Code-based cryptography : new privacy-preserving constructions

Zeng, Neng

2020

Zeng, N. (2020). Code-based cryptography : new privacy-preserving constructions. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/142777>

<https://doi.org/10.32657/10356/142777>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 17:40:31 SGT

**CODE-BASED CRYPTOGRAPHY: NEW
PRIVACY-PRESERVING CONSTRUCTIONS**

NENG ZENG

**SCHOOL OF PHYSICAL AND MATHEMATICAL
SCIENCES**

2020

CODE-BASED CRYPTOGRAPHY: NEW PRIVACY-PRESERVING CONSTRUCTIONS

NENG ZENG

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

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research done by me except where otherwise stated in this thesis. The thesis work has not been submitted for a degree or professional qualification to any other university or institution. I declare that this thesis is written by myself and is free of plagiarism and of sufficient grammatical clarity to be examined. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

11/June/2020

11/June/2020

Zeng Neng

NENG ZENG

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it of sufficient grammatical clarity to be examined. To the best of my knowledge, the thesis is free of plagiarism and the research and writing are those of the candidate's except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

11/June/2020

11/June/2020

A handwritten signature in black ink, consisting of a stylized 'H' followed by a series of loops and a horizontal stroke.

Huaxiong Wang

Authorship Attribution Statement

This thesis contains material from 1 paper under submission to the following peer-reviewed journal and 1 paper accepted at conference in which I am listed as an author.

Chapter 3, 4, 5 and 6 are published as Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng. New code-based privacy-preserving cryptographic constructions. In *ASIACRYPT 2019*, volume 11922 of *Lecture Notes in Computer Science*, pages 25–55. Springer, 2019.

The contributions of the co-authors are as follows:

All authors joined in discussions and edited the manuscript in different levels.

- Dr Khoa Nguyen proposed the initial idea, drafted and revised the paper.
- Prof Huaxiong Wang guided the project, involved in discussions and held meetings.
- Hanh Tang involved in discussions, joined in editing and revising the manuscript.
- Neng Zeng took part in discussions, wrote the initial manuscript and revised the manuscript.

Chapter 7 is in submission to “Designs, Codes and Cryptography”. Khoa Nguyen, Thuc D. Nguyen, Huaxiong Wang, and Neng Zeng.

The contributions of the co-authors are as follows:

All authors took part in discussions during the project.

- Dr Khoa Nguyen proposed the initial idea and revised the manuscript.
- Prof Huaxiong Wang and Prof Thuc D. Nguyen supported the project and took part in discussions.
- Neng Zeng drafted and wrote the manuscripts, as well as revised the manuscript.

11/June/2020

11/June/2020

Zeng Neng

NENG ZENG

Abstract

Code-based cryptography is one of the promising candidates of post-quantum cryptography. It has a long history, dated back to the seminal work by McEliece in 1978, but did suffer from a relatively slow development. Recently, it has witnessed resurgence in building theoretical or practical constructions such as identity-based encryptions, public key encryptions and so on. However, its subfield of privacy-preserving constructions is not fully developed. On one hand, many primitives still need to improve, e.g., accumulators, ring signatures, e-cash systems and so on. On the other hand, many important building blocks are still missing, such as proofs of knowledge of a hash preimage/image and set membership proofs.

Observing the underdevelopment of code-based cryptography, we are aiming to introduce several privacy-preserving constructions to help enrich the diversity and advance the state-of-affairs of this field. Specifically, we put forth five main contributions:

- We design a code-based commitment scheme which is statistically hiding and computationally binding with companion proofs of a valid opening using zero knowledge argument. A good feature is that it can be easily composed with other cryptographic primitives and relations.
- We construct the first logarithmic code-based Merkle-tree accumulator and zero knowledge proofs for set membership which have numerous privacy-preserving applications in ring signature, group signatures, e-cash and so on. An accumu-

lator is able to compute a short accumulated value by hashing a set of elements and generate a short witness for each element to show that it belonging to the set. An honest prover who possesses a valid element-witness pair can convince a verifier in a zero knowledge manner that a specific input element is included in the hashed set. Our accumulator can be utilized in membership proofs, the prover follows and develops Stern’s zero knowledge arguments to demonstrate that he knows a hidden hash chain from the element to the accumulated root value without revealing any information of it.

- We present the first logarithmic-size code-based ring signatures. All constructions prior to ours suffered from linear signature size. Generally speaking, it is non-trivial to design logarithmic size ring signatures since a powerful supporting technique is needed. We design such scheme based on our first logarithmic-size code-based accumulator.
- We construct the first group signatures with logarithmic signature size in code-based assumptions. Our scheme follows the framework of group signature in static model (Bellare et al. - EUROCRYPT 2003) and has shorter signature compared to the lattice constructions and the previous code-based proposals. Furthermore, it achieves CCA2-anonymity.
- We obtain the first code-based verifier-local revocation group signature with backward unlinkability. In 2004, Boneh and Shacham first formalized the concept of Verifier-Local Revocation (VLR) group signatures, in which the up-to-date revocation information only sent to verifiers who can use it to check if a signer has been revoked. Backward unlinkability is a desirable functionality which allows to retain the anonymity of the past signatures of revoked users. This property was put forth by Song in 2001. All existing VLR group signatures with backward unlinkability are based on number-theoretic assumptions that subject to quantum attacks. Our scheme achieves BU-anonymity and trace-

ability by assuming the hardness of the DLPN problem and 2-RNSD problem. Furthermore, we propose the first decentralized e-cash system which relies on code-based assumptions that are conjectured to be quantum resistant and no such scheme appeared in code-based cryptography.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Prof. Huaxiong Wang, for providing me the opportunity to join the excellent cryptography group at NTU, for supporting and helping me during my PhD study, for being such a great role model to me in his immense knowledge and positive attitude towards research.

I would like to show my gratitude to my co-supervisor Dr. Khoa Nguyen, for his insightful discussions and useful advice, for his patience and motivation. I still remember the “10000 hours rule” he shared with me and have been motivated to study hard and pursue high.

I am immensely grateful to all my co-authors: Prof. San Ling, Prof. Huaxiong Wang, Dr. Khoa Nguyen, Dr. Jingwei Hu, Dr. Macro Baldi, Dr. Paolo Santini, Dr. Fuchun Lin, Prof. Thuc D. Nguyen and Hanh Tang. I am very honored to work with all of them. I have two other published papers that are not inserted as chapters in this thesis. I would like to acknowledge the contributions of my co-authors here. In “Threshold Changeable Ramp Secret Sharing”, Prof. Huaxiong Wang and Prof. San Ling gave the initial ideas and took part in the discussions, Dr. Fuchun Lin wrote the initial manuscript and revised the paper and I revised the manuscript and edited proofs. In “Lightweight Key Encapsulation Using LDPC Codes on FPGAs”, Prof. Huaxiong Wang and Prof. San Ling led the project and joined the discussions, Dr. Jingwei Hu edited the manuscript and implementations, Dr. Macro Baldi, Dr. Paolo Santini involved in the discussions and the implementation parts and I revised the

manuscript and joined the discussions.

I would like to thank all crypto group members, for all the stimulating discussions and excellent seminars we had together. Thank you Prof. Huaxiong Wang, Dr. Khoa Nguyen, Dr. Juanyang Zhang, Dr. Yanhong Xu, Dr. Hyung Tae Lee, Dr. Tan Hong Meng Benjamin, Hanh Tang, Dr. Jingnan He, Prof. Xianhui Lu, Dr. Jingwei Hu, as well as all the visiting scholars and exchange PhD students. I also would like to thank all my friends for their patience, kindness and support, for all the amazing time we have had together. Thank you Yi Chen, Qiyun He, Jianing Li, Xiaoman Lu, Rui Huang, Yiming Liu and many amazing friends I met during my study in NTU.

I would also like to thank Nanyang Technological University for all the financial supports including my stipends and travel grants in the last four years.

Most importantly, I am forever grateful to my mother and father for their unconditional and deepest love. They will always be there no matter what happened to me or what decisions I made and let me chase my dream. None of this could have happened without their encouragement and support.

List of Works

The following is the list of works that published or submitted during my PhD study.

1. Fuchun Lin, San Ling, Huaxiong Wang, and Neng Zeng. Threshold Changeable Ramp Secret Sharing. In Cryptology and Network Security - CANS 2019, volume 11829 of Lecture Notes in Computer Science, pages 308-327. Springer, 2019.
2. Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng. New Code-Based Privacy-Preserving Cryptographic Constructions. In Advances in Cryptology - ASIACRYPT 2019, volume 11922 of Lecture Notes in Computer Science, pages 25-55. Springer, 2019.
3. Jingwei Hu, Macro Baldi, Paolo Santini, Neng Zeng, San Ling, and Huaxiong Wang. Lightweight Key Encapsulation Using LDPC Codes on FPGAs. In IEEE Transactions on Computers, volume 69, pages 327-341. IEEE 2019.
4. Khoa Nguyen, Thuc D. Nguyen, Huaxiong Wang, and Neng Zeng. Code-Based Verifier-Local Revocation Group Signatures with Backward Unlinkability and Applications to E-cash. In submission to “Designs, Codes and Cryptography”, March 2020.

Contents

Abstract	1
Acknowledgements	4
1 Introduction	11
1.1 Code-Based Cryptography	11
1.2 Contributions of the Thesis	15
1.3 Organization of the Thesis	20
2 Preliminaries	22
2.1 Notations	22
2.2 Coding Theory	23
2.2.1 Linear Codes	23
2.2.2 Binary Goppa Codes	27
2.3 Code-Based Cryptography	28
2.3.1 Hard Problems	28
2.3.2 Code-Based Cryptographic Primitives	31
2.3.3 Zero-Knowledge Proofs and Arguments	37
2.3.4 Stern's Protocol	41
3 Code-Based Statistically Hiding Commitments With Companion Zero-Knowledge Protocol	46

3.1	Overview of Our Main Techniques	47
3.2	Previous Work	48
3.3	Commitment Schemes	49
3.3.1	Definition and Security Requirements	49
3.4	Description of Our Construction	50
3.5	Extension and Permutating Techniques	52
3.6	ZKAoK of a Valid Opening	54
4	Code-Based Accumulators and Logarithmic-size Zero-Knowledge Arguments for Set Membership	56
4.1	Introduction	56
4.2	Overview of Our Techniques	57
4.3	Cryptographic Accumulators	57
4.4	Hashing with Two Inputs	58
4.5	Code-Based Merkle-tree Accumulators	59
4.6	Logarithmic-size Arguments of Set Membership	62
5	Code-Based Logarithmic-size Ring Signatures	67
5.1	Overview of our Techniques	67
5.2	Definitions and Security Requirements of Ring Signatures	68
5.3	Description of Our Ring Signature	70
5.4	The Underlying Zero-Knowledge Argument System	72
5.5	Analysis of the Scheme	75
6	Code-Based Logarithmic-size Group Signatures	79
6.1	Overview of Our Techniques	79
6.2	Definitions and Security Requirements of Group Signatures	80
6.3	The Randomized McEliece Encryption Scheme	83
6.4	Our Logarithmic-Size Group Signature Scheme	85

6.5	The Supporting Zero-Knowledge Argument System	88
6.6	Analysis of the Scheme	90
7	Verifier-Local Revocation Group Signature with Backward Unlinkability and Code-Based Decentralized E-cash System	96
7.1	Overview of Our Techniques	96
7.2	VLR Group Signatures with Backward Unlinkability	99
7.2.1	Syntax of VLR Group Signature	99
7.2.2	Notions of Correctness and Security Requirements	99
7.3	Our VLR Group Signature Scheme	102
7.4	The Supporting Zero-Knowledge Argument System	105
7.5	Analysis of the Scheme	110
7.6	Decentralized E-cash Scheme	118
7.6.1	Syntax of the Decentralized E-cash Scheme	118
7.6.2	Our Code-Based Decentralized E-cash Scheme	120
7.7	The Supporting Zero-Knowledge Argument System	122
7.7.1	Security Analysis	125
8	Conclusions and Open Problems	130
	Bibliography	133
	Appendices	149
	Appendix A Proof of Theorem 2.3.13	150

List of Figures

2.1	Stern-like ZKAoK for R_{abstract}	44
4.1	A Merkle tree with $2^3 = 8$ leaves	61
5.1	A ring with $2^3 = 8$ members	72

Chapter 1

Introduction

1.1 Code-Based Cryptography

In nowadays computer society, electronic interactions and confidential information raise growing privacy and security concerns and call for more privacy-preserving primitives or constructions. In our daily life, it may cause severe problems if the privacy of the confidential information for patients in a hospital is disclosed. Take another example, it is inconceivable that at a time social media is being hacked and a leakage of online information emerges. As we can see, it is essential to seek more solutions to avoid these situations.

In 1976, public-key cryptography was invented and published by Diffie and Hellman [45]. A large number of currently used public-key infrastructures rely on integer factorization problem or discrete logarithm problem. It has been shown that these two problems are conjectured to be hard in classic computation model, but they are considered to be vulnerable to the quantum algorithms. In the early 80's, Manin [89] and Feynman [49] introduced the concept of quantum computers, which are able to perform simultaneous and sufficiently large computations by using quantum mechanics. In 1994, Shor [118] showed that powerful quantum computers are capable of solving these two problems in polynomial time. It would put our currently used public key

cryptosystems into a risk once efficient and fast quantum computings are available in the “not too distance” future and increasing number of researchers turned to dive into a new direction of “post quantum cryptography” , which studies quantum resistant cryptosystems. In 2016, National Institute of Standards and Technology (NIST) initiated a global post-quantum cryptography standardization project which received extensive attentions and moved into second round earlier this year.

Code-based cryptography has a long history as the public-key cryptography and has been considered as a promising candidate that is immune against quantum attacks, many constructions and primitives are constructed based on the NP hard coding problem of [17] decoding general linear codes. For instance, in an encryption scheme like McEliece cryptosystem [92], one can choose a specific code with efficient decoding algorithm serve as secret key to decrypt the ciphertext and a general random linear code that derived from the secret key serve as public key to encrypt a message. The idea here is that it is hard to decode a random linear code in order to get messages without secret key. Some variants of this problem have been proposed, such as Syndrome Decoding (SD) problem and Learning Parity with Noise (LPN) problem.

Random Linear Codes Decoding problem is a well-known problem in coding theory that has been studied for a long time [17,20,115], it has many applications in cryptography. There are many variants of this problem that are intensively used in building different cryptosystems. One of them called syndrome decoding problem, which asks to find a vector $\mathbf{e} \in \mathbb{F}_q^n$ with small hamming weight ω such that $\mathbf{H}\mathbf{e} = \mathbf{s}$, given a parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and $\omega \in \mathbb{N}$. In 1962, Prange first introduced the information set decoding algorithm by applying linear algebra to reduce the search space to deal with this problem. After that, many algorithms have been proposed to optimize and improve the original algorithm [11, 52, 76, 77, 91, 120] and up to date, the best algorithm still remains on average $\mathcal{O}(2^{n/20})$ operations in decoding random binary linear codes, which is exponential on the length of $\mathcal{C}$.

Another well-known code-based problem is the search LPN problem which defined

to find a secret $\mathbf{s} \in \mathbb{F}_2^n$, given a sample in the form of $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \in \mathbb{F}_2^{m \times n} \times \mathbb{F}_2^m$, where $\mathbf{e}$ is a noise vector and $\mathbf{A}$ is a uniformly random matrix. There is an equivalent version to be the decisional LPN problem which defined to distinguish noisy inner products from random and thus it is also equivalent to the random linear codes decoding problem. Many applications based on search/decisional LPN problem have been found in provably secure cryptosystems since they could be secure even after sufficiently large quantum computers appears, such as symmetric encryption [4, 56], secret key authentication schemes [69, 72] and public key authentication schemes [121]. In [20], Blum et al. first suggest a BKW algorithm for solving LPN problem. They utilize Gaussian elimination to sum up the entries to eliminate more information bits and use the remaining information bit to find the secret in a exhausted way. Later, Leveil and Fouque [78] further optimize the BKW algorithm by applying fast Walsh-Hadamard transformation. Kirchner later transfer the problem into a systematic form and while it is still very efficient in practice. In 2014, [64] suggest a algorithm by using the covering codes. The best known algorithms still run in exponential time.

McEliece [92] is the first one to propose an encryption scheme using Goppa codes, assuming the hardness of a generic decoding problem, it enjoys quantum resistant and has a fast and efficient encryption and decryption procedure compared to the conventional constructions which rely on the number-theoretic assumptions. Since then, many different kinds of constructions in code-based cryptography appeared, such as the Niederreiter encryption scheme [103], the first identification scheme proposed by Stern [121], hash functions [7, 8], a CFS signature scheme [41], random number generators [54]. Unfortunately, almost all subsequently proposed code-based cryptosystems have a common disadvantage of rather large public key size that may cause inefficiency, the field thus did not attract as much attention as lattice-based cryptography and went through periods of slow development. In recent years, a resurgence has been witnessed and many theoretical or practical constructions have been proposed to considerably enrich code-based cryptography. On the one hand, some identity-

based encryption schemes have been introduced [25, 55] and the worst-case hardness for LPN problem has been explored [26] which highly promote the theoretical development. On the other hand, as we can notice, an encouraging result in the second round of the NIST PQC Standardization process is that 7 code-based PKE/KEM algorithms have been accepted. It shown that code-based cryptography has still attracted attentions in practical constructions in recent years. Nevertheless, many open questions are still waiting for exploring.

A subfield of privacy-preserving cryptography is aiming to protect the user's privacy and security, the main code-based tools for constructing such schemes are zero knowledge proofs [61] and argument [29, 57], which allows a prover to prove the validity of a statement to the verifier without revealing any other information. Since Stern proposed an identification scheme in 1996 [122], the following code-based zero knowledge proof/argument systems are mostly influenced by the idea of the Stern's protocol. Some privacy-preserving constructions that employ on the variants of Stern's protocol, such as proofs of plaintext knowledge for PKE [96], proofs of valid commitment openings for commitment schemes have been analysed in [68], proofs leading to anonymity-oriented applications, such as serving as an important building block in ring signatures [28, 93] and group signatures [3, 47], and proofs for general relations [68]. Compared to lattice-based cryptography, the development in code-based cryptography is quite unsatisfied since many important building blocks, primitives and constructions are still unknown to us. The fundamental constructions that may need to prove a hash preimage in zero knowledge and some advanced cryptographic schemes which allow to prove in zero knowledge set membership proofs are still missing in code-based assumptions. It is urgent and interesting to fill these gaps to enrich and advance the code-based privacy-preserving world.

1.2 Contributions of the Thesis

We present five code-based privacy preserving primitives and constructions in this thesis: commitment, accumulator, ring signature, group signature and electronic cash. They all rely on well-studied hard problems in coding theory and each contribution may trigger various other applications. The contributions are as follow.

Motivation 1.

Commitment schemes have two properties: hiding property and binding property. These two properties can be either statistical or computational. In the code-based setting, a computationally hiding commitment scheme was proposed [68], but designing a scheme achieving the hiding property in the statistical sense remains an open question. In this thesis, we aim to fill in this gap.

Contribution 1. We propose a code-based commitment scheme that is statistically hiding and present proofs of valid openings to a committed value.

Our construction is quite standard by extending the input of a collision resistant hash function proposed by Augot, Finiasz and Sendrier (AFS) from one input to two inputs by padding a message with a randomness of sufficient min-entropy. The non-trivial part in our construction lies in the zero knowledge argument system of proving a valid opening that also make our commitment a useful building block in privacy enhancing protocols. In many advanced cryptographic protocols, they allows the building layer to be connected by some common secrets and some supporting tools are needed to connect these layers. Our commitment scheme with supporting zero knowledge proofs/arguments can be easily composable with these layers. Previous work only achieved computationally hiding in code-based setting was given by Jain et al. in [68], which relies on the LPN problem. Our scheme based on the hardness of the 2-Regular Null Syndrome (2-RNSD) problem, which asks to find a specific codeword with small hamming weight and can be considered as a similar version of Syndrome Decoding problem. Our new techniques are specific to the code setting, for which

we prove that hash preimages are properly encoded to sparse and regular vectors. The techniques preserve the structure of regular words and ensure that all bits of the secret vector are kept under control, so that we can prove that these bits satisfy other relations.

Motivation 2.

Cryptographic accumulators were initially put forward by Benaloh and de Mare [14] in 1993. They have numerous applications in privacy-preserving constructions, such as anonymous credentials [1, 84], group signatures [123], set membership proofs [111], authenticated data structures [108], digital cash [35, 95], and so on. From conventional setting (number-theoretic assumptions) [102], lattice setting [81] and symmetric-key primitives [23, 44], many accumulators with zero knowledge proofs have been proposed. However, such constructions have not been considered in code-based assumptions.

