption schemes in the lattice setting often involves noise terms. We observe that the encryption scheme proposed by Lyubashevsky, Peikert, Regev (LPR) [130] is a good candidate. In fact, the public key and the ciphertext of the LPR scheme have the same dimension, which is analogous to the ElGamal cryptosystem [77]. Setting the parameters carefully, we are safe to adapt the idea of [99] to the lattice setting.

To tackle with the third question, we need a zero-knowledge protocol for proving well-formedness of the ciphertext under a hidden encryption key, which seems to be much more challenging in the lattice setting. There are two main lines of zero-knowledge (ZK) protocols from lattices. One line of research [23, 24, 124, 125, 129, 132] designed very elegant approximate ZK proofs for (R)LWE and (R)SIS relations by employing “Fiat-Shamir with Aborts” technique. While these proofs are quite efficient and compact, they only handle linear relations. In other words, they can only prove knowledge of a short vector $\mathbf{x}$ satisfying $\mathbf{y} = \mathbf{A} \cdot \mathbf{x} \bmod q$. This seems not suitable for our scenario. Another line of research [106–109, 116, 117, 120] then designed Stern-like protocols [173] by developing decomposition, extension, and permutation techniques. Even though less practical than the former approximate ZK protocols, Stern-like protocols are much more versatile and can even deal with quadratic relations [106]. More

precisely, Libert et al. [106] design a ZK protocol to prove knowledge of secret $\mathbf{A}$ and short vector $\mathbf{x}$ satisfying $\mathbf{y} = \mathbf{A} \cdot \mathbf{x} \bmod q$. This appears to be the right ZK protocols for our case. Nevertheless, Libert et al. [106] proved this quadratic relation only in the setting of general lattices while we have to deal with ideal lattices, which seems to be more difficult. By adapting their techniques [106] to the ideal lattices, we managed to obtain the desired protocol. In the process, we also demonstrate a new zero-knowledge protocol showing that a secret integer y is a product of secret integers $a \in \{-1, 0, 1\}$ and $g \in \{-1, 0, 1\}$. Furthermore, this protocol is compatible and extendable so that we can additionally prove that the same a and g satisfy other relations.

As discussed so far, we have identified the LPR encryption scheme and Stern-like zero-knowledge protocols in constructing our ATS scheme. For the underlying signature scheme, it is required to sign some ring elements. We observe that Ling et al. [120] used the LPR encryption scheme [130] and Ducas and Micciancio signature scheme [72, 73] for constructing their constant size group signatures. Therefore, we employed the same signature scheme as in [120]. As a byproduct, we achieve constant size signatures as well.

The thing that remains is how should we ensure the accountability of the group manager. To this end, the group manager reveals the choice (either traceable or non-traceable) for the user and the randomness used to obtain the randomized public key. The user then checks whether his **epk** is computed from the public key according to group manager's choice. However, the group manager may claim a traceable user to be non-traceable by giving away malicious randomness and accusing that the user had changed **epk** by himself. To ensure non-repudiation, group manager is required to sign **epk** and **upk** when registering the user into the group. This mechanism in fact also prevents dishonest users from choosing (non-traceable) **epk** by themselves.

6.2 Accountable Tracing Signatures

An ATS scheme involves a group manager (GM), who manages the group, a set of potential group users. As a standard group signature scheme (e.g. [20, 21]), GM can discover the signer of any signature. However, if GM is able to do so, there is an additional *accounting* mechanism that later reveals which user he chose to trace (traceable user). Specifically, if a user suspects that he was traceable by group manager who had claimed non-traceability of this user, then the user can resort to this mechanism to check whether group manager is honest/accountable or not.

6.2.1 Syntax of Accountable Tracing Signatures

An ATS scheme contains polynomial-time algorithms described below.

Setup(λ): On input the security parameter λ , it outputs public parameter $\mathbf{pp}$. $\mathbf{pp}$ is implicit for all algorithms below if not explicitly mentioned.

GKeyGen($\mathbf{pp}$): This algorithm is run by GM. On input $\mathbf{pp}$, GM generates group public key $\mathbf{gpk}$ and group secret keys: issue key $\mathbf{ik}$ and opening key $\mathbf{ok}$.

UKeyGen($\mathbf{pp}$): Given input $\mathbf{pp}$, it outputs a user key pair $(\mathbf{upk}, \mathbf{usk})$.

Enroll($\mathbf{gpk}, \mathbf{ik}, \mathbf{upk}, \mathbf{tr}$): This algorithm is run by GM. Upon receiving a user public key $\mathbf{upk}$ from a user, GM determines the value of the bit $\mathbf{tr} \in \{0, 1\}$, indicating whether the user is traceable ($\mathbf{tr} = 1$) or not. He then produces a certificate $\mathbf{cert}$ for this user according to his choice of $\mathbf{tr}$. GM then registers this user to the group and stores the registration information and the witness w^{escrw} to the bit $\mathbf{tr}$, and sends $\mathbf{cert}$ to the user.

Sign($\mathbf{gpk}, \mathbf{cert}, \mathbf{usk}, M$): Given the inputs $\mathbf{gpk}$, $\mathbf{cert}$, $\mathbf{usk}$ and message M , this algorithm outputs a signature Σ on this message M .

Verify(gpk, M , Σ): Given the inputs **gpk** and the message-signature pair (M, Σ) , this algorithm outputs 1/0 indicating whether the signature is valid or not.

Open(gpk, ok, M , Σ): Given the inputs **gpk**, **ok** and the pair (M, Σ) , it returns a user public key **upk'** and a proof Π_{open} demonstrating that user **upk'** indeed generated the signature Σ . In case of **upk'** = $\perp$, $\Pi_{\text{open}} = \perp$.

Judge(gpk, M , Σ , **upk', Π_{open}):** Given all the inputs, this algorithm outputs 1/0 indicating whether it accepts the opening result or not.

Account(gpk, cert, w^{escrw} , **tr):** Given all the inputs, this algorithm returns 1 confirming the choice of **tr** and 0 otherwise.

Remark 6.2.1. There is a minor difference between the syntax we describe here and that presented by Kohlweiss and Miers [99]. Specifically, we omit the time epoch when the user joins the group, since we do not consider forward and backward tracing scenarios as in [99].

6.2.2 Definitions of Correctness and Security Requirements

The above **ATS** scheme should satisfy *correctness*, *anonymity under tracing* (**AuT**), *traceability* (**Trace**), and *non-frameability* (**NF**), *anonymity with accountability* (**AwA**), and *trace-obliviousness* (**TO**).

Correctness requires that: for any honestly generated signature, the **Verify** algorithm always outputs 1. Furthermore, if the user is traceable, then **Account** algorithm outputs 1 when **tr** = 1, and the **Open** algorithm can identify the signer and generate a proof Π_{open} that will be accepted by the **Judge** algorithm. On the other hand, if the user is non-traceable, then the **Account** algorithm outputs 1 when **tr** = 0, and the **Open** algorithm outputs $\perp$.

Anonymity under tracing is the standard anonymity requirement of group signatures (e.g., [20, 21]). It guarantees that even when being traced, users are anonymous to

the adversary who does not hold the opening key.

Definition 6.2.2 (Anonymity under tracing). For any security parameter λ and PPT algorithm $\mathcal{A}$, we define the experiment $\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-b}(\lambda)$ and some oracles accessible to $\mathcal{A}$ as follows.

Experiment $\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-b}(\lambda)$ // $b \leftarrow \{0, 1\}$.

$\text{pp} \leftarrow \text{Setup}(\lambda)$, $(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKeyGen}(\text{pp})$.

$b' \leftarrow \mathcal{A}^{\text{Ch}, \text{Open}}(\text{gpk}, \text{ik})$.

Return b' .

Oracle $\text{Open}(M, \Sigma)$

If $\Sigma \in Q$, then return $\perp$; Else return $(\text{upk}, \Pi) \leftarrow \text{Open}(\text{ok}, M, \Sigma)$.

Oracle $\text{Ch}(\text{cert}_0, \text{cert}_1, \text{usk}_0, \text{usk}_1, M, w_0^{\text{escrw}}, w_1^{\text{escrw}}, 1)$

$\Sigma_0 \leftarrow \text{Sign}(\text{gpk}, \text{cert}_0, \text{usk}_0, M)$, $\Sigma_1 \leftarrow \text{Sign}(\text{gpk}, \text{cert}_1, \text{usk}_1, M)$.

If $\Sigma_0 \neq \perp \wedge \Sigma_1 \neq \perp \wedge \text{Account}(\text{gpk}, \text{cert}_0, w_0^{\text{escrw}}, 1) \wedge \text{Account}(\text{gpk}, \text{cert}_1, w_1^{\text{escrw}}, 1)$

Then $Q \leftarrow Q \cup \{\Sigma_b\}$, and return Σ_b .

Else return $\perp$.

Define the advantage $\mathbf{Adv}_{\text{ATS}, \mathcal{A}}^{\text{AuT}}(\lambda)$ of adversary $\mathcal{A}$ against anonymity under tracing of the ATS scheme as $|\Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-0}(\lambda) = 1]|$. An ATS scheme is anonymous under tracing if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Traceability demands that every valid signature is attached to someone as long as the adversary does not hold both the certificate and user secret key of a user who is not traceable (non-traceable user). As pointed out by Kohlweiss and Miers [99], this is slightly different from the standard traceability game (e.g. [20, 21]), where all users are being traced by GM. In an ATS scheme, when adversary queries certificate of a user of his choice, challenger will always generate a certificate according to $\text{tr} = 1$. In other words, the user of the adversary's choice is a traceable user. This ensures that the adversary does not hold both certificate and user secret key for a non-traceable user.

Definition 6.2.3 (Traceability). For any security parameter λ and PPT algorithm $\mathcal{A}$, define the experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{Trace}}(\lambda)$ and some oracles accessible to $\mathcal{A}$ in the following.

Experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{Trace}}(\lambda)$

$\text{pp} \leftarrow \text{Setup}(\lambda)$, $(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKeyGen}(\text{pp})$.

$(M, \Sigma) \leftarrow \mathcal{A}^{\text{UKG}, \text{Enroll}, \text{Sign}, \text{Open}}(\text{gpk})$.

If $(M, \Sigma) \in Q$ or $\text{Verify}(\text{gpk}, M, \Sigma) = 0$, output 0.

$(\text{upk}, \Pi) \leftarrow \text{Open}(\text{ok}, m, \Sigma)$.

Output 1 if $\text{upk} = \perp$ or $\text{Judge}(\text{gpk}, M, \Sigma, \text{upk}, \Pi) = 0$.

Else output 0.

Oracle $\text{UKG}(\text{pp})$

$(\text{upk}, \text{usk}) \leftarrow \text{UKeyGen}(\text{pp})$. $S[\text{upk}] = \text{usk}$. Return upk .

Oracle $\text{Enroll}(\text{upk}, \text{tr})$

Let $\text{tr}' = (\text{upk} \notin \text{dom } S) \in \{0, 1\}$. $(\text{cert}, w^{\text{escrw}}) \leftarrow \text{Enroll}(\text{ik}, \text{upk}, \text{tr} \vee \text{tr}')$.

Return cert .

Oracle $\text{Sign}(\text{cert}, M)$

$\text{usk} = S[\text{cert.upk}]$. If $\text{usk} = \perp$, return $\perp$.

$\Sigma \leftarrow \text{Sign}(\text{gpk}, \text{cert}, \text{usk}, M)$. $Q = Q \cup \{(m, \Sigma)\}$. Return Σ .

Oracle $\text{Open}(M, \Sigma)$

$(\text{upk}, \Pi) \leftarrow \text{Open}(\text{ok}, M, \Sigma)$. Return (upk, Π) .

Let the advantage $\mathbf{Adv}_{\text{ATS},\mathcal{A}}^{\text{Trace}}(\lambda)$ of adversary $\mathcal{A}$ against traceability of the ATS scheme be $\Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{Trace}}(\lambda) = 1]$. An ATS scheme is traceable if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Non-frameability demands that the adversary cannot sign messages on behalf of honest users, even though the adversary can corrupt GM and all other users. This ensures that signatures signed by a traceable user (traceable signatures) are non-repudiated.

Definition 6.2.4 (Non-frameability). For any security parameter λ and PPT algorithm $\mathcal{A}$, define the experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{NF}}(\lambda)$ below. The two oracles UKG and Sign

are defined as in Definition 6.2.3.

Experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{NF}}(\lambda)$

$\text{pp} \leftarrow \text{Setup}(\lambda)$. $(\text{gpk}, \text{st}) \leftarrow \mathcal{A}(\text{pp})$. If $\text{gpk.pp} \neq \text{pp}$, return $\perp$.

$(M, \Sigma, \text{upk}, \Pi) \leftarrow \mathcal{A}^{\text{UKG}, \text{Sign}}(\text{st})$.

If $(M, \Sigma) \in Q$ or $\text{Verify}(\text{gpk}, M, \Sigma) = 0$ or $\text{upk} \notin \text{dom}(S)$, return 0.

Return 1 if $\text{Judge}(\text{gpk}, M, \Sigma, \text{upk}, \Pi) = 1$. Else return 0.

Define the advantage $\mathbf{Adv}_{\text{ATS},\mathcal{A}}^{\text{NF}}(\lambda)$ of adversary $\mathcal{A}$ against non-frameability of the ATS scheme as $\Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{NF}}(\lambda) = 1]$. An ATS scheme is non-frameable if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Anonymity with accountability requests that a user is anonymous even from a corrupted group manager that is in charge of the whole system as long as this user is non-traceable. In other words, the certificate is generated according to $\text{tr} = 0$.

Definition 6.2.5 (Anonymity with Accountability). For security parameter λ and PPT algorithm $\mathcal{A}$, the experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AwA}-b}(\lambda)$ and the oracle Ch are defined in the following way.

Experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AwA}-b}(\lambda)$ // $b \leftarrow \{0, 1\}$

$\text{pp} \leftarrow \text{Setup}(\lambda)$. $(\text{gpk}, \text{st}) \leftarrow \mathcal{A}(\text{pp})$. If $\text{gpk.pp} \neq \text{pp}$, return $\perp$.

$b' \leftarrow \mathcal{A}^{\text{Ch}}(\text{st})$.

Return b' .

Oracle $\text{Ch}(\text{cert}_0, \text{cert}_1, \text{usk}_0, \text{usk}_1, M, w_0^{\text{escrw}}, w_1^{\text{escrw}}, 0)$

$\Sigma_0 \leftarrow \text{Sign}(\text{gpk}, \text{cert}_0, \text{usk}_0, M)$, $\Sigma_1 \leftarrow \text{Sign}(\text{gpk}, \text{cert}_1, \text{usk}_1, M)$.

If $\Sigma_0 \neq \perp \wedge \Sigma_1 \neq \perp \wedge \text{Account}(\text{gpk}, \text{cert}_0, w_0^{\text{escrw}}, 0) \wedge \text{Account}(\text{gpk}, \text{cert}_1, w_1^{\text{escrw}}, 0)$

Return Σ_b .

Else return $\perp$.

Define the advantage $\mathbf{Adv}_{\text{ATS},\mathcal{A}}^{\text{AwA}}(\lambda)$ of $\mathcal{A}$ against anonymity with accountability of the ATS scheme as $|\Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AwA}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AwA}-0}(\lambda) = 1]|$. An ATS scheme is anonymous with accountability if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Trace-obliviousness requests that each user is not able to decide whether he is being traced or not.

Definition 6.2.6 (Trace-obliviousness). For any security parameter λ and PPT algorithm $\mathcal{A}$, define the experiments $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{TO}-b}(\lambda)$ and the oracles as follows.

Experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{TO}-b}(\lambda)$ // $b \leftarrow \{0, 1\}$.

$\text{pp} \leftarrow \text{Setup}(\lambda)$. $(\text{gpk}, \text{ik}, \text{ok}) \leftarrow \text{GKeyGen}(\text{pp})$.

$b' \leftarrow \mathcal{A}^{\text{Ch}, \text{Enroll}, \text{Open}}(\text{gpk})$.

Output b' .

Oracle $\text{Enroll}(\text{upk}, \text{tr})$

$(\text{cert}, w^{\text{escrw}}) \leftarrow \text{Enroll}(\text{ik}, \text{upk}, \text{tr})$. Output cert .

Oracle $\text{Ch}(\text{upk})$

$(\text{cert}, w^{\text{escrw}}) \leftarrow \text{Enroll}(\text{ik}, \text{upk}, b)$. $U = U \cup \{\text{upk}\}$. Output cert .

Oracle $\text{Open}(M, \Sigma)$

$(\text{upk}, \Pi) \leftarrow \text{Open}(\text{ok}, M, \Sigma)$. If $\text{upk} \in U$, then output $\perp$; Else output (upk, Π) .

Define the advantage $\mathbf{Adv}_{\text{ATS},\mathcal{A}}^{\text{TO}}(\lambda)$ of adversary $\mathcal{A}$ against trace-obliviousness of the ATS scheme as $|\Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{TO}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{TO}-0}(\lambda) = 1]|$. An ATS scheme is trace-oblivious if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

6.3 Key-Oblivious Encryption from Lattices

In Section 6.3.1, we review the model, correctness and security notions of *key-oblivious encryption* (KOE) scheme suggested by Kohlweiss and Miers [99]. In Section 6.3.2, we give a concrete instantiation from lattices. The security of our key-oblivious encryption scheme is analyze in Section 6.3.3.

6.3.1 Key-Oblivious Encryption

A KOE scheme consists of polynomial-time algorithms specified below.

Setup(λ): On input the security parameter λ , it outputs public parameter pp . pp is implicit for all algorithms below if not explicitly mentioned.

KeyGen(pp): On input pp , it generates a key pair (pk, sk) .

KeyRand(pk): On input the public key pk , it outputs a new public key pk' for the same secret key.

Enc(pk, m): On inputs pk and a message m , it outputs a ciphertext ct on this message.

Dec(sk, ct): On inputs sk and ct , it outputs the decrypted message m' .

Correctness of the above scheme must satisfy the following requirement: For all λ , all $\text{pp} \leftarrow \text{Setup}(\lambda)$, all $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp})$, all $\text{pk}' \leftarrow \text{KeyRand}(\text{pk})$, all m ,

$$\text{Dec}(\text{sk}, \text{Enc}(\text{pk}', \text{m})) = \text{m}.$$

The security requirements of a KOE scheme consist of *key randomizability* (KR), *plaintext indistinguishability under key randomization* (INDr), and *key privacy under key randomization* (KPr), which we describe in more detail below.

Key Randomizability requires that any PPT adversary cannot determine how public keys are related to each other without possession of secret keys.

Definition 6.3.1 (Key Randomizability). Let λ be the security parameter and $\mathcal{A}$ be any PPT adversary. Define the experiment $\text{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KR}}(\lambda)$ below.

Experiment $\text{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KR}}(\lambda)$

$b \leftarrow \{0, 1\}$, $\text{pp} \leftarrow \text{Setup}(\lambda)$, $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp})$.

$\text{pk}_0 \leftarrow \text{KeyRand}(\text{pk})$, $(\text{pk}_1, \text{sk}_1) \leftarrow \text{KeyGen}(\text{pp})$.

$b' \leftarrow \mathcal{A}(\text{pk}, \text{pk}_b)$. Return $(b' = b)$.

Define the advantage $\text{Adv}_{\text{KOE}, \mathcal{A}}^{\text{KR}}(\lambda)$ of adversary $\mathcal{A}$ against KR of the KOE scheme as $|2\Pr[\text{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KR}}(\lambda) = 1] - 1|$. A KOE scheme is key randomizable if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Plaintext indistinguishability under key randomization requires that any adversary cannot distinguish ciphertext of one message from ciphertext of another one even though the adversary is allowed to choose the two messages and to randomize the public key.

Definition 6.3.2 (Plaintext indistinguishability under key randomization). Let λ be the security parameter and $\mathcal{A}$ be any PPT adversary. Define the experiment $\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{INDr}}(\lambda)$ below.

Experiment $\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{INDr}}(\lambda)$

$b \leftarrow \{0, 1\}$, $\text{pp} \leftarrow \text{Setup}(\lambda)$, $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp})$.

$(\text{pk}', r, \text{m}_0, \text{m}_1, st) \leftarrow \mathcal{A}(\text{pk})$.

If $\text{pk}' \neq \text{KeyRand}(\text{pk}, r)$, then return $\perp$; else $\text{ct} \leftarrow \text{Enc}(\text{pk}', \text{m}_b)$.

$b' \leftarrow \mathcal{A}(\text{ct}, st)$. Return $(b' = b)$.

Define the advantage $\mathbf{Adv}_{\text{KOE}, \mathcal{A}}^{\text{INDr}}(\lambda)$ of adversary $\mathcal{A}$ against INDr of the KOE scheme as $|\Pr[\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{INDr}}(\lambda) = 1] - 1/2|$. A KOE scheme is plaintext indistinguishable under key randomization if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

Key privacy under key randomization requires that any adversary cannot distinguish ciphertext of a message under one public key from ciphertext of the same message under another public key even though the adversary is allowed to choose the message and to randomize the two public keys.

Definition 6.3.3 (Key privacy under key randomization). Let λ be the security parameter and $\mathcal{A}$ be any PPT adversary. Define the experiment $\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KPr}}(\lambda)$ below.

$\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KPr}}(\lambda)$

$b \leftarrow \{0, 1\}$, $\text{pp} \leftarrow \text{Setup}(\lambda)$; $(\text{pk}_0, \text{sk}_0) \leftarrow \text{KeyGen}(\text{pp})$, $(\text{pk}_1, \text{sk}_1) \leftarrow \text{KeyGen}(\text{pp})$.

$(\text{m}, \text{pk}'_0, r_0, \text{pk}'_1, r_1, st) \leftarrow \mathcal{A}(\text{pk}_0, \text{pk}_1)$.

If $\exists c$ such that $\text{pk}'_c \neq \text{KeyRand}(\text{pk}_c, r_c)$, then return $\perp$; else $\text{ct} \leftarrow \text{Enc}(\text{pk}'_b, \text{m})$.

$b' \leftarrow \mathcal{A}(\text{ct}, st)$. Return $(b' = b)$.

Define the advantage $\mathbf{Adv}_{\text{KOE}, \mathcal{A}}^{\text{KPr}}(\lambda)$ of adversary $\mathcal{A}$ against **INDr** of the **KOE** scheme as $|2\Pr[\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KPr}}(\lambda) = 1] - 1|$. A **KOE** scheme is key private under key randomization if the advantage of any PPT adversary $\mathcal{A}$ is negligible.

6.3.2 Description of Our **KOE** Scheme

In the work by Kohlweiss and Miers [99], they constructed a **KOE** scheme based on ElGamal cryptosystem [77]. As an application in [130], a ring-based public key encryption scheme was proposed, which is analogous to ElGamal cryptosystem. Applying the idea of [99] to a variant of the scheme in [130] and carefully setting the parameters, we obtain our **KOE** scheme from lattices, which is specified below.

Setup(λ): Taken λ as input, it chooses $n = \mathcal{O}(\lambda)$ be a power of 2 and $q = \tilde{\mathcal{O}}(n^4)$.

Also let $\ell = \lfloor \log \frac{q-1}{2} \rfloor + 1$. Define the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$.

Let the integer bound B be of order $\tilde{\mathcal{O}}(\sqrt{n})$ and χ be a distribution over the ring R that is B -bounded. This algorithm then outputs public parameter $\mathbf{pp} = \{n, q, \ell, R, R_q, B, \chi\}$.

KeyGen($\mathbf{pp}$): Given the input $\mathbf{pp}$, this algorithm samples $s \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^\ell$ and $\mathbf{a} \xleftarrow{\$} R_q^\ell$.

Set $\mathbf{pk} = (\mathbf{a}, \mathbf{b}) = (\mathbf{a}, \mathbf{a} \cdot s + \mathbf{e}) \in R_q^\ell \times R_q^\ell$ and $\mathbf{sk} = s$. It then returns $(\mathbf{pk}, \mathbf{sk})$.

KeyRand($\mathbf{pk}$): Given the public key $\mathbf{pk} = (\mathbf{a}, \mathbf{b})$, it samples $g \leftarrow \chi$, $\mathbf{e}_1 \leftarrow \chi^\ell$ and $\mathbf{e}_2 \leftarrow \chi^\ell$. Compute

$$(\mathbf{a}', \mathbf{b}') = (\mathbf{a} \cdot g + \mathbf{e}_1, \mathbf{b} \cdot g + \mathbf{e}_2) \in R_q^\ell \times R_q^\ell.$$

This algorithm then outputs randomized public key as $\mathbf{pk}' = (\mathbf{a}', \mathbf{b}')$.

Enc($\mathbf{pk}', p$): Taken the public key $\mathbf{pk}' = (\mathbf{a}', \mathbf{b}')$ and a message $p \in R_q$ as input, it samples $g' \in \chi$, $\mathbf{e}'_1 \in \chi^\ell$ and $\mathbf{e}'_2 \in \chi^\ell$. Compute

$$(\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{a}' \cdot g' + \mathbf{e}'_1, \mathbf{b}' \cdot g' + \mathbf{e}'_2 + \lfloor q/4 \rfloor \cdot \text{rdec}(p)) \in R_q^\ell \times R_q^\ell.$$

This algorithm then returns ciphertext as $\mathbf{ct} = (\mathbf{c}_1, \mathbf{c}_2)$.

Dec(sk, ct): Given $\mathbf{sk} = s$ and $\mathbf{ct} = (\mathbf{c}_1, \mathbf{c}_2)$, the algorithm acts as follows.

1. It computes

$$\mathbf{p}'' = \frac{\mathbf{c}_2 - \mathbf{c}_1 \cdot s}{\lfloor q/4 \rfloor}.$$

2. For each coefficient of $\mathbf{p}''$,

- round it to -1 if it is closer to -1 than to 0 and 1 ;
- round it to 0 if it is closer to 0 than to -1 and 1 ;
- round it to 1 if it is closer to 1 than to 0 and -1 .

3. Let the rounded polynomial be $\mathbf{p}' \in R_q^\ell$ such that each coefficient is in the set $\{-1, 0, 1\}$.

4. Compute $p' \in R_q$ satisfying $\tau(p') = \mathbf{H} \cdot \tau(\mathbf{p}')$. Note that $\mathbf{H} \in \mathbb{Z}_q^{n \times n\ell}$ is the decomposition matrix for elements in the ring R_q (see Section 2.4.3).

CORRECTNESS. Note that

$$\begin{aligned} \mathbf{c}_2 - \mathbf{c}_1 \cdot s &= \mathbf{b}' \cdot g' + \mathbf{e}'_2 + \lfloor q/4 \rfloor \cdot \mathbf{rdec}(p) - (\mathbf{a}' \cdot g' + \mathbf{e}'_1) \cdot s \\ &= \mathbf{e} \cdot g \cdot g' + \mathbf{e}_2 \cdot g' - \mathbf{e}_1 \cdot s \cdot g' + \mathbf{e}'_2 - \mathbf{e}'_1 \cdot s + \lfloor q/4 \rfloor \cdot \mathbf{rdec}(p) \end{aligned}$$

where $s, g, g', \mathbf{e}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}'_1, \mathbf{e}'_2$ are B -bounded. Hence we have:

$$\|\mathbf{e} \cdot g \cdot g' + \mathbf{e}_2 \cdot g' - \mathbf{e}_1 \cdot s \cdot g' + \mathbf{e}'_2 - \mathbf{e}'_1 \cdot s\|_\infty \leq 3n^2 \cdot B^3 = \tilde{\mathcal{O}}(n^{3.5}) \leq \left\lceil \frac{q}{10} \right\rceil = \tilde{\mathcal{O}}(n^4).$$

Except with negligible probability, the rounding procedure specified in the **Dec** algorithm recovers $\mathbf{rdec}(p)$ and hence returns p . Therefore, our **KOE** scheme is correct.

6.3.3 Security Analysis of Our KOE Scheme

Our KOE scheme is secure assumption the hardness of the RLWE assumption and security is established by the following lemmas.

Lemma 6.3.4. The key-oblivious encryption scheme depicted in Section 6.3.2 is key randomizable defined in Section 6.3.1 under RLWE assumption.

Proof. Notice that the samples chosen according to $\mathcal{A}_{s,\chi}$ for some $s \leftarrow \chi$ are indistinguishable from random under the RLWE assumption. Therefore, the honestly generated public key $\mathbf{pk} = (\mathbf{a}, \mathbf{b}) \in R_q^\ell \times R_q^\ell$ is indistinguishable from truly random pair $\widetilde{\mathbf{pk}} = (\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}) \in R_q^\ell \times R_q^\ell$. Hence, we may replace $\mathbf{pk}$ with $\widetilde{\mathbf{pk}}$ and this modification is negligible to the adversary.

Let $\mathbf{pk}_0 = (\widetilde{\mathbf{a}} \cdot g + \mathbf{e}_1, \widetilde{\mathbf{b}} \cdot g + \mathbf{e}_2)$ and $\mathbf{pk}_1 = (\mathbf{a}', \mathbf{a}' \cdot s' + \mathbf{e}')$, where $\mathbf{pk}_1$ is independent of $\widetilde{\mathbf{pk}}$. When $b = 0$, adversary is given $(\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}, \widetilde{\mathbf{a}} \cdot g + \mathbf{e}_1, \widetilde{\mathbf{b}} \cdot g + \mathbf{e}_2)$, which are 2ℓ samples chosen according to $\mathcal{A}_{g,\chi}$. Therefore, $(\widetilde{\mathbf{pk}}, \mathbf{pk}_0)$ is indistinguishable from 2ℓ samples chosen according to $U(R_q \times R_q)$. When $b = 1$, adversary is given $(\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}, \mathbf{a}', \mathbf{a}' \cdot s' + \mathbf{e}')$. Since $\mathbf{pk}_1$ is independent of $\widetilde{\mathbf{pk}}$, so we can replace $\mathbf{pk}_1$ with a truly random pair. Hence, $(\widetilde{\mathbf{pk}}, \mathbf{pk}_1)$ is also indistinguishable from 2ℓ samples chosen according to $U(R_q \times R_q)$. Therefore, the adversary cannot distinguish the case $b = 0$ from the case $b = 1$.

It then follows that the advantage of any PPT adversary in $\mathbf{Exp}_{\text{KOE}, \mathcal{A}}^{\text{KR}}(\lambda)$ is negligible and hence our KOE scheme is key randomizable. $\square$

Lemma 6.3.5. The key-oblivious encryption scheme described in Section 6.3.2 is plaintext indistinguishable under key randomization defined in Section 6.3.1 under RLWE assumption.

Proof. Let $\mathcal{A}$ be any PPT algorithm against the plaintext indistinguishability under key randomization with advantage ϵ , we then show $\epsilon = \text{negl}(\lambda)$ assuming the hardness of the RLWE problem. Concretely, we construct a sequence of indistinguishable games G_0, G_1, G_2, G_3, G_4 , such that, $\mathbf{Adv}_{\mathcal{A}}(G_0) = \epsilon$ and $\mathbf{Adv}_{\mathcal{A}}(G_4) = 0$.

Game G_0 : This is the real experiment $\text{Exp}_{\text{KOE}, \mathcal{A}}^{\text{INDr}}(\lambda)$. The challenger generates a public key $\mathbf{pk} = (\mathbf{a}, \mathbf{b}) = (\mathbf{a}, \mathbf{a} \cdot s + \mathbf{e})$ honestly, sends it to the adversary $\mathcal{A}$, receives back a randomized key pair $\mathbf{pk}' = (\mathbf{a} \cdot g + \mathbf{e}_1, \mathbf{b} \cdot g + \mathbf{e}_2)$, the randomness used to generate $\mathbf{pk}'$, and two messages $p_0, p_1 \in R_q$. The challenger first checks whether $\mathbf{pk}'$ is generated from the randomness or not. If not, the challenger returns $\perp$. Otherwise, he samples $b \xleftarrow{\$} \{0, 1\}$ and encrypts the message p_b to ciphertext $(\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{a}' \cdot g' + \mathbf{e}'_1, \mathbf{b}' \cdot g' + \mathbf{e}'_2 + \lfloor q/4 \rfloor \cdot \text{rdec}(p_b))$ and sends $(\mathbf{c}_1, \mathbf{c}_2)$ to the adversary $\mathcal{A}$, who then outputs $b' \in \{0, 1\}$. This game returns 1 if $b' = b$ or 0 otherwise. By assumption, $\mathcal{A}$ has advantage ϵ in this game.

Game G_1 : In this game, we make a slight modification to the Game G_0 : the public key $\mathbf{pk}$ is replaced with a truly random pair $\widetilde{\mathbf{pk}} = (\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}})$. By the $\text{RLWE}_{n,q,\ell,\chi}$ assumption, the adversary cannot distinguish $\mathbf{pk} = (\mathbf{a}, \mathbf{b})$ from uniform. It then follows that G_0 is indistinguishable from G_1 . We additionally remark that $\mathbf{pk}'$ obtained from randomizing $\widetilde{\mathbf{pk}}$ is indistinguishable from random by the same assumption.

Game G_2 : In this game, we modify G_1 as follows: instead of generating $(\mathbf{c}_1, \mathbf{c}_2)$ faithfully using the randomized public key $\mathbf{pk}'$, we generate ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ as $(\widetilde{\mathbf{a}}' \cdot g' + \mathbf{e}'_1, \widetilde{\mathbf{b}}' \cdot g' + \mathbf{e}'_2 + \lfloor q/4 \rfloor \cdot \text{rdec}(p_b))$, where $\widetilde{\mathbf{pk}}' = (\widetilde{\mathbf{a}}', \widetilde{\mathbf{b}}')$ is uniformly chosen over $R_q^\ell \times R_q^\ell$. Since $\mathbf{pk}'$ obtained from randomizing $\widetilde{\mathbf{pk}}$ is indistinguishable from random, this modification is indistinguishable to adversary $\mathcal{A}$.

Game G_3 : In this game, we generate $(\mathbf{c}_1, \mathbf{c}_2)$ as $(\mathbf{z}_1, \mathbf{z}_2 + \lfloor q/4 \rfloor \cdot \text{rdec}(p_b))$, where $(\mathbf{z}_1, \mathbf{z}_2) \in R_q^\ell \times R_q^\ell$ are uniformly random. The assumed hardness of the $\text{RLWE}_{n,q,\ell,\chi}$ problem implies that G_2 and G_3 are computationally indistinguishable.

Game G_4 : In the game, we make a conceptual modification to G_3 . Namely, we sample uniformly random $\mathbf{z}'_1 \in R_q^\ell$ and $\mathbf{z}'_2 \in R_q^\ell$ and let $(\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{z}'_1, \mathbf{z}'_2)$. It is clear that G_3 and G_4 are statistically indistinguishable. Moreover, since G_4

does not depend on the challenger's bit b any more, the advantage of $\mathcal{A}$ in this game is 0.

It then follows that the advantage ϵ of the adversary $\mathcal{A}$ is negligible. This ends the proof. $\square$

Lemma 6.3.6. The key-oblivious encryption scheme described in Section 6.3.2 is key private under key randomization defined in Section 6.3.1 under $\text{RLWE}_{n,q,\chi}$ assumption.

Proof. The proof of Lemma 6.3.6 is similar to that of Lemma 6.3.5, we briefly describe it here. As in Lemma 6.3.5, we construct a sequence of indistinguishable games G_0, G_1, G_2, G_3 , such that, $\text{Adv}_{\mathcal{A}}(G_0) = \text{Adv}_{\text{KOE},\mathcal{A}}^{\text{KPr}}(\lambda)$ and $\text{Adv}_{\mathcal{A}}(G_3) = 0$.

Game G_0 is the experiment $\text{Exp}_{\text{KOE},\mathcal{A}}^{\text{KPr}}(\lambda)$, Game G_1 modifies Game G_0 by replacing public key pk_0 with truly random pair $\widetilde{\text{pk}}_0$ while Game G_2 modifies Game G_1 by replacing public key pk_1 with another independent and random pair $\widetilde{\text{pk}}_1$. By the hardness of the $\text{RLWE}_{n,q,\ell,\chi}$ problem, these two modifications are indistinguishable to any PPT adversary. In Game G_3 , we further modify Game G_2 by generating the ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ using $\widetilde{\text{pk}}'$ chosen uniformly over $R_q^\ell \times R_q^\ell$ as in Lemma 6.3.5. By the same argument, this change is negligible to any PPT adversary. Furthermore, since G_3 does not rely on the challenger's bit b any more, the advantage of adversary in this game is 0. This ends our brief description. $\square$

6.4 Proving the RLWE Relation with Hidden Vector

In Section 6.4.1, we extend the permutation technique described in Section 4.3.1 to prove in zero-knowledge that a secret integer y is a product of secret integers $a \in \{-1, 0, 1\}$ and $g \in \{-1, 0, 1\}$. We then employ this technique, together with the techniques in Section 4.3.1, to design our zero-knowledge protocol for proving the RLWE relation with hidden vector in Section 6.4.2. Specifically, we describe how to

prove that a given vector $\mathbf{c}$ is a correct RLWE evaluation in a zero-knowledge manner, i.e., $\mathbf{c} = \mathbf{a} \cdot g + \mathbf{e}$, where the hidden vectors $\mathbf{a}, \mathbf{e}$ and element g may involve in other equations.

6.4.1 An Extended Permutation Technique

Now we are going to describe our permutation technique to demonstrate knowledge of $y \in \{-1, 0, 1\}$ such that $y = a \cdot g$ with $a \in \{-1, 0, 1\}$ and $g \in \{-1, 0, 1\}$.