There are three families in accumulator setting. The first one is based on the hardness of solving computational problems in groups of hidden order, such as the strong RSA problem. Accumulators in this family typically have small public parameters and can be extended to universal accumulators. The second family is based on pairings and bilinear maps, they have a large parameter setting as linear in the hashed elements and in this kind of constructions, they need to limit the number of the hashed elements to optimize the public parameters. The third one utilizes the advantage of Merkle tree to construction accumulators, and lattice-based Merkle tree accumulators were given following Libert et al.’s framework [81]. In this kind of setting, we need to make it compatible with efficient zero knowledge proof, which is still a challenging problem. It motivates us to build a code-based accumulator in Merkle tree structure and give its companion zero knowledge argument.

Contribution 2. The first logarithmic-size code-based accumulator with underlying zero knowledge arguments for set membership.

Our accumulator is built upon a Merkle-tree [94] and prove in zero knowledge arguments of knowledge of a hash chain in order to show the inclusion of an element

in the accumulate set. Our construction resembles and develops the techniques of the lattice-based accumulators proposed by Libert et al. in 2016 [81], however, some challenging problems emerges when we apply the permutation techniques in zero knowledge argument systems. In their scheme, the output of each level easily serve as the input of the next level by decomposing into binary representation but in our scheme, output must be non-linearly encoded the into a regular word before involving in the next hash level and then prove all processing are done correctly. Luckily, this challenge can be solved by applying our zero knowledge techniques. It is highly non-trivial to construct underlying ZK protocol of our Merkle-tree accumulator, since at each level of the tree, we need to construct proofs of preimage and image in a hash function.

Motivation 3.

Rivest, Shamir, and Tauman [116] proposed the first ring signature in 2001, in which a user is capable of issuing a signature on behalf of a ring set without revealing his identity. In 2006, Bender, Katz and Morselli [15] put forward stronger security definitions. Unlike group signatures [40], ring signature does not require authority managers to control the ring or any coordination among the users. It has been intensively studied in [28, 81]. However, all the previous code-based ring signatures have linear signature size.

Contribution 3. The first code-based logarithmic-size ring signatures.

We present a code-based ring signature achieves signature size logarithmic in the cardinality of the ring. Our construction relies on the code-based logarithmic-size accumulator from Contribution 2. Furthermore, our construction satisfies the stronger security definitions in [15]. The size of our ring signature is around 555KB for 2 users, a good feature is that the our signature size increases slowly for a large group of users. For example, the signature size can be less than 20 MB for a rings of 2^{30} users. Compared to the first lattice-based logarithmic-size ring signature from [81], which has signature size greater than 50 MB for 2^{10} users, our construction is more efficient

and produces shorter signatures.

Motivation 4.

Chaum and van Heyst proposed the concept of group signatures in 1991 [40]. For a group managed by a group manager, every group user is allowed to anonymously sign messages on behalf of the whole group (anonymity) and his identity can be revealed since the group manager has the ability to trace to the actual signer in case of dispute (traceability). Thanks to these nice features, group signatures have many applications in anonymous credentials [34], electronic cash [36], digital rights management [32], and so on. What's more, designing group signature requires to combine many cryptographic ingredients which make it theoretically interesting. Numerous group signatures have been proposed, such as number-theoretic based group signature schemes [22, 82] which may subject to post-quantum computers [118] and lattice-based group signatures [74, 75, 85], which are secure against quantum computers but have large key and signature sizes. For the purposes of not putting all eggs in one basket and bringing in more diversity for post-quantum cryptography, some group signature schemes that rely on coding problems have also been introduced in recent years [3, 47].

Contribution 4. The first code-based logarithmic-size group signatures.

We present the first group signature in static model [13] that produces logarithmic-size signatures from code-based assumptions. Prior to our construction, only linear size and sub-linear code-based group signature were proposed [3, 47]. Furthermore, we reduce the signature size compared to lattice-based constructions (for large group) and solve an open question for achieving CCA-anonymity. A signature compatible with ZK proofs in standard model is traditionally used as an indispensable ingredient for building a CCA-anonymous group signature scheme. However, ours is the first one that achieves CCA-anonymity prior to a standard model one in all branches of cryptography.

Motivation 5.

Group signature has many useful applications and some of them [6, 22, 24] support user revocation, where the signing ability of revoked users is disabled. Membership revocation allows the manager to securely revoke misbehaving members from the group, which is a practically useful feature. Since such situations quite often happen in our daily life what make it a desirable property for actual group signature. There are two main revocation approaches, one processes in the way of reissuing new keys to all users except the revoked users and another approach allows only verifiers to download the information of the revocation list, which is called verifier-local revocation. We only focus on VLR group signatures which satisfy verifier-local revocation in the second method, it allows all verifiers make use of a revocation list (RL) that all the revoked users are recorded to determine if a signature was signed by a revoked one.

Since Boneh and Shacham [24] formalised VLR group signatures in 2004, many follow-up works have proposed concrete instantiations in the bilinear map setting [83, 99] and the lattice setting [75]. An important issue in VLR group signature is whether the anonymity of the signatures signed by a revoked user before his revocation still achieved. The property to make the signatures unlinkable and retain the anonymity of the signatures signed before or after the user revoked is called backward unlinkability. In some constructions, when a user has been revoked, the verification algorithm no longer accepts his signatures and can reveal his previous signature by applying the newest RL which make all his signatures linkable. This property is useful in some cases and can help trace all signatures signed by some revoked but misbehaving person. However, if a user is voluntarily revoked from the group or temporarily leave, he may want his previous signatures still achieve anonymity and unlinkability and then the property of linkability is not desirable here. Song [119] first introduced the *Backward unlinkability* property. Later on, many VLR group signatures with backward unlinkability from bilinear maps were put forth [31, 99, 127]. However, all existing schemes are based on conventional assumptions that can be broken by quan-

tum computing algorithms. It is imperative and desirable to have a quantum-resistant alternative instantiations which are based on code-based assumptions.

Electronic cash was first introduced by Chaum [38, 39] and Bitcoin [9, 97] has gained intensively attentions in the recent years. The advantages of Bitcoin is that it requires no central banks and is fully decentralized, while their anonymity relies on cryptographically pseudonyms and all the information of Bitcoin transactions are public made the transactions might be tracked. In [95], the Zerocoin which achieves the anonymity of the coin transactions was first proposed. The existing proposed Bitcoin systems are either based on RSA assumption [95] or lattice assumption [87], while no such Bitcoin systems rely on code-based assumptions.

Contribution 5. The first code-based VLR group signature with backward unlinkability.

We introduce the first code-based VLR group signature with backward unlinkability. Besides, our scheme achieve BU-anonymity and traceability based on the DLPN problem and 2-RNSD problem. The building blocks that are need in our scheme are a code-based accumulator, a code-based commitment scheme, an LPN instance and a Stern-like [122] argument of knowledge. Additionally, we construct the first code-based decentralized e-cash system.

1.3 Organization of the Thesis

Following is the organization of the thesis.

In Chapter 2, we introduce some general notions in coding theory and some hard problems and primitives in code based cryptography .

In Chapter 3, we describe our code-based commitment scheme and propose proofs of zero knowledge of a hash preimage.

In Chapter 4, we introduce our first code based logarithmic-size accumulators and prove in ZKAoK of set membership.

In Chapter 5, the first code based logarithmic size ring signature was given.

In Chapter 6, we present our code based logarithmic size group signature.

In Chapter 7, the first code-based VLR group signature with backward unlinkability was given and a decentralized e-cash system was proposed.

In Chapter 8, we give a conclusion and present some open problems for further interests.

Chapter 2

Preliminaries

First, we revisit some notions and background on coding theory in Sections 2.1 and 2.2, as well as some hard problems and primitives in code-based cryptography in Section 2.3.

2.1 Notations

Sets and Matrices Throughout this thesis, the classic notations for numbers are sustained, e.g. integer set $\mathbb{Z}$, non-negative integer set $\mathbb{Z}_{\geq 0}$, natural number set $\mathbb{N}$, real number set $\mathbb{R}$ and non-negative real number set $\mathbb{R}_{\geq 0}$. We identify $\mathbb{Z}_2$ as the set $\{0, 1\}$ and $\mathbb{F}_q$ as the finite field, where q is a power of a prime. Denote $\log x$ as logarithm of x with base 2, where $x \in \mathbb{R}$. Denote $[a, b]$ as $\{a, \dots, b\}$, where $a, b \in \mathbb{Z}$. We simply write $[b]$ for $\{1, \dots, b\}$. For a finite set S , denote $x \xleftarrow{\$} S$ as x is chosen uniformly at random from S . All vectors in the thesis are column vectors. Denote $\mathbf{x}^\top$ as the transpose of $\mathbf{x}$. Column vectors in this thesis are represented by bold lower-case letters, matrices use bold upper-case letters, and sets use upper-case letters. $\oplus$ denotes the bit-wise addition of two binary vectors. Denote $(\mathbf{x}^\top \parallel \mathbf{y}^\top)^\top$ as the concatenation of vectors $\mathbf{x} \in \mathbb{Z}_2^m$ and $\mathbf{y} \in \mathbb{Z}_2^k$ and in this thesis we use $(\mathbf{x} \parallel \mathbf{y}) \in \mathbb{Z}_2^{m+k}$ for simplicity. We denote $[\mathbf{A} \parallel \mathbf{B}] \in \mathbb{Z}_2^{n \times (m_0 + m_1)}$ as the concatenation of two matrices $\mathbf{A} \in \mathbb{Z}_2^{n \times m_0}$

and $\mathbf{B} \in \mathbb{Z}_2^{n \times m_1}$. For a string $\mathbf{x}$, denote $wt(\mathbf{x})$ as the Hamming weight of $\mathbf{x}$, $\mathbf{B}(n, \omega)$ as the set of all binary vectors of length n with Hamming weight ω . S_n denotes all symmetric group of all permutations of n elements. $\mathbf{B}_{2^c}^1$ denote all the elements in $\{0, 1\}^{2^c}$ with Hamming weight exactly to be 1. Define distribution distance between D_1, D_2 as $\Delta(D_1, D_2) = \frac{1}{2} \sum |\Pr[X = a] - \Pr[Y = a]|$, if the distance between two distribution is negligible, denote $X \approx_{\text{statistical}} Y$ as X, Y are statistical close.

Other Notations Let λ be a security parameter. A PPT algorithm (efficient algorithm) is denoted to be a probabilistic polynomial time algorithm with respect to its input size. A function f which maps elements in $\mathbb{N}$ to elements in $\mathbf{R}_{\geq 0}$ is *negligible*, denote as $f(\lambda) = \text{negl}(\lambda)$, if $\forall c \in \mathbb{N}, \exists n \in \mathbb{N}$ such that $\forall \lambda \geq n, \|f(\lambda)\| \leq 1/\lambda^c$. A function f is *overwhelming* if $1 - f$ is negligible. If an event happens with negligible probability, then we say the complementary occurs with overwhelming probability. We also use notation $\mathcal{O}$ to estimate the communication cost in this thesis. Let f and g be functions maps elements in $\mathbb{N}$ to real numbers in $\mathbb{R}$, the definitions of $\mathcal{O}$ follows Bachmann-Landau notations [109]. Denote $f(\lambda) = \mathcal{O}(g(\lambda))$ if $\exists c > 0, \exists n \in \mathbb{N}$ s.t. $\forall \lambda > n, f(\lambda) \leq c \cdot g(\lambda)$;

2.2 Coding Theory

Linear codes are widely used in many constructions. We first review some basic definitions and theorems in coding theory following [117], then give a specific code that we will use in our thesis: binary Goppa codes.

2.2.1 Linear Codes

In this section, we will review the definitions for a linear code, which is an error correcting codes.

Definition 2.2.1. (Linear Code) Let $n, k \in \mathbb{Z}_{\geq 0}$, a finite set $\mathbb{F}_q$, where q is a power of a prime integer. An $[n, k]$ linear code $\mathcal{C}$ is defined as a k -dimension subspace of the

n vector space $\mathbb{F}_q^n$.

A codeword is an element in the code and any linear combination of codewords also leads to a codeword. e.g. Any linear combination of any two codewords $\mathbf{a}, \mathbf{b} \in \mathcal{C}$ with any $\alpha, \beta \in \mathbb{F}_q$, we have $\alpha\mathbf{a} + \beta\mathbf{b} \in \mathcal{C}$. Some parameters notations: n denote the length of the code, k represents the dimension of the code, the subtraction of $n - k$ denote the redundancy of the code and the ratio of k/n is the code rate which is the proportion of transmitted data that is non-redundant and useful. Linear codes can be represented by matrices following the definition and we can use some linearly independent vectors to form a basis of $\mathcal{C}$.

Definition 2.2.2. (Hamming Distance) Any two linear codewords $\mathbf{a} = (a_1, \dots, a_n), \mathbf{b} = (b_1, \dots, b_n) \in \mathcal{C}$, the Hamming distance between the codewords denote as $\mathbf{d}_H(\mathbf{a}, \mathbf{b}) = |\{i \mid a_i \neq b_i; i \in [n]\}|$.

Definition 2.2.3. (Hamming Weight) For a linear code $\mathcal{C}$ and any element $\mathbf{a} = (a_1, \dots, a_n) \in \mathcal{C}$, the Hamming weight of the codeword denote as

$$\mathbf{wt}_H(\mathbf{a}) = |\{i \mid a_i \neq 0; i \in [n]\}|.$$

The relations between the hamming weight and hamming distance in a linear code can be captured: for any two codewords $\mathbf{a}$ and $\mathbf{b}$, we have $\mathbf{c} = \mathbf{a} - \mathbf{b} \in \mathcal{C}$, the hamming weight of $\mathbf{c}$ can be easily found to be the hamming distance between these two elements $\mathbf{d}_H(\mathbf{a}, \mathbf{b}) = \mathbf{wt}_H(\mathbf{a} - \mathbf{b}) = \mathbf{wt}_H(\mathbf{c})$. The hamming weight of any codeword is the hamming distance between itself and $\mathbf{0}$, $\mathbf{wt}_H(\mathbf{a}) = \mathbf{d}_H(\mathbf{a}, \mathbf{0})$. In our thesis, let wt be the hamming weight for simplicity instead of $\mathbf{wt}_H$.

Definition 2.2.4. (Minimal Distance) Let d be a minimal distance $\mathcal{C}$,

$$d = \min \{\mathbf{d}_H(\mathbf{a}, \mathbf{b}) : \mathbf{a}, \mathbf{b} \in \mathcal{C}; \mathbf{a} \neq \mathbf{b}\}.$$

The minimal distance of a linear code $\mathcal{C}$ is measured by the minimal of hamming distances between any two codewords in $\mathcal{C}$, which equivalently is the minimal hamming weight of all non-zero codewords $d = \min \{\text{wt}(\mathbf{a} - \mathbf{b}) : \mathbf{a} - \mathbf{b} \neq \mathbf{0}\}$.

If $\mathcal{C}$ is an $[n, k]$ linear code with minimal distance d , we simply denote it as $[n, k, d]$ linear code. The minimal distance of a code is essential to provide its capability of error-correction.

Considering a transmission of a codeword $\mathbf{a}$ passes through a noisy channel and some errors occur leads to the received word be $\mathbf{b} = \mathbf{a} + \mathbf{e}$, where $\mathbf{e}$ is a error vector adding in the transmission with hamming weight t , here we say $\mathbf{b}$ contains t errors. A linear code can detect t errors, for $0 < t < d$. Since the minimal distance of $\mathcal{C}$ is d , the codeword nearest to $\mathbf{a}$ is at least distance d away. It turns out that the received word $\mathbf{b}$ cannot be a codeword as the distance is $t < d$ away from $\mathbf{a}$. Then the receiver can detect errors occur during transmission. When $t \geq d$, no error would be detected since the received word can be distance d away from $\mathbf{a}$ if enough errors introduced in the channel.

A code $\mathcal{C}$ could correct t errors if, given noisy word $\mathbf{b}$, find $\mathbf{e}$ from $\mathbf{b}$ and recover the codeword $\mathbf{a}$, namely, it is able to detect and correct t errors introduced during the codeword transmission.

Theorem 2.2.5. An $[n, k, d]$ linear code $\mathcal{C}$ is able to correct at most $t = \lfloor \frac{d-1}{2} \rfloor$ errors.

Suppose that at most $t = \lfloor \frac{d-1}{2} \rfloor$ errors introduced to a codeword during a noisy channel transmission, on one hand, we want to detect errors, it can be easily done since $t < d$. On the other hand, we want to correct errors to find out the *unique* nearest codeword and specifically prove the uniqueness. For a received word $\mathbf{b} \in \mathbb{F}_q^n$, assuming that there are two nearest codewords $\mathbf{a}, \mathbf{c} \in \mathcal{C}$ s.t. $d(\mathbf{a}, \mathbf{b}) \leq t$ and $d(\mathbf{c}, \mathbf{b}) \leq t$. Then the minimal distance $d_H(\mathbf{a}, \mathbf{c}) \leq d_H(\mathbf{a}, \mathbf{b}) + d_H(\mathbf{c}, \mathbf{b}) \leq 2t \leq d - 1 < d$, since the hamming distance between these two codewords is less than the minimal distance of $\mathcal{C}$, we can deduct that $\mathbf{a} = \mathbf{c}$.

It is desirable that a linear code with as large as possible minimal distance d in order to correct more errors. The theorem below gives an upper bound of d .

Theorem 2.2.6. For an $[n, k, d]$ linear code $\mathcal{C}$, we have $d < n - k + 1$.

It states the upper bound for k which is computed using Singleton bound, which shows that the maximum number of the possible codewords in $\mathcal{C}$ would be less than or equal to q^{n-d+1} , namely, $q^k \leq q^{n-d+1}$, thus $k \leq n - d + 1$ or $d < n - k + 1$.

Definition 2.2.7. (Generator Matrix) For an $[n, k]$ linear code $\mathcal{C}$. Let k linearly independent vectors $\mathbf{a}_1, \dots, \mathbf{a}_k$ form a basis of $\mathcal{C}$. Define a $k \times n$ matrix as the generator matrix of $\mathcal{C}$:

$$\mathbf{G} = \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_k \end{pmatrix}$$

That is, for any message $\mathbf{m} \in \mathbb{F}_q^k$,

$$\mathcal{C} = \{\mathbf{m}\mathbf{G}; \mathbf{m} \in \mathbb{F}_q^k\}$$

The generator matrix for $\mathcal{C}$ is not unique since the choice of basis of the vector subspace is not unique.

Definition 2.2.8. (Parity Check Matrix) An $(n - k) \times k$ matrix $\mathbf{H}$ is defined as a parity check matrix of $\mathcal{C}$, if for any $\mathbf{x} \in \mathbb{F}^n$:

$$\mathbf{x} \in \mathcal{C} \Leftrightarrow \mathbf{H}\mathbf{x}^T = 0$$

Given the generator matrix $\mathbf{G}$ and parity check matrix $\mathbf{H}$ of a linear code, we have $\mathbf{GH} = \mathbf{HG} = 0$.

Definition 2.2.9. (Syndrome) Let $\mathbf{H}$ be the parity check matrix of an $[n, k]$ linear

code $\mathcal{C}$. For $\mathbf{y} \in \mathbb{F}_q^n$, define the syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$ of $\mathbf{y}$ as:

$$\mathbf{s}^T = \mathbf{H}\mathbf{y}^T$$

If $\mathbf{y} \in \mathcal{F}_q^n$ and $\mathbf{y} \in \mathcal{C} \Leftrightarrow \mathbf{s}^T = \mathbf{H}\mathbf{y}^T = 0$

2.2.2 Binary Goppa Codes

McEliece suggested an encryption scheme using Goppa codes in 1978 [92]. Goppa codes were named after the inventor V. D Goppa in 1970s [62]. In this thesis, we only consider the binary Goppa code case for cryptographic purposes.

Let $n, m, t \in \mathbb{N}$, $n \leq 2^m$ and $t < n/m$. Denote a monic Goppa polynomial $g(x)$ over the finite field $\mathbb{F}_{2^m}$ as

$$g(x) = \sum_{i=0}^t g_i x^i \in \mathbb{F}_{2^m}[x]$$

and t is the degree. Define its support $L = \{\alpha_1, \dots, \alpha_n\} \in \mathbb{F}_{2^m}^n$ contains all the elements s.t

$$g(\alpha_i) \neq 0, \forall i \in \{1, \dots, n\}$$

We define a Goppa code $\Gamma(g, L)$ over $\mathbb{F}_{2^m}$ as a set that concludes of all elements $c = (c_1, \dots, c_n) \in \mathbb{F}_2^n$.

$$\Gamma(g, L) = \{c \in \mathbb{F}_2^n \mid \sum_{i=1}^n \frac{c_i}{x - \alpha_i} \equiv 0 \text{ mod } g(x)\}$$

After some simply computations, we obtain:

$$\Gamma(g, L) = \{c \in \mathbb{F}_2^n \mid \sum_{i=1}^n c_i \frac{g(x) - g(\alpha_i)}{x - \alpha_i} (g(\alpha_i)^{-1})\} \quad (2.1)$$

Goppa code is a linear code with dimension $k \geq n - tm$ and minimal distance at least

$t + 1$. The parity check matrix $\mathbf{H}$ for Goppa code $\Gamma(g, L)$, such that, $\mathbf{H}\mathbf{c} = 0$, can be observed in equation 2.1.

$$\mathbf{H} = \begin{pmatrix} \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_n)} \\ \frac{\alpha_1}{g(\alpha_1)} & \cdots & \frac{\alpha_n}{g(\alpha_n)} \\ \vdots & \ddots & \vdots \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix}$$

where $\frac{1}{g(\alpha_i)}$ denote as the inverse element in $\mathbb{F}_{2^m}$.