Proving knowledge of y so that $y = a \cdot g$. For any $a, g \in \{-1, 0, 1\}$, define a 9-dimensional integer vector $\mathbf{mult}_3(a, g) \in \{-1, 0, 1\}^9$ in the following manner:

$$\mathbf{mult}_3(a, g) = \left([a+1]_3 \cdot [g+1]_3, [a]_3 \cdot [g+1]_3, [a-1]_3 \cdot [g+1]_3, [a+1]_3 \cdot [g]_3, \right. \\ \left. [a]_3 \cdot [g]_3, [a-1]_3 \cdot [g]_3, [a+1]_3 \cdot [g-1]_3, [a]_3 \cdot [g-1]_3, [a-1]_3 \cdot [g-1]_3 \right)^\top.$$

Now, we define a permutation $\phi_{b,e}(\cdot)$ associated to any $b, e \in \{-1, 0, 1\}$ in the way that it permutes a vector

$$\mathbf{w} = \left(w^{(-1,-1)}, w^{(0,-1)}, w^{(1,-1)}, w^{(-1,0)}, w^{(0,0)}, w^{(1,0)}, w^{(-1,1)}, w^{(0,1)}, w^{(1,1)} \right)^\top \in \mathbb{Z}^9$$

to vector

$$\phi_{b,e}(\mathbf{w}) = \left(w^{([-b-1]_3, [-e-1]_3)}, w^{([-b]_3, [-e-1]_3)}, w^{([-b+1]_3, [-e-1]_3)}, \right. \\ w^{([-b-1]_3, [-e]_3)}, w^{([-b]_3, [-e]_3)}, w^{([-b+1]_3, [-e]_3)}, \\ \left. w^{([-b-1]_3, [-e+1]_3)}, w^{([-b]_3, [-e+1]_3)}, w^{([-b+1]_3, [-e+1]_3)} \right)^\top$$

For any $a, b, g, e \in \{-1, 0, 1\}$, we identify the following equivalence:

$$\mathbf{w} = \mathbf{mult}_3(a, g) \iff \phi_{b,e}(\mathbf{w}) = \mathbf{mult}_3([a+b]_3, [g+e]_3). \quad (6.1)$$

In the setting of Stern's protocol, the above extension and permutation techniques are used to prove that $y = a \cdot g$. Specifically, one first extends y to $\mathbf{w} = \text{mult}_3(a, g)$. Next, to show the well formedness of $\mathbf{w}$, one can choose $b \xleftarrow{\$} \{-1, 0, 1\}$, $e \xleftarrow{\$} \{-1, 0, 1\}$, and prove to the verifier that $\phi_{b,e}(\mathbf{w}) = \text{mult}_3([a+b]_3, [g+e]_3)$. Due to the equivalence in (6.1), the verifier will be convinced of the well formedness of $\mathbf{w}$. At the same time, the random integers b, e perfectly hide the values of a and g , respectively, acting as "one-time pads". In addition, if a, g appear in other equations, we can use the same randomness b, e at those places.

Now we generalize the above core technique to show knowledge of a vector of the following expansion form. Based on the above discussions, our target is to obtain equivalence similar to (6.1), which is useful in Stern's framework.

Handling an expansion vector. We now tackle an expansion vector $\mathbf{y} = \text{expd}(\mathbf{a}, \mathbf{g})$ of the form $\mathbf{y} = (\mathbf{y}_0 \parallel \dots \parallel \mathbf{y}_{n-1}) \in \{-1, 0, 1\}^{n^2 \ell \delta_B}$, where

$$\mathbf{y}_i = (a_1 \cdot g_{i,1}, a_1 \cdot g_{i,2}, \dots, a_1 \cdot g_{i,\delta_B}, \dots, a_{n\ell} \cdot g_{i,1}, \dots, a_{n\ell} \cdot g_{i,\delta_B}),$$

the vector $\mathbf{g}$ is of the form $\mathbf{g} = (g_{0,1}, g_{0,2}, \dots, g_{0,\delta_B}, \dots, g_{n-1,1}, g_{n-1,2}, \dots, g_{n-1,\delta_B})^\top \in \{-1, 0, 1\}^{n\delta_B}$, vector $\mathbf{a}$ is of the form $\mathbf{a} = (a_1, \dots, a_{n\ell})^\top \in \{-1, 0, 1\}^{n\ell}$ for some positive integers n, ℓ, δ_B .

Denote $\mathbf{y} = (a_i \cdot g_{j,k})_{i \in [n\ell], j \in \{0,1,\dots,n-1\}, k \in [\delta_B]}$, we then define extension of the expansion vector $\mathbf{y}$ as

$$\text{mult}(\mathbf{a}, \mathbf{g}) = (\text{mult}_3(a_i, g_{j,k}))_{i \in [n\ell], j \in \{0,1,\dots,n-1\}, k \in [\delta_B]} \in \{-1, 0, 1\}^{9n^2 \ell \delta_B}.$$

For vector $\mathbf{e} = (e_{0,1}, e_{0,2}, \dots, e_{0,\delta_B}, \dots, e_{n-1,1}, e_{n-1,2}, \dots, e_{n-1,\delta_B})^\top \in \{-1, 0, 1\}^{n\delta_B}$ and vector $\mathbf{b} = (b_1, \dots, b_{n\ell})^\top \in \{-1, 0, 1\}^{n\ell}$, associate a permutation $\Phi_{\mathbf{b},\mathbf{e}}(\cdot)$ to them that works in the following manner. When applying to vector $\mathbf{w} \in \mathbb{Z}^{9n^2 \ell \delta_B}$ of the following

form:

$$\begin{aligned}
& \left(\mathbf{w}_{1,0,1} \|\mathbf{w}_{1,0,2}\| \cdots \|\mathbf{w}_{1,0,\delta_B}\| \cdots \|\mathbf{w}_{n\ell,0,1}\| \|\mathbf{w}_{n\ell,0,2}\| \cdots \|\mathbf{w}_{n\ell,0,\delta_B}\| \right. \\
& \quad \mathbf{w}_{1,1,1} \|\mathbf{w}_{1,1,2}\| \cdots \|\mathbf{w}_{1,1,\delta_B}\| \cdots \|\mathbf{w}_{n\ell,1,1}\| \|\mathbf{w}_{n\ell,1,2}\| \cdots \|\mathbf{w}_{n\ell,1,\delta_B}\| \\
& \quad \dots \dots \\
& \quad \left. \mathbf{w}_{1,n-1,1} \|\mathbf{w}_{1,n-1,2}\| \cdots \|\mathbf{w}_{1,n-1,\delta_B}\| \cdots \|\mathbf{w}_{n\ell,n-1,1}\| \|\mathbf{w}_{n\ell,n-1,2}\| \cdots \|\mathbf{w}_{n\ell,n-1,\delta_B}\| \right)
\end{aligned}$$

which consists of blocks of size 9, it transforms $\mathbf{w}$ into vector $\Phi_{\mathbf{b},\mathbf{e}}(\mathbf{w})$ of the following form:

$$\begin{aligned}
& \left(\phi_{b_1,e_0,1}(\mathbf{w}_{1,0,1}) \|\phi_{b_1,e_0,2}(\mathbf{w}_{1,0,2})\| \cdots \|\phi_{b_1,e_0,\delta_B}(\mathbf{w}_{1,0,\delta_B})\| \cdots \|\right. \\
& \quad \phi_{b_{n\ell},e_0,1}(\mathbf{w}_{n\ell,0,1}) \|\phi_{b_{n\ell},e_0,2}(\mathbf{w}_{n\ell,0,2})\| \cdots \|\phi_{b_{n\ell},e_0,\delta_B}(\mathbf{w}_{n\ell,0,\delta_B})\| \\
& \quad \phi_{b_1,e_1,1}(\mathbf{w}_{1,1,1}) \|\phi_{b_1,e_1,2}(\mathbf{w}_{1,1,2})\| \cdots \|\phi_{b_1,e_1,\delta_B}(\mathbf{w}_{1,1,\delta_B})\| \cdots \|\right. \\
& \quad \phi_{b_{n\ell},e_1,1}(\mathbf{w}_{n\ell,1,1}) \|\phi_{b_{n\ell},e_1,2}(\mathbf{w}_{n\ell,1,2})\| \cdots \|\phi_{b_{n\ell},e_1,\delta_B}(\mathbf{w}_{n\ell,1,\delta_B})\| \\
& \quad \dots \dots \\
& \quad \left. \phi_{b_1,e_{n-1},1}(\mathbf{w}_{1,n-1,1}) \|\phi_{b_1,e_{n-1},2}(\mathbf{w}_{1,n-1,2})\| \cdots \|\phi_{b_1,e_{n-1},\delta_B}(\mathbf{w}_{1,n-1,\delta_B})\| \cdots \|\right. \\
& \quad \left. \phi_{b_{n\ell},e_{n-1},1}(\mathbf{w}_{n\ell,n-1,1}) \|\phi_{b_{n\ell},e_{n-1},2}(\mathbf{w}_{n\ell,n-1,2})\| \cdots \|\phi_{b_{n\ell},e_{n-1},\delta_B}(\mathbf{w}_{n\ell,n-1,\delta_B})\| \right)
\end{aligned}$$

For any $\mathbf{a}, \mathbf{b} \in \{-1, 0, 1\}^{n\ell}$ and any $\mathbf{g}, \mathbf{e} \in \{-1, 0, 1\}^{n\delta_B}$, the following equivalence holds.

$$\mathbf{w} = \text{mult}(\mathbf{a}, \mathbf{g}) \iff \Phi_{\mathbf{b},\mathbf{e}}(\mathbf{w}) = \text{mult}([\mathbf{a} + \mathbf{b}]_3, [\mathbf{g} + \mathbf{e}]_3). \quad (6.2)$$

6.4.2 Proving the RLWE Relation with Hidden Vector

Coupled with the techniques described in Section 6.4.1 and Section 4.3.1, we are prepared to describe our zero-knowledge protocol for the RLWE relation with hidden vector. This protocol is adapted from the protocol for the LWE relation with hidden

matrices, first proposed by Libert et al. [106].

Let q, ℓ, B be some integers and R, R_q be two rings, which are specified as in Section 6.3.2. We consider a zero-knowledge protocol that enables a prover $\mathcal{P}$ to persuade a verifier $\mathcal{V}$ on input $\mathbf{c} \in R_q^\ell$ that $\mathcal{P}$ knows secrets $\mathbf{a} \in R_q^\ell$, $g \in R_q$ and $\mathbf{e} \in R_q^\ell$ such that g and $\mathbf{e}$ are B -bounded and

$$\mathbf{c} = \mathbf{a} \cdot g + \mathbf{e}. \quad (6.3)$$

Furthermore, this protocol should be flexible such that we can show that the secrets $\mathbf{a}, g, \mathbf{e}$ satisfy other relations.

Our strategy is to reduce the considered statement to an instance of the abstract protocol from Section 2.4.2. The reduction consists of 2 steps.

DECOMPOSING-UNIFYING. We first utilize the decomposition technique described in Section 2.4.3 together with the notations **rot** and τ in Section 2.2.5 to transform equation (6.3) into equation of the form $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u} \bmod q$, where $\mathbf{M}_0, \mathbf{u}$ are public, and the coefficients of vector $\mathbf{w}_0$ are in the set $\{-1, 0, 1\}$.

Let $\mathbf{a} = (a_1, a_2, \dots, a_\ell)^\top$, $\tau(g) = (g_0, \dots, g_{n-1})^\top$, $\mathbf{a}_i^* = \tau(\text{rdec}(a_i)) \in \{-1, 0, 1\}^{n\ell}$ $\forall i \in [\ell]$, $\mathbf{g}^* = \tau(\text{rdec}_B(g)) \in \{-1, 0, 1\}^{n\delta_B}$. Let $\mathbf{a}_i^* = (a_{i,1}, a_{i,2}, \dots, a_{i,n\ell})^\top \forall i \in [\ell]$, $\mathbf{g}^* = (g_{0,1}, \dots, g_{0,\delta_B}, \dots, g_{n-1,1}, \dots, g_{n-1,\delta_B})^\top$. We then have the following:

$$\begin{aligned} \tau(a_i \cdot g) &= \text{rot}(a_i) \cdot \tau(g) = [\tau(a_i) | \tau(a_i \cdot X) | \dots | \tau(a_i \cdot X^{n-1})] \cdot \tau(g) \\ &= \sum_{j=0}^{n-1} \tau(a_i \cdot X^j) \cdot g_j = \sum_{j=0}^{n-1} \text{rot}(X^j) \cdot \tau(a_i) \cdot g_j = \sum_{j=0}^{n-1} \text{rot}(X^j) \cdot \mathbf{H} \cdot \mathbf{a}_i^* \cdot g_j \\ &= \sum_{j=0}^{n-1} \text{rot}(X^j) \cdot \mathbf{H} \cdot (a_{i,1} \cdot g_j, \dots, a_{i,n\ell} \cdot g_j)^\top \bmod q \end{aligned}$$

Observe that, for each $k \in [n\ell]$, we have

$$\begin{aligned} a_{i,k} \cdot g_j &= a_{i,k} \cdot (B_1, \dots, B_{\delta_B}) \cdot (g_{j,1}, \dots, g_{j,\delta_B})^\top \\ &= (B_1, \dots, B_{\delta_B}) \cdot (a_{i,k} \cdot g_{j,1}, \dots, a_{i,k} \cdot g_{j,\delta_B})^\top \end{aligned}$$

Denote the following:

$$\mathbf{y}_{i,j} = (a_{i,1} \cdot g_{j,1}, \dots, a_{i,1} \cdot g_{j,\delta_B}, \dots, a_{i,n\ell} \cdot g_{j,1}, \dots, a_{i,n\ell} \cdot g_{j,\delta_B})^\top \in \{-1, 0, 1\}^{n\ell\delta_B},$$

we then obtain

$$(a_{i,1} \cdot g_j, \dots, a_{i,n\ell} \cdot g_j)^\top = \mathbf{H}_{\ell,B} \cdot \mathbf{y}_{i,j} \pmod{q}.$$

Define the following:

$$\mathbf{Q}_0 = [\text{rot}(X^0) \cdot \mathbf{H} \cdot \mathbf{H}_{\ell,B} \mid \dots \mid \text{rot}(X^{n-1}) \cdot \mathbf{H} \cdot \mathbf{H}_{\ell,B}] \in \mathbb{Z}_q^{n \times n^2\ell\delta_B},$$

$$\mathbf{y}_i = (\mathbf{y}_{i,0} \parallel \dots \parallel \mathbf{y}_{i,n-1}) = \text{expd}(\mathbf{a}_i^*, \mathbf{g}^*) \in \{-1, 0, 1\}^{n^2\ell\delta_B},$$

we then obtain:

$$\tau(a_i \cdot g) = \mathbf{Q}_0 \cdot \mathbf{y}_i \pmod{q}.$$

Let $\mathbf{e}^* = \tau(\text{rdec}_B(\mathbf{e})) \in \{-1, 0, 1\}^{n\ell\delta_B}$, $\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_0 & & & \\ & \mathbf{Q}_0 & & \\ & & \ddots & \\ & & & \mathbf{Q}_0 \end{pmatrix} \in \mathbb{Z}_q^{n\ell \times n^2\ell^2\delta_B}$. Now

equation (6.3) is equivalent to

$$\begin{aligned} \tau(\mathbf{c}) &= (\tau(a_1 \cdot g), \dots, \tau(a_\ell \cdot g))^\top + \tau(\mathbf{e}) \\ &= \mathbf{Q} \cdot (\mathbf{y}_1 \parallel \dots \parallel \mathbf{y}_\ell) + \mathbf{H}_{\ell,B} \cdot \mathbf{e}^* \pmod{q} \end{aligned}$$

Using some basic algebra, we can manipulate the above equivalent form and obtain

an unifying equation of the following form:

$$\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u} \pmod{q},$$

where $\mathbf{M}_0$ is built from the public matrices $\mathbf{Q}$ and $\mathbf{H}_{\ell,B}$, $\mathbf{u}$ is the vector $\tau(\mathbf{c})$, while $\mathbf{w}_0 = (\mathbf{y}_1 \| \cdots \| \mathbf{y}_\ell \| \mathbf{e}^*) \in \{-1, 0, 1\}^{n^2 \ell^2 \delta_B + n \ell \delta_B}$.

EXTENDING-PERMUTING. In this second step, we will transform the equation $\mathbf{M}_0 \cdot \mathbf{w}_0 = \mathbf{u} \pmod{q}$ obtained in the first step into an equation of the form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \pmod{q}$, where the secret vector $\mathbf{w}$ satisfies the conditions required by the abstract protocol. In the process, we will employ the techniques introduced in Section 6.4.1 and in Section 4.3.1.

Specifically, we extend the blocks of vector $\mathbf{w}_0 = (\mathbf{y}_1 \| \cdots \| \mathbf{y}_\ell \| \mathbf{e}^*)$ as follows.

$$\begin{aligned} \mathbf{y}_i &\mapsto \mathbf{y}'_i = \text{mult}(\mathbf{a}_i^*, \mathbf{g}^*) \in \{-1, 0, 1\}^{9n^2 \ell \delta_B}, \text{ for each } i \in [\ell]; \\ \mathbf{e}^* &\mapsto \mathbf{e}'^* = \text{enc}(\mathbf{e}^*) \in \{-1, 0, 1\}^{L_2}. \end{aligned}$$

Notice that for each $i \in [\ell]$, we have $\mathbf{y}_i = \text{expd}(\mathbf{a}_i^*, \mathbf{g}^*)$. We then form vector $\mathbf{w} = (\mathbf{y}'_1 \| \cdots \| \mathbf{y}'_\ell \| \mathbf{e}'^*) \in \{-1, 0, 1\}^L$, where

$$L = L_1 + L_2; \quad L_1 = 9n^2 \ell^2 \delta_B; \quad L_2 = 3n \ell \delta_B.$$

At the same time, we insert suitable zero-columns to matrix $\mathbf{M}_0$ to obtain matrix $\mathbf{M} \in \mathbb{Z}_q^{n \ell \times L}$ such that $\mathbf{M} \cdot \mathbf{w} = \mathbf{M}_0 \cdot \mathbf{w}_0$.

Up to this point, we have transformed the considered relations into one equation of the desired form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \pmod{q}$. We now specify the set **VALID** that contains the obtained vector $\mathbf{w}$, the set $\mathcal{S}$ and permutations $\{\Gamma_\eta : \eta \in \mathcal{S}\}$, such that the conditions in (2.1) hold.

Define **VALID** as the set of all vectors $\mathbf{v}' = (\mathbf{v}'_1 \| \cdots \| \mathbf{v}'_\ell \| \mathbf{v}'_{\ell+1}) \in \{-1, 0, 1\}^L$, satisfying

the following:

- There exist $\mathbf{a}_i^* \in \{-1, 0, 1\}^{n\ell}$ for each $i \in [\ell]$ and $\mathbf{g}^* \in \{-1, 0, 1\}^{n\delta_B}$ such that $\mathbf{v}'_i = \text{mult}(\mathbf{a}_i^*, \mathbf{g}^*)$.
- There exists $\mathbf{e}^* \in \{-1, 0, 1\}^{n\ell\delta_B}$ such that $\mathbf{v}'_{\ell+1} = \text{enc}(\mathbf{e}^*)$.

Clearly, our vector $\mathbf{w}$ belongs to this tailored set **VALID**.

Now let $\mathcal{S} = (\{-1, 0, 1\}^{n\ell})^\ell \times \{-1, 0, 1\}^{n\delta_B} \times \{-1, 0, 1\}^{n\ell\delta_B}$, and associate every element $\eta = (\mathbf{b}_1, \dots, \mathbf{b}_\ell, \mathbf{f}_1, \mathbf{f}_2) \in \mathcal{S}$ with permutation Γ_η that acts as follows. When applied to vector $\mathbf{v} = (\mathbf{v}_1 \parallel \dots \parallel \mathbf{v}_\ell \parallel \mathbf{v}_{\ell+1}) \in \mathbb{Z}^L$, where $\mathbf{v}_i \in \mathbb{Z}^{9n^2\ell\delta_B}$ for each $i \in [\ell]$ and $\mathbf{v}_{\ell+1} \in \mathbb{Z}^{L_2}$, it transforms $\mathbf{v}$ into vector

$$\Gamma_\eta(\mathbf{v}) = \left(\Phi_{\mathbf{b}_1, \mathbf{f}_1}(\mathbf{v}_1) \parallel \dots \parallel \Phi_{\mathbf{b}_\ell, \mathbf{f}_1}(\mathbf{v}_\ell) \parallel \Pi_{\mathbf{f}_2}(\mathbf{v}_{\ell+1}) \right).$$

Based on the equivalences observed in (4.4) and (6.2), it can be checked that **VALID**, $\mathcal{S}$ and Γ_η satisfy the conditions specified in (2.1). In other words, we have reduced the considered statement to an instance of the abstract protocol from Section 2.4.2. Therefore, both parties run the protocol of Figure 2.1 and we obtain a statistical ZKAoK for the considered statement, with perfect completeness, soundness error $2/3$ and communication cost $\mathcal{O}(L \cdot \log q)$, which is of order $\mathcal{O}(n^2 \cdot \log^4 n) = \tilde{\mathcal{O}}(\lambda^2)$ for the setting of parameters we use in our scheme from Section 6.3.2.

6.5 Accountable Tracing Signatures from Lattices

In this section, we construct our ATS scheme from the Ducas-Micciancio signature scheme as recalled in Section 2.3.3 and the KOE scheme as described in Section 6.3.2. We assume there is a trusted setup such that it generates parameters of the scheme. Specifically, it generates a public matrix $\mathbf{B}$, which is used for users to generate their user secret-public key pair, and two secret-public key pairs of our KOE scheme such

that the secret keys are discarded and not known by any party. The group public key then consists of three parts: (i) the parameters from the trusted setup, (ii) a verification key from the Ducas-Micciancio signature scheme, (iii) two public keys of our KOE scheme such that the group manager knows both secret keys. The issue key is the corresponding signing key of the verification key while the opening key is any one of the corresponding secret keys of the two public keys. Note that both the issue key and the opening key are generated by the group manager.

When a user joins the group, it first generates a secret-public key pair $(\mathbf{x}, p)$ such that $\mathbf{B} \cdot \mathbf{x} = p$. It then interacts with the group manager, who will determine whether user p is traceable or not. If the user is traceable, group manager sets a bit $\mathbf{tr} = 1$, randomizes the two public key generated by himself, and then generates a Ducas-Micciancio signature σ_{cert} on user public key p and the two randomized public keys $(\text{epk}_1, \text{epk}_2)$. If the user is non-traceable, group manager sets a bit $\mathbf{tr} = 0$, randomizes the two public key generated from the trusted setup, and then generates a signature on p and $\text{epk}_1, \text{epk}_2$. If it completely successfully, the group manager sends certificate $\text{cert} = (p, \text{epk}_1, \text{epk}_2, \sigma_{\text{cert}})$ to user p , registers this user to the group, and keeps himself the witness w^{escrw} that was ever used for randomization.

Once registered as a group member, the user can sign messages on behalf of the group. When signing a message, it first encrypts the public key p twice using the two randomized public keys, where the ciphertexts are denoted as $\mathbf{c}_1, \mathbf{c}_2$. The user then generates a ZKAoK such that (i) he has a valid secret key $\mathbf{x}$ corresponding to p ; (ii) he possesses a Ducas-Micciancio signature on p and $\text{epk}_1, \text{epk}_2$; and (iii) $\mathbf{c}_1, \mathbf{c}_2$ are correct ciphertexts of p under the randomized keys $\text{epk}_1, \text{epk}_2$, respectively. This ZKAoK protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via Fiat-Shamir transform [75]. The signature then consists of the NIZKAoK Π_{gs} and the two ciphertexts. Note that the ZK argument together with double encryption enables CCA-security of the underlying encryption scheme, which is known as the Naor-Yung transformation [146].

When one needs to know the validity of a signature, one simply verifies Π_{gs} . Should the need arises, the group manager can decrypt using his opening key. If a user is traceable, the opening key group manager possesses can be used to correctly identify the signer and hence group manager can revoke anonymity on any message he signed. However, if a user is non-traceable, then the user is even anonymous to the group manager.

To prevent corrupted opening, group manager is required to generate a NIZKAoK of correct opening Π_{open} . Only when Π_{open} is a valid proof, we then accept the opening result.

Furthermore, there is an additional accounting mechanism for group manager to reveal which users he had chosen to be traceable. This is done by checking the consistency of tr and the randomized public keys in user's certificate with the help of the witness w^{escrw} .

6.5.1 Description of Our ATS Scheme

Details of the scheme are described below.

Setup(λ): Given the security parameter λ , it generates the following public parameter.

- Choose parameter $n = \mathcal{O}(\lambda)$ being a power of 2, and modulus $q = \tilde{\mathcal{O}}(n^4)$, where $q = 3^k$ for some positive integer k . Let $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$.
Also, let $\ell = \lfloor \log \frac{q-1}{2} \rfloor + 1$, $m \geq 2\lceil \log q \rceil + 2$, $m_s = 4\ell + 1$, and $\bar{m} = m + k$ and $\bar{m}_s = m_s \cdot \ell$.
- Choose integer d and sequence $c_0, \dots, c_d$ as described in Section 2.3.3.
- Choose integer bounds $\beta = \tilde{\mathcal{O}}(n)$, $B = \tilde{\mathcal{O}}(\sqrt{n})$, and let χ be a B -bounded distribution over R .
- Let $\mathcal{H}_{\text{FS}} : \{0, 1\}^* \rightarrow \{1, 2, 3\}^\kappa$, where $\kappa = \omega(\log \lambda)$, be a collision-resistant

hash function, to be modelled as a random oracle in the Fiat-Shamir transformations [75].

- Let **COM** be the statistically hiding and computationally binding commitment scheme from [94], to be used in our zero-knowledge argument systems.
- Let $\mathbf{B} \xleftarrow{\$} R_q^{1 \times m}$, $\mathbf{a}_1^{(0)} \xleftarrow{\$} R_q^\ell$, $\mathbf{a}_2^{(0)} \xleftarrow{\$} R_q^\ell$, $s_{-1}, s_{-2} \leftarrow \chi$, $\mathbf{e}_{-1}, \mathbf{e}_{-2} \leftarrow \chi^\ell$.
Compute

$$\mathbf{b}_1^{(0)} = \mathbf{a}_1^{(0)} \cdot s_{-1} + \mathbf{e}_{-1} \in R_q^\ell; \quad \mathbf{b}_2^{(0)} = \mathbf{a}_2^{(0)} \cdot s_{-2} + \mathbf{e}_{-2} \in R_q^\ell.$$

This algorithm outputs the public parameter **pp**:

$$\{ n, q, k, R, R_q, \ell, m, m_s, \overline{m}, \overline{m}_s, d, c_0, \dots, c_d, \beta, B, \chi, \\ \mathcal{H}_{\text{FS}}, \kappa, \text{COM}, \mathbf{B}, \{\mathbf{a}_i^{(0)}, \mathbf{b}_i^{(0)}\}_{i \in \{1,2\}} \}.$$

pp is implicit for all algorithms below if not explicitly mentioned.

GKeyGen(pp): On input **pp**, **GM** proceeds as follows.

- Generate verification key

$$\mathbf{A}, \mathbf{F}_0 \in R_q^{1 \times \overline{m}}; \quad \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]} \in R_q^{1 \times k}; \quad \mathbf{F} \in R_q^{1 \times \ell}; \quad \mathbf{F}_1 \in R_q^{1 \times \overline{m}_s}; \quad u \in R_q$$

and signing key $\mathbf{R} \in R_q^{m \times k}$ for the Ducas-Micciancio signature scheme as described in Section 2.3.3.

- Initialize the Naor-Yung double-encryption mechanism [146] with the key-oblivious encryption scheme described in Section 6.3.2. Specifically, sample $s_1, s_2 \leftarrow \chi$, $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi^\ell$, $\mathbf{a}_1^{(1)} \xleftarrow{\$} R_q^\ell$, $\mathbf{a}_2^{(1)} \xleftarrow{\$} R_q^\ell$ and compute

$$\mathbf{b}_1^{(1)} = \mathbf{a}_1^{(1)} \cdot s_1 + \mathbf{e}_1 \in R_q^\ell; \quad \mathbf{b}_2^{(1)} = \mathbf{a}_2^{(1)} \cdot s_2 + \mathbf{e}_2 \in R_q^\ell.$$

Set the group public key **gpk**, the issue key **ik** and the opening key **ok** as follows:

$$\mathbf{gpk} = \{\mathbf{pp}, \mathbf{A}, \{\mathbf{A}_{[j]}\}_{j=0}^d, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u, \mathbf{a}_1^{(1)}, \mathbf{b}_1^{(1)}, \mathbf{a}_2^{(1)}, \mathbf{b}_2^{(1)}\},$$

$$\mathbf{ik} = \mathbf{R}, \quad \mathbf{ok} = (s_1, \mathbf{e}_1).$$

GM then makes **gpk** public, initializes his internal state $S = 0$ and the registration table **reg** = $\emptyset$.

UKeyGen(pp): The user samples $\mathbf{x} \in R^m$, whose coefficients are uniformly random in the set $\{-1, 0, 1\}$. Then he computes $p = \mathbf{B} \cdot \mathbf{x} \in R_q$. Set **upk** = p and **usk** = $\mathbf{x}$.

Enroll(gpk, ik, upk, tr): Upon receiving a user public key **upk** from a user, GM determines the value of the bit $\mathbf{tr} \in \{0, 1\}$, indicating whether the user is traceable or not. He then does the following:

- Randomize two pairs of public keys $(\mathbf{a}_1^{(\mathbf{tr})}, \mathbf{b}_1^{(\mathbf{tr})})$ and $(\mathbf{a}_2^{(\mathbf{tr})}, \mathbf{b}_2^{(\mathbf{tr})})$ as described in Section 6.3.2. Specifically, sample $g_1, g_2 \leftarrow \chi$, $\mathbf{e}_{1,1}, \mathbf{e}_{1,2} \leftarrow \chi^\ell$, $\mathbf{e}_{2,1}, \mathbf{e}_{2,2} \leftarrow \chi^\ell$. For each $i \in \{1, 2\}$, compute

$$\mathbf{epk}_i = (\mathbf{a}'_i, \mathbf{b}'_i) = (\mathbf{a}_i^{(\mathbf{tr})} \cdot g_i + \mathbf{e}_{i,1}, \mathbf{b}_i^{(\mathbf{tr})} \cdot g_i + \mathbf{e}_{i,2}) \in R_q^\ell \times R_q^\ell. \quad (6.4)$$

- Set the tag $t = (t_0, t_1, \dots, t_{c_d-1})^\top \in \mathcal{T}_d$, where $S = \sum_{j=0}^{c_d-1} 2^j \cdot t_j$, and compute $\mathbf{A}_t = [\mathbf{A} | \mathbf{A}_{[0]} + \sum_{i=1}^d t_{[i]} \mathbf{A}_{[i]}] \in R_q^{1 \times (\overline{m}+k)}$.
- Let $\mathbf{m} = (p \| \mathbf{a}'_1 \| \mathbf{b}'_1 \| \mathbf{a}'_2 \| \mathbf{b}'_2) \in R_q^{m_s}$.
- Using the issue key $\mathbf{ik} = \mathbf{R}$, generate a Ducas-Micciancio signature $\sigma_{\text{cert}} = (t, \mathbf{r}, \mathbf{v})$ on message $\mathbf{rdec}(\mathbf{m}) \in R^{\overline{m}_s}$ - whose coefficients are in $\{-1, 0, 1\}$.

As described in Section 2.3.3, one has $\mathbf{r} \in R^{\overline{m}}$, $\mathbf{v} \in R^{\overline{m}+k}$ and

$$\begin{cases} \mathbf{A}_t \cdot \mathbf{v} = \mathbf{F} \cdot \text{rdec}(\mathbf{F}_0 \cdot \mathbf{r} + \mathbf{F}_1 \cdot \text{rdec}(\mathbf{m})) + u, \\ \|\mathbf{r}\|_\infty \leq \beta, \quad \|\mathbf{v}\|_\infty \leq \beta. \end{cases} \quad (6.5)$$

Set certificate $\mathbf{cert}$ and w^{escrw} as follows:

$$\mathbf{cert} = (p, \mathbf{a}'_1, \mathbf{b}'_1, \mathbf{a}'_2, \mathbf{b}'_2, t, \mathbf{r}, \mathbf{v}), \quad w^{\text{escrw}} = (g_1, \mathbf{e}_{1,1}, \mathbf{e}_{1,2}, g_2, \mathbf{e}_{2,1}, \mathbf{e}_{2,2}).$$

GM sends $\mathbf{cert}$ to the user p , stores $\mathbf{reg}[S] = (p, \mathbf{tr}, w^{\text{escrw}})$ and updates S to $S + 1$.

Sign(gpk, cert, usk, M): To sign a message $M \in \{0, 1\}^*$ using the certificate $\mathbf{cert} = (p, \mathbf{a}'_1, \mathbf{b}'_1, \mathbf{a}'_2, \mathbf{b}'_2, t, \mathbf{r}, \mathbf{v})$ and $\mathbf{usk} = \mathbf{x}$, the user proceeds as follows.

- Encrypt the ring vector $\text{rdec}(p) \in R_q^\ell$ whose coefficients are in $\{-1, 0, 1\}$ twice. Namely, sample $g'_1, g'_2 \leftarrow \chi$, $\mathbf{e}'_{1,1}, \mathbf{e}'_{1,2} \leftarrow \chi^\ell$, and $\mathbf{e}'_{2,1}, \mathbf{e}'_{2,2} \leftarrow \chi^\ell$. For each $i \in \{1, 2\}$, compute

$$\begin{aligned} \mathbf{c}_i &= (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}) \\ &= (\mathbf{a}'_i \cdot g'_i + \mathbf{e}'_{i,1}, \quad \mathbf{b}'_i \cdot g'_i + \mathbf{e}'_{i,2} + \lfloor q/4 \rfloor \cdot \text{rdec}(p)) \in R_q^\ell \times R_q^\ell. \end{aligned}$$

- Generate a NIZKAoK Π_{gs} to demonstrate the possession of a valid tuple

$$\zeta = (p, \mathbf{a}'_1, \mathbf{b}'_1, \mathbf{a}'_2, \mathbf{b}'_2, t, \mathbf{r}, \mathbf{v}, \mathbf{x}, g'_1, \mathbf{e}'_{1,1}, \mathbf{e}'_{1,2}, g'_2, \mathbf{e}'_{2,1}, \mathbf{e}'_{2,2}) \quad (6.6)$$

such that

- (i) The conditions from (6.5) hold.
- (ii) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\text{rdec}(p)$ with B -bounded randomness $g'_1, \mathbf{e}'_{1,1}, \mathbf{e}'_{1,2}$ and $g'_2, \mathbf{e}'_{2,1}, \mathbf{e}'_{2,2}$, respectively.

(iii) $\|\mathbf{x}\|_\infty \leq 1$ and $\mathbf{B} \cdot \mathbf{x} = p$.

This is done by running the argument system described in Section 6.5.3. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via Fiat-Shamir heuristic [75] as a triple $\Pi_{\text{gs}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}_i\}_{i=1}^\kappa)$ where the challenge CH is generated as $\text{CH} = \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^\kappa, \xi)$ with

$$\xi = (\mathbf{A}, \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]}, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u, \mathbf{B}, \mathbf{c}_1, \mathbf{c}_2) \quad (6.7)$$

- Output the group signature $\Sigma = (\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$.

Verify(gpk, M , Σ): Given the inputs, this algorithm proceeds as follows.

- Parse Σ as $\Sigma = (\{\text{CMT}_i\}_{i=1}^\kappa, (Ch_1, \dots, Ch_\kappa), \{\text{RSP}_i\}_{i=1}^\kappa, \mathbf{c}_1, \mathbf{c}_2)$.
If $(Ch_1, \dots, Ch_\kappa) \neq \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^\kappa, \xi)$, then return 0, where ξ is as in (6.7).
- For each $i \in [\kappa]$, run the verification phase of the protocol in Section 6.5.3 to check the validity of RSP_i with respect to CMT_i and Ch_i . If any of the conditions does not hold, then return 0.
- Return 1.

Open(gpk, ok, M , Σ): Let $\text{ok} = (s_1, \mathbf{e}_1)$ and $\Sigma = (\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$. This algorithm then does the following.

- Use s_1 to decrypt $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ as the decryption algorithm described in Section 6.3.2, which outputs $p' \in R_q$.
- It then searches the registration information. If **reg** does not include an element p' , then return $\perp$.

- Otherwise, generate a NIZKAoK Π_{open} to demonstrate the possession of a tuple $(s_1, \mathbf{e}_1, \mathbf{y}) \in R_q \times R_q^\ell \times R_q^\ell$ such that the following hold.

$$\begin{cases} \|s_1\|_\infty \leq B; \|\mathbf{e}_1\|_\infty \leq B; \|\mathbf{y}\|_\infty \leq \lceil q/10 \rceil; \\ \mathbf{a}_1^{(1)} \cdot s_1 + \mathbf{e}_1 = \mathbf{b}_1^{(1)}; \\ \mathbf{c}_{1,2} - \mathbf{c}_{1,1} \cdot s_1 = \mathbf{y} + \lfloor q/4 \rfloor \cdot \text{rdec}(p'). \end{cases} \quad (6.8)$$

We remark that conditions in (6.8) involve only linear secret objects with bounded norms, and can be handled using the Stern-like techniques from Sections 6.4.2 and 6.5.3. As a result, we can obtain a statistical ZKAoK for the considered statement. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\text{open}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}\}_{i=1}^\kappa)$, where

$$\text{CH} = \mathcal{H}_{\text{FS}}(\{\text{CMT}_i\}_{i=1}^\kappa, \mathbf{a}_1^{(1)}, \mathbf{b}_1^{(1)}, M, \Sigma, p') \in \{1, 2, 3\}^\kappa. \quad (6.9)$$

- Output (p', Π_{open}) .

Judge(gpk, $M, \Sigma, p', \Pi_{\text{open}}$): Given all the inputs, this algorithm does the following.