Definition 2.2.10. (GV Bound) The Gilbert-Varshamov (GV) distance in a linear code $\mathcal{C}$ is the largest integer d_0 s.t.

$$\sum_{j=0}^{d_0-1} \binom{n}{j} (q-1)^j \leq q^{n-k}.$$

2.3 Code-Based Cryptography

Code-based cryptography are all the cryptosystems rely on some computational or decisional coding problems, which also have many applications. We will present some code-based hard problems in Section 2.3.1 and their regarding complexity results. Then we present some code based primitives in Section 2.3.2.

2.3.1 Hard Problems

We present some general hard problems in coding theory and some variants that will be used in later chapters in this section. Many provably secure constructions are constructed based on these hard problems.

Definition 2.3.1. (Bounded Distance Decoding Problem) $\mathcal{C}$ is an $[n, k, d]$ linear code. Given a word $\mathbf{y} \in \mathbb{F}_q^n$ and $t \in \mathbb{N}_{\neq 0}$, find a codeword $\mathbf{x} \in \mathcal{C}$ s.t.

$$d(\mathbf{y}, \mathbf{x}) \leq t$$

Berlekamp, McEliece and van Tilborg first studied the bound distance decoding problem and proved it to be NP-complete [17]. It has two variants called general decoding problem in generator matrix and syndrome decoding problem by considering parity check matrix.

Definition 2.3.2. (General Decoding Problem) Given $\mathbf{G} \leftarrow \mathbb{F}_q^{k \times n}$, $\mathbf{y} \leftarrow \mathbb{F}_q^n$ and $t \in \mathbb{N}_{\neq 0}$. The general decoding problem is to find $\mathbf{x} \in \mathbb{F}_q^k$ and $\mathbf{e} \in \mathbb{F}_q^n$ s.t. $\mathbf{xG} + \mathbf{e} = \mathbf{y}$ and $\text{wt}_H(\mathbf{e}) \leq t$.

Definition 2.3.3. (Syndrome Decoding Problem) Let $\mathbf{H} \leftarrow \mathbb{F}_q^{(n-k) \times n}$, $\mathbf{y} \leftarrow \mathbb{F}_q^{n-k}$ and $t \in \mathbb{N}_{\neq 0}$. The syndrome decoding problem is to find $\mathbf{e} \in \mathbb{F}_q^n$ s.t. $\mathbf{He} = \mathbf{y}$ and $\text{wt}_H(\mathbf{e}) \leq t$.

In the syndrome decoding problem, if $t \leq d_0$, we are able to find a unique solution for the problem. In [106] shows that multiples solutions exists if $t > d_0$. If the given syndrome $\mathbf{y}$ is random, then it guarantees the hardness of the problem. Moreover, the weight of $\mathbf{e}$ would be close to the GV bound. But in practical aspect, the t would be much smaller than bounded by the correction range of the code.

The results for complexities of this problem started from information set decoding algorithm, which is proposed by Prange in 1962 [112], after that, several algorithms have been optimized [11, 52, 76, 77, 91, 120] and the best algorithm is still in $\mathcal{O}(2^{n/20})$ operations in $\mathbb{F}_2$, which is exponential on the length of $\mathcal{C}$.

Next, We present some problems that serve as the hard assumptions for later chapters.

The 2-Regular Null Syndrome Decoding (2-RNSD) problem was suggested by Augot, Finiasz and Sendrier [7, 8]. This problem is similar to the well-studied Syndrome Decoding Problem, Small Codeword Problem [88] and binary Shortest Vector Problem [5] of finding small weight codewords, with a strong constraint that it must be 2-regular words.

Before directly going to this problem, we give some notations that are needed. For

$c \in \mathbb{N}$ and $c|k$, a regular word $\mathbf{x} \in \text{Regular}(k, c)$ is denoted with $\mathbf{x} = (\mathbf{x}_1 \| \dots \| \mathbf{x}_{k/c}) \in \{0, 1\}^{2^c \cdot k/c}$, where $wt(\mathbf{x}_i) = 1 \ \forall i \in [k/c]$ and the length of each block is 2^c .

Define a regular encoding function $\text{RE}: \{0, 1\}^k \rightarrow \{0, 1\}^{2^c \cdot k/c}$ that maps any vector $\mathbf{y}$ into a regular word $\text{RE}(\mathbf{y})$ in the following way. Dividing $\mathbf{y}$ into k/c blocks s.t. $\mathbf{y} = (\mathbf{y}_1 \| \dots \| \mathbf{y}_{k/c})$ with each block length $\mathbf{y}_i$ to be c . The encoding processing for each block works as follows. Take $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,c})^\top$ for $i \in [k/c]$, compute $t_i = \sum_{j=1}^c 2^{c-j} \cdot y_{i,j} \in [0, 2^c - 1]$ based on the entries in $\mathbf{y}_i$ and then put 1 at the t_i -th position in a string of length 2^c to obtain $\mathbf{w}_i$, thus each block $\mathbf{y}_i$ is encoded into a larger string block $\mathbf{w}_i$, where $wt(\mathbf{w}_i) = 1$. The encoding value $\text{RE}(\mathbf{x})$ is then in the form of $(\mathbf{w}_1 \| \dots \| \mathbf{w}_{k/c}) \in \text{Regular}(k, c)$.

Remark: Take a block with length $c = 3$ as an example, for block 000, one encodes it into a larger one with length 8 of hamming weight exactly to be 1, the corresponding decimal value for 000 is 0, so we get 10000000 as the encoded value; for block 011, the corresponding decimal value is 3, then it should be converted into 0001000.

We denote the addition of two regular words as **2-Regular** words. For any two regular words $\text{RE}(\mathbf{x}), \text{RE}(\mathbf{y})$, we have $\text{RE}(\mathbf{x}) + \text{RE}(\mathbf{y}) \in \text{2-Regular}(k, c)$. It straightforward to check that the hamming weight of each block in a **2-Regular** word is either 0 or 2.

Definition 2.3.4. (The $\text{2-RNSD}_{n,k,c}$ Problem) Let $n, k, c \in \mathbb{N}$. Given a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$, where $m = 2^c \cdot k/c$, the $\text{2-RNSD}_{n,k,c}$ problem asks to find a non-zero $\text{2-Regular}(k, c)$ vector $\mathbf{z} \in \{0, 1\}^m$ such that $\mathbf{B} \cdot \mathbf{z} = \mathbf{0}$.

The problem is known to be **NP**-complete in the worst case [7]. Intuitively speaking, the problem is at least as hard as Small Codeword Finding problem. Furthermore, it demands to find a vector that is not only of small weight, but also satisfy specific structure of the hamming weight in each block is either 0 or 2. In [18], there are some known attacks to this problem and give a thorough discussion on the parameter settings. Complexity for solving this problem still requires exponential times even in the best algorithm up to date. We can achieve practical security by appropriately

selecting values for parameters n, k, c to make our constructions that based on this problem to be practical.

Definition 2.3.5. (Decisional Learning Parity with Noise (DLPN) Problem)

The problem asks to determine whether a pair $(\mathbf{A}, \mathbf{s}) \in \mathbb{Z}_2^{n \times k} \times \mathbb{Z}_2^n$ is uniformly chosen from $\mathbb{Z}_2^{n \times k} \times \mathbb{Z}_2^n$ or is obtained by choosing $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n \times k}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_2^k$, $\mathbf{e} \in \mathcal{B}(n, t)$ and outputting $(\mathbf{A}, \mathbf{A} \cdot \mathbf{u} \oplus \mathbf{e})$.

The DLPN problem is hard, if for any PPT distinguisher, the success probability is not better than guessing, namely the probability is at most $1/2 + \text{negl}(\lambda)$.

Definition 2.3.6. (Search LPN Problem) The problem asks to find the secret $\mathbf{u} \in \mathbb{Z}_2^k$, given sample of form $(\mathbf{A}, \mathbf{A} \cdot \mathbf{u} \oplus \mathbf{e})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n \times k}$, $\mathbf{e} \xleftarrow{\$} \mathcal{B}(n, t)$.

In [20], Blum et al first suggest a BKW algorithm for solving LPN problem. They utilize Gaussian elimination to sum up the entries to eliminate more information bits and use the remaining information bit to find the secret in an exhausted way. Later, Leveil and Fouque [78] further optimize the BKW algorithm by applying fast Walsh-Hadamard transformation. Kirchner later transfer the problem into a systematic form and while it is still very efficient in practice. In 2014, [64] suggest an algorithm by using the covering codes. The best known algorithms still run in exponential time.

There is a search to decision reduction for LPN problem [70].

2.3.2 Code-Based Cryptographic Primitives

We will present some cryptographic primitives, such as public-key encryption, hash function and zero knowledge protocol, as well as some code-based constructions that we may use in our thesis.

Cryptographic Hash Function

We first present the definition and three security requirements for hash function and review a specific AFS hash function that will be used in later chapters.

A cryptographic hash function $f : \{0, 1\}^* \rightarrow \{0, 1\}^n$, is defined to map binary strings with arbitrary length into shorter fixed length binary strings. We now demonstrate three security requirements.

Preimage resistance It requires to find preimage $\mathbf{x} \in \{0, 1\}^*$ for a given $\mathbf{y} \in \{0, 1\}^n$, s.t.

$$f(\mathbf{x}) = \mathbf{y}.$$

Second preimage resistance It asks to find another distinct preimage $\mathbf{x}' \neq \mathbf{x} \in \{0, 1\}^*$, for a given $\mathbf{x} \in \{0, 1\}^*$ s.t.

$$f(\mathbf{x}) = f(\mathbf{x}').$$

Collision resistance It demands to find two distinct preimages $\mathbf{x} \neq \mathbf{x}' \in \{0, 1\}^*$ s.t.

$$f(\mathbf{x}) = f(\mathbf{x}').$$

AFS Hash Function

AFS hash function was first put forward by Augot, Augot, Finiasz and Senderier (AFS) [7, 8], which can be proved to be collision resistant under the hardness of the 2-RNSD problem. Designing AFS hash functions are quite similar to some other latest proposed code-based hash functions [5, 27, 126] that are based on different assumptions. The results in [18, 19] allow us to choose practical parameters that guarantee the security of AFS hash function. In this thesis, our commitment presented in Section 3.4 and accumulator in Section 4.5 constructions are all based on this code-based hash functions.

The construction of AFS hash function is as follows. Let λ be the security parameter, $n, k = \Omega(\lambda)$, $k > n$ and $c|k$. It is defined as a function $\mathcal{H}_{\text{afs}}$ which maps $\{0, 1\}^k$ to $\{0, 1\}^n$ and precisely speaking, the elements can be described as keyed functions $h_{\mathbf{P}}$, namely, $\mathcal{H}_{\text{afs}} = \{h_{\mathbf{P}} : \{0, 1\}^k \rightarrow \{0, 1\}^n\}$, where $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_2^{n \times 2^{c \cdot k/c}}$. On input $\mathbf{x} \in \{0, 1\}^k$,

one encodes it into $\text{RE}(\mathbf{x})$ before hashing it to $h_{\mathbf{P}}(\mathbf{x}) = \mathbf{P} \cdot \text{RE}(\mathbf{x})$.

The AFS hash function is given as follows.

- **Parameter:** $n, k = \Omega(\lambda)$, $k > n$ and $c|k$.
- **Key Matrix:** A hash matrix $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_2^{n \times 2^c \cdot k/c}$.
- **Hash Function:** For $\mathbf{x} \in \{0, 1\}^k$, $h_{\mathbf{P}}(\mathbf{x}) = \mathbf{P} \cdot \text{RE}(\mathbf{x}) \in \{0, 1\}^n$.

Theorem 2.3.7. *The AFS hash function is collision-resistant if the $2\text{-RNSD}_{n,k,c}$ problem is hard.*

Assuming that the adversary finds a collision $\mathbf{P} \cdot \text{RE}(\mathbf{x}_0) = \mathbf{P} \cdot \text{RE}(\mathbf{x}_1)$ for some $\mathbf{x}_0 \neq \mathbf{x}_1$, from which yields a $2\text{-Regular}(k, c)$ word $\mathbf{y} = \text{RE}(\mathbf{x}_0) \oplus \text{RE}(\mathbf{x}_1) \neq \mathbf{0}$ such that $\mathbf{P} \cdot \mathbf{y} = \mathbf{0}$. We can deduct if we can find a collision for the AFS hash function, which means we obtain a solution to $2\text{-RNSD}_{n,k,c}$ problem associated with matrix $\mathbf{P}$.

Public-Key Encryption Scheme

We now present an important cryptographic primitive in modern cryptography, the public-key encryption scheme (PKE). The definition and security notions for a PKE are as follows.

Formally, a PKE consists of triple algorithms $(\text{Gen}, \text{Enc}, \text{Dec})$:

- **Gen(λ):** Take the security parameter $\lambda \in \mathbb{N}$, this PPT key generation algorithm **Gen** outputs $(\mathbf{pk}, \mathbf{sk})$, where $\mathbf{pk}$ is the public key and $\mathbf{sk}$ is the secret key.
- **Enc($\mathbf{pk}, m, r$):** Take $\mathbf{pk}$, message $m \in \mathcal{M}$, this PPT encryption algorithm outputs a ciphertext $c \in \mathcal{C}$.
- **Dec($\mathbf{sk}, c$):** Take $\mathbf{sk}$ and a ciphertext c . This deterministic polynomial-time decryption algorithm outputs m or $\perp$ otherwise.

We now demonstrate correctness and three security requirements for PKE schemes: IND-CPA , IND-CCA1 and IND-CCA2 .

Correctness PKE require that any encrypted ciphertext can be retrieved to the message with the decryption algorithm. For all keys pair $(\mathbf{pk}, \mathbf{sk})$ that generated by $\text{Gen}(1^\lambda)$ and message $m \in \mathcal{M}$, the following conditions hold with overwhelming probability:

$$\Pr[\text{Dec}(\mathbf{sk}, \text{Enc}(\mathbf{pk}, m, r)) = m] > 1 - \text{negl}(\lambda).$$

One-wayness under chosen-plaintext attacks (OW-CPA)

It is a basic security notion for PKE. Given $\mathbf{pk}$ and $c \in \mathcal{C}$, it is infeasible for a PPT adversary $\mathcal{A}$ who does not hold the corresponding secret key $\mathbf{sk}$ to recover the message m .

In 1982, Goldwasser and Micali [59] first mentioned semantic security notion for an encryption scheme, which requires that no PPT adversary $\mathcal{A}$ is able to learn anything about the plaintext for a given ciphertext, however no method was given to prove an encryption scheme is semantic security. In 1984, they [60] formulated a security notion which is called ciphertext indistinguishability under chosen plaintext attack by giving public key to $\mathcal{A}$, and showed that this security notion was equivalent to semantic security. IND-CPA describes that given two distinct messages and an encryption of one of them, there is no advantage better than random guessing for any adversary to distinguish which one of the messages was the actual plaintext. In 1990, Naor and Yung [100] formalized the notion of IND-CCA1, in which the adversary can additionally get an access to a decryption oracle and make queries to the oracle before receiving the challenge ciphertext c^* . In 1991, Rackoff and Simon [114] then formalized a stronger security notion, IND-CCA2, in which the adversary $\mathcal{A}$ can not only make decryption queries before receiving the challenge ciphertext c^* but also even after being given c^* under restriction that c^* has not been queried.

Next we present these three standard security notions and give the game definitions as follows [65].

Indistinguishability against chosen-plaintext attacks (IND-CPA) [60]

Given a PPT adversary $\mathcal{A}$ and define $\mathbf{Exp}_{\text{PKE},\mathcal{A}}^{\text{IND-CPA}}(\lambda)$ in game-based experiment.

$\mathbf{Exp}_{\text{PKE},\mathcal{A}}^{\text{IND-CPA}}(\lambda)$
 $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(\lambda)$
 $(\text{state}, m_0, m_1) \leftarrow \mathcal{A}(\text{pk})$
 $b \leftarrow \{0, 1\}$
 $c^* \leftarrow \text{Enc}(\text{pk}, m_b)$
 $b' \leftarrow \mathcal{A}(\text{st}, c^*)$
Return b'

The advantage of the adversary $\mathcal{A}$ against IND – CPA denoted as $\mathbf{Adv}_{\text{PKE},\mathcal{A}}^{\text{IND-CPA}}(\lambda)$.

$$\mathbf{Adv}_{\text{PKE},\mathcal{A}}^{\text{IND-CPA}}(\lambda) = \left| \Pr[\mathbf{Exp}_{\text{PKE},\mathcal{A}}^{\text{IND-CPA}}(\lambda) = 1] - 1/2 \right|.$$

PKE satisfies IND – CPA if for any PPT adversaries $\mathcal{A}$, the advantage in the above experiment game is $\text{negl}(\lambda)$.

Indistinguishability against chosen-ciphertext attacks (IND-CCA1) [100]

This standard security notion $\mathbf{Exp}_{\text{PKE},\mathcal{A}}^{\text{IND-CCA1}}(\lambda)$ is described as below.

$\mathbf{Exp}_{\text{PKE},\mathcal{A}}^{\text{IND-CCA1}}(\lambda)$
 $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$
 $(\text{state}, m_0, m_1) \leftarrow \mathcal{A}^{\text{Dec}(\text{sk}, \cdot)}(\text{pk})$
 $b \leftarrow \{0, 1\}$
 $c^* \leftarrow \text{Enc}(\text{pk}, m_b)$
 $b' \leftarrow \mathcal{A}(\text{st}, c^*)$
Return b'

The advantage of the adversary $\mathcal{A}$ against IND – CCA1 is denoted as $\mathbf{Adv}_{\text{PKE},\mathcal{A}}^{\text{IND-CCA1}}(\lambda)$ is defined as follows:

$$\mathbf{Adv}_{\text{PKE},\mathcal{A}}^{\text{IND-CCA1}}(\lambda) = \left| \Pr[\mathbf{Exp}_{\text{PKE},\mathcal{A}}^{\text{IND-CCA1}}(\lambda) = 1] - 1/2 \right|.$$

PKE is said to be IND – CCA1 if the above advantage is $\text{negl}(\lambda)$.

Indistinguishability against chosen-ciphertext attacks (IND-CCA2) [114]

This standard security notion $\mathbf{Exp}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA2}}(\lambda)$ is described as below.

$\mathbf{Exp}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA2}}(\lambda)$
 $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$
 $(\text{state}, m_0, m_1) \leftarrow \mathcal{A}^{\text{Dec}(\text{sk}, \cdot)}(\text{pk})$
 $b \leftarrow \{0, 1\}$
 $c^* \leftarrow \text{Enc}(\text{pk}, m_b)$
 $b' \leftarrow \mathcal{A}^{\text{Dec}(\text{sk}, \cdot)}(\text{st}, c^*)$, c^* not queried to $\text{Dec}(\text{sk}, \cdot)$
Return b' Denote $\mathbf{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA2}}(\lambda)$ as the advantage of the adversary $\mathcal{A}$ against IND – CCA2 security.

$$\mathbf{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA2}}(\lambda) = \left| \Pr[\mathbf{Exp}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA2}}(\lambda) = 1] - 1/2 \right|.$$

PKE achieves IND – CCA2 if for PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above experiment game to be

$$\max \mathbf{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA2}}(\lambda) < \text{negl}(\lambda)$$

McEliece Cryptosystem

McEliece first suggest an encryption scheme constructed from Goppa codes [92] in 1978. However, this original version is not IND-CPA. Later in [104], a randomized McEliece provides this security. In this part, we review the original McEliece encryption scheme and leave the randomized one in Chapter 6.

The McEliece encryption scheme consists of the following three probabilistic algorithms (Gen_{ME} , Enc_{ME} , Dec_{ME}):

- $\text{Gen}_{ME}(\lambda)$: The probabilistic polynomial-time key generation algorithm Gen_{ME} works as follows:
 1. Generate a $k \times n$ generator matrix $\mathbf{G}$ of a binary Goppa code with an efficient

decoding algorithm **Decode** that capable of correcting up to t errors.

2. Compute the permuted and scrambled matrix $\mathbf{G}' = \mathbf{SGP}$, where $\mathbf{P}$ is a $n \times n$ random permutation matrix and $\mathbf{S}$ is a $k \times k$ random non-singular matrix. Output secret key $\mathbf{sk} = (\mathbf{S}, \mathbf{G}, \mathbf{P})$ and public key $\mathbf{pk} = (\mathbf{G}', t)$.

- $\text{Enc}_{ME}(\mathbf{pk}, \mathbf{m}, \mathbf{e})$: On inputs $\mathbf{pk}$, a message $\mathbf{m} \in \mathcal{M}$ and output a ciphertext $\mathbf{c} = \mathbf{mG}' + \mathbf{e}$, where $\mathbf{e} \leftarrow \mathcal{B}(n, t)$.
- $\text{Dec}_{ME}(\mathbf{sk}, \mathbf{c})$: Take $\mathbf{sk}$ and $\mathbf{c}$, this deterministic polynomial-time decryption algorithm decrypts follows:
 1. Multiply the inverse matrix of $\mathbf{P}$ to $\mathbf{c}$ to obtain $\mathbf{cP}^{-1} = (\mathbf{mS})\mathbf{G} \oplus \mathbf{eP}^{-1}$.
 2. Decoder correct up to t errors and obtain $\mathbf{mS} = \text{Decode}(\mathbf{cP}^{-1})$, then easily obtain $\mathbf{m} = (\mathbf{mS}) \cdot \mathbf{S}^{-1}$.

In Chapter 6, we will make use of the randomized McEliece encryption which is constructed based on the original construction described above by padding a randomness with the message to achieve IND-CPA.

2.3.3 Zero-Knowledge Proofs and Arguments

In 1989, Goldwasser, Micali and Rackoff [61] first put forward zero-knowledge, which is considered as an interactive protocol between two parties, a prover and a verifier. It is an important primitive that serve as a building in many privacy-enhancing constructions. A *Prover* wants to convince a *Verifier* that he possesses a valid statement and prove to the verifier only the fact that the statement without revealing any other information. It consists of a series of challenge and response phases, the verifier outputs “1” if it is convinced that the statement is valid and “0” otherwise. Take Σ protocol as an example, the prover first computes a commitment and gives it to the verifier. The verifier then chooses a random challenge value and sends back to the prover who then computes a corresponding response according to the challenge

and returns it to the verifier. At last the verifier outputs “0” or “1” depending on the commitment and challenge. We require three properties for a zero knowledge protocol: completeness, soundness and zero knowledge. Loosely speaking, completeness means that an honest prover (with the secret) is capable of convincing the verifier with overwhelming probability. Soundness indicates that a malicious prover (without secret) should not be able to convince the verifier. Zero knowledge property indicates that the verifier learns nothing about the prover’s secret from the interaction .

Definition 2.3.8. (NP Language) An relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ is a polynomial time bounded binary relation, where $|w| \leq \text{poly}(|x|)$ and $R(x, w) = 1$ or $(x, w) \in R$ can be recognized in polynomial time. An NP language L_R is a set of all statements x that there exists witness w such that $R(x, w) = 1$ and it is associate with an NP relation R .

Definition 2.3.9. (Zero Knowledge Proofs and Arguments) Denote $\langle P(w), V(x) \rangle$ as a two party interactive proof system with soundness error s , where s is a function maps element in $\mathbb{N}$ to a fraction number in $[0, 1]$ for a NP language L_R if:

- (Perfect) Completeness The verifier always output “1” for a valid statement. Formally speaking, for all $x \in L$,

$$\Pr [\text{out}_V \langle P(x) \leftrightarrow V(x) \rangle = 1] = 1.$$

- Soundness For all $x \notin L$ ($\forall w', R(x, w') = 0$) and for any prover P^* . In $\langle P^*(x), V(x) \rangle$, verifier rejects with overwhelming probability at least $1 - s$.

$$\Pr [\text{out}_V \langle P^*(x) \leftrightarrow V(x) \rangle = 1 \mid \forall x \in L_R] \leq s.$$

Remark 1. If it holds for a PPT prover P^* , it is called zero knowledge arguments; it is called zero knowledge proofs for computationally unbounded P^* . In our thesis, the proofs used in later chapters are all focused on zero knowledge arguments system.

Remark 2. The soundness error $s(\lambda)$ is a polynomial time functions of a security parameter λ .

Zero knowledge property ensures that a verifier V^* gains no extra knowledge besides the statement if without any help from the prover P . We need to describe this property using a simulator that act as V^* by simulating an interaction that is statistically close to the real interactions between the honest P and V^* on a common input. The simulatability requires that whatever V^* computes by interacting with P can be directly computed by an efficient algorithm on the same input. Here a probabilistic polynomial time algorithm can be used as a simulator to simulate the verifier's view. In [61], a relaxation definition has been given which allows the simulator performs in expected polynomial time. We call a simulator fail if it outputs a symbol $\perp$, we only consider a useful simulator which is conditioned on non-failure whose probability of outputting $\perp$ is no more than $1/2$. Denote $Sim_{V^*}(x)$ as the output of Sim simulates the view of verifier V^* on input of $x \in L$.

We will give definition of statical zero knowledge argument systems.

Definition 2.3.10. (Statistical Zero Knowledge Arguments System [61, 105])

For all PPT verifier V^* , there exists a PPT simulator Sim s.t. for every $x \in L_R$ with $(x, w) \in R$,

$$View_{V^*}\langle P(w) \leftrightarrow V^*(x, inf) \rangle \approx_{\text{statistical}} Sim_{V^*}(x, inf),$$

here inf denotes any information about the statement.

A zero knowledge argument allows the prover P shows that he has a statement that is valid, which means a witness for the statement exists. Moreover, there is a stronger notion called argument of knowledge [12, 61], which allows the prover additionally proves that he knows regarding witness. This property can be captured by an knowledge extractor Ext . For every PPT prover P^* s.t. $\langle P^*(w) \leftrightarrow V(x) \rangle = 1$, there exists an efficient extractor Ext that is able to output a witness such that $R(x, w) = 1$

through an oracle access to P^* .

Definition 2.3.11. (Argument of Knowledge [12]) $\langle P, V \rangle$ is a zero knowledge argument of knowledge with knowledge error κ for an NP relation R if for every PPT prover P^* , there exists a PPT extractor Ext s.t. for every $x \in L_R$,

$$\Pr [v \leftarrow Ext^{\mathcal{O}(P^*)} : R(x, v) = 1] \geq \text{poly}(p - \kappa),$$

where $p = \Pr [out_V[P^* \leftrightarrow V(x)] = 1]$.

Witness indistinguishability [48] shows that even a computationally unbounded verifier cannot distinguish a proof from which witness among w_1 and w_2 is used for a statement x . This security is weaker than zero knowledge. Thus a zero knowledge scheme is witness indistinguishable. In the contrapositive way, it is easy to show that a scheme is not zero knowledge by indicating it does not hold the weaker notion of witness indistinguishability.

Definition 2.3.12. (Σ -Protocol) A Σ -protocol for relation R consists of the following three moves.

1. The prover sends a commitment $c = \alpha(x, w, r_P)$, where r_P is a random tape, to the verifier.
 2. The verifier computes a random challenge $e = \beta(r_V)$, where r_V is a random tape, and return it back to the prover.
 3. The prover outputs a response $s = \gamma(x, w, r_P, e)$.
- **Completeness:** If $(x, w) \in R$, the prover and the verifier honestly follow the protocol, then verifier always accepts in $\langle P, V \rangle$.

$$\Pr [out_V \langle P(w) \leftrightarrow V(x) \rangle = 1 \mid \forall (x, w) \in R] = 1.$$

- **Special soundness:** For any x , any two accepting scripts (c, e, s) , (c, e', s') , where

$e \neq e'$, there exist a PPT algorithm to compute witness v such that $(x, v) \in R$.

It requires that a valid witness can be computed by giving any two accepting scripts with same commitment and different challenge values.

- **Honest verifier zero knowledge:** For any x , challenge e , there exists a PPT simulator produces an accepting script (c, e, s) , which has the same distribution as the interaction between an honest verifier and a prover on input x and challenge e .

Here a malicious prover has a probability at most $1/|\beta(\cdot)|$ succeed in guessing the challenge.

In Σ -protocol, the existence of a simulator only required for honest verifier, while zero knowledge proofs require for any (potentially malicious) verifier. Σ protocol can be converted into zero knowledge proofs by applying Damgård et al.'s generic standard techniques [43].

2.3.4 Stern's Protocol

Stern-like protocols. In 1996, Stern [122] proposed a three move Zero knowledge protocol for the Syndrome Decoding relation. Random masking and permutations are two main techniques used in this protocol. The high level ideas are hiding the secrets with random maskings and proving the constraints of the secrets through permutations. In this section, we present a Stern-like statistical ZKAoK protocols which satisfy perfect completeness, constant soundness error $2/3$. They follow Σ -protocols with three interactive moves of commitment phase, challenge phase and response phase as defined in Definition 2.3.12. We apply our code-based statistically hiding commitment in Chapter 3 to the first phase to obtain a statistical ZKoAK. Moreover, it has perfect completeness as Σ -protocol and there exist a PPT simulator that can simulate as the verifier to interact with the prover as the real one with

soundness error $2/3$. However, the soundness error can be obtained negligibly small when we apply some repetition techniques. In practice, One can achieve soundness error 2^{-80} by repeating the protocol for at least 137 times.

An abstraction of Stern-like protocols. We revisit an abstraction of Stern's protocol in [79]. This abstraction works well when many complicated relations are given in a zero knowledge proof, it can unify all of them into a simple yet versatile equation and develop the main idea of the original Stern's to unify the permutation set and the secret constraints set. It enjoys a much clear explanation and provides easier understanding for the zero knowledge. All the relations in our ZKAoK systems can be converted into instances of the abstract protocol.

For positive integers k, l , with $k \leq l$. Define a set $\text{VALID} \subseteq \{0, 1\}^l$ and a finite set $\mathcal{S}$ and a permutation Γ_ϕ of l elements for any $\phi \in \mathcal{S}$, such that satisfy the following conditions:

$$\begin{cases} \mathbf{w} \text{ in the VALID set} \iff \Gamma_\phi(\mathbf{w}) \text{ in the VALID set,} \\ \text{If } \mathbf{w} \in \text{VALID and } \phi \text{ is uniformly chosen in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in VALID.} \end{cases} \quad (2.2)$$

This protocol is a statistical ZKAoK for the following relation with public input $\mathbf{M}$ and $\mathbf{v}$ and secret input $\mathbf{w}$:

$$R_{\text{abstract}} = \left\{ (\mathbf{M}, \mathbf{v}) \in \mathbb{Z}_2^{k \times l} \times \mathbb{Z}_2^k; \mathbf{w} \in \text{VALID} : \mathbf{M} \cdot \mathbf{w} = \mathbf{v} \right\}.$$

$\mathbf{w}$ denote all the secrets which are in the VALID set. $\mathbf{M}$ can be obtained using simple algebra techniques. Take the original Stern's protocol as an example, we have $R_{\text{Stern}} = \left\{ (\mathbf{A}, \mathbf{s} \in \mathbb{Z}_2^{k \times l} \times \mathbb{Z}_2^k; \mathbf{e} \in \text{VALID} : \mathbf{A} \cdot \mathbf{e} = \mathbf{s}) \right\}$. where $\text{VALID} = \{\mathbf{x} \in \{0, 1\}^l | wt(\mathbf{x}) = t\}$ and ϕ is uniformly chosen in $\mathcal{S}_l$ and $\Gamma_\phi(\mathbf{e}) = \phi(\mathbf{e})$ with hamming weight t in the set VALID.

Following the conditions in (2.2), One can prove in zero knowledge of $\mathbf{w} \in \text{VALID}$. The main ideas are randomization and permutations. On one hand, using random

masking to prove the linear equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{v}$ in zero knowledge, the prover samples a mask $\mathbf{r}_w$ for $\mathbf{w}$ to show that $\mathbf{M} \cdot (\mathbf{w} \oplus \mathbf{r}_w) = \mathbf{M} \cdot \mathbf{r}_w \oplus \mathbf{v}$. Since $\mathbf{r}_w$ is a randomness, it can perfectly hide the information of the secret while proving the linear equation. On the other hand, using random permutations to prove the constraints for the secrets in zero knowledge, the prover samples $\phi \xleftarrow{\$} \mathcal{S}$ and send it to the verifier who goes to check that $\Gamma_\phi(\mathbf{w}) \in \text{VALID}$, it guarantees that the verifier learns nothing no more than $\Gamma_\phi(\mathbf{w})$.

A commitment that is statistically hiding and computationally binding is an important component in building a statistical ZKAoK protocol. In this thesis, we apply our first construction to obtain a code-based statistical ZKAoK. The details of the interaction between $\mathcal{P}$ and $\mathcal{V}$ is given in Figure 2.1.

We need to prove the zero knowledge property and proof of knowledge property by constructing a PPT simulator and PPT extractor, respectively. A statistically hiding and computationally binding commitment are highly required in the proof. We only give a brief overview here. The hiding property of the commitment guarantees the commitments send in the first phase reveal nothing about the secret for constructing the simulator. The binding property is used for extracting the secret. We present the following Theorem 2.3.13 to demonstrate the details. The detailed proofs are attached in Appendix A.

Theorem 2.3.13. [79] Let COM be a statistically hiding and computationally binding string commitment scheme. Then, the interactive protocol in Figure 2.1 is complete with perfect completeness, statistical zero knowledge, with constant soundness error $2/3$, and communication cost $\mathcal{O}(L)$. Specifically:

- There is a PPT simulator that output an accepted transcript statistically close to that produced by the real prover on $(\mathbf{M}, \mathbf{v})$.
- There is a PPT knowledge extractor that output $\mathbf{z} \in \text{VALID}$ such that $\mathbf{M} \cdot \mathbf{z} = \mathbf{v}$, under the conditions of a commitment CMT and 3 valid responses

1. **Commitment:** Prover samples a mask vector $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, permutation $\phi \xleftarrow{\$} \mathcal{S}$ and randomness ρ_1, ρ_2, ρ_3 for COM. Then he sends $\text{CMT} = (C_1, C_2, C_3)$ to the verifier, where

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

2. **Challenge:** The verifier uniformly chosen a challenge Ch in $\{1, 2, 3\}$ and sends it to the prover.
3. **Response:** The prover computes RSP with respect to the commitments and challenge:
 - $Ch = 1$: Let $\mathbf{t}_w = \Gamma_\phi(\mathbf{w})$, $\mathbf{t}_r = \Gamma_\phi(\mathbf{r}_w)$, output $\text{RSP} = (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$. Compute openings for C_2 and C_3 for verification, which are $\pi_2 = \text{COM}(\mathbf{t}_r; \rho_2)$, $\pi_3 = \text{COM}(\mathbf{t}_w \oplus \mathbf{t}_r; \rho_3)$. Check $\mathbf{t}_w \in \text{VALID}$.
 - $Ch = 2$: Let $\phi_2 = \phi$, $\mathbf{w}_2 = \mathbf{w} \oplus \mathbf{r}_w$, output $\text{RSP} = (\phi_2, \mathbf{w}_2, \rho_1, \rho_3)$. Compute the openings for C_1 and C_3 , $\pi_1 = \text{COM}(\phi_2, \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v}; \rho_1)$, $\pi_3 = \text{COM}(\Gamma_{\phi_2}(\mathbf{w}_2); \rho_3)$.
 - $Ch = 3$: Let $\phi_3 = \phi$, $\mathbf{w}_3 = \mathbf{r}_w$, set $\text{RSP} = (\phi_3, \mathbf{w}_3, \rho_1, \rho_2)$. Compute the openings for C_1 and C_2 , $\pi_1 = \text{COM}(\phi_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1)$, $\pi_2 = \text{COM}(\Gamma_{\phi_3}(\mathbf{w}_3); \rho_2)$.

Verification: Receiving RSP, the verifier proceeds as follows:

- $Ch = 1$: Check that $\mathbf{t}_w \in \text{VALID}$, $C_2 = \pi_2$, $C_3 = \pi_3$.
- $Ch = 2$: Check that $C_1 = \pi_1$, $C_3 = \pi_3$.
- $Ch = 3$: Check that $C_1 = \pi_1$, $C_2 = \pi_2$.

Figure 2.1: Stern-like ZKAoK for $\mathbf{R}_{\text{abstract}}$.

$(\text{RSP}_1, \text{RSP}_2, \text{RSP}_3)$ to all 3 possible values of the challenge Ch are given.

The detailed proofs refer to Appendix A.

We will conclude this chapter, we present some basic notation and definitions, then we present some hard problems and primitives in code based cryptography, including some code-based constructions that we will use in the thesis. In Chapter 3, 4, 5, 6 and 7, we will describe our contributions.

Chapter 3

Code-Based Statistically Hiding Commitments With Companion Zero-Knowledge Protocol

Commitment scheme was introduced by Blum in 1983 [21]. It is a basic building block in many cryptographic protocols, especially in zero-knowledge proofs [67, 107]. In this chapter, we present our code based statistically hiding and computationally binding commitment with companion zero knowledge argument of knowledge of a valid opening.

First, we give an overview of our technique in Section 3.1. In Section 3.2, we briefly review a previous work which is in code-based setting. The definition and security requirements for commitment scheme is given in Section 3.3 and in Section 3.4, we develop the AFS hash function to obtain a code-based computationally binding and statistically binding commitment scheme. Then, in Section 3.5, we build up our techniques for proving in zero-knowledge the correct encoding of binary strings into regular words. Relying on these techniques, we put forward in Section 3.6 a ZKAoK of a valid opening for the given commitment scheme. This building block will further be used in other advanced constructions, which will be presented later in the paper.

3.1 Overview of Our Main Techniques

Our commitment scheme is built based on the AFS hash function by extending its one input which is a preimage of a hash to two inputs. Let us first discuss our basic techniques for proving in zero-knowledge the knowledge of a preimage of a hash, computed via the AFS hash function $\mathcal{H}_{\text{afs}} : \{0, 1\}^k \rightarrow \{0, 1\}^n$. Let $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times 2^c \cdot k/c}$, for some constant c dividing k . Let $\text{RE} : \{0, 1\}^k \rightarrow \{0, 1\}^{2^c \cdot k/c}$ be an encoding function that maps $\mathbf{x}$ to $\text{RE}(\mathbf{x})$, defined as follows. First, write $\mathbf{x}$ block-wise as $\mathbf{x} = (\mathbf{x}_1 \| \dots \| \mathbf{x}_{k/c})$, where $\mathbf{x}_j = (x_{j,1}, \dots, x_{j,c})^\top$ for $j \in [k/c]$. Then compute $t_j = \sum_{i=1}^c 2^{c-i} \cdot x_{j,i}$. Denote by $\Delta_{2^c}(\mathbf{x}_j)$ the element in $\mathbb{B}_{2^c}^1$ whose sole 1 entry is at the t_j -th position for $j \in [0, 2^c - 1]$. Then $\text{RE}(\mathbf{x})$ is defined to be $(\Delta_{2^c}(\mathbf{x}_1) \| \dots \| \Delta_{2^c}(\mathbf{x}_{k/c}))$, and the hash output is set as $\mathbf{u} = \mathbf{B} \cdot \text{RE}(\mathbf{x})$. Given $(\mathbf{B}, \mathbf{u})$, to prove that we know $\mathbf{x}$ such that $\mathcal{H}_{\text{afs}}(\mathbf{x}) = \mathbf{u}$, we have to demonstrate that the encoding $\text{RE}(\cdot)$ is done correctly for $\mathbf{x}$. To this end, we introduce the following permuting technique.

For every vector $\mathbf{s} = (s_1, \dots, s_c) \in \{0, 1\}^c$, define the permutation $E_{\mathbf{s}}$ that transforms vector $\mathbf{x} = (x_{0,0,\dots,0}, \dots, x_{i_1,\dots,i_c}, \dots, x_{1,1,\dots,1}) \in \{0, 1\}^{2^c}$ into vector $E_{\mathbf{s}}(\mathbf{x}) = (x'_{0,0,\dots,0}, \dots, x'_{i_1,\dots,i_c}, \dots, x'_{1,1,\dots,1})$, where for each $(i_1, \dots, i_c) \in \{0, 1\}^c$, we have $x_{i_1,\dots,i_c} = x'_{i_1 \oplus s_1, \dots, i_c \oplus s_c}$.