- If **Verify** algorithm outputs 0 or $p' = \perp$, return 0.
- This algorithm then verifies the argument Π_{open} with respect to common input $(\mathbf{a}_1^{(1)}, \mathbf{b}_1^{(1)}, M, \Sigma, p')$, in a similar manner as in algorithm **Verify**. If Π_{open} does not verify, then return 0.
- Else return 1.

Account(gpk, cert, w^{escrw} , tr): Let the certificate be $\text{cert} = (p, \mathbf{a}'_1, \mathbf{b}'_1, \mathbf{a}'_2, \mathbf{b}'_2, t, \mathbf{r}, \mathbf{v})$ and witness be $w^{\text{escrw}} = (g_1, \mathbf{e}_{1,1}, \mathbf{e}_{1,2}, g_2, \mathbf{e}_{2,1}, \mathbf{e}_{2,2})$ and the bit **tr**, this algorithm proceeds as follows.

- It checks whether $(t, \mathbf{r}, \mathbf{v})$ is a valid Ducas-Micciancio signature on the message $(p, \mathbf{a}'_1, \mathbf{b}'_1, \mathbf{a}'_2, \mathbf{b}'_2)$ or not. Specifically, it verifies whether **cert** satisfies the conditions in (6.5) or not. If not, return 0.
- Otherwise, it then checks that $(\mathbf{a}'_1, \mathbf{b}'_1)$ and $(\mathbf{a}'_2, \mathbf{b}'_2)$ are randomization of $(\mathbf{a}_1^{(\text{tr})}, \mathbf{b}_1^{(\text{tr})})$ and $(\mathbf{a}_2^{(\text{tr})}, \mathbf{b}_2^{(\text{tr})})$ with respect to randomness $(g_1, \mathbf{e}_{1,1}, \mathbf{e}_{1,2})$ and $(g_2, \mathbf{e}_{2,1}, \mathbf{e}_{2,2})$, respectively. Specifically, it verifies whether the conditions in (6.4) hold or not. If not, return 0.
- Else return 1.

6.5.2 Security Analysis of Our Scheme

EFFICIENCY. We first analyze the efficiency of the scheme described in Section 6.5.1, with respect to security parameter λ .

- The public key **gpk** has bit-size $\mathcal{O}(\lambda \cdot \log^3 \lambda) = \tilde{\mathcal{O}}(\lambda)$.
- The certificate **cert** has bit-size $\mathcal{O}(\lambda \cdot \log^2 \lambda) = \tilde{\mathcal{O}}(\lambda)$.
- The size of a signature Σ is dominated by that of the Stern-like NIZKAoK Π_{gs} , which is $\mathcal{O}(L \cdot \log q) \cdot \omega(\log \lambda)$, where L denotes the bit-size of a vector $\mathbf{w} \in \text{VALID}$ as described in Section 6.5.3. Recall $\mathcal{O}(L \cdot \log q) = \mathcal{O}(\lambda^2 \cdot \log^3 \lambda)$. As a result, Σ has bit-size $\mathcal{O}(\lambda^2 \cdot \log^3 \lambda) \cdot \omega(\log \lambda) = \tilde{\mathcal{O}}(\lambda^2)$.
- The Stern-like NIZKAoK Π_{open} has bit-size $\mathcal{O}(\lambda \cdot \log^3 \lambda) \cdot \omega(\log \lambda) = \tilde{\mathcal{O}}(\lambda)$.

CORRECTNESS. For an honestly generated signature Σ for message M , we first show that the **Verify** algorithm always outputs 1. Due to the honest behavior of the user, when signing a message on behalf of the group, this user is able to demonstrate the possession of a valid tuple ζ of the form (6.6). Then, by the perfect completeness of our argument system, Π_{gs} will be accepted by the **Verify** algorithm with probability 1.

If an honest user is traceable, then $\text{Account}(\text{gpk}, \text{cert}, w^{\text{escrw}}, 1)$ will output 1, implied by the correctness of Ducas-Micciancio signature scheme and honest behaviour of group manager. As for the correctness of the **Open** algorithm, note that $\mathbf{c}_{1,2} - \mathbf{c}_{1,1} \cdot s_1 =$

$$(\mathbf{b}_1^{(\text{tr})} - \mathbf{a}_1^{(\text{tr})} \cdot s_1) \cdot g_1 \cdot g'_1 + \mathbf{e}_{1,2} \cdot g'_1 - \mathbf{e}_{1,1} \cdot s_1 \cdot g'_1 + \mathbf{e}'_{1,2} - \mathbf{e}'_{1,1} \cdot s_1 + \lfloor q/4 \rfloor \cdot \text{rdec}(p),$$

denoted as $\tilde{\mathbf{e}} + \lfloor q/4 \rfloor \cdot \text{rdec}(p)$. In this case, $\text{tr} = 1$, $\mathbf{b}_1^{(\text{tr})} - \mathbf{a}_1^{(\text{tr})} \cdot s_1 = \mathbf{e}_1$, and $\|\tilde{\mathbf{e}}\|_\infty \leq \lceil \frac{q}{10} \rceil$. By the correctness of our key-oblivious encryption scheme from Section 6.3.2, the decryption can recover $\text{rdec}(p)$ and hence the real signer. Thus, correctness of the **Open** algorithm follows. What is more, by the perfect completeness of our argument systems, Π_{open} will be accepted by the **Judge** algorithm with probability 1.

If an honest user is non-traceable, then again $\text{Account}(\text{gpk}, \text{cert}, w^{\text{escrw}}, 1)$ will output 1. For the **Open** algorithm, since $\mathbf{b}_1^{(0)} - \mathbf{a}_1^{(0)} \cdot s_1 = \mathbf{a}_1^{(0)} \cdot (s_{-1} - s_1) + \mathbf{e}_{-1}$, denote

$$\mathbf{c}_{1,2} - \mathbf{c}_{1,1} \cdot s_1 = \mathbf{a}_1^{(0)} \cdot (s_{-1} - s_1) \cdot g_1 \cdot g'_1 + \tilde{\mathbf{e}} + \lfloor q/4 \rfloor \cdot \text{rdec}(p),$$

where $\|\tilde{\mathbf{e}}\|_\infty \leq \lceil \frac{q}{10} \rceil$. Note that $\mathbf{a}_1^{(0)} \xleftarrow{\$} R_q^\ell$, and $s_{-1} \neq s_1$ with overwhelming probability. Over the randomness of g_1, g'_1 , the decryption algorithm described in Section 6.3.2 will output a random element $p' \in R_q$. Then, with overwhelming probability, p' is not in the registration table and the **Open** algorithm outputs $\perp$. It then follows that our scheme is correct.

SECURITY. In Theorem 6.5.2, we prove that our scheme satisfies the security requirements of accountable tracing signatures described in Section 6.2.2. For the proof of traceability and non-frameability, we will use the Lemma 4.4.1 described in Section 4.4.2, which we recall here for convenience.

Lemma 6.5.1. Let $\mathbf{B} \in R_q^{1 \times m}$, where $m \geq 2\lceil \log q \rceil + 2$. If $\mathbf{x}$ is a uniformly random element of R^m such that $\|\mathbf{x}\|_\infty \leq 1$, then with probability at least $1 - 2^{-n}$, there exists another $\mathbf{x}' \in R^m$ such that $\|\mathbf{x}'\|_\infty \leq 1$ and $\mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{x}' \in R_q$.

Theorem 6.5.2. Assume that the Stern-like argument systems used in Section 6.5.1 are simulation-sound. Then, in the random oracle model, the given group signature scheme satisfies anonymity under tracing, traceability, non-frameability, anonymity with accountability, and trace-obliviousness under the RLWE and RSIS assumptions.

The proof of Theorem 6.5.2 is established by Lemmas 6.5.3-6.5.7 given below.

Lemma 6.5.3. Assume the RLWE problem is hard, then the given accountable tracing scheme is anonymous under tracing in the random oracle model.

Proof. We prove this lemma using a series of indistinguishable games. In the first game, the challenger runs the experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AuT}-0}(\lambda)$ while in the last game, the challenger runs the experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AuT}-1}(\lambda)$. Denote W_i be the event that the adversary outputs 1 in Game i .

Game 0: This is exactly the experiment $\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AuT}-0}(\lambda)$, where the adversary receives a challenged signature $(\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*) \leftarrow \text{Sign}(\text{gpk}, \text{cert}_0, \text{usk}_0, M)$ in the challenge phase with $p_0 = \mathbf{B} \cdot \text{usk}_0$. So $\Pr[W_0] = \Pr[\mathbf{Exp}_{\text{ATS},\mathcal{A}}^{\text{AuT}-0}(\lambda) = 1]$.

Game 1: This game is the same as Game 0 with one exception: the challenger will keep decryption key $(s_2, \mathbf{e}_2)$ secret (by himself) instead of erasing it. However, the view of the adversary $\mathcal{A}$ is still the same as in Game 0. Therefore, we obtain that $\Pr[W_0] = \Pr[W_1]$.

Game 2: This game is identical to Game 1 except that it generates simulated proofs for the opening oracle queries by programming the random oracle $\mathcal{H}_{\text{FS}}$. Note that the challenger still follows the original game (that is, it uses s_1 to decrypt $\mathbf{c}_1$) to identify the real signer. The view of the adversary, however, is statistically indistinguishable between Game 1 and Game 2 by the statistical zero-knowledge property of our argument system generating Π_{Open} . Therefore we have the following: $|\Pr[W_1] - \Pr[W_2]| = \text{negl}(\lambda)$.

Game 3: This game modifies Game 2 as follows. It uses s_2 instead of s_1 to answer the opening oracle queries. In other words, it now uses $\mathbf{s}_2$ to decrypt $\mathbf{c}_2$ to identify the signer. The view of $\mathcal{A}$ is the same as in Game 2 until event F_1 , where $\mathcal{A}$ queries the opening oracle a valid signature $(\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting distinct messages, happens. Since F_1 breaks the soundness of our argument generating Π_{gs} , we have $|\Pr[W_2] - \Pr[W_3]| \leq \Pr[F_1] \leq \mathbf{Adv}_{\Pi_{\text{gs}}}^{\text{sound}}(\lambda) = \text{negl}(\lambda)$.

Game 4: This game changes Game 3 in the following way. It generates a simulated proof Π_{gs}^* in the challenge phase by programming the random oracle $\mathcal{H}_{\text{FS}}$ even though the challenger has the correct witness to generate a real proof. The view is indistinguishable to $\mathcal{A}$ by the statistical zero-knowledge property of our argument system generating Π_{gs}^* . Therefore we have $|\Pr[W_3] - \Pr[W_4]| = \text{negl}(\lambda)$.

Game 5: In this game, we modify Game 4 by modifying the distribution of the challenged signature $\Sigma^* = (\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ as follows. For $i \in \{0, 1\}$, parse $\text{cert}_i = (p_i, \mathbf{a}'_{1,i}, \mathbf{b}'_{1,i}, \mathbf{a}'_{2,i}, \mathbf{b}'_{2,i}, t_i, \mathbf{r}_i, \mathbf{v}_i)$. Recall that in Game 4, both $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ encrypt the same message, i.e., $\text{rdec}(p_0)$, under the randomized key $(\mathbf{a}'_{1,0}, \mathbf{b}'_{1,0})$ and $(\mathbf{a}'_{2,0}, \mathbf{b}'_{2,0})$, respectively. Here we change $\mathbf{c}_1^*$ to be encryption of $\text{rdec}(p_1)$ and keep $\mathbf{c}_2^*$ unchanged. By the semantic security under key randomization of our key oblivious encryption scheme for public key $(\mathbf{a}_1^{(1)}, \mathbf{b}_1^{(1)})$ (which is implied by the RLWE assumption since we no longer use s_1 to open signatures), the change made in this game is negligible to the adversary. Therefore we have $|\Pr[W_4] - \Pr[W_5]| = \text{negl}(\lambda)$.

Game 6: In this game, we further modify the distribution of the challenged signature Σ^* . We change $\mathbf{c}_1^*$ to be encryption of $\text{rdec}(p_1)$ under a fresh and then randomized key. By the property of key privacy under key randomization of our key-oblivious encryption scheme, the change made in this game is negligible to the adversary. Therefore we have $|\Pr[W_5] - \Pr[W_6]| = \text{negl}(\lambda)$.

Game 7: In this game, we again modify the distribution of the challenged signature Σ^* . We change $\mathbf{c}_1^*$ to be encryption of $\text{rdec}(p_1)$ under the randomized key $(\mathbf{a}'_{1,1}, \mathbf{b}'_{1,1})$. By the same argument of indistinguishability between Game 6 and Game 5, we have $|\Pr[W_6] - \Pr[W_7]| = \text{negl}(\lambda)$.

Game 8: This game follows Game 7 with one change: it switches back to use s_1 for the opening oracle queries and erases $(s_2, \mathbf{e}_2)$ again. This modification is indistinguishable to the adversary until event F_2 , where $\mathcal{A}$ queries the opening oracle a valid signature $(\Pi_{\text{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting different messages, happens. However event F_2 breaks the simulation soundness of the argument system generating Π_{gs} . Therefore we have $|\Pr[W_7] - \Pr[W_8]| \leq \text{Adv}_{\Pi_{\text{gs}}}^{\text{ss}}(\lambda) = \text{negl}(\lambda)$.

Game 9: In this game, we modify Game 8 by modifying the distribution of the challenged signature $\Sigma^* = (\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ again. It changes $\mathbf{c}_2^*$ to be encryption of $\text{rdec}(p_1)$ under the randomized key $(\mathbf{a}'_{2,1}, \mathbf{b}'_{2,1})$ in the challenge phase. By the same argument of indistinguishability from Game 4 to Game 7, we have $|\Pr[W_8] - \Pr[W_9]| = \text{negl}(\lambda)$.

Game 10: Note that in Game 9, both $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ encrypt the same message, i.e., $\text{rdec}(p_1)$, under the randomized key $(\mathbf{a}'_{1,1}, \mathbf{b}'_{1,1})$ and $(\mathbf{a}'_{2,1}, \mathbf{b}'_{2,1})$, respectively. Thus, the challenger has correct witness to generate Π_{gs}^* . In this game, we modify Game 9 by switching back to a real proof Π_{gs}^* in the challenge phase. By the statistical zero-knowledge property of our argument system generating Π_{gs}^* , the view of $\mathcal{A}$ is indistinguishable between Game 9 and Game 10. Therefore we have $|\Pr[W_9] - \Pr[W_{10}]| = \text{negl}(\lambda)$.

Game 11: This game changes Game 10 in one aspect. It produces real proofs for the opening oracle queries. By the statistical zero-knowledge property of our argument system generating Π_{Open} , we have Game 10 and Game 11 are statistically

indistinguishable to the adversary. Hence $|\Pr[W_{10}] - \Pr[W_{11}]| = \text{negl}(\lambda)$. This is actually the experiment $\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-1}(\lambda)$. Therefore, the following equation holds:

$$\Pr[W_{11}] = \Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-1}(\lambda) = 1].$$

As a result, we have that

$$|\Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{AuT}-0}(\lambda) = 1]| = \text{negl}(\lambda),$$

and hence our scheme is anonymous under tracing. $\square$

Lemma 6.5.4. Assume the RSIS problem is hard, then the given accountable tracing scheme is traceable in the random oracle model.

Proof. We show that the success probability ϵ of $\mathcal{A}$ against traceability is negligible by the unforgeability of the Ducas-Micciancio signature scheme from Section 2.3.3, which is based on the hardness of RSIS problem, or by the hardness of solving a RSIS instance directly.

Let $\mathcal{C}$ be the challenger and honestly run the experiment $\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{Trace}}(\lambda)$. When $\mathcal{A}$ halts, it outputs $(M^*, \Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$. Let us consider the case that $\mathcal{A}$ wins the experiment. Parse $\Pi_{\text{gs}}^* = (\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}_i^*\}_{i=1}^\kappa)$. Let

$$\xi^* = (\mathbf{A}, \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]}, \mathbf{F}, \mathbf{F}_0, \mathbf{F}_1, u, \mathbf{B}, \mathbf{c}_1^*, \mathbf{c}_2^*).$$

Then $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ and RSP_i^* is a valid response w.r.t. CMT_i^* and CH_i^* for $i \in [\kappa]$ by the fact that $\mathcal{A}$ wins and hence $(\Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ is a valid signature on message M^* .

We claim that $\mathcal{A}$ had queried $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ to the hash oracle $\mathcal{H}_{\text{FS}}$ with overwhelming probability. Otherwise, the probability of guessing correctly the value of $\mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ is at most $3^{-\kappa}$, which is negligible. Therefore, with probability $\epsilon' = \epsilon - 3^{-\kappa}$, $\mathcal{A}$ had queried the hash oracle $\mathcal{H}_{\text{FS}}$. Denote by $\theta^* \in \{1, 2, \dots, Q_H\}$

the index of this specific query, where Q_H is the total number of hash queries made by $\mathcal{A}$.

Challenger $\mathcal{C}$ then runs at most $32 \cdot Q_H / \epsilon'$ executions of $\mathcal{A}$. For each new run, it is exactly the same as the original run until the point of θ^* -th query to the hash oracle $\mathcal{H}_{FS}$. From this point on, $\mathcal{C}$ replies $\mathcal{A}$'s hash queries with uniformly random and independent values for each new run. This guarantees that the input of θ^* -th query $\mathcal{A}$ made to $\mathcal{H}_{FS}$ is the tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ for each new run while the output of this hash query is uniformly random and independent for each new run. To this point, by the forking lemma of Brickell et al. [47], with probability $\geq 1/2$, $\mathcal{B}$ obtains 3-fork involving the same tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \xi^*)$ with pairwise distinct hash values $\text{CH}_{\theta^*}^{(1)}, \text{CH}_{\theta^*}^{(2)}, \text{CH}_{\theta^*}^{(3)} \in \{1, 2, 3\}^\kappa$ and corresponding valid responses $\text{RSP}_{\theta^*}^{(1)}, \text{RSP}_{\theta^*}^{(2)}, \text{RSP}_{\theta^*}^{(3)}$. A simple calculation shows that with probability $1 - (\frac{7}{9})^\kappa$, we have $\{\text{CH}_{\theta^*,j}^{(1)}, \text{CH}_{\theta^*,j}^{(2)}, \text{CH}_{\theta^*,j}^{(3)}\} = \{1, 2, 3\}$ for some $j \in \{1, 2, \dots, \kappa\}$.