Note that, for any $\mathbf{s}, \mathbf{v} \in \{0, 1\}^c$, we have:

$$\mathbf{x} = \Delta_{2^c}(\mathbf{v}) \iff E_{\mathbf{s}}(\mathbf{x}) = \Delta_{2^c}(\mathbf{v} \oplus \mathbf{s}). \quad (3.1)$$

For $\mathbf{t} = (\mathbf{t}_1 \| \dots \| \mathbf{t}_{k/c}) \in \{0, 1\}^k$ consisting of k/c blocks of length c , define the permutation $E'_{\mathbf{t}}$ that transforms vector $\mathbf{y} = (\mathbf{y}_1 \| \dots \| \mathbf{y}_{k/c}) \in \{0, 1\}^{2^c \cdot k/c}$ consisting of k/c blocks of length 2^c into vector of the following form $E'_{\mathbf{t}}(\mathbf{y}) = (E_{\mathbf{t}_1}(\mathbf{y}_1) \| \dots \| E_{\mathbf{t}_{k/c}}(\mathbf{y}_{k/c}))$. Note that, for any $\mathbf{t}, \mathbf{x} \in \{0, 1\}^k$, we have:

$$\mathbf{y} = \text{RE}(\mathbf{x}) \iff E'_{\mathbf{t}}(\mathbf{y}) = \text{RE}(\mathbf{x} \oplus \mathbf{t}). \quad (3.2)$$

In the framework of Stern’s protocol [122], the equivalence in (3.2) enables us to prove that $\mathbf{y}$ is the correct encoding of $\mathbf{x}$, as follows. The prover samples uniformly random $\mathbf{t}$ and demonstrates to the verifier that the right-hand side of (3.2) holds. The verifier is thus convinced that its left-hand side also holds, while learning no additional information about $\mathbf{x}$, thanks to the “one-time pad” $\mathbf{t}$. Moreover, this permuting technique allows us to keep control over the bits of $\mathbf{x}$, in order to prove that they satisfy other relations. To this end, it suffices to use the same “one-time pad” at other appearances of $\mathbf{x}$.

Our discussed above technique then readily extends to handle the case when there are two inputs to the hash function, i.e., $\mathcal{H}_{\text{afs}} : \{0, 1\}^\ell \times \{0, 1\}^k \rightarrow \{0, 1\}^n$. If we set k sufficiently large so that the leftover hash lemma [58] applies, then we obtain a statistically hiding commitment scheme that is supported by our ZK technique.

3.2 Previous Work

In code-based setting, Jain et al. [68] put forward a computationally hiding and perfectly binding construction that relies on the hardness of the LPN problem. We briefly review this scheme here. Their scheme commits to bit strings $\mathbf{m}$ and obtain a commitment $\mathbf{A}(\mathbf{s} \parallel \mathbf{m}) \oplus \mathbf{e}$ which is a LPN sample with a secret to be the concatenation of a uniformly random string $\mathbf{s}$ and the message $\mathbf{m}$, for a fixed public random matrix $\mathbf{A}$ and a random low weight vector $\mathbf{e}$. Its computationally hiding property is based on the exact learning parity with error problem (xLPN problem), which is a variant of the LPN problem and the binding property guaranteed by the large distance of the code generated by the random matrix $\mathbf{A}$. It also motivates us to build our statistically hiding commitment scheme from code-based assumptions.

3.3 Commitment Schemes

3.3.1 Definition and Security Requirements

Now we recall the standard definition and security requirements for commitment scheme, as put forward in [68].

Definition 3.3.1. A commitment scheme consists of a triple of polynomial-time algorithms (CSetup , Com , CVer).

$\text{CSetup}(1^\lambda)$: On input the security parameter 1^λ , this algorithm outputs the public parameter pp .

$\text{Com}(pp, m, r)$: On input public parameters pp , a message $m \in \mathcal{M}$ and a randomness r , outputs a commitment c and an opening value d .

$\text{CVer}(pp, m, d, c)$: Take public parameters pp , a message m , a commitment c and an opening value d , it outputs 1 to accept or 0 to reject.

We now recall the requirements for commitment scheme: correctness, *hiding property* and *binding property*.

Correctness requires that the verification algorithm CVer always output 1 if the inputs are computed honestly. For all $m \in \mathcal{M}$, the following holds:

$$\Pr[pp \leftarrow \text{CSetup}(1^\lambda); (c, d) \leftarrow \text{Com}(pp, m, r) : \text{CVer}(pp, m, d, c) = 1] = 1$$

Definition 3.3.2 (Binding property). A commitment scheme is **statistically** (respectively, computationally) binding if the probability for any computationally unbounded (respectively, PPT) forger prover come up with a commitment with two different valid opening is negligible. A commitment scheme satisfies binding if, for any (possibly un-

bounded) adversary $\mathcal{P}$, for two distinct $m, m' \in \mathcal{M}$,

$$\Pr \left[\begin{array}{l} pp \leftarrow \text{CSetup}(1^\lambda); \\ (c, m, d, m', d') \leftarrow \mathcal{P}(pp); m \neq m'; \\ (c, d) \leftarrow \text{Com}(pp, m, r); \\ (c, d') \leftarrow \text{Com}(pp, m', r') \end{array} : \text{CVer}(pp, m, d, c) = \text{CVer}(pp, m', d', c) = 1 \right] \leq \text{negl}(\lambda),$$

Definition 3.3.3 (Hiding property). A commitment scheme provides **statistically** (respectively, **computationally**) hiding if, for any computationally unbounded (respectively, PPT) adversary $\mathcal{V}$, for all $m, m' \in \mathcal{M}$ with $|m| = |m'|$,

$$\Pr \left[\begin{array}{l} pp \leftarrow \text{CSetup}(1^\lambda); b \leftarrow \{0, 1\} \\ (c, d) \leftarrow \text{Com}(pp, m_b, r) \end{array} : b \leftarrow \mathcal{V}(c) \right] < 1/2 + \text{negl}(\lambda),$$

In this thesis, we consider statistically hiding and computationally binding commitment scheme as a building block in our code-based zero-knowledge proof and argument, as mentioned in Section 2.3.3.

3.4 Description of Our Construction

Given security parameter λ , choose $n = \mathcal{O}(\lambda)$, $k \geq n + 2\lambda + \mathcal{O}(1)$. Let the message space be $\mathcal{M} = \{0, 1\}^L$, and let c be a constant dividing L and k . Let $m_0 = 2^c \cdot L/c$, $m_1 = 2^c \cdot k/c$ and $m = m_0 + m_1$. Our scheme works as follows.

- **KGen**: Sample $\mathbf{C}_0 \xleftarrow{\$} \mathbb{Z}_2^{n \times m_0}$, $\mathbf{C}_1 \xleftarrow{\$} \mathbb{Z}_2^{n \times m_1}$, and output commitment key $pk = \mathbf{C} = [\mathbf{C}_0 \mid \mathbf{C}_1] \in \mathbb{Z}_2^{n \times m}$.
- **Com**: On input a message $\mathbf{x} \in \{0, 1\}^L$ and commitment key pk , sample randomness $\mathbf{s} \xleftarrow{\$} \{0, 1\}^k$, compute $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) \in \mathbb{Z}_2^n$, and output commitment

$\mathbf{c}$ together with opening $\mathbf{s}$. Here, $\text{RE}(\cdot)$ is the regular encoding function from Section 2.3.1.

- **Open:** On input commitment key pk , a commitment $\mathbf{c} \in \mathbb{Z}_2^n$, a message $\mathbf{x} \in \{0, 1\}^L$ and an opening $\mathbf{s} \in \{0, 1\}^k$, output 1 if $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s})$, or 0 otherwise.

One can check that the proposed scheme is correct. Let us now prove the computationally binding and statistically hiding properties.

Lemma 3.4.1. The scheme is computationally binding, assuming the hardness of the $2\text{-RNSD}_{n, L+k, c}$ problem.

Proof. Suppose that the adversary outputs $\mathbf{c}, \mathbf{x}, \mathbf{x}', \mathbf{s}, \mathbf{s}'$ such that $\mathbf{x} \neq \mathbf{x}'$ and

$$\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}') \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}').$$

Let $\mathbf{z} = \text{RE}(\mathbf{x}) \oplus \text{RE}(\mathbf{x}') \neq \mathbf{0}$ and $\mathbf{y} = \text{RE}(\mathbf{s}) \oplus \text{RE}(\mathbf{s}')$. Then $\mathbf{C}_0 \cdot \mathbf{z} + \mathbf{C}_1 \cdot \mathbf{y} = \mathbf{0}$. Next, let $\mathbf{t} = (\mathbf{z} \parallel \mathbf{y})$ then we have $\mathbf{t} \neq \mathbf{0}$, $\mathbf{t} \in 2\text{-Regular}(L+k, c)$ and $\mathbf{C} \cdot \mathbf{t} = \mathbf{0}$. In other words, $\mathbf{t}$ is a solution to the $2\text{-RNSD}_{n, L+k, c}$ problem associated with uniformly random matrix $\mathbf{C}$. $\square$

The statistically hiding property of the scheme is based on the following leftover hash lemma.

Lemma 3.4.2 (Leftover hash lemma, adapted from [58]). Let D be a distribution over $\{0, 1\}^t$ with min-entropy k . For any $\epsilon > 0$ and $n \leq k - 2 \log(1/\epsilon) - \mathcal{O}(1)$, the statistical distance between the joint distribution of $(\mathbf{C}, \mathbf{C} \cdot \mathbf{t})$, where $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{n \times t}$ and $\mathbf{t} \in \{0, 1\}^t$ is drawn from distribution D , and the uniform distribution over $\mathbb{Z}_2^{n \times t} \times \mathbb{Z}_2^n$ is at most ϵ .

If $\mathbf{t}$ is uniformly random over $\{0, 1\}^k$, then the distribution of $\text{RE}(\mathbf{t})$ over $\{0, 1\}^{m_1}$ has min-entropy exactly k . Since $k \geq n + 2\lambda + \mathcal{O}(1)$, the distribution of $\mathbf{C}_1 \cdot \text{RE}(\mathbf{s})$ is at most $2^{-\lambda}$ -far from the uniform distribution over $\mathbb{Z}_2^n$. Then, for any $\mathbf{x} \in \{0, 1\}^L$,

the distribution of $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s})$ is statistically close to uniform over $\mathbb{Z}_2^n$. As a result, the scheme is statistically hiding.

Remark 3.4.3. As for the lattice-based commitment scheme from [71], we can extend the message space of our scheme to $\{0, 1\}^L$ for arbitrary $L = \text{poly}(\lambda)$ using the Merkle-Damgard technique together with the AFS hash function.

3.5 Extension and Permutating Techniques

We next recall some extension and permutating techniques for proving in Stern's framework the knowledge of a single secret bit x and a product of 2 secret bits $x_1 \cdot x_2$, presented in [80, 81]. What's more, in our ZKAoK of a valid opening for the given commitment scheme, which will be presented in Section 3.6, as well as in all subsequent argument systems of this paper, we need a special mechanism allowing to the correctness of the (non-linear) encoding process from $\mathbf{v} \in \{0, 1\}^m$, for some $m \in \mathbb{Z}^+$, to regular word $\mathbf{y} = \text{RE}(\mathbf{v}) \in \text{Regular}(m, c)$, where c divides m . To this end, we introduce the following notations and techniques.

Let $\oplus$ denote the bit-wise addition operation modulo 2. Let $c \in \mathbb{Z}^+$. For any bit $b \in \{0, 1\}$, denote by $\bar{b}$ the bit $b = b \oplus 1$. Note that, for any $b, d \in \{0, 1\}$, we have $\overline{b \oplus d} = b \oplus d \oplus 1 = \bar{b} \oplus d$. For any bit b , let $\text{enc}(b) = (\bar{b}, b) \in \{0, 1\}^2$.

- For any bit $d \in \{0, 1\}$, define F_d as the permutation that transforms integer vector $\mathbf{v} = (v_0, v_1) \in \mathbb{Z}^2$ into vector $F_d(\mathbf{v}) = (v_d, v_{\bar{d}})$. Namely, if $d = 0$ then F_d keeps the arrangement the coordinates of $\mathbf{v}$; or swaps them if $d = 1$. Note that:

$$\mathbf{v} = \text{enc}(b) \iff F_d(\mathbf{v}) = \text{enc}(b \oplus d). \quad (3.3)$$

- For any two bits b_1, b_2 , define the vector

$$\text{ext}(b_1, b_2) = (\bar{b}_1 \cdot \bar{b}_2, \bar{b}_1 \cdot b_2, b_1 \cdot \bar{b}_2, b_1 \cdot b_2) \in \{0, 1\}^4,$$

that is an extension of the bit product $b_1 \cdot b_2$.

The authors of [81] showed that the equivalence (3.3) is helpful for proving knowledge of a secret bit x that may appear in several correlated linear equations. To this end, one extends x to $\text{enc}(x) \in \{0, 1\}^2$, and permutes the latter using F_c , where c is a uniformly random bit. Then one demonstrates to the verifier that the permuted vector is $\text{enc}(x \oplus c)$, which implies that the original vector $\text{enc}(x)$ is well-formed - which in turn implies knowledge of some bit x . Meanwhile, the bit c acts as a “one-time pad” that completely hides x .

In [80], Libert et al. proposed a method for proving the well-formedness of the product of two secret bits x_1, x_2 , based on the following technique.

- For any two bits $d_1, d_2 \in \{0, 1\}$, define T_{d_1, d_2} as the permutation that transforms vector $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$ into vector

$$T_{d_1, d_2}(\mathbf{v}) = (v_{d_1, d_2}, v_{d_1, \bar{d}_2}, v_{\bar{d}_1, d_2}, v_{\bar{d}_1, \bar{d}_2}) \in \mathbb{Z}^4.$$

- For any bits b_1, b_2, d_1, d_2 and any vector $\mathbf{v} = (v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}) \in \mathbb{Z}^4$,

$$\mathbf{v} = \text{ext}(b_1, b_2) \iff T_{d_1, d_2}(\mathbf{v}) = \text{ext}(b_1 \oplus d_1, b_2 \oplus d_2). \quad (3.4)$$

As a result, to prove that a bit has the form $x_1 \cdot x_2$, one can extend it to vector $\text{ext}(x_1, x_2)$, then permute the latter using T_{c_1, c_2} , where c_1, c_2 are uniformly random bits. One then demonstrates to the verifier that the permuted vector is $\text{ext}(x_1 \oplus c_1, x_2 \oplus c_2)$. This convinces the verifier that the original vector, i.e., $\text{ext}(x_1, x_2)$, is well-formed, while learning no additional information about x_1 and x_2 , thanks to the randomness of c_1 and c_2 .

- For every $\mathbf{s} = (s_1, \dots, s_d) \in \{0, 1\}^d$, define the permutation $E_{\mathbf{s}}$ that transforms vector $\mathbf{x} = (x_{0,0,\dots,0}, \dots, x_{i_1,\dots,i_d}, \dots, x_{1,1,\dots,1}) \in \{0, 1\}^{2^c}$ into vector $E_{\mathbf{s}}(\mathbf{x}) =$

$(x'_{0,0,\dots,0}, \dots, x'_{1,1,\dots,1})$, where for each $(i_1, \dots, i_c) \in \{0, 1\}^d$, we have $x_{i_1, \dots, i_c} = x'_{i_1 \oplus s_1, \dots, i_c \oplus s_c}$.

- For any $\mathbf{s}, \mathbf{v} \in \{0, 1\}^d$, we have:

$$\mathbf{x} = \Delta_{2^d}(\mathbf{v}) \iff E_{\mathbf{s}}(\mathbf{x}) = \Delta_{2^c}(\mathbf{v} \oplus \mathbf{s}). \quad (3.5)$$

- For $\mathbf{t} = (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_{n/c}) \in \{0, 1\}^n$ consisting of n/c blocks of length c , define the permutation $E'_{\mathbf{t}}$ that transforms vector $\mathbf{y} = (\mathbf{y}_1 \parallel \dots \parallel \mathbf{y}_{n/c}) \in \{0, 1\}^{2^c \cdot n/c}$ consisting of n/c blocks of length 2^c into vector of the following form $E'_{\mathbf{t}}(\mathbf{y}) = (E_{\mathbf{t}_1}(\mathbf{y}_1) \parallel \dots \parallel E_{\mathbf{t}_{n/c}}(\mathbf{y}_{n/c}))$. Note that, for any $\mathbf{t}, \mathbf{v} \in \{0, 1\}^n$, we have:

$$\mathbf{y} = \text{RE}(\mathbf{v}) \iff E'_{\mathbf{t}}(\mathbf{y}) = \text{RE}(\mathbf{v} \oplus \mathbf{t}). \quad (3.6)$$

The equivalence in (3.6) enables us to prove that $\mathbf{y}$ is the correct encoding of $\mathbf{v}$, as follows. The prover samples uniformly random $\mathbf{t}$ and demonstrates to the verifier that the right-hand side of (3.6) holds. The verifier is thus convinced that its left-hand side also holds, while learning no additional information about $\mathbf{v}$, thanks to the “one-time pad” $\mathbf{t}$.

3.6 ZKAoK of a Valid Opening

We now describe a ZKAoK of a valid opening for the commitment scheme from Section 3.4. Specifically, we consider the relation R_{com} , defined as:

$$R_{\text{com}} = \left\{ \left((\mathbf{C} = [\mathbf{C}_0 \parallel \mathbf{C}_1], \mathbf{c}), \mathbf{x}, \mathbf{s} \right) : \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) + \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{c} \right\}.$$

The protocol is realized based on the permuting technique of Section 3.5. Let $\mathbf{z} = (\mathbf{x} \parallel \mathbf{s}) \in \{0, 1\}^{L+k}$ and $\mathbf{w}_{\text{com}} = \text{RE}(\mathbf{z}) \in \text{Regular}(L+k, c) \subset \{0, 1\}^{2^c \cdot (L+k)/c}$. Then the

equation $\mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) + \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{c}$ can be written as $\mathbf{C} \cdot \mathbf{w}_{\text{com}} = \mathbf{c}$.

Next, define the sets $\text{VALID}_{\text{com}} = \text{Regular}(L + k, c)$ and $\mathcal{S} = \{0, 1\}^{L+k}$. For $\mathbf{t} = (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_{(L+k)/c}) \in \mathcal{S}$ consisting of $(L + k)/c$ blocks of length c , it follows from (3.6) that we have

$$\mathbf{w}_{\text{com}} = \text{RE}(\mathbf{z}) \iff E'_{\mathbf{t}}(\mathbf{w}_{\text{com}}) = \text{RE}(\mathbf{z} \oplus \mathbf{t}). \quad (3.7)$$

Moreover, if $\mathbf{t}$ is chosen uniformly at random in $\mathcal{S}$, then $E'_{\mathbf{t}}(\mathbf{w}_{\text{com}})$ is uniformly random in $\text{VALID}_{\text{com}}$. In other words, the conditions of (2.2) hold, and relation R_{com} can be reduced to an instance of R_{abstract} in Section 2.3.4. As a result, we can run the interactive protocol in Figure 2.1 with public input $(\mathbf{C}, \mathbf{c})$ and prover's witness $\mathbf{w}_{\text{com}}$, and obtain a ZKAoK for R_{com} .

Chapter 4

Code-Based Accumulators and Logarithmic-size Zero-Knowledge Arguments for Set Membership

4.1 Introduction

In this chapter, we provide a code-based accumulator scheme supported by zero-knowledge argument of an accumulated value. Such an argument system is essentially an argument of set membership, in which the prover convinces the verifier in zero-knowledge that his data item (e.g., his public key) belongs to a given set, and is a highly desirable building block in various privacy-preserving applications. Our accumulator relies on a Merkle hash tree that is built from a suitable variant of the AFS hash function. To design a supporting zero-knowledge protocol for the hash tree, we first use the techniques for handling regular words from Section 3.5 to prove knowledge of hash preimages in the path from a leaf to the tree root, and then adapt Libert et al.'s method [81] to hide the bits determining steps in the path. As the tree depth is logarithmic in its size, we obtain an argument system for set membership with size logarithmic in the cardinality of the set.

We first give an overview of our techniques in Section 4.2, then in Section 4.3, we recall the definition and security requirement of cryptographic accumulators. In Section 4.4, we modify the AFS hash function to support hashing with two inputs, upon which we build our Merkle-tree accumulator in Section 4.5. In Section 4.6, we describe our zero-knowledge argument system.

4.2 Overview of Our Techniques

Our accumulator proceeds by computing a Merkle tree using the AFS hash function based on the $2\text{-RNSD}_{n,k,c}$ problem. We modify our statistically hiding commitment scheme by setting two equally length inputs to get a function that compresses two child-inputs of n bits to one parent output, which is then can be used to build a Merkle hash tree. Then, by combining the ZK techniques from [81] and our techniques for proving correctness of re-encoding at each step in a tree path, we obtain a method for demonstrating knowledge of a Merkle tree hash chain leads to get a Merkle-tree-accumulator supported by logarithmic-size zero-knowledge arguments.

4.3 Cryptographic Accumulators

We recall the definition of accumulators in the following.

Definition 4.3.1. An accumulator scheme is a tuple of polynomial-time algorithms (Setup, Accu, WitGen, Verify):

- **Setup**(1^λ) Given a security parameter 1^λ , outputs the public parameter pp .
- **Accu** $_{pp}(R)$ Take as input a set R with n data values as $R = \{\mathbf{d}_0, \dots, \mathbf{d}_{N-1}\}$, outputs an accumulator value $\mathbf{u}$.
- **WitGen** $_{pp}(R, \mathbf{d})$ Take as input the set R and a value $\mathbf{d}$, outputs a witness w such that $\mathbf{d}$ is accumulated in $\text{TAcc}(R)$, otherwise returns $\perp$ if $\mathbf{d} \notin R$.

- $\text{Verify}_{pp}(\mathbf{u}, (\mathbf{d}, w))$ This deterministic algorithm takes as inputs the accumulator value $\mathbf{u}$ and $(\mathbf{d}, w)$, outputs 1 if $(\mathbf{d}, w)$ is valid for the accumulator $\mathbf{u}$, otherwise returns 0 if invalid.

Correctness. The correctness requires that the correctly computed accumulator value has valid verifying witnesses. In other words, for all $pp \leftarrow \text{Setup}(n)$, the following holds:

$$\text{Verify}_{pp}(\text{Accu}_{pp}(R), \mathbf{d}, \text{WitGen}_{pp}(R, \mathbf{d})) = 1 \text{ for } \mathbf{d} \in R.$$

An accumulator is secure, as defined in [10, 33], if it is infeasible to output a valid witness for a value $\mathbf{d}^*$ that is not chosen from the data value set.

Definition 4.3.2. An accumulator scheme is secure if for all PPT adversaries $\mathcal{A}$:

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

4.4 Hashing with Two Inputs

We aim to build a Merkle-tree accumulator based on the AFS family of hash functions. Since in Merkle trees, every internal node is the hash of its two children nodes, we slightly modify the AFS hash functions so that the function takes two inputs instead of just one.

Definition 4.4.1. Let $q = 2 \cdot 2^c \cdot n / c$. The function family $\mathcal{H}$ mapping $\{0, 1\}^n \times \{0, 1\}^n$ to $\{0, 1\}^n$ is defined as $\mathcal{H} = \{h_{\mathbf{B}} \mid \mathbf{B} \in \mathbb{Z}_2^{n \times q}\}$, where for $\mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1]$ with $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_2^{n \times q/2}$, and for any $(\mathbf{u}_0, \mathbf{u}_1) \in \{0, 1\}^n \times \{0, 1\}^n$, we have:

$$h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{B}_0 \cdot \text{RE}(\mathbf{u}_0) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{u}_1) \in \{0, 1\}^n.$$

Lemma 4.4.2. The function family $\mathcal{H}$, defined in Definition 4.4.1 is collision-resistant, assuming the hardness of the $2\text{-RNSD}_{n,2n,c}$ problem.

Proof. Given $\mathbf{B} = [\mathbf{B}_0|\mathbf{B}_1] \xleftarrow{\$} \mathbb{Z}_2^{n \times q}$, if one can find two distinct pairs $(\mathbf{u}_0, \mathbf{u}_1) \in (\{0, 1\}^n)^2$ and $(\mathbf{v}_0, \mathbf{v}_1) \in (\{0, 1\}^n)^2$ such that $h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) = h_{\mathbf{B}}(\mathbf{v}_0, \mathbf{v}_1)$, then one can obtain a non-zero vector $\mathbf{z} = \begin{pmatrix} \text{RE}(\mathbf{u}_0) \oplus \text{RE}(\mathbf{v}_0) \\ \text{RE}(\mathbf{u}_1) \oplus \text{RE}(\mathbf{v}_1) \end{pmatrix} \in 2\text{-Regular}(2n, c)$ such that

$$\mathbf{B} \cdot \mathbf{z} = h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) \oplus h_{\mathbf{B}}(\mathbf{v}_0, \mathbf{v}_1) = \mathbf{0}.$$

In other words, $\mathbf{z}$ is a valid solution to the $2\text{-RNSD}_{n,2n,c}$ problem associated with matrix $\mathbf{B}$. □

4.5 Code-Based Merkle-tree Accumulators

We now describe our Merkle-tree accumulator based on the code-based hash functions $\mathcal{H}$ in Definition 4.4.1. The construction is adapted from the blueprint by Libert et al. [81].

Setup(λ). Given $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ and $q = 2 \cdot 2^c \cdot n/c$. Sample $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times q}$, and output the public parameter $pp = \mathbf{B}$.

Accu_B($R = \{\mathbf{d}_0, \dots, \mathbf{d}_{N-1}\} \subseteq (\{0, 1\}^n)^N$). Let the binary representation of j be a ℓ length string $(j_1, \dots, j_\ell) \in \{0, 1\}^\ell$, re-write $\mathbf{d}_j$ as $\mathbf{u}_{j_1, \dots, j_\ell}$. Build a binary tree with $N = 2^\ell$ leaves $\mathbf{u}_{0,0,\dots,0}, \dots, \mathbf{u}_{1,1,\dots,1}$ in the following way:

1. At depth $i \in [1, \ell - 1]$, for the nodes $\mathbf{u}_{a_1, \dots, a_i, 0} \in \{0, 1\}^n$ and $\mathbf{u}_{a_1, \dots, a_i, 1} \in \{0, 1\}^n$, compute $h_{\mathbf{B}}(\mathbf{u}_{a_1, \dots, a_i, 0}, \mathbf{u}_{a_1, \dots, a_i, 1})$ and define it to be $\mathbf{u}_{a_1, \dots, a_i}$ for all $(a_1, \dots, a_i) \in \{0, 1\}^i$.
2. At depth 0, for the nodes $\mathbf{u}_0 \in \{0, 1\}^n$ and $\mathbf{u}_1 \in \{0, 1\}^n$, compute $h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1)$ and define it to be the root value $\mathbf{u}$.

Output the accumulated value $\mathbf{u}$.

$\text{WitGen}_{\mathbf{B}}(R, \mathbf{d})$. If $\mathbf{d} \notin R$, the algorithm outputs $\perp$. Otherwise, it outputs the witness w for $\mathbf{d}$ as follows.

1. Set $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N - 1]$. Re-write $\mathbf{d}_j$ as $\mathbf{u}_{j_1, \dots, j_\ell}$ using the binary representation of the index j .
2. Consider the path from $\mathbf{u}_{j_1, \dots, j_\ell}$ to the root $\mathbf{u}$, the witness w then consists of the binary representation $(j_1, \dots, j_\ell)$ for j as well as all the sibling nodes of the path. Specifically,

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

$\text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w)$. Let w be of the following form:

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

This algorithm then computes $\mathbf{v}_\ell, \dots, \mathbf{v}_0$. Let $\mathbf{v}_\ell = \mathbf{d}$ and

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

Output 1 if $\mathbf{v}_0 = \mathbf{u}$ or 0 otherwise.

Take a Merkle Tree with 2^3 leaves in Figure 4.1 as an example. A Merkle tree with $2^3 = 8$ leaves $\mathbf{d}_0, \dots, \mathbf{d}_7$. It recursively accumulates the leaf values into value $\mathbf{u}$. The witness of $\mathbf{d}_5$ contains the binary representation of 5, which is (101), and all the gray nodes.

The correctness of the Merkle-tree accumulator scheme follows immediately from the construction. In terms of the security, it is guaranteed by the collision-resistance

Figure 4.1: A Merkle tree with $2^3 = 8$ leaves

hash function family $\mathcal{H}$, which means that if the adversary can break the security of the accumulator, then it can find a solution to the $2\text{-RNSD}_{n,2n,c}$ problem.

Theorem 4.5.1. Assume that the $2\text{-RNSD}_{n,2n,c}$ problem is hard, then the given accumulator scheme is secure.

Proof. Assume that there exists a probabilistic polynomial time adversary $\mathcal{B}$ who breaks the security of accumulator scheme with non-negligible probability. By the security definition, $\mathcal{B}$ first receives a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times q}$ that output by $\text{Setup}(1^\lambda)$, and then outputs $(R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1}), \mathbf{d}^*, w^*)$ such that $\mathbf{d}^* \notin R$ and $\text{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*) = 1$, where $\mathbf{u}^* = \text{Accu}_{\mathbf{B}}(R)$.

Let $w^* = ((j_1^*, \dots, j_\ell^*), (\mathbf{w}_\ell^*, \dots, \mathbf{w}_1^*))$ and $(j_1^*, \dots, j_\ell^*)$ be the binary representation of some index $j^* \in [0, N-1]$, then we can find a path that starts from $\mathbf{d}_{j^*}$ to the accumulated value $\mathbf{u}$: $\mathbf{u}_{j_1^*, \dots, j_\ell^*} = \mathbf{d}_{j^*}, \mathbf{u}_{j_1^*, \dots, j_{\ell-1}^*}, \dots, \mathbf{u}_{j_1^*}, \mathbf{u}^*$.

On the other hand, the algorithm $\text{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*)$ can compute another path: $\mathbf{k}_\ell^* = \mathbf{d}^*, \mathbf{k}_{\ell-1}^*, \dots, \mathbf{k}_1^*, \mathbf{k}_0^* = \mathbf{u}^*$.

Since $\mathbf{d}^* \notin R$, then $\mathbf{d}^* \neq \mathbf{d}_{j^*}$. Comparing two paths $\mathbf{d}_{j^*}, \mathbf{u}_{j_1^*, \dots, j_{\ell-1}^*}, \dots, \mathbf{u}_{j_1^*}, \mathbf{u}^*$ and path $\mathbf{d}^*, \mathbf{k}_{\ell-1}^*, \dots, \mathbf{k}_1^*, \mathbf{u}^*$, we can find the smallest integer $i \in [\ell]$, such that $\mathbf{k}_i^* \neq \mathbf{u}_{j_1^*, \dots, j_i^*}$. So we obtain two distinct solutions to form a collision solution to the hash function $h_{\mathbf{B}}$ at the parent node of $\mathbf{u}_{j_1^*, \dots, j_i^*}$. $\square$

4.6 Logarithmic-size Arguments of Set Membership

In this section, we describe a statistical zero-knowledge argument that allows prover $\mathcal{P}$ to convince verifier $\mathcal{V}$ in zero-knowledge that $\mathcal{P}$ knows a value that was correctly accumulated into the root of the above code-based Merkle tree. Our protocol directly implies a logarithmic-size argument of set membership, with respect to the set of all data items accumulated into the tree root.

Given a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times q}$ and the accumulated value $\mathbf{u} \in \{0, 1\}^n$ as input, the goal of $\mathcal{P}$ is to convince $\mathcal{V}$ that it possesses a value $\mathbf{d}$ and a valid witness w . We define the associated relation R_{acc} as follows:

$$R_{\text{acc}} = \left\{ \left((\mathbf{B}, \mathbf{u}) \in \mathbb{Z}_2^{n \times q} \times \{0, 1\}^n; \mathbf{d} \in \{0, 1\}^n, w \in \{0, 1\}^\ell \times (\{0, 1\}^n)^\ell \right) : \right. \\ \left. \text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \right\}.$$

Before constructing a ZKAoK for the above relation, we first present several additional permuting techniques, which are developed from [81] and from our permuting technique presented Section 3.5.

- For vector $\mathbf{b} = (b_1, \dots, b_n) \in \{0, 1\}^n$, where $n \in \mathbb{Z}^+$, denote by $\text{Encode}(\mathbf{b})$ the vector $(\bar{b}_1, b_1, \dots, \bar{b}_n, b_n) \in \{0, 1\}^{2n}$.
- Let $\mathbf{I}_n^* \in \mathbb{Z}_2^{n \times 2n}$ be an extension of the identity matrix $\mathbf{I}_n$, obtained by inserting a zero-column $\mathbf{0}^n$ right before each of the columns of $\mathbf{I}_n$. Note that if $\mathbf{b} \in \{0, 1\}^n$, then $\mathbf{b} = \mathbf{I}_n^* \cdot \text{Encode}(\mathbf{b})$.
- For $\mathbf{t} = (t_1, \dots, t_n)^\top \in \{0, 1\}^n$, define the permutation $F'_{\mathbf{t}}$ that transforms vector $\mathbf{w} = (w_{1,0}, w_{1,1}, \dots, w_{n,0}, w_{n,1})^\top \in \{0, 1\}^{2n}$ into:

$$F'_{\mathbf{t}}(\mathbf{w}) = (w_{1,t_1}, w_{1,\bar{t}_1}, \dots, w_{n,t_n}, w_{n,\bar{t}_n})^\top.$$

Note that, for any $\mathbf{t}, \mathbf{v} \in \{0, 1\}^n$, we have:

$$\mathbf{w} = \text{Encode}(\mathbf{b}) \iff F'_{\mathbf{t}}(\mathbf{w}) = \text{Encode}(\mathbf{b} \oplus \mathbf{t}). \quad (4.1)$$

- For $b \in \{0, 1\}$ and $\mathbf{v} \in \{0, 1\}^{m/2}$, we denote $\text{Ext}(b, \mathbf{v})$ as $\begin{pmatrix} \bar{b} \cdot \mathbf{v} \\ b \cdot \mathbf{v} \end{pmatrix}$.
- For $e \in \{0, 1\}$, for $\mathbf{t} \in \{0, 1\}^n$, define the permutation $\Psi_{e, \mathbf{t}}$ that works on $\mathbf{z} = \begin{pmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \end{pmatrix} \in \{0, 1\}^m$ as follows, where $\mathbf{z}_0, \mathbf{z}_1 \in \{0, 1\}^{m/2}$. It transforms $\mathbf{z}$ to $\Psi_{e, \mathbf{t}}(\mathbf{z}) = \begin{pmatrix} E'_{\mathbf{t}}(\mathbf{z}_e) \\ E'_{\mathbf{t}}(\mathbf{z}_{\bar{e}}) \end{pmatrix}$. Precisely, it rearranges the blocks of $\mathbf{z}$ according to e and permutes each block using $E'_{\mathbf{t}}$.
- For any $b, e \in \{0, 1\}$ and $\mathbf{v}, \mathbf{w}, \mathbf{t} \in \{0, 1\}^n$, it follows from (3.6) that the following equivalences hold:

$$\begin{cases} \mathbf{z} = \text{Ext}(b, \text{RE}(\mathbf{v})) \iff \Psi_{e, \mathbf{t}}(\mathbf{z}) = \text{Ext}(b \oplus e, \text{RE}(\mathbf{v} \oplus \mathbf{t})) \\ \mathbf{y} = \text{Ext}(\bar{b}, \text{RE}(\mathbf{w})) \iff \Psi_{\bar{e}, \mathbf{t}}(\mathbf{y}) = \text{Ext}(\bar{b} \oplus \bar{e}, \text{RE}(\mathbf{w} \oplus \mathbf{t})). \end{cases} \quad (4.2)$$

Now let us look the the equations associated with the relation R_{acc} . This algorithm **Verify** computes the path $\mathbf{v}_\ell = \mathbf{d}, \mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1, \mathbf{v}_0 = \mathbf{u}$, where $\mathbf{v}_i$ for $i \in \{\ell-1, \dots, 1, 0\}$ is computed as follows:

$$\mathbf{v}_i = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), & \text{if } j_{i+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), & \text{if } j_{i+1} = 1. \end{cases} \quad (4.3)$$

Rewrite equation (4.3) into the following equivalent form.

$$\begin{aligned}
\mathbf{v}_i &= \bar{j}_{i+1} \cdot h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}) \oplus j_{i+1} \cdot h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}) \\
&= \bar{j}_{i+1} \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{v}_{i+1}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{w}_{i+1})) \oplus j_{i+1} \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{w}_{i+1}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{v}_{i+1})) \\
&= \mathbf{B} \cdot \begin{pmatrix} \bar{j}_{i+1} \cdot \text{RE}(\mathbf{v}_{i+1}) \\ j_{i+1} \cdot \text{RE}(\mathbf{v}_{i+1}) \end{pmatrix} \oplus \mathbf{B} \cdot \begin{pmatrix} j_{i+1} \cdot \text{RE}(\mathbf{w}_{i+1}) \\ \bar{j}_{i+1} \cdot \text{RE}(\mathbf{w}_{i+1}) \end{pmatrix} \\
&= \mathbf{B} \cdot \text{Ext}(j_{i+1}, \text{RE}(\mathbf{v}_{i+1})) \oplus \mathbf{B} \cdot \text{Ext}(\bar{j}_{i+1}, \text{RE}(\mathbf{w}_{i+1}))
\end{aligned}$$

Therefore, it is necessary and sufficient to construct an argument system in which $\mathcal{P}$ convinces $\mathcal{V}$ that $\mathcal{P}$ knows $j_1, \dots, j_\ell \in \{0, 1\}^\ell$ and $\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{w}_1, \dots, \mathbf{w}_\ell \in \{0, 1\}^n$ satisfying

$$\begin{cases} \mathbf{B} \cdot \text{Ext}(j_1, \text{RE}(\mathbf{v}_1)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{j}_1, \text{RE}(\mathbf{w}_1)) = \mathbf{u}; \\ \mathbf{B} \cdot \text{Ext}(j_2, \text{RE}(\mathbf{v}_2)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{j}_2, \text{RE}(\mathbf{w}_2)) \oplus \mathbf{v}_1 = \mathbf{0}. \\ \dots\dots\dots \\ \mathbf{B} \cdot \text{Ext}(j_\ell, \text{RE}(\mathbf{v}_\ell)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{j}_\ell, \text{RE}(\mathbf{w}_\ell)) \oplus \mathbf{v}_{\ell-1} = \mathbf{0}. \end{cases} \quad (4.4)$$

Next, we apply the function `Encode` defined above to vectors $\mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1$. Let $\mathbf{x}_i = \text{Encode}(\mathbf{v}_i) \in \{0, 1\}^{2n}$ for $i \in [\ell - 1]$. Then we have $\mathbf{v}_i = \mathbf{I}_n^* \cdot \mathbf{x}_i$ for $i \in [\ell - 1]$. For ease of notation, for $i \in [\ell]$, denote

$$\begin{cases} \mathbf{y}_i = \text{Ext}(j_i, \text{RE}(\mathbf{v}_i)) \in \{0, 1\}^m \\ \mathbf{z}_i = \text{Ext}(\bar{j}_i, \text{RE}(\mathbf{w}_i)) \in \{0, 1\}^m \end{cases} \quad (4.5)$$

Therefore, the equations in (4.4) is equivalent to the following.

$$\left\{ \begin{array}{l} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u}; \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{x}_1 = \mathbf{0}. \\ \dots\dots\dots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{x}_{\ell-1} = \mathbf{0}. \end{array} \right. \quad (4.6)$$

Now through some basic algebra, we can transform the equations in (5.3) into an unifying equation of the form $\mathbf{M}_A \cdot \mathbf{w}_A = \mathbf{v}_A$, where $\mathbf{M}_A \in \mathbb{Z}_2^{\ell n \times L}$, $\mathbf{v}_A \in \mathbb{Z}_2^{\ell n}$ are public and $\mathbf{w}_A \in \{0, 1\}^L$ is secret with $L = 2\ell m + 2(\ell - 1)n$ and

$$\mathbf{w}_A = \left(\mathbf{y}_1 \parallel \dots \parallel \mathbf{y}_\ell \parallel \mathbf{z}_1 \parallel \dots \parallel \mathbf{z}_\ell \parallel \mathbf{x}_1 \parallel \dots \parallel \mathbf{x}_{\ell-1} \right) \quad (4.7)$$

Now we specify the set VALID_A that contains our secret vector $\mathbf{w}_A$, the set $\mathcal{S}_A$ and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_A\}$ such that the conditions in (2.2) hold. Let VALID_A contains all vectors $\mathbf{w}'_A = \left(\mathbf{y}'_1 \parallel \dots \parallel \mathbf{y}'_\ell \parallel \mathbf{z}'_1 \parallel \dots \parallel \mathbf{z}'_\ell \parallel \mathbf{x}'_1 \parallel \dots \parallel \mathbf{x}'_{\ell-1} \right) \in \{0, 1\}^L$ satisfying the following conditions:

- For $i \in [\ell]$, there exists $\mathbf{v}'_i, \mathbf{w}'_i \in \{0, 1\}^n$, $j'_i \in \{0, 1\}$ such that

$$\mathbf{y}'_i = \text{Ext}(j'_i, \text{RE}(\mathbf{v}'_i)) \in \{0, 1\}^m, \quad \text{and} \quad \mathbf{z}'_i = \text{Ext}(\bar{j}'_i, \text{RE}(\mathbf{w}'_i)) \in \{0, 1\}^m.$$

- For $i \in [\ell - 1]$, $\mathbf{x}'_i = \text{Encode}(\mathbf{v}'_i) \in \{0, 1\}^{2n}$.

Let $\mathcal{S}_A = \left(\{0, 1\}^n \right)^\ell \times \left(\{0, 1\}^n \right)^\ell \times \{0, 1\}^\ell$.