Therefore, $\text{RSP}_{\theta^*,j}^{(1)}, \text{RSP}_{\theta^*,j}^{(2)}, \text{RSP}_{\theta^*,j}^{(3)}$ are 3 valid responses for all the challenges 1, 2, 3 w.r.t. the same commitment CMT_j^* . Since COM is computationally binding, $\mathcal{C}$ is able to extract the witness

$$\zeta^* = (p^*, \mathbf{a}_1^*, \mathbf{b}_1^*, \mathbf{a}_2^*, \mathbf{b}_2^*, t^*, \mathbf{r}^*, \mathbf{v}^*, \mathbf{x}^*, g_1^*, \mathbf{e}_{1,1}^*, \mathbf{e}_{1,2}^*, g_2^*, \mathbf{e}_{2,1}^*, \mathbf{e}_{2,2}^*)$$

such that $t^* \in \mathcal{T}_d$, $\mathbf{r}^*, \mathbf{v}^*$ have infinity bound β , $\mathbf{g}_1^*, \mathbf{e}_{1,1}^*, \mathbf{e}_{1,2}^*, g_2^*, \mathbf{e}_{2,1}^*, \mathbf{e}_{2,2}^*$ have infinity bound B , $\mathbf{x}^*$ has infinity bound 1; equations $\mathbf{B} \cdot \mathbf{x}^* = p^*$ and

$$\mathbf{A}_{t^*} \cdot \mathbf{v}^* = u + \mathbf{F} \cdot \text{rdec}(\mathbf{F}_0 \cdot \mathbf{r}^* + \mathbf{F}_1 \cdot \text{rdec}(p^* \|\mathbf{a}_1^* \|\mathbf{b}_1^* \|\mathbf{a}_2^* \|\mathbf{b}_2^*))$$

hold; $\mathbf{c}_1^*, \mathbf{c}_2^*$ are encryptions of $\text{rdec}(p^*)$ with randomness $(g_1^*, \mathbf{e}_{1,1}^*, \mathbf{e}_{1,2}^*), (g_2^*, \mathbf{e}_{2,1}^*, \mathbf{e}_{2,2}^*)$, respectively.

Since $\mathcal{A}$ wins the game, either we have (i) the **Open** algorithm outputs $\perp$ or (ii) the **Open** algorithm output $(p', \Pi_{\text{open}}^*)$ with $p' \neq \perp$ but the **Judge** algorithm rejects the

opening result.

By the unforgeability of the underlying signature scheme, with overwhelming probability, $(p^*, \mathbf{a}_1^*, \mathbf{b}_1^*, \mathbf{a}_2^*, \mathbf{b}_2^*, t^*, \mathbf{r}^*, \mathbf{v}^*)$ is a certificate returned by the **Enroll** oracle. In other words, p^* is a registered user. If p^* is a non-traceable user, then $\mathcal{A}$ does not hold the user secret key of p^* , denoted as $\mathbf{x}^*$. Note that this is ensured by the definition of traceability described in Section 6.2.2. With probability $\geq 1/2$, $\mathbf{x}^* \neq \mathbf{x}^*$ by Lemma 6.5.1. In the case they are not equal, we obtain a non-zero vector $\mathbf{y} = \mathbf{x}^* - \mathbf{x}^*$ such that $\mathbf{B} \cdot \mathbf{y} = 0$ and $\mathbf{y}$ has infinity bound 2, since both $\mathbf{x}^*$ and $\mathbf{x}^*$ have infinity bound 1. This solves a **RSIS** instance. Therefore, the **Open** algorithm outputs $\perp$ with negligible probability. In other words, case (i) happens with negligible probability. On the other hand, if p^* is a traceable user. Then by the correctness of the underlying encryption scheme, the **Open** algorithm will output p^* . Furthermore, by the honest behaviour of decryption (performed by the honest challenger), the **Judge** algorithm always outputs 1. This implies case (ii) happens with negligible probability. This concludes the proof. $\square$

Lemma 6.5.5. Assume the **RSIS** problem is hard, then the given accountable tracing scheme is non-frameable in the random oracle model.

Proof. We show that the success probability ϵ of $\mathcal{A}$ against non-frameability is negligible by the hardness of solving a **RSIS** instance directly.

Let $\mathcal{C}$ be the challenger and faithfully run the experiment $\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{NF}}(\lambda)$. When $\mathcal{A}$ halts, it outputs $(M^*, \Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*, p^*, \Pi_{\text{open}}^*)$. Let us consider the case that $\mathcal{A}$ wins the experiment.

The fact that $\mathcal{A}$ wins the game implies $(M^*, \Pi_{\text{gs}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ is a valid message-signature pair that was not queried before. By the same extraction technique as in Lemma 6.5.4, we can extract witness $\mathbf{x}' \in R_q^m$ and $p' \in R_q$ such that $\mathbf{x}'$ has infinity bound 1, $\mathbf{B} \cdot \mathbf{x}' = p'$ and $\mathbf{c}_1^*, \mathbf{c}_2^*$ are correct encryptions of $\text{rdec}(p')$. By the correctness of the underlying encryption scheme, $\mathbf{c}_1^*$ will be decrypted to p' .

The fact that $\mathcal{A}$ wins the game also implies $(p^*, \Pi_{\text{open}}^*)$ is accepted by the **Judge** algorithm. It follows from the soundness of the argument system used to generate Π_{open}^* that $\mathbf{c}_1^*$ will be decrypted to p^* . Therefore, we have $p' = p^*$. Note that $\mathcal{A}$ wins the game also implies that $\mathcal{A}$ does not know the user secret key $\mathbf{x}^*$ that corresponds to p^* . Thus we obtain: $\mathbf{B} \cdot \mathbf{x}' = p' = p^* = \mathbf{B} \cdot \mathbf{x}^*$, where $\mathbf{x}^*$ has infinity bound 1. By Lemma 6.5.1, $\mathbf{x}' \neq \mathbf{x}^*$ with probability at least $1/2$. In the case they are not equal, we obtain a non-zero vector $\mathbf{y} = \mathbf{x}' - \mathbf{x}^*$ such that $\mathbf{B} \cdot \mathbf{y} = 0$ and $\mathbf{y}$ has infinity bound 2. However, under the hardness of the RSIS problem, the success probability of $\mathcal{A}$ is negligible. This concludes the proof. $\square$

Lemma 6.5.6. Assume the RLWE problem is hard, then the given accountable tracing scheme is anonymous with accountability in the random oracle model.

Proof. The proof of this lemma is similar to Lemma 6.5.3 except that we do not need to switch between two decryption keys. This is because the randomized keys in the certificate of the challenged users are obtained from the pairs $(\mathbf{a}_1^{(0)}, \mathbf{b}_1^{(0)})$ and $(\mathbf{a}_2^{(0)}, \mathbf{b}_2^{(0)})$, which are not related to the opening key. The details are omitted here. $\square$

Lemma 6.5.7. Assume the RLWE problem is hard, then the given accountable tracing scheme is trace-oblivious in the random oracle model.

Proof. We proceed through a sequence of hybrids. Let W_i be the event that adversary outputs 1 in Game i .

Game 0: Let this game be the experiment $\text{Exp}_{\text{ATS}, \mathcal{A}}^{\text{TO}-0}(\lambda)$, where the adversary receives **cert** for user p of his choice. Parse **cert** as $(p, \mathbf{a}'_1, \mathbf{b}'_1, \mathbf{a}'_2, \mathbf{b}'_2, t, \mathbf{r}, \mathbf{v})$. Note that $(\mathbf{a}'_1, \mathbf{b}'_1)$ and $(\mathbf{a}'_2, \mathbf{b}'_2)$ are randomized keys from $(\mathbf{a}_1^{(0)}, \mathbf{b}_1^{(0)})$ and $(\mathbf{a}_2^{(0)}, \mathbf{b}_2^{(0)})$, respectively. We then have $\Pr[W_0] = \Pr[\text{Exp}_{\text{ATS}, \mathcal{A}}^{\text{TO}-0}(\lambda) = 1]$.

Game 1: We modify Game 0 by replacing $(\mathbf{a}'_1, \mathbf{b}'_1)$ with a new fresh key $(\tilde{\mathbf{a}}_1, \tilde{\mathbf{b}}_1)$ generated by the **KeyGen** algorithm of our KOE scheme. It then follows from

the key randomizability of our encryption scheme, this modification is negligible to the adversary. Therefore, we have $|\Pr[W_0] - \Pr[W_1]| = \text{negl}(\lambda)$.

Game 2: We modify Game 1 by replacing $(\mathbf{a}'_2, \mathbf{b}'_2)$ with a new fresh key $(\tilde{\mathbf{a}}_2, \tilde{\mathbf{b}}_2)$ as in Game 1. By the same argument, we have $|\Pr[W_1] - \Pr[W_2]| = \text{negl}(\lambda)$.

Game 3: We change Game 2 by replacing $(\tilde{\mathbf{a}}_2, \tilde{\mathbf{b}}_2)$ with $(\mathbf{a}'_2, \mathbf{b}'_2)$ that are randomized key from $(\mathbf{a}_2^{(1)}, \mathbf{b}_2^{(1)})$. By the key randomizability of our encryption scheme, we have $|\Pr[W_2] - \Pr[W_3]| = \text{negl}(\lambda)$.

Game 4: We change Game 3 by replacing $(\tilde{\mathbf{a}}_1, \tilde{\mathbf{b}}_1)$ with $(\mathbf{a}'_1, \mathbf{b}'_1)$ that are randomized key from $(\mathbf{a}_1^{(1)}, \mathbf{b}_1^{(1)})$. We then have $|\Pr[W_3] - \Pr[W_4]| = \text{negl}(\lambda)$. This is exactly the experiment $\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{TO}-1}(\lambda)$. Therefore, $\Pr[W_4] = \Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{TO}-1}(\lambda) = 1]$.

As a result, we have $|\Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{TO}-1}(\lambda) = 1] - \Pr[\mathbf{Exp}_{\text{ATS}, \mathcal{A}}^{\text{TO}-0}(\lambda) = 1]| = \text{negl}(\lambda)$. This implies that our scheme is trace-oblivious. $\square$

6.5.3 The Main Zero-Knowledge Argument System

In this section, we present the statistical ZKAoK that will be invoked by the signer when generating group signatures. The protocol is an extension of the ones for Ducas-Micciancio signature from Section 4.3.2 and for RLWE relation with hidden vectors from Section 6.4.2. Let $n, q, k, \ell, m, m_s, \bar{m}, \bar{m}_s, d, c_0, \dots, c_d, \beta, B$ be parameters as specified in Section 6.5.1. The protocol is summarized as follows.

- The public input consists of

$$\begin{aligned} \mathbf{A}, \mathbf{F}_0 &\in R_q^{1 \times \bar{m}}; \quad \mathbf{A}_{[0]}, \dots, \mathbf{A}_{[d]} \in R_q^{1 \times k}; \quad \mathbf{F} \in R_q^{1 \times \ell}; \\ \mathbf{F}_1 &\in R_q^{1 \times \bar{m}_s}; \quad u \in R_q; \quad \mathbf{B} \in R_q^m, \quad \mathbf{c}_{1,1}, \mathbf{c}_{1,2} \in R_q^\ell, \quad \mathbf{c}_{2,1}, \mathbf{c}_{2,2} \in R_q^\ell. \end{aligned}$$

- The prover's secret input consists of message $\mathbf{m} = (p \|\mathbf{a}'_1 \|\mathbf{b}'_1 \|\mathbf{a}'_2 \|\mathbf{b}'_2)$ and the corresponding signature $(t, \mathbf{r}, \mathbf{v})$ for the Ducas-Micciancio signature scheme, a

user secret key $\mathbf{x}$ corresponding to the public key p , encryption randomness

$g'_1, g'_2, \mathbf{e}'_{1,1}, \mathbf{e}'_{1,2}, \mathbf{e}'_{2,1}, \mathbf{e}'_{2,2}$, where

$$\begin{cases} p \in R_q; & \mathbf{a}'_1 \in R_q^\ell; & \mathbf{b}'_1 \in R_q^\ell; & \mathbf{a}'_2 \in R_q^\ell; & \mathbf{b}'_2 \in R_q^\ell; \\ t = (t_0, \dots, t_{c_1-1}, \dots, t_{c_d-1}, \dots, t_{c_d-1})^\top \in \{0, 1\}^{c_d}; \\ \mathbf{r} \in R^{\bar{m}}; & \mathbf{v} = (\mathbf{s} \parallel \mathbf{z}) \in R^{\bar{m}+k}; & \mathbf{s} \in R^{\bar{m}}; & \mathbf{z} \in R^k; \\ \mathbf{x} \in R^m; & g'_1, g'_2 \in R; & \mathbf{e}'_{1,1}, \mathbf{e}'_{1,2}, \mathbf{e}'_{2,1}, \mathbf{e}'_{2,2} \in R^\ell. \end{cases}$$

- The prover's goal is to prove in zero-knowledge that $\|\mathbf{r}\|_\infty \leq \beta$, $\|\mathbf{v}\|_\infty \leq \beta$, $\|\mathbf{x}\|_\infty \leq 1$, $\|g'_i\|_\infty \leq B$, $\|\mathbf{e}_{i,1}\|_\infty \leq B$, $\|\mathbf{e}_{i,2}\|_\infty \leq B$ and that the following equations hold:

$$\mathbf{A}_t \cdot \mathbf{v} = \mathbf{F} \cdot \text{rdec}(\mathbf{F}_0 \cdot \mathbf{r} + \mathbf{F}_1 \cdot \text{rdec}(\mathbf{m})) + u,$$

$$\mathbf{B} \cdot \mathbf{x} = p,$$

$$\text{for } i \in \{1, 2\}, \quad \mathbf{c}_{i,1} = \mathbf{a}'_i \cdot g'_i + \mathbf{e}'_{i,1}, \quad \mathbf{c}_{i,2} = \mathbf{b}'_i \cdot g'_i + \mathbf{e}'_{i,2} + \lfloor q/4 \rfloor \cdot \text{rdec}(p). \quad (6.10)$$

Since the transformations for the Ducas-Micciancio signature scheme have been established¹ in Section 4.3.2, in the following, we only specify the transformations for other relations.

Let $\mathbf{a}'_i = (a'_{i,1}, \dots, a'_{i,\ell})^\top$, $\mathbf{b}'_i = (b'_{i,1}, \dots, b'_{i,\ell})^\top$ for each $i \in \{1, 2\}$. First, we apply the decomposition techniques in Section 2.4.3 to the following secrets.

- Let $\mathbf{x}^* = \tau(\mathbf{x}) \in \{-1, 0, 1\}^{nm}$.
- For each $i \in \{1, 2\}$, each $j \in [\ell]$, compute $\mathbf{a}_{i,j}^* = \tau(\text{rdec}(a'_{i,j})) \in \{-1, 0, 1\}^{n\ell}$,
 $\mathbf{b}_{i,j}^* = \tau(\text{rdec}(b'_{i,j})) \in \{-1, 0, 1\}^{n\ell}$.

¹Note that in Section 4.3.2, the signed message is a ring element in R_q while here we deal with a vector in $R_q^{m_s}$. However, this does not matter too much for the transformations.

- For $i \in \{1, 2\}$, compute $\mathbf{g}_i^* = \tau(\text{rdec}_B(g'_i)) \in \{-1, 0, 1\}^{n\delta_B}$.
- For $i \in \{1, 2\}$, compute $\mathbf{e}_{i,1}^* = \tau(\text{rdec}_B(\mathbf{e}'_{i,1}))$ and $\mathbf{e}_{i,2}^* = \tau(\text{rdec}_B(\mathbf{e}'_{i,2}))$. Note that they are vectors in $\{-1, 0, 1\}^{n\ell\delta_B}$.

Then the equation $\mathbf{B} \cdot \mathbf{x} = p$ can be translated as

$$[\text{rot}(\mathbf{B})] \cdot \mathbf{x}^* - [\mathbf{H}] \cdot \tau(\text{rdec}(p)) = \mathbf{0}^n \bmod q. \quad (6.11)$$

For each $i \in \{1, 2\}$, each $j \in [\ell]$, let

$$\mathbf{y}_{i,j} = \text{expd}(\mathbf{a}_{i,j}^*, \mathbf{g}_i^*), \quad \mathbf{z}_{i,j} = \text{expd}(\mathbf{b}_{i,j}^*, \mathbf{g}_i^*) \in \{-1, 0, 1\}^{n^2\ell\delta_B}. \quad (6.12)$$

From Section 6.4.2, we know that equations in (6.10) can be written as, for $i \in \{1, 2\}$,

$$\tau(\mathbf{c}_{i,1}) = [\mathbf{Q}] \cdot (\mathbf{y}_{i,1} \parallel \cdots \parallel \mathbf{y}_{i,\ell}) + [\mathbf{H}_{\ell,B}] \cdot \mathbf{e}_{i,1}^* \bmod q; \quad (6.13)$$

$$\tau(\mathbf{c}_{i,2}) = [\mathbf{Q}] \cdot (\mathbf{z}_{i,1} \parallel \cdots \parallel \mathbf{z}_{i,\ell}) + [\mathbf{H}_{\ell,B}] \cdot \mathbf{e}_{i,2}^* + \lfloor q/4 \rfloor \cdot \tau(\text{rdec}(p)) \bmod q. \quad (6.14)$$

Before proceeding further, let us recall that, in the protocol for the Ducas-Micciancio signature from Section 4.3.2, at the end of the Decomposing-Unifying step, we combined the secret objects into vectors $\mathbf{w}_1, \mathbf{w}_2$ of the form:

$$\begin{cases} \mathbf{w}_1 = (\mathbf{z}^* \parallel t_0 \cdot \mathbf{z}^* \parallel \cdots \parallel t_{c_d-1} \cdot \mathbf{z}^*) \in \{-1, 0, 1\}^{(k\delta_\beta + c_d k\delta_\beta)n}; \\ \mathbf{w}_2 = (\mathbf{s}^* \parallel \mathbf{r}^* \parallel \tau(\mathbf{y}) \parallel \tau(\text{rdec}(\mathbf{m}))) \in \{-1, 0, 1\}^{2n\overline{m}\delta_\beta + n\ell + n\overline{m}s}. \end{cases}$$

Note that $\tau(\text{rdec}(\mathbf{m})) =$

$$\begin{aligned} & (\tau(\text{rdec}(p)) \parallel \tau(\text{rdec}(\mathbf{a}'_1)) \parallel \tau(\text{rdec}(\mathbf{b}'_1)) \parallel \tau(\text{rdec}(\mathbf{a}'_2)) \parallel \tau(\text{rdec}(\mathbf{b}'_2))) \\ &= (\tau(\text{rdec}(p)) \parallel \mathbf{a}_{1,1}^* \parallel \cdots \parallel \mathbf{a}_{1,\ell}^* \parallel \mathbf{b}_{1,1}^* \parallel \cdots \parallel \mathbf{b}_{1,\ell}^* \parallel \mathbf{a}_{2,1}^* \parallel \cdots \parallel \mathbf{a}_{2,\ell}^* \parallel \mathbf{b}_{2,1}^* \parallel \cdots \parallel \mathbf{b}_{2,\ell}^*) \end{aligned}$$

Since $\tau(\text{rdec}(\mathbf{m}))$ has been counted as a block of vectors $\mathbf{w}_2$, we now combine the newly appeared secret vectors in equations (6.11), (6.13), (6.14) into vectors

$$\begin{aligned}\mathbf{w}_3 &= (\mathbf{x}^* \parallel \mathbf{e}_{1,1}^* \parallel \mathbf{e}_{1,2}^* \parallel \mathbf{e}_{2,1}^* \parallel \mathbf{e}_{2,2}^*) \in \{-1, 0, 1\}^{nm+4n\ell\delta_B}; \\ \mathbf{w}_4 &= (\mathbf{y}_{1,1} \parallel \cdots \parallel \mathbf{y}_{1,\ell} \parallel \mathbf{z}_{1,1} \parallel \cdots \parallel \mathbf{z}_{1,\ell} \parallel \mathbf{y}_{2,1} \parallel \cdots \parallel \mathbf{y}_{2,\ell} \parallel \mathbf{z}_{2,1} \parallel \cdots \parallel \mathbf{z}_{2,\ell})\end{aligned}\quad (6.15)$$

with $\mathbf{w}_4 \in \{-1, 0, 1\}^{4n^2\ell^2\delta_B}$ such that $i \in \{1, 2\}$, and $j \in [\ell]$, $\mathbf{y}_{i,j}, \mathbf{z}_{i,j}$ satisfy the equations in (6.12).