Then, for each $\phi = \left(\mathbf{b}_1 \parallel \dots \parallel \mathbf{b}_\ell \parallel \mathbf{e}_1 \parallel \dots \parallel \mathbf{e}_\ell \parallel g_1 \parallel \dots \parallel g_\ell \right)$, define the permutation Γ_ϕ that transforms $\mathbf{w}_A^* = \left(\mathbf{y}_1^* \parallel \dots \parallel \mathbf{y}_\ell^* \parallel \mathbf{z}_1^* \parallel \dots \parallel \mathbf{z}_\ell^* \parallel \mathbf{x}_1^* \parallel \dots \parallel \mathbf{x}_{\ell-1}^* \right) \in \{0, 1\}^L$ with

$\mathbf{y}_i^*, \mathbf{z}_i^* \in \{0, 1\}^m$ for $i \in [\ell]$ and $\mathbf{x}_i^* \in \{0, 1\}^{2n}$ for $i \in [\ell - 1]$ to

$$\Gamma_\phi(\mathbf{w}_A^*) = \left(\Psi_{g_1, \mathbf{b}_1}(\mathbf{y}_1^*) \parallel \dots \parallel \Psi_{g_\ell, \mathbf{b}_\ell}(\mathbf{y}_\ell^*) \parallel \Psi_{\bar{g}_1, \mathbf{e}_1}(\mathbf{z}_1^*) \parallel \dots \parallel \Psi_{\bar{g}_\ell, \mathbf{e}_\ell}(\mathbf{z}_\ell^*) \parallel \right. \\ \left. F'_{\mathbf{b}_1}(\mathbf{x}_1^*) \parallel \dots \parallel F'_{\mathbf{b}_{\ell-1}}(\mathbf{x}_{\ell-1}^*) \right)$$

Based on the equivalences observed in (4.1) and (4.2), it can be checked that the conditions in (2.2) are satisfied. We thus have reduced the considered relation into an instance of R_{abstract} .

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

- The public input consists of matrix $\mathbf{M}_A$ and vector $\mathbf{v}_A$, which are constructed from the original public input, as discussed above.
- The prover's witness consists of vector $\mathbf{w}_A \in \text{VALID}_A$, which is built from the initial secret input, as described above.

The prover and the verifier then interact as in Figure 2.1. The protocol utilizes the statistically hiding and computationally binding string commitment scheme from Section 3.4 to obtain the desired statistical ZKAoK. The protocol has communication cost $\mathcal{O}(L) = \ell \cdot \mathcal{O}(m + n) = \mathcal{O}(\log N)$ bits.

Chapter 5

Code-Based Logarithmic-size Ring Signatures

In this chapter, we present a code-based ring signature scheme [116] with signature size logarithmic in the cardinality of the ring. The construction employs our accumulator which is based on Merkle hash tree and its associated ZKAoK, given in Section 4.5, as the building blocks.

First, we present an overview of our technique in Section 5.1, then we recall the definitions of ring signatures in Section 5.2. Our code-based instantiation is then described in Section 5.3, while its supporting zero-knowledge argument system is presented in Section 5.4. Finally, Section 5.5 provides the analysis of the scheme.

5.1 Overview of our Techniques

The starting point of our ring signature construction is based on the code based accumulator described in Section 4.5. The design is quite simple by adding one more level of secret under every leaf in the Merkle tree, so that each leaf corresponds to a user's public key, and define the signing process as the process of proving knowledge of an extended path from beneath a leaf up to the root of the tree. Each user's secret

key is a random vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \xleftarrow{\$} \{0, 1\}^{2n}$, and the corresponding public key is the hash value of $\mathbf{x}$. In order to sign the message, the user consider an accumulated value $\mathbf{u} \in \{0, 1\}^n$ of all user's public keys $R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$, which dynamically form a ring R in the ring signature. Then the user generates a Stern-type argument to show that the ownership of his secret key which is corresponding to a public key in the ring set. Our ring signatures feature logarithmic-size signatures, thanks to the Merkle tree structure.

5.2 Definitions and Security Requirements of Ring Signatures

Now we recall the standard definition and security requirements for ring signatures, as put forward in [16, 63].

Definition 5.2.1. A ring signature scheme consists of a tuple of polynomial-time algorithms (RSetup, RKgen, RSign, RVerify).

RSetup(1^λ): On input the security parameter 1^λ , this algorithm outputs the public parameter pp , which are available to all users.

RKgen(pp): On input public parameter, it generates a pair of public key and the corresponding secret signing key (pk, sk) .

RSign $_{pp}(sk, M, R)$: Take public parameter, secret key sk , a message $M \in \{0, 1\}^*$ and $R = (pk_0, \dots, pk_{N-1})$ as inputs, it outputs a signature Σ on the message M with respect to the ring R . Here, (pk, sk) is a valid key pair output by RKgen(pp) and $pk \in R$.

RVerify $_{pp}(M, R, \Sigma)$: This deterministic algorithm verifies a purported ring signature Σ on the message M with respect to the ring of public keys R , it outputs 1 if

the signature is valid or 0 otherwise.

The correctness requirement says that signatures generated by honest users can always be accepted as valid ones. This is formalized as follows.

Definition 5.2.2 (Correctness). *A ring signature $(\text{RSetup}, \text{RKgen}, \text{RSign}, \text{RVerify})$ is correct if for any $pp \leftarrow \text{RSetup}(1^\lambda)$, any $(pk, sk) \leftarrow \text{RKgen}(pp)$, any R such that $pk \in R$, any $M \in \{0, 1\}^*$, we have $\text{RVerify}_{pp}(M, R, \text{RSign}_{pp}(sk, M, R)) = 1$.*

A ring signature scheme is said to be secure if it satisfies unforgeability with respect to insider corruption, and statistical anonymity.

A ring signature is unforgeable with respect to insider corruption if it is infeasible to forge a ring signature without controlling one of the ring members. It is said to be statistically anonymous if signatures generated by two adversarially chosen keys are statistically indistinguishable. The formal definitions are given below.

Definition 5.2.3 (Unforgeability w.r.t. insider corruption). A ring signature scheme $(\text{RSetup}, \text{RKgen}, \text{RSign}, \text{RVerify})$ is unforgeable w.r.t. insider corruption if for all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ who is given access to the following oracles succeeds is negligible:

- PKGen on the j -th query runs $(pk_j, sk_j) \leftarrow \text{RKgen}(pp)$ and returns pk_j to $\mathcal{A}$.
- $\mathcal{A}$ is given access to $\text{Sign}(j, M, R)$, which returns $\text{RSign}_{pp}(sk_j, M, R)$ conditioned on the fact that (pk_j, sk_j) has been generated by PKGen and $pk_j \in R$.
- $\mathcal{A}$ is given access to a corrupted oracle $\text{Corrupt}(j)$, which returns sk_j , provided that (pk_j, sk_j) has been generated by PKGen.
- $\mathcal{A}$ outputs (M^*, R^*, Σ^*) such that $\text{Sign}(\cdot, M^*, R^*)$ has not been queried. Moreover, R^* is non-empty and only contains public keys pk_j generated by PKGen for which j has not been corrupted.

the probability is

$$\Pr[pp \leftarrow \text{RSetup}(1^\lambda); (M^*, R^*, \Sigma^*) \leftarrow \mathcal{A}^{\text{PKGen, Sign, Corrupt}}(pp) : \\ \text{RVerify}_{pp}(M^*, R^*, \Sigma^*) = 1] \in \text{negl}(\lambda),$$

Definition 5.2.4 (Statistical anonymity). *A ring signature scheme provides statistical anonymity if, for any (possibly unbounded) adversary $\mathcal{A}$,*

$$\Pr \left[\begin{array}{l} pp \leftarrow \text{RSetup}(1^\lambda); (M^*, j_0, j_1, R^*) \leftarrow \mathcal{A}^{\text{RKgen}(pp)}(pp) \\ b \xleftarrow{\$} \{0, 1\}; \Sigma^* \leftarrow \text{RSign}_{pp}(sk_{j_b}, M^*, R^*) \end{array} : \mathcal{A}(\Sigma^*) = b \right] \\ = 1/2 + \text{negl}(\lambda),$$

where $pk_{j_0}, pk_{j_1} \in R^*$.

5.3 Description of Our Ring Signature

The scheme works as follows.

RSetup(1^λ) : Let λ be the security parameter, choose $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ and $q = 2 \cdot 2^c \cdot n/c$. This algorithm outputs a uniformly random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times q}$, and returns $pp = \mathbf{B}$.

RKgen($pp = \mathbf{B}$) : On input $pp = \mathbf{B}$, choose a uniformly random vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \xleftarrow{\$} \{0, 1\}^{2n}$, generates $\mathbf{d} = h_{\mathbf{B}}(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \in \{0, 1\}^n$, and outputs $(pk, sk) = (\mathbf{d}, \mathbf{x})$.

RSign _{pp} (sk, M, R) : On input pp , sk , a message $M \in \{0, 1\}^*$ and a ring $R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$, where $\mathbf{d}_i \in \{0, 1\}^n$ for every $i \in [0, N-1]$, and $sk = \mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \in \{0, 1\}^{2n}$ such that $\mathbf{d} = h_{\mathbf{B}}(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \in R$, outputs a ring signature Σ on $M \in \{0, 1\}^*$ with respect to the ring as follows:

1. First, On input the ring R , $\text{Accu}_{\mathbf{B}}(R)$ is able to build a Merkle tree based on the ring to obtain the accumulated value $\mathbf{u} \in \{0, 1\}^n$.
2. On input ring R and $\mathbf{d}$, we aim to get the witnesses of $\mathbf{d}$ with which to find a path from $\mathbf{d}$ to the accumulated value $\mathbf{u}$ in the Merkle tree. Run algorithm $\text{WitGen}_{\mathbf{B}}(R, \mathbf{d})$, it outputs the witness

$$w = \left((j_1, \dots, j_\ell) \in \{0, 1\}^\ell, (\mathbf{w}_\ell, \dots, \mathbf{w}_1) \in (\{0, 1\}^n)^\ell \right)$$

3. Generate a NIZKAoK Π_{ring} to show the possession of a valid pair $(pk, sk) = (\mathbf{d}, \mathbf{x})$ such that $\mathbf{d}$ is correctly accumulated in $\mathbf{u}$ in the Merkle tree. The details of the protocol is given in Section 5.4. The interactive protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligibly small soundness error and then made non-interactive via the Fiat-Shamir heuristic as $\Pi_{\text{ring}} = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}\}_{i=1}^\kappa)$, where

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

4. Outputs $\Sigma = \Pi_{\text{ring}}$.

$\text{RVerify}_{pp}(M, R, \Sigma)$: On input $pp = \mathbf{B}$, a message M , a ring $R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$, and a signature Σ , the algorithm outputs 0/1 as follows:

1. Build a Merkle tree based on the ring R , then the algorithm $\text{Accu}_{\mathbf{B}}(R)$ computes the root $\mathbf{u}$ of the tree.
2. Parse Σ as $\Sigma = (\{\text{CMT}_i\}_{i=1}^\kappa, (Ch_1, \dots, Ch_\kappa), \{\text{RSP}\}_{i=1}^\kappa)$. Return 0 if $(Ch_1, \dots, Ch_\kappa) \neq \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, R)$. For each $i = 1$ to κ , run the verification phase of the protocol from Section 5.4 with public input $(\mathbf{B}, \mathbf{u})$ 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. Otherwise, return 1.

Take a ring with 2^3 ring members in Figure 5.1 as an example.

Figure 5.1: A ring with $2^3 = 8$ members

5.4 The Underlying Zero-Knowledge Argument System

In this section, we describe the statistical ZKAoK that is invoked by the signer when producing ring signatures. The protocol is an extension of the one for the accumulator from Section 4.6. Here, the prover not only convinces the verifier that a secret value $\mathbf{d}$ is properly accumulated to the root of the tree, but it also shows that he knows $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \in \{0, 1\}^n \times \{0, 1\}^n$ satisfying

$$\mathbf{B}_0 \cdot \text{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}^{(1)}) = \mathbf{d}. \quad (5.1)$$

The associated relation $\mathbf{R}_{\text{ring}}$ is as follows:

$$\mathbf{R}_{\text{ring}} = \left\{ \left((\mathbf{B}, \mathbf{u}); (\mathbf{d}, w, \mathbf{x}) \right) : \text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \wedge \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}^{(1)}) = \mathbf{d} \right\}.$$

Since the transformation layer for the accumulator has been established in Section 4.6, we only need to consider the new equation (5.1). Recall that $\mathbf{v}_\ell = \mathbf{d}$, define $\mathbf{x}_\ell = \text{Encode}(\mathbf{v}_\ell) \in \{0, 1\}^{2n}$, then we have $\mathbf{v}_\ell = \mathbf{I}_n^* \cdot \mathbf{x}_\ell$. Hence, the equation (5.1) is equivalent

to the following:

$$\mathbf{B}_0 \cdot \text{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}^{(1)}) \oplus \mathbf{I}_n^* \cdot \mathbf{x}_\ell = \mathbf{0}. \quad (5.2)$$

Up to this point, one can check that it suffices for $\mathcal{P}$ to convince $\mathcal{V}$ that he knows $j_1, \dots, j_\ell \in \{0, 1\}^\ell$, $\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{w}_1, \dots, \mathbf{w}_\ell \in \{0, 1\}^n$, and $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \in \{0, 1\}^n$ satisfying

$$\left\{ \begin{array}{l} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u} \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{x}_1 = \mathbf{0} \\ \dots\dots\dots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{x}_{\ell-1} = \mathbf{0} \\ \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}^{(1)}) \oplus \mathbf{I}_n^* \cdot \mathbf{x}_\ell = \mathbf{0}, \end{array} \right. \quad (5.3)$$

where

$$\left\{ \begin{array}{l} \mathbf{y}_i = \text{Ext}(j_i, \text{RE}(\mathbf{v}_i)) \in \{0, 1\}^q, i \in [\ell] \\ \mathbf{z}_i = \text{Ext}(\bar{j}_i, \text{RE}(\mathbf{w}_i)) \in \{0, 1\}^q, i \in [\ell] \\ \mathbf{x}_i = \text{Encode}(\mathbf{v}_i) \in \{0, 1\}^{2n}, i \in [\ell]. \end{array} \right. \quad (5.4)$$

By suitably concatenating and extending the matrices and vectors from public input, we can obtain public matrix $\mathbf{M}_R$, public vector $\mathbf{v}_R$ such that $\mathbf{M}_R \cdot \mathbf{w}_R = \mathbf{v}_R$ with secret vector $\mathbf{w}_R \in \{0, 1\}^{L_R}$ with $L_R = (2\ell + 1)q + 2\ell n$ and

$$\mathbf{w}_R = (\mathbf{y}_1 \parallel \dots \parallel \mathbf{y}_\ell \parallel \mathbf{z}_1 \parallel \dots \parallel \mathbf{z}_\ell \parallel \mathbf{x}_1 \parallel \dots \parallel \mathbf{x}_\ell \parallel \text{RE}(\mathbf{x}^{(0)}) \parallel \text{RE}(\mathbf{x}^{(1)})) \quad (5.5)$$

With the desired unifying equation, we now define the set VALID_R that contains our secret vector $\mathbf{w}_R$, the set $\mathcal{S}_R$ and permutations $\{\Gamma_{\phi_R} : \phi_R \in \mathcal{S}_R\}$ such that the

conditions in (2.2) hold. Let VALID_R contain all vectors of the form

$$\mathbf{w}'_R = (\mathbf{y}'_1 \parallel \dots \parallel \mathbf{y}'_\ell \parallel \mathbf{z}'_1 \parallel \dots \parallel \mathbf{z}'_\ell \parallel \mathbf{x}'_1 \parallel \dots \parallel \mathbf{x}'_\ell \parallel \mathbf{u}'_0 \parallel \mathbf{u}'_1) \in \{0, 1\}^{L_R} \quad (5.6)$$

satisfying the following conditions:

- For $i \in [\ell]$, there exists $\mathbf{v}'_i, \mathbf{w}'_i \in \{0, 1\}^n, j'_i \in \{0, 1\}$ such that

$$\mathbf{y}'_i = \text{Ext}(j'_i, \text{RE}(\mathbf{v}'_i)) \in \{0, 1\}^q, \quad \text{and} \quad \mathbf{z}'_i = \text{Ext}(\bar{j}'_i, \text{RE}(\mathbf{w}'_i)) \in \{0, 1\}^q.$$

- For $i \in [\ell]$, $\mathbf{x}'_i = \text{Encode}(\mathbf{v}'_i) \in \{0, 1\}^{2n}$.
- For $i \in \{0, 1\}$, there exists $\mathbf{x}^{(i)'} \in \{0, 1\}^n$ such that $\mathbf{u}'_i = \text{RE}(\mathbf{x}^{(i)'}) \in \{0, 1\}^{q/2}$.

Let $\mathcal{S}_R$ be of the following form:

$$\mathcal{S}_R = (\{0, 1\}^n)^\ell \times (\{0, 1\}^n)^\ell \times \{0, 1\}^\ell \times (\{0, 1\}^n)^2.$$

Then, for each $\phi_R = (\mathbf{b}_1, \dots, \mathbf{b}_\ell, \mathbf{e}_1, \dots, \mathbf{e}_\ell, g_1, \dots, g_\ell, \mathbf{p}_0, \mathbf{p}_1) \in \mathcal{S}_R$, define the permutation Γ_{ϕ_R} that transforms

$$\mathbf{w}^*_R = (\mathbf{y}^*_1 \parallel \dots \parallel \mathbf{y}^*_\ell \parallel \mathbf{z}^*_1 \parallel \dots \parallel \mathbf{z}^*_\ell \parallel \mathbf{x}^*_1 \parallel \dots \parallel \mathbf{x}^*_\ell \parallel \mathbf{u}^*_0 \parallel \mathbf{u}^*_1) \in \{0, 1\}^{L_R} \quad (5.7)$$

with $\mathbf{y}^*_i, \mathbf{z}^*_i \in \{0, 1\}^q, \mathbf{x}^*_i \in \{0, 1\}^{2n}$ for $i \in [\ell]$ and $\mathbf{u}^*_i \in \{0, 1\}^{q/2}$ for $i \in \mathbb{Z}_2$ to

$$\Gamma_{\phi_R}(\mathbf{w}^*_R) = \left(\Psi_{g_1, \mathbf{b}_1}(\mathbf{y}^*_1) \parallel \dots \parallel \Psi_{g_\ell, \mathbf{b}_\ell}(\mathbf{y}^*_\ell) \parallel \Psi_{\bar{g}_1, \mathbf{e}_1}(\mathbf{z}^*_1) \parallel \dots \parallel \Psi_{\bar{g}_\ell, \mathbf{e}_\ell}(\mathbf{z}^*_\ell) \parallel \right. \\ \left. F'_{\mathbf{b}_1}(\mathbf{x}^*_1) \parallel \dots \parallel F'_{\mathbf{b}_\ell}(\mathbf{x}^*_\ell) \parallel E'_{\mathbf{p}_0}(\mathbf{u}^*_0) \parallel E'_{\mathbf{p}_1}(\mathbf{u}^*_1) \right).$$

Based on the equivalences observed in (4.2), (4.1) and (3.6), it can be checked that the conditions in (2.2) are satisfied. We thus have reduced the considered relation into an instance of $\text{R}_{\text{abstract}}$ and the desired statistical ZKAoK protocol can be obtained by

running the protocol in Figure 2.1. The protocol has communication cost $\mathcal{O}(L_R) = \ell \cdot \mathcal{O}(q + n)$ bits.

5.5 Analysis of the Scheme

We summarize the properties of the given ring signature scheme in the following theorem.

Theorem 5.5.1. The ring signature scheme described in Section 5.3 is correct, and produces signatures of bit-size $\mathcal{O}(n \cdot \log N)$. The scheme is unforgeable w.r.t. insider corruption based on the hardness of the $2\text{-RNSD}_{n,2n,c}$ problem, and it is statistically anonymous.

The proof of unforgeability employs the following lemma.

Lemma 5.5.2. Let $h : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be a hash function. Then for $\mathbf{x} \xleftarrow{\$} \{0, 1\}^{2n}$, the probability that there exists $\mathbf{x}' \in \{0, 1\}^{2n}$ such that $h(\mathbf{x}') = h(\mathbf{x})$ is at least $1 - 2^{-n}$.

Proof. There are in total 2^{2n} elements $\mathbf{x} \in \{0, 1\}^{2n}$. Among them, there exist at most $2^n - 1$ elements that do not have $\mathbf{x}' \in \{0, 1\}^{2n}$ such that $h(\mathbf{x}') = h(\mathbf{x})$. Thus, the probability that a uniformly random element $\mathbf{x}$ has a corresponding $\mathbf{x}'$ such that $h(\mathbf{x}') = h(\mathbf{x})$ is at least $\frac{2^{2n} - (2^n - 1)}{2^{2n}} > 1 - 2^{-n}$. $\square$

With $m = 2n$ and $\mathbf{x} \xleftarrow{\$} \{0, 1\}^m$, there exists $\mathbf{x}' \in \{0, 1\}^m \setminus \{\mathbf{x}\}$ such that $\mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{x}'$ with overwhelming probability $1 - 2^{-n}$.

Theorem 5.5.3. The scheme provides unforgeability w.r.t. insider corruption in the random oracle model if the $2\text{-RNSD}_{n,2n,c}$ problem is hard.