For the sake of simplicity when defining our tailored set **VALID** and permutation Γ_η , we rearrange our secret vectors $\mathbf{w}_2, \mathbf{w}_3$ as follows.

$$\begin{aligned}\bar{\mathbf{w}}_2 &= (\mathbf{s}^* \parallel \mathbf{r}^* \parallel \tau(\mathbf{y}) \parallel \tau(\text{rdec}(p)) \parallel \mathbf{x}^* \parallel \mathbf{e}_{1,1}^* \parallel \mathbf{e}_{1,2}^* \parallel \mathbf{e}_{2,1}^* \parallel \mathbf{e}_{2,2}^*); \\ \bar{\mathbf{w}}_3 &= (\mathbf{a}_{1,1}^* \parallel \cdots \parallel \mathbf{a}_{1,\ell}^* \parallel \mathbf{b}_{1,1}^* \parallel \cdots \parallel \mathbf{b}_{1,\ell}^* \parallel \mathbf{a}_{2,1}^* \parallel \cdots \parallel \mathbf{a}_{2,\ell}^* \parallel \mathbf{b}_{2,1}^* \parallel \cdots \parallel \mathbf{b}_{2,\ell}^*)\end{aligned}\quad (6.16)$$

with $\bar{\mathbf{w}}_2 \in \{-1, 0, 1\}^{L'_2}$, $\bar{\mathbf{w}}_3 \in \{-1, 0, 1\}^{4n\ell^2}$, and $L'_2 = 2n\bar{m}\delta_\beta + 2n\ell + nm + 4n\ell\delta_B$. Now we form our secret vector as $\mathbf{w}_0 = (\mathbf{w}_1 \parallel \bar{\mathbf{w}}_2 \parallel \bar{\mathbf{w}}_3 \parallel \mathbf{w}_4)$.

Next, we apply the extension and permutation techniques from Section 4.3.1 and Section 6.4.1 to our secret vectors $\mathbf{w}_0$. Let $\mathbf{w}'_1 = \text{mix}(t, \mathbf{z}^*) \in \{-1, 0, 1\}^{L_1}$ be the “mixing” vector obtained in equation (4.8), $\mathbf{w}'_2 = \text{enc}(\bar{\mathbf{w}}_2) \in \{-1, 0, 1\}^{L_2}$, $\mathbf{w}'_3 = \text{enc}(\bar{\mathbf{w}}_3) \in \{-1, 0, 1\}^{L_3}$, and $\mathbf{w}'_4 = \text{Mult}(\mathbf{w}_4) \in \{-1, 0, 1\}^{L_4}$ be of the following form:

$$\begin{aligned}&(\text{mult}(\mathbf{a}_{1,1}^*, \mathbf{g}_1^*) \parallel \cdots \parallel \text{mult}(\mathbf{a}_{1,\ell}^*, \mathbf{g}_1^*) \parallel \text{mult}(\mathbf{b}_{1,1}^*, \mathbf{g}_1^*) \parallel \cdots \parallel \text{mult}(\mathbf{b}_{1,\ell}^*, \mathbf{g}_1^*) \parallel \\ &\text{mult}(\mathbf{a}_{2,1}^*, \mathbf{g}_2^*) \parallel \cdots \parallel \text{mult}(\mathbf{a}_{2,\ell}^*, \mathbf{g}_2^*) \parallel \text{mult}(\mathbf{b}_{2,1}^*, \mathbf{g}_2^*) \parallel \cdots \parallel \text{mult}(\mathbf{b}_{2,\ell}^*, \mathbf{g}_2^*)),\end{aligned}$$

Where $L_1 = 3k\delta_\beta + 6nk\delta_\beta c_d$, $L_2 = 3L'_2$, $L_3 = 12n\ell^2$, and $L_4 = 36n^2\ell^2\delta_B$. Denote $L = L_1 + L_2 + L_3 + L_4$. Form our extended vector $\mathbf{w} = (\mathbf{w}'_1 \parallel \mathbf{w}'_2 \parallel \mathbf{w}'_3 \parallel \mathbf{w}'_4) \in \{-1, 0, 1\}^L$.

We remark that, by suitably concatenating/extending the matrices and vectors derived from the public input, we can obtain public matrix $\mathbf{M}$ and public vector $\mathbf{u}$ such that $\mathbf{M} \cdot \mathbf{w} = \mathbf{u} \bmod q$. Having obtained this desired equation, we now proceed as in Section 6.4.2.

Define **VALID** as the set of all vectors $\mathbf{v}' = (\mathbf{v}'_1 \| \mathbf{v}'_2 \| \mathbf{v}'_3 \| \mathbf{v}'_4) \in \{-1, 0, 1\}^L$, satisfying the following:

- There exist $t \in \{0, 1\}^{cd}$ and $\mathbf{z}^* \in \{-1, 0, 1\}^{nk\delta_\beta}$ such that $\mathbf{v}'_1 = \text{mix}(t, \mathbf{z}^*)$.
- There exists $\bar{\mathbf{w}}_2 \in \{-1, 0, 1\}^{L'_2}$ such that $\mathbf{v}'_2 = \text{enc}(\bar{\mathbf{w}}_2)$.
- For $j \in [4\ell]$, there exists $\bar{\mathbf{w}}_{3,j} \in \{-1, 0, 1\}^{n\ell}$ and $\bar{\mathbf{w}}_3 = (\bar{\mathbf{w}}_{3,1} \cdots \| \bar{\mathbf{w}}_{3,4\ell}) \in \{-1, 0, 1\}^{4n\ell^2}$ such that $\mathbf{v}'_3 = (\text{enc}(\bar{\mathbf{w}}_{3,1}) \| \cdots \| \text{enc}(\bar{\mathbf{w}}_{3,4\ell})) = \text{enc}(\bar{\mathbf{w}}_3)$.
- There exists $\mathbf{g}_1^*, \mathbf{g}_2^* \in \{-1, 0, 1\}^{n\delta_B}$ and $\mathbf{w}_4 \in \{-1, 0, 1\}^{4n^2\ell^2\delta_B}$ be of the form:

$$(\text{expd}(\bar{\mathbf{w}}_{3,1}, \mathbf{g}_1^*) \| \cdots \| \text{expd}(\bar{\mathbf{w}}_{3,2\ell}, \mathbf{g}_1^*) \| \text{expd}(\bar{\mathbf{w}}_{3,2\ell+1}, \mathbf{g}_2^*) \| \cdots \| \text{expd}(\bar{\mathbf{w}}_{3,4\ell}, \mathbf{g}_2^*))$$

such that $\mathbf{v}'_4 = \text{Mult}(\mathbf{w}_4)$.

It can be seen that vector $\mathbf{w}$ belongs to **VALID**.

Now, let $\mathcal{S} = \{0, 1\}^{cd} \times \{-1, 0, 1\}^{nk\delta_\beta} \times \{-1, 0, 1\}^{L'_2} \times (\{-1, 0, 1\}^{n\ell})^{4\ell} \times (\{-1, 0, 1\}^{n\delta_B})^2$, and associate every element

$$\eta = (\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_{4,1}, \dots, \mathbf{f}_{4,2\ell}, \mathbf{f}_{5,1}, \dots, \mathbf{f}_{5,2\ell}, \mathbf{f}_6, \mathbf{f}_7) \in \mathcal{S}$$

with Γ_η that acts as follows. When applied to vector $\mathbf{v}^* = (\mathbf{v}_1^* \| \mathbf{v}_2^* \| \mathbf{v}_3^* \| \mathbf{v}_4^*) \in \mathbb{Z}^L$, where $\mathbf{v}_i^* \in \mathbb{Z}^{L_i}$ for $i \in \{1, 2\}$, $\mathbf{v}_3^* = (\mathbf{v}_{3,1}^* \| \cdots \| \mathbf{v}_{3,4\ell}^*)$ with $\mathbf{v}_{3,j}^* \in \mathbb{Z}^{3n\ell}$, and $\mathbf{v}_4^* =$

$(\mathbf{v}_{4,1}^* \cdots \|\mathbf{v}_{4,4\ell}^*)$ with $\mathbf{v}_{4,j}^* \in \mathbb{Z}^{9n^2\ell\delta_B}$, it transforms $\mathbf{v}^*$ into vector $\Gamma_\eta(\mathbf{v}^*)$

$$\begin{aligned} & (\Psi_{\mathbf{f}_1, \mathbf{f}_2}(\mathbf{v}_1^*) \|\Pi_{\mathbf{f}_3}(\mathbf{v}_2^*) \|\Pi_{\mathbf{f}_{4,1}}(\mathbf{v}_{3,1}^*) \|\cdots \|\Pi_{\mathbf{f}_{4,2\ell}}(\mathbf{v}_{3,2\ell}^*) \|\Pi_{\mathbf{f}_{5,1}}(\mathbf{v}_{3,2\ell+1}^*) \|\cdots \|\Pi_{\mathbf{f}_{5,2\ell}}(\mathbf{v}_{3,4\ell}^*) \|\Phi_{\mathbf{f}_{4,1}, \mathbf{f}_6}(\mathbf{v}_{4,1}^*) \|\cdots \|\Phi_{\mathbf{f}_{4,2\ell}, \mathbf{f}_6}(\mathbf{v}_{4,2\ell}^*) \|\Phi_{\mathbf{f}_{5,1}, \mathbf{f}_7}(\mathbf{v}_{4,2\ell+1}^*) \|\cdots \|\Phi_{\mathbf{f}_{5,2\ell}, \mathbf{f}_7}(\mathbf{v}_{4,4\ell}^*)) \end{aligned}$$

Based on the equivalences observed in (4.4), (4.5), and (6.2), it can be checked that **VALID**, $\mathcal{S}$ and Γ_η satisfy the conditions specified in (2.1). In other words, we have reduced the considered statement to an instance of the abstract protocol from Section 2.4.2. Therefore, both parties run the protocol of Figure 2.1 and we obtain a statistical **ZKAoK** for the considered statement, with perfect completeness, soundness error $2/3$ and communication cost $\mathcal{O}(L \cdot \log q)$, which is of order $\mathcal{O}(n^2 \cdot \log^3 n) = \tilde{\mathcal{O}}(\lambda^2)$ for the setting of parameters we use in our scheme from Section 6.5.1.

Chapter 7

Conclusions and Open Problems

In this thesis, we have shown how to construct lattice-based group signature schemes that achieve full dynamicity, deniability, constant-size signatures, forward-security, and accountability of the group manager. As we already see, construction of group signatures usually requires complex combination of signature schemes, encryption schemes, and zero-knowledge protocols. While signature schemes and encryption schemes are available from lattices, the major challenge in constructing lattice-based group signatures is the underlying zero-knowledge protocols. Fortunately, Ling et al. [116] designed a flexible zero-knowledge argument of knowledge protocol that is Stern-type. Towards constructing group signatures with advanced features, we are confronted with much more complex statements that should be proven in a zero-knowledge manner. Nevertheless, we managed to reduce the required statements to instances of the abstracted Stern-like protocol, reviewed in Section 2.4.2, and thus by running the protocol described in Figure 2.1, we can obtain the desired zero-knowledge argument of knowledge protocols.

However, a drawback of the above zero-knowledge protocols is that they have soundness error $2/3$. To achieve negligible soundness error, one should repeat the protocol $\kappa = \omega(\log \lambda)$ times based on the theoretical definition of the security parameter λ . This in turn results in the high overhead of our group signature schemes. If

we insist on the practical definition of λ , the repetition should be implemented $\Omega(\lambda)$ times, which further leads to inefficiency of our constructions in this thesis. It thus remains an open question to design practical group signature schemes that at the same time admit the above advanced features. Designing efficient and compatible zero-knowledge protocols might be a direction.

There are some other open questions regarding some of the constructions presented in this thesis. In Chapter 5, we have constructed a lattice-based group signature scheme with forward-security. However, a drawback of this scheme is that it is designed only for static groups. As we have advocated in this thesis, dynamic user enrollment and revocation of some misbehaving/leaving users are two desirable features. Therefore, it is left as an open question to construct forward-secure group signatures from lattices that enable admission of new users or revocation of misbehaving/leaving users, or even both features.

In Chapter 6, we have presented our accountable tracing signature scheme from lattices. Recall that when a user joins the group, the group manager first determines whether this user is traceable or non-traceable and then generates the corresponding certificate. This in fact monitors the future actions of the suspects. However, in real-life applications, there exists situations where we cannot determine that a user is the suspect until we have already detected some suspicious behaviours of this user. Under this circumstance, a mechanism that enables backward-tracing would be quite helpful. Indeed, Kohlweiss and Miers [99] already considered this situation and extended their accountable tracing signature scheme to achieve backward-tracing. Thus, an open question is whether we can also achieve backward-tracing from lattices.

Note that among all of the constructions we presented in this thesis, two of them (Chapter 4 and Chapter 6) are based on ideal lattices. However, we believe that the full potential of ideal lattices has not been completely utilized. Therefore, exploring the further potential of ideal lattices is left for future investigations.

Bibliography

- [1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, 2010.
- [2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, 2010.
- [3] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, 2011.
- [4] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC 1996*, pages 99–108. ACM, 1996.
- [5] Miklós Ajtai. The shortest vector problem in L_2 is *NP*-hard for randomized reductions (extended abstract). In *STOC 1998*, pages 10–19. ACM, 1998.
- [6] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC 1997*, pages 284–293. ACM, 1997.
- [7] Miklós Ajtai and Cynthia Dwork. The first and fourth public-key cryptosystems with worst-case/average-case equivalence. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(097), 2007.

- [8] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC 2001*, pages 601–610. ACM, 2001.
- [9] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *Computational Complexity 2002*, pages 53–57. IEEE, 2002.
- [10] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, 2009.
- [11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS 2009*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
- [12] Ross Anderson. Two Remarks on Public Key Cryptology. Technical report, University of Cambridge, Computer Laboratory, 2002.
- [13] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, 2009.
- [14] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:385, 2005.
- [15] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.
- [16] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In *Financial Cryptography 1999*, volume 1648 of *LNCS*, pages 196–211. Springer, 1999.

- [17] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 480–494. Springer, 1997.
- [18] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *PKC 2014*, volume 8383 of *LNCS*, pages 520–537. Springer, 2014.
- [19] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO 1992*, volume 740 of *LNCS*, pages 390–420. Springer, 1992.
- [20] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
- [21] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.
- [22] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In *EUROCRYPT 1993*, volume 765 of *LNCS*, pages 274–285. Springer, 1993.
- [23] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 551–572. Springer, 2014.
- [24] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS 2015*, volume 9326 of *LNCS*, pages 305–325. Springer, 2015.

- [25] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *SCN 2010*, volume 6280 of *LNCS*, pages 381–398. Springer, 2010.
- [26] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, and Christoph Striecks. Confined guessing: New signatures from standard assumptions. *J. Cryptology*, 28(1):176–208, 2015.
- [27] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.
- [28] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
- [29] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, 2014.
- [30] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *ACM CCS 2004*, pages 168–177. ACM, 2004.
- [31] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *ACNS 2016*, volume 9696 of *LNCS*, pages 117–136. Springer, 2016.
- [32] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *ESORICS 2015*, volume 9326 of *LNCS*, pages 243–265. Springer, 2015.

- [33] Cecilia Boschini, Jan Camenisch, and Gregory Neven. Floppy-sized group signatures from lattices. In *ACNS 2018*, volume 10892 of *LNCS*, pages 163–182. Springer, 2018.
- [34] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, 2010.
- [35] Xavier Boyen, Hovav Shacham, Emily Shen, and Brent Waters. Forward-secure signatures with untrusted update. In *ACM-CCS 2006*, pages 191–200. ACM, 2006.
- [36] Xavier Boyen and Brent Waters. Compact Group Signatures without Random Oracles. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, 2006.
- [37] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.
- [38] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, 2012.
- [39] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in lwe-based homomorphic encryption. In *PKC 2013*, volume 7778 of *LNCS*, pages 1–13. Springer, 2013.
- [40] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012*, pages 309–325. ACM, 2012.

- [41] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC 2013*, pages 575–584. ACM, 2013.
- [42] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS 2011*, pages 97–106. IEEE, 2011.
- [43] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, 2011.
- [44] Emmanuel Bresson and Jacques Stern. Efficient revocation in group signatures. In *PKC 2001*, volume 1992 of *LNCS*, pages 190–206. Springer, 2001.
- [45] Ernest Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In *PKC 2000*, pages 276–292. Springer, 2000.
- [46] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *CCS 2004*, pages 132–145. ACM, 2004.
- [47] Ernest F. Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In *PKC 2000*, volume 1751 of *LNCS*, pages 276–292. Springer, 2000.
- [48] Jin-yi Cai and Ajay Nerurkar. An improved worst-case to average-case connection for lattice problems. In *FOCS 1997*, pages 468–477. IEEE, 1997.
- [49] Jin-yi Cai and Ajay Nerurkar. Approximating the SVP to within a factor $(1+1/\dim^{\text{xi}})$ is np-hard under randomized reductions. *J. Comput. Syst. Sci.*, 59(2):221–239, 1999.
- [50] Jan Camenisch. Efficient anonymous fingerprinting with group signatures. In *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 415–428. Springer, 2000.

- [51] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *SCN 2004*, volume 3352 of *LNCS*, pages 120–133. Springer, 2004.
- [52] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *CCS 2002*, pages 21–30. ACM, 2002.
- [53] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.
- [54] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.
- [55] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.
- [56] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.
- [57] Jan Camenisch, Gregory Neven, and Markus Rückert. Fully anonymous attribute tokens from lattices. In *SCN 2012*, volume 7485 of *LNCS*, pages 57–75. Springer, 2012.
- [58] Sébastien Canard and Jacques Traoré. On fair e-cash systems based on group signature schemes. In *ACISP 2003*, volume 2727 of *LNCS*, pages 237–248. Springer, 2003.
- [59] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, 2003.

- [60] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.
- [61] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, 2010.
- [62] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
- [63] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Khoa Nguyen. Revocable identity-based encryption from lattices. In *ACISP 2012*, volume 7372 of *LNCS*, pages 390–403. Springer, 2012.
- [64] Lidong Chen and Torben P. Pedersen. New group signature schemes (extended abstract). In *EUROCRYPT 1994*, volume 950 of *LNCS*, pages 171–181. Springer, 1994.
- [65] Shantian Cheng, Khoa Nguyen, and Huaxiong Wang. Policy-based signature scheme from lattices. *Des. Codes Cryptography*, 81(1):43–74, 2016.
- [66] Ivan Damgård. On Sigma-protocols. Lecture on Cryptologic Protocol Theory; Faculty of Science, University of Aarhus, 2010.
- [67] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM-CCS 2018*, pages 574–591. ACM, 2018.
- [68] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 193–210. Springer, 2006.

- [69] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003.
- [70] Irit Dinur, Guy Kindler, and Shmuel Safra. Approximating-cvp to within almost-polynomial factors is np-hard. In *FOCS 1998*, pages 99–111. IEEE, 1998.
- [71] Léo Ducas. Advances on quantum cryptanalysis of ideal lattices. *Nieuw Archief voor Wiskunde*, 5:184–189, 2017.
- [72] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014*, volume 8616 of *LNCS*, pages 335–352. Springer, 2014.
- [73] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. *IACR Cryptology ePrint Archive*, 2014:495, 2014.
- [74] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A provably secure group signature scheme from code-based assumptions. In *ASIACRYPT 2015*, volume 9452 of *LNCS*, pages 260–285. Springer, 2015.
- [75] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
- [76] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO 1997*, volume 1294 of *LNCS*, pages 16–30. Springer, 1997.

- [77] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18. Springer, 1984.
- [78] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009*, pages 169–178. ACM, 2009.
- [79] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS 2011*, pages 107–109. IEEE, 2011.
- [80] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from LWE. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 506–522. Springer, 2010.
- [81] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206. ACM, 2008.
- [82] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013*, volume 8042 of *LNCS*, pages 75–92. Springer, 2013.
- [83] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.
- [84] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(42), 1996.
- [85] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS 2010*, pages 230–240. Tsinghua University Press, 2010.

- [86] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
- [87] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC 2013*, pages 545–554. ACM, 2013.
- [88] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO 2015*, volume 9216 of *LNCS*, pages 503–523. Springer, 2015.
- [89] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 395–412. Springer, 2010.
- [90] Jens Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, 2007.
- [91] Ai Ishida, Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Keisuke Tanaka. Group signature with deniability: How to disavow a signature. In *CANS 2016*, volume 10052 of *LNCS*, pages 228–244, 2016.
- [92] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 663–680. Springer, 2012.
- [93] Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.
- [94] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 372–389. Springer, 2008.
- [95] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *FOCS 2004*, pages 126–135. IEEE, 2004.

- [96] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, 2004.
- [97] Aggelos Kiayias and Moti Yung. Extracting group signatures from traitor tracing schemes. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 630–648. Springer, 2003.
- [98] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *Int. Journal of Security and Networks*, 1(1):24–45, 2006.
- [99] Markulf Kohlweiss and Ian Miers. Accountable metadata-hiding escrow: A group signature case study. *PoPETs*, 2015(2):206–221, 2015.
- [100] Ravi Kumar and Dandapani Sivakumar. On polynomial-factor approximations to the shortest lattice vector length. *SIAM J. Discrete Math.*, 16(3):422–425, 2003.
- [101] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 41–61. Springer, 2013.
- [102] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *PKC 2014*, volume 8383 of *LNCS*, pages 345–361. Springer, 2014. Corrected full version: <http://eprint.iacr.org/2014/033>.
- [103] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.
- [104] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

- [105] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 373–403. Springer, 2016.
- [106] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 101–131. Springer, 2016.
- [107] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Adaptive oblivious transfer with access control from lattice assumptions. In *ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 533–563. Springer, 2017.
- [108] 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 *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 1–31. Springer, 2016.
- [109] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based prfs and applications to e-cash. In *ASIACRYPT 2017*, volume 10626 of *LNCS*, pages 304–335. Springer, 2017.
- [110] Benoît Libert, Fabrice Mouhartem, and Khoa Nguyen. A lattice-based group signature scheme with message-dependent opening. In *ACNS 2016*, volume 9696 of *LNCS*, pages 137–155. Springer, 2016.
- [111] Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In *CRYPTO 2012*, volume 7417 of *LNCS*, pages 571–589. Springer, 2012.

- [112] Benoît Libert, Thomas Peters, and Moti Yung. Scalable group signatures with revocation. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 609–627. Springer, 2012.
- [113] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In *CRYPTO 2015*, volume 9216 of *LNCS*, pages 296–316. Springer, 2015.
- [114] Benoît Libert and Moti Yung. Dynamic fully forward-secure group signatures. In *Asia-CCS 2010*, pages 70–81. ACM, 2010.
- [115] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, 2011.
- [116] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, 2013.
- [117] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *PKC 2015*, volume 9020 of *LNCS*, pages 427–449. Springer, 2015.
- [118] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Accountable tracing signatures from lattices. To appear in CT-RSA 2019.
- [119] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS 2017*, volume 10355 of *LNCS*, pages 293–312. Springer, 2017.
- [120] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Constant-size group signatures from lattices. In *PKC 2018*, volume 10770 of *LNCS*, pages 58–88. Springer, 2018.

- [121] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Forward-secure group signatures from lattices. *CoRR*, abs/1801.08323, 2018. Submitted to PQCrypto 2019.
- [122] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity (and deniability) with ease. *CoRR*, abs/1801.08737, 2018. Submitted to Theoretical Computer Science.
- [123] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC 2008*, volume 4939 of *LNCS*, pages 162–179. Springer, 2008.
- [124] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009.
- [125] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.
- [126] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP 2006*, volume 4052 of *LNCS*, pages 144–155. Springer, 2006.
- [127] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC 2008*, volume 4948 of *LNCS*, pages 37–54. Springer, 2008.
- [128] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE 2008*, volume 5086 of *LNCS*, pages 54–72. Springer, 2008.
- [129] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT 2017*, volume 10210 of *LNCS*, pages 293–323. Springer, 2017.

- [130] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010.
- [131] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.
- [132] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT 2018*, volume 10820 of *LNCS*, pages 204–224. Springer, 2018.
- [133] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. In *FOCS 1998*, pages 92–98. IEEE, 1998.
- [134] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *FOCS 2002*, pages 356–365. IEEE, 2002.
- [135] Daniele Micciancio. Improved cryptographic hash functions with worst-case/average-case connection. In *STOC 2002*, pages 609–618. ACM, 2002.
- [136] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.
- [137] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 465–484. Springer, 2011.
- [138] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012.

- [139] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO 2013*, volume 8042 of *LNCS*, pages 21–39. Springer, 2013.
- [140] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. In *FOCS 2004*, pages 372–381. IEEE, 2004.
- [141] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.
- [142] Daniele Micciancio and Salil P. Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 282–298. Springer, 2003.
- [143] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *PKC 2009*, volume 5443 of *LNCS*, pages 463–480. Springer, 2009.
- [144] Toru Nakanishi and Nobuo Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 533–548. Springer, 2005.
- [145] Toru Nakanishi, Yuta Hira, and Nobuo Funabiki. Forward-secure group signatures from pairings. In *Pairing 2009*, pages 171–186. Springer, 2009.
- [146] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *ACM 1990*, pages 427–437. ACM, 1990.
- [147] Khoa Nguyen, Huaxiong Wang, and Juanyang Zhang. Server-aided revocable identity-based encryption from lattices. In *CANS 2016*, volume 10052 of *LNCS*, pages 107–123, 2016.
- [148] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.

- [149] Lan Nguyen and Reihaneh Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 372–386. Springer, 2004.
- [150] Phong Q. Nguyen and Damien Stehlé. Floating-point LLL revisited. In *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2005.
- [151] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In *CaLC 2001*, volume 2146 of *LNCS*, pages 146–180. Springer, 2001.
- [152] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *PKC 2015*, volume 9020 of *LNCS*, pages 401–426. Springer, 2015.
- [153] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC 2009*, pages 333–342. ACM, 2009.
- [154] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-lwe for any ring and modulus. In *STOC 2017*, pages 461–473. ACM, 2017.
- [155] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC 2006*, volume 3876 of *LNCS*, pages 145–166. Springer, 2006.
- [156] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, 2008.
- [157] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC 2008*, pages 187–196. ACM, 2008.

- [158] Van Emde Boas Peter. Another np-complete partition problem and the complexity of computing short vectors in a lattice. Technical report, Technical Report 81-04, 1981.
- [159] Oded Regev. New lattice based cryptographic constructions. In *STOC 2003*, pages 407–416. ACM, 2003.
- [160] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pages 84–93. ACM, 2005.
- [161] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34, 2009.
- [162] Markus Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. In *PQCrypto 2010*, volume 6061 of *LNCS*, pages 182–200. Springer, 2010.
- [163] Markus Rückert. *Lattice-based signature schemes with additional features*. PhD thesis, Darmstadt University of Technology, 2011.
- [164] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, Takahiro Matsuda, and Kazumasa Omote. Group signatures with message-dependent opening. In *Pairing 2012*, volume 7708 of *LNCS*, pages 270–294. Springer, 2012.
- [165] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In *PKC 2012*, volume 7293 of *LNCS*, pages 715–732. Springer, 2012.
- [166] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987.
- [167] Claus-Peter Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of algorithms*, 9(1):47–62, 1988.

- [168] Claus Peter Schnorr. Fast llr-type lattice reduction. *Information and Computation*, 204(1):1–25, 2006.
- [169] Claus-Peter Schnorr and Horst Helmut Hörner. Attacking the chor-rivest cryptosystem by improved lattice reduction. In *EUROCRYPT 1995*, volume 921 of *LNCS*, pages 1–12. Springer, 1995.
- [170] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS 1994*, pages 124–134. IEEE Computer Society, 1994.
- [171] Dawn Xiaodong Song. Practical forward secure group signature schemes. In *ACM-CCS 2001*, pages 225–234. ACM, 2001.
- [172] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, 2009.
- [173] Jacques Stern. A new paradigm for public key identification. *IEEE Trans. Information Theory*, 42(6):1757–1768, 1996.
- [174] Gene Tsudik and Shouhuai Xu. Accumulating composites and improved group signing. In *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 269–286. Springer, 2003.
- [175] Keita Xagawa. Improved (hierarchical) inner-product encryption from lattices. In *PKC 2013*, volume 7778 of *LNCS*, pages 235–252. Springer, 2013.

Appendices

Appendix A

Proof of Theorem 2.4.4

We provide the proof of Theorem 2.4.4, as appeared in [105]. We first restate the theorem.

Theorem A.0.8. Assume that COM is a statistically hiding and computationally binding string commitment scheme. Then, the protocol in Figure 2.1 is a statistical *ZKAoK* with perfect completeness, soundness error $2/3$, knowledge error $2/3$, and communication cost $\mathcal{O}(L \log q)$. In particular:

- There exists a PPT simulator that on input $(\mathbf{M}, \mathbf{u})$, with probability $2/3$ it outputs an accepted transcript that is within statistical distance from the one produced by an honest prover who knows the witness.
- There exists a polynomial-time algorithm $\mathcal{E}$ that, takes as inputs $(\mathbf{M}, \mathbf{u})$ and three accepting transcripts on $(\mathbf{M}, \mathbf{u})$, $(\text{CMT}, 1, \text{RSP}_1)$, $(\text{CMT}, 2, \text{RSP}_2)$, and $(\text{CMT}, 3, \text{RSP}_3)$, outputs $\mathbf{w}' \in \text{VALID}$ such that $\mathbf{M} \cdot \mathbf{w}' = \mathbf{u} \bmod q$.

Proof. It can be checked that the protocol has perfect completeness: If an honest prover follows the protocol, then he always gets accepted by the verifier. It is also easy to see that the communication cost is bounded by $\mathcal{O}(L \log q)$.

We now prove that the protocol is a statistical zero-knowledge argument of knowledge.

Zero-Knowledge Property. We construct a PPT simulator **SIM** interacting with a (possibly dishonest) verifier $\hat{\mathcal{V}}$, such that, given only the public input, **SIM** outputs with probability negligibly close to $2/3$ a simulated transcript that is statistically close to the one produced by the honest prover in the real interaction.

The simulator first chooses a random $\overline{Ch} \in \{1, 2, 3\}$ as a prediction of the challenge value that $\hat{\mathcal{V}}$ will *not* choose.

Case $\overline{Ch} = 1$: Using basic linear algebra over $\mathbb{Z}_q$, **SIM** computes a vector $\mathbf{w}' \in \mathbb{Z}_q^L$ such that $\mathbf{M} \cdot \mathbf{w}' = \mathbf{u} \bmod q$. Next, it samples $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_q^L$, $\eta \xleftarrow{\$} \mathcal{S}$, and randomness ρ_1, ρ_2, ρ_3 for **COM**. Then, it sends the commitment $\text{CMT} = (C'_1, C'_2, C'_3)$ to $\hat{\mathcal{V}}$, where

$$\begin{aligned} C'_1 &= \text{COM}(\eta, \mathbf{M} \cdot \mathbf{r}_w; \rho_1), \\ C'_2 &= \text{COM}(\Gamma_\eta(\mathbf{r}_w); \rho_2), \quad C'_3 = \text{COM}(\Gamma_\eta(\mathbf{w}' + \mathbf{r}_w); \rho_3). \end{aligned}$$

Receiving a challenge Ch from $\hat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Output $\perp$ and abort.
- If $Ch = 2$: Send $\text{RSP} = (\eta, \mathbf{w}' + \mathbf{r}_w, \rho_1, \rho_3)$.
- If $Ch = 3$: Send $\text{RSP} = (\eta, \mathbf{r}_w, \rho_1, \rho_2)$.

Case $\overline{Ch} = 2$: **SIM** samples $\mathbf{w}' \xleftarrow{\$} \text{VALID}$, $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_q^L$, $\eta \xleftarrow{\$} \mathcal{S}$, and randomness ρ_1, ρ_2, ρ_3 for **COM**. Then it sends the commitment $\text{CMT} = (C'_1, C'_2, C'_3)$ to $\hat{\mathcal{V}}$, where

$$\begin{aligned} C'_1 &= \text{COM}(\eta, \mathbf{M} \cdot \mathbf{r}_w; \rho_1), \\ C'_2 &= \text{COM}(\Gamma_\eta(\mathbf{r}_w); \rho_2), \quad C'_3 = \text{COM}(\Gamma_\eta(\mathbf{w}' + \mathbf{r}_w); \rho_3). \end{aligned}$$

Receiving a challenge Ch from $\hat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Send $\text{RSP} = (\Gamma_\eta(\mathbf{w}'), \Gamma_\eta(\mathbf{r}_w), \rho_2, \rho_3)$.
- If $Ch = 2$: Output $\perp$ and abort.

- If $Ch = 3$: Send $\text{RSP} = (\eta, \mathbf{r}_w, \rho_1, \rho_2)$.

Case $\overline{Ch} = 3$: SIM samples $\mathbf{w}' \xleftarrow{\$} \text{VALID}$, $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_q^L$, $\eta \xleftarrow{\$} \mathcal{S}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then it sends the commitment $\text{CMT} = (C'_1, C'_2, C'_3)$ to $\hat{\mathcal{V}}$, where $C'_2 = \text{COM}(\Gamma_\eta(\mathbf{r}_w); \rho_2)$, $C'_3 = \text{COM}(\Gamma_\eta(\mathbf{w}' + \mathbf{r}_w); \rho_3)$ as in the previous two cases, while

$$C'_1 = \text{COM}(\eta, \mathbf{M} \cdot (\mathbf{w}' + \mathbf{r}_w) - \mathbf{u}; \rho_1).$$

Receiving a challenge Ch from $\hat{\mathcal{V}}$, it responds as follows:

- If $Ch = 1$: Send RSP computed as in the case $(\overline{Ch} = 2, Ch = 1)$.
- If $Ch = 2$: Send RSP computed as in the case $(\overline{Ch} = 1, Ch = 2)$.
- If $Ch = 3$: Output $\perp$ and abort.

We observe that, in every case we have considered above, since COM is statistically hiding, the distribution of the commitment CMT and the distribution of the challenge Ch from $\hat{\mathcal{V}}$ are statistically close to those in the real interaction. Hence, the probability that the simulator outputs $\perp$ is negligibly close to $1/3$. Moreover, one can check that whenever the simulator does not halt, it will provide an accepted transcript, the distribution of which is statistically close to that of the prover in the real interaction. In other words, we have constructed a simulator that can successfully impersonate the honest prover with probability negligibly close to $2/3$.

Argument of Knowledge. Suppose $\text{RSP}_1 = (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$, $\text{RSP}_2 = (\eta_2, \mathbf{w}_2, \rho_1, \rho_3)$, $\text{RSP}_3 = (\eta_3, \mathbf{w}_3, \rho_1, \rho_2)$ are 3 valid responses to the same commitment $\text{CMT} = (C_1, C_2, C_3)$, with respect to all 3 possible values of the challenge. The validity of

these responses implies that:

$$\begin{cases} \mathbf{t}_w \in \text{VALID}; \\ C_1 = \text{COM}(\eta_2, \mathbf{M} \cdot \mathbf{w}_2 - \mathbf{u} \bmod q; \rho_1) = \text{COM}(\eta_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1); \\ C_2 = \text{COM}(\mathbf{t}_r; \rho_2) = \text{COM}(\Gamma_{\eta_3}(\mathbf{w}_3); \rho_2); \\ C_3 = \text{COM}(\mathbf{t}_w + \mathbf{t}_r \bmod q; \rho_3) = \text{COM}(\Gamma_{\eta_2}(\mathbf{w}_2); \rho_3). \end{cases}$$

Since COM is computationally binding, we can deduce that

$$\begin{cases} \mathbf{t}_w \in \text{VALID}; \eta_2 = \eta_3; \mathbf{t}_r = \Gamma_{\eta_3}(\mathbf{w}_3); \mathbf{t}_w + \mathbf{t}_r = \Gamma_{\eta_2}(\mathbf{w}_2) \bmod q; \\ \mathbf{M} \cdot \mathbf{w}_2 - \mathbf{u} = \mathbf{M} \cdot \mathbf{w}_3 \bmod q. \end{cases} \quad (\text{A.1})$$

Since $\mathbf{t}_w \in \text{VALID}$, if we let $\mathbf{w}' = [\Gamma_{\eta_2}]^{-1}(\mathbf{t}_w)$, then $\mathbf{w}' \in \text{VALID}$. Furthermore, we have

$$\Gamma_{\eta_2}(\mathbf{w}') + \Gamma_{\eta_2}(\mathbf{w}_3) = \Gamma_{\eta_2}(\mathbf{w}_2) \bmod q,$$

which implies that $\mathbf{w}' + \mathbf{w}_3 = \mathbf{w}_2 \bmod q$, and that $\mathbf{M} \cdot \mathbf{w}' + \mathbf{M} \cdot \mathbf{w}_3 = \mathbf{M} \cdot \mathbf{w}_2 \bmod q$.

As a result, we have $\mathbf{M} \cdot \mathbf{w}' = \mathbf{u} \bmod q$. This concludes the proof. $\square$