Proof. We prove unforgeability w.r.t. insider corruption by contraposition. Suppose that an adversary $\mathcal{A}$ succeeds with non-negligible advantage ϵ in breaking unforgeability property, then we are able to construct a PPT algorithm $\mathcal{B}$ that either breaks

the security of the accumulator, or breaks the computational soundness of the zero-knowledge protocol, or directly solves an $2\text{-RNSD}_{n,n+k,c}$ problem with non-negligible probability.

Towards this goal, $\mathcal{B}$ first defines the public parameter $pp = \mathbf{B}$. When $\mathcal{A}$ makes queries to the PKGen oracle, $\mathcal{B}$ chooses a uniformly random vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \end{pmatrix} \in \{0, 1\}^{2n}$ and computes $\mathbf{d} = h_{\mathbf{B}}(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$. It then provides $\mathcal{A}$ with public key $pk = \mathbf{d}$ while keeping the secret key $sk = \mathbf{x}$. For the $\text{Corrupt}(j)$ oracle and $\text{Sign}(j, M, R)$ oracle, $\mathcal{B}$ is able to answer all the queries made by $\mathcal{A}$ since $\mathcal{B}$ possesses all the secret keys. When $\mathcal{A}$ halts and outputs (M^*, R^*, Σ^*) . Consider the case that $\mathcal{A}$ succeeds in breaking the unforgeability w.r.t. insider corruption. It then follows that Σ^* is a valid signature on message M^* , $\text{Sign}(\cdot, M^*, R^*)$ has not been queried, and R^* contains all the public keys pk_j generated by PKGen in which j has not been corrupted. Denote $R^* = (pk_{i_1}, \dots, pk_{i_{|R^*|}})$ and rewrite it as a set of binary vectors $(\mathbf{d}_0, \dots, \mathbf{d}_{|R^*|-1})$. Let $\Sigma^* = \Pi_{\text{ring}}^* = (\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}_i^*\}_{i=1}^\kappa)$, where $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, R^*)$ and RSP^* is a valid response w.r.t. CMT_i^* and CH^* . Note that the oracle $\mathcal{H}_{\text{FS}}(\cdot)$ outputs uniformly random elements in $\{1, 2, 3\}^\kappa$ and the same answer is output when a hash query is made more than once.

We claim that $\mathcal{A}$ had queried $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}^*, R^*)$ to the hash oracle $\mathcal{H}_{\text{FS}}$ with overwhelming probability, where $\mathbf{u}^* = \text{TAcc}_{\mathbf{A}}(R^*)$. Otherwise, the probability of guessing the value of $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}^*, R^*)$ correctly would be at most $3^{-\kappa}$, which is negligible. Therefore, with probability at least $\epsilon' := \epsilon - 3^{-\kappa}$, $\mathcal{A}$ had queried the hash oracle $\mathcal{H}_{\text{FS}}$ and we denote $t^* \in \{1, \dots, Q_H\}$ as 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 $32 \cdot Q_H / (\epsilon - 3^{-\kappa})$ extra executions of the adversary $\mathcal{A}$. In each new run, all queries receive exactly the same answers as in the original run until the point of t^* -th query to the hash oracle. From the t^* -th query on, $\mathcal{B}$ replies $\mathcal{A}$ with uniformly random and independent values for each new run. As a result, the

input of t^* -th query is the same as in the initial run while the output is uniformly random and independent from the original run. By the Forking Lemma of Brickell et al. [30], with probability at least $1/2$, $\mathcal{B}$ can obtain a 3-fork involving the tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}^*, R^*)$ with pairwise distinct hash values $\text{CH}_{t^*}^{(1)}, \text{CH}_{t^*}^{(2)}, \text{CH}_{t^*}^{(3)} \in \{1, 2, 3\}^\kappa$ and corresponding valid responses $\text{RSP}_{t^*}^{(1)}, \text{RSP}_{t^*}^{(2)}, \text{RSP}_{t^*}^{(3)}$. With probability $1 - (7/9)^\kappa$, the results of [30] imply that there exists some $j \in \{1, \dots, \kappa\}$ such that $\{\text{CH}_{t^*,j}^{(1)}, \text{CH}_{t^*,j}^{(2)}, \text{CH}_{t^*,j}^{(3)}\} = \{1, 2, 3\}$.

From 3 valid responses $(\text{RSP}_{t^*,j}^{(1)}, \text{RSP}_{t^*,j}^{(2)}, \text{RSP}_{t^*,j}^{(3)})$ w.r.t. the same commitment CMT_j^* and challenges 1, 2, 3, Theorem 2.3.13 ensures that $\mathcal{B}$ is able to extract witnesses $(\mathbf{x}^*, \mathbf{d}^*, w^*)$, where $\mathbf{x}^* = \begin{pmatrix} \mathbf{x}^{*(0)} \\ \mathbf{x}^{*(1)} \end{pmatrix}$, $w^* = ((j_1^*, \dots, j_\ell^*), (\mathbf{w}_\ell^*, \dots, \mathbf{w}_1^*))$ such that $(j_1^*, \dots, j_\ell^*) \in \{0, 1\}^\ell$ is the binary expansion of some index $j^* \in \{0, \dots, |R^*| - 1\}$ and

$$h_{\mathbf{B}}(\mathbf{x}^{*(0)}, \mathbf{x}^{*(1)}) = \mathbf{d}^*, \quad (5.8)$$

$$\text{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*) = 1. \quad (5.9)$$

Since $\mathcal{A}$ wins the game, either we have (i) $\mathbf{d}^* \notin R^* = (\mathbf{d}_0, \dots, \mathbf{d}_{|R^*|-1})$ or (ii) $\mathbf{d}^* \in R^* = (\mathbf{d}_0, \dots, \mathbf{d}_{|R^*|-1})$ and $\mathbf{d}^* = \mathbf{d}_{j^*} = pk_{j^*}$.

Case (i) implies that, $\mathcal{B}$ can use $(\mathbf{d}^*, R^*, \mathbf{u}^*)$ to break the security of the accumulator from equation (5.9).

Case (ii) implies that, the extracted witnesses $(\mathbf{d}_{j^*}, \mathbf{x}^*)$ satisfy equation (5.8) by the soundness of the argument system. When $\mathcal{A}$ queried the PKGen oracle, $\mathcal{B}$ chose $sk_{j^*} = \mathbf{x}_{j^*} = \begin{pmatrix} \mathbf{x}_{j^*}^{(0)} \\ \mathbf{x}_{j^*}^{(1)} \end{pmatrix} \in \{0, 1\}^{2n}$ satisfying

$$\mathbf{d}_{j^*} = \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}_{j^*}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}_{j^*}^{(1)}). \quad (5.10)$$

Since R^* contains only uncorrupted public keys, we have that $\mathbf{x}_{j^*} \neq \mathbf{x}^*$ with probability at least $1/2$ according to Lemma 5.5.2. Furthermore, the statistical Witness Indistinguishability (WI) of the argument system implies that only a negligible amount of information regarding which witness among $\mathbf{x}^*$ and $\mathbf{x}_{j^*}$ is leaked. Combining equation (5.8) and equation (5.10), we obtain $\mathbf{B}_0 \cdot (\text{RE}(\mathbf{x}_{j^*}^{(0)}) \oplus \text{RE}(\mathbf{x}^{*(0)})) \oplus \mathbf{B}_1 \cdot (\text{RE}(\mathbf{x}_{j^*}^{(1)}) \oplus \text{RE}(\mathbf{x}^{*(1)})) = \mathbf{0}$, which yields a valid $2\text{-RNSD}_{n,2n,c}$ solution $\mathbf{w} = \text{RE}(\mathbf{x}_{j^*}) \oplus \text{RE}(\mathbf{x}^*)$.

Therefore, with probability at least $1/2 \cdot (\epsilon - 3^{-\kappa}) \cdot (1 - (7/9)^\kappa) \cdot 1/2$, which is non-negligible, $\mathcal{B}$ solves an instance of the $2\text{-RNSD}_{n,2n,c}$ problem.

□

Theorem 5.5.4. The scheme provides statistical anonymity in the random oracle model.

The proof of the above theorem relies on the statistical witness indistinguishability of the underlying argument system. The proof is straightforward and omitted.

Chapter 6

Code-Based Logarithmic-size Group Signatures

This chapter presents our construction of logarithmic-size group signature from code-based assumptions.

We first give an overview of our technique in Section 6.1, then recall the definitions and security requirements of group signatures in Section 6.2 and the randomized McEliece encryption scheme in Section 6.3. Our group signature construction is then described in Section 6.4. Its supporting zero-knowledge argument system is presented in Section 6.5 and its security proofs are provided in Section 6.6.

6.1 Overview of Our Techniques

Our group signature follows the static group signature model proposed by Bellare et al [13], which typically require the following 3 layers. A signature layer requires a signature scheme to sign messages. The existing code-based signatures consist of two categories: the “hash-and-sign” paradigm and the “Fiat-Shamir” category. The first paradigm consists of the CFS signature [41] and its modified version [42, 51, 113]. The second category are those schemes derived from Stern’s identification protocol

[122] and its variants [2, 37, 47, 124] via the Fiat-Shamir transformation [53]. In our construction, we follow the second Fiat-Shamir category. An encryption layer enable the tracing mechanism. The signer additionally encryption his identifying information and send the ciphertext as part of group signature. The group manager can decrypt the ciphertext in order to trace back to the actual signer. The code-based encryption schemes consist of McEliece cryptosystems [92] and its variants [46, 104, 110] as well as Niederreiter cryptosystems [103] and its variant [90]. In our construction, we use the randomized McEliece encryption scheme [92] and apply for the Naor-Yung double-encryption method to achieve CCA2 secure in the random oracle model. A Zero-knowledge protocol layer serves as the bridge between the first two layers. The zero-knowledge demonstrates that the given signature is signed by a certificated group member who also encrypts the message. We have the ZK proofs of the plaintext knowledge for the McEliece cryptosystem [47, 66].

The scheme is based on the ring signature of Section 5.3, where an encryption layer is introduced to enable the tracing capability. To this end, the signer is constrained to encrypt his identity, which is the essentially the bits determining the path from his respective leaf in the Merkle tree to the tree root. To enable anonymity in the strongest sense, we apply the Naor-Yung double encryption technique [100] to a randomized variant of the McEliece encryption scheme [92], suggested in [104], and obtain a CCA2-secure encryption mechanism.

6.2 Definitions and Security Requirements of Group Signatures

In this section, we give the syntax and the security model of the static group signatures [13]. A static group signature means that the group size is determined when the group is setup and no other new members can be added in, the group members can

issue signatures on behalf of the group anonymously and the opener is able to trace back the actual signer.

Definition 6.2.1. A group signature scheme consists of the following 4 polynomial-time algorithms (**GKgen**, **GSign**, **GVerify**, **GOpen**):

- **GKgen**: The probabilistic group key generation algorithm **GKgen** takes input $1^\lambda, 1^N$, where λ is the security parameter and $N \in \mathbb{N}$ is the number of group users, outputs the group public key **gpk**, the group manager's secret key **gmsk** and **gsk**[k] the secret signing key for each group user of index $k \in \{0, \dots, N-1\}$, that is, returns a triple (**gpk**, **gmsk**, **gsk**).
- **GSign**: This probabilistic group signing algorithm on inputs the group public key **gpk**, a secret signing key **gsk**[k] for some signer $k \in \{0, \dots, N-1\}$, and a message M , outputs a group signature Σ on M .
- **GVerify**: This deterministic group verify algorithm on inputs **gpk**, M , a signature Σ , and outputs either 1 if the signature is valid or 0 otherwise.
- **GOpen**: On inputs **gpk**, **gmsk**, a message M , a signature Σ , and this deterministic opening algorithm provides an index k when it succeeds or $\perp$ otherwise.

We now recall the requirements for group signature scheme: correctness, *full anonymity* and *traceability*.

Correctness requires that signatures generated by honest group members are always deemed valid and the opener can always correctly identify the originator of any signature. Formally, for all $\lambda, N \in \mathbb{N}$, all (**gpk**, **gmsk**, **gsk**) generated by **GKgen**($1^\lambda, 1^N$), for any $k \in [0, N-1]$, and any message $M \in \{0, 1\}^*$, the following conditions hold:

$$\begin{aligned} \text{GVerify}(\text{gpk}, M, \text{GSign}(\text{gpk}, \text{gsk}[k], M)) &= 1 \text{ and} \\ \text{GOpen}(\text{gpk}, \text{gmsk}, M, \text{GSign}(\text{gsk}[k], M)) &= k. \end{aligned}$$

Full anonymity requires that it is infeasible for any PPT adversary to distinguish which of two signers of its choice signed a targeted message even if the adversary is accessible to all group user's secret keys, can choose that message and can query to the opening oracle for any signature except the challenged one. The requirement is modelled in the experiment game $\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-b}}(\lambda, N)$ as below.

$\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-b}}(\lambda, N)$
 $(\text{gpk}, \text{gmsk}, \text{gsk}) \leftarrow \text{GKeyGen}(1^\lambda, 1^N)$
 $(\text{st}, j_0, j_1, M^*) \leftarrow \mathcal{A}_1^{\mathcal{GS}.\text{GOpen}(\text{gpk}, \text{msk}, \dots)}(\text{gpk}, \text{gsk})$
 $\Sigma^* \leftarrow \text{GSign}(\text{gpk}, \text{gsk}[j_b], M^*)$
 $b' \leftarrow \mathcal{A}_2^{\mathcal{GS}.\text{GOpen}(\text{gpk}, \text{msk}, \dots), \neg(M^*, \Sigma^*)}(\text{st}, \Sigma^*)$
 Return b'

The advantage of the adversary $\mathcal{A}$ against full anonymity denoted as $\mathbf{Adv}_{\mathcal{GS}, \mathcal{A}}^{\text{anon}}(\lambda, N)$ is defined as follows:

$$\mathbf{Adv}_{\mathcal{GS}, \mathcal{A}}^{\text{anon}}(\lambda, N) = \left| \Pr[\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-1}}(\lambda, N) = 1] - \Pr[\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-0}}(\lambda, N) = 1] \right|.$$

A group signature is *fully anonymous* if for any PPT adversary $\mathcal{A}$, advantage $\mathbf{Adv}_{\mathcal{GS}, \mathcal{A}}^{\text{anon}}(\lambda, N)$ is negligible in the security parameter λ .

Full traceability demands that it is infeasible for any PPT adversary to output a valid group signature that either fails in the opening algorithm or that was traced to a user who is not in the coalition set even if the adversary could corrupt the group manager. The requirement is modeled in the experiment game $\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{trace}}(\lambda, N)$ as below.

$\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{trace}}(\lambda, N)$
 $(\text{gpk}, \text{gmsk}, \text{gsk}) \leftarrow \text{GKeygen}(1^n, 1^N)$
 $\text{st} \leftarrow (\text{gmsk}, \text{gpk})$
 $\mathcal{C} \leftarrow \emptyset ; K \leftarrow \varepsilon ; \text{Cont} \leftarrow \text{true}$
 while $(\text{Cont} = \text{true})$ do
 $(\text{Cont}, \text{st}, j) \leftarrow \mathcal{A}_1^{\mathcal{GS}.\text{GSign}(\text{gpk}, \text{gsk}[\cdot, \cdot])}(\text{st}, K)$

```

    if  $Cont = \text{true}$  then  $\mathcal{C} \leftarrow \mathcal{C} \cup \{j\}$ ;
     $K \leftarrow \text{gsk}[j]$ 
end if
end while;
 $(M^*, \Sigma^*) \leftarrow \mathcal{A}_2^{\text{GS.GSign}(\text{gpk}, \text{gsk}[\cdot], \cdot)}(\text{st})$ 
if  $\text{GVerify}(\text{gpk}, M^*, \Sigma^*) = 0$ , Return 0
if  $\text{GOpen}(\text{gpk}, \text{gmsk}, M^*, \Sigma^*) = \perp$ ,
    Return 1
if  $\text{GOpen}(\text{gpk}, \text{gmsk}, M^*, \Sigma^*) = j^* \quad \wedge \quad (j^* \in \{0, \dots, N-1\} \setminus \mathcal{C})$ 
     $\wedge$  (no signing query involved( $j^*, M^*$ ))
then Return 1 else Return 0

```

The advantage of the adversary $\mathcal{A}$ against full traceability is defined as

$$\mathbf{Succ}_{\mathcal{GS}, \mathcal{A}}^{\text{trace}}(\lambda, N) = \Pr[\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{trace}}(\lambda, N) = 1].$$

A group signature scheme is *fully traceable* if for any PPT adversary $\mathcal{A}$, the probability $\mathbf{Succ}_{\mathcal{GS}, \mathcal{A}}^{\text{trace}}(\lambda, N)$ is negligible in λ .

6.3 The Randomized McEliece Encryption Scheme

Now we recall a randomized variant of the McEliece [92] encryption scheme as suggested in [104], which is CPA-secure. The scheme is summarized as follows. It consists of the following algorithms ME.Setup , ME.KeyGen , ME.Enc , and ME.Dec .

- $\text{ME.Setup}(1^\lambda)$: Let $n_e = n_e(\lambda)$, $k_e = k_e(\lambda)$, $t_e = t_e(\lambda)$ be the parameters for a binary $[n_e, k_e, 2t_e + 1]$ Goppa code. Choose $k_1, k_2 \in \mathbb{Z}$ such that $k_e = k_1 + k_2$. Let $\mathbb{Z}_2^{k_2}$ be the plaintext space.

- **ME.KeyGen**(n_e, k_e, t_e): This algorithm outputs the encryption key and decryption key for the randomized McEliece encryption scheme. It works as follows:
 1. Choose a generator matrix $\mathbf{G}' \in \mathbb{Z}_2^{n_e \times k_e}$ of a randomly selected $[n_e, k_e, 2t_e + 1]$ Goppa code. Let $\mathbf{S} \in \mathbb{Z}_2^{k_e \times k_e}$ be a random invertible matrix and $\mathbf{P} \in \mathbb{Z}_2^{n_e \times n_e}$ be a random permutation matrix, then compute $\mathbf{G} = \mathbf{P}\mathbf{G}'\mathbf{S} \in \mathbb{Z}_2^{n_e \times k_e}$. Output encryption key $\mathbf{pk}_{\text{ME}} = \mathbf{G}$ and decryption key $\mathbf{sk}_{\text{ME}} = (\mathbf{S}, \mathbf{G}', \mathbf{P})$.
- **ME.Enc**($\mathbf{pk}_{\text{ME}}, \mathbf{m}$): On input a message $\mathbf{m} \in \mathbb{Z}_2^{k_2}$ and $\mathbf{pk}_{\text{ME}}$, sample $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_2^{k_1}$ and $\mathbf{e} \xleftarrow{\$} \mathcal{B}(n_e, t_e)$, and then output the ciphertext $\mathbf{c} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix} \oplus \mathbf{e} \in \mathbb{Z}_2^{n_e}$.
- **ME.Dec**($\mathbf{sk}_{\text{ME}}, \mathbf{c}$): On input the ciphertext $\mathbf{c}$ and decryption key $\mathbf{sk}_{\text{ME}}$, it works as follows:
 1. Multiply $\mathbf{P}^{-1}$ to the left of the ciphertext $\mathbf{c}$, then apply an error-correcting algorithm. Obtain $\mathbf{m}'' = \text{Decode}_{\mathbf{G}'}(\mathbf{c} \cdot \mathbf{P}^{-1})$ where *Decode* is an error-correcting algorithm with respect to $\mathbf{G}'$. Returns $\perp$ if *Decode* fails.
 2. Multiply $\mathbf{S}^{-1}$ to the right of the ciphertext $\mathbf{m}''$, then $\mathbf{m}' = \mathbf{S}^{-1} \cdot \mathbf{m}''$, parse $\mathbf{m}' = \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix}$, where $\mathbf{u} \in \mathbb{Z}_2^{k_1}$ and $\mathbf{m} \in \mathbb{Z}_2^{k_2}$, and return $\mathbf{m}$.

Assuming the hardness of the Decisional McEliece problem **DMcE** and the Decisional Learning Parity with (fixed-weight) Noise problem **DLPN** [104], we have the CPA-security of the randomized McEliece scheme in the standard model. The definitions of the two problems are given below.

Definition 6.3.1. The **DMcE**(n_e, k_e, t_e) problem asks to determine whether a matrix $\mathbf{G} \in \mathbb{Z}_2^{n_e \times k_e}$ is uniformly chosen from $\mathbb{Z}_2^{n_e \times k_e}$ or is generated by the algorithm **ME.KeyGen**(n_e, k_e, t_e) described above.

When $n_e = n_e(\lambda), k_e = k_e(\lambda), t_e = t_e(\lambda)$, we say that the **DMcE**(n_e, k_e, t_e) problem is hard, if the success probability of any PPT distinguisher is at most $1/2 + \text{negl}(\lambda)$.

Definition 6.3.2. The $\text{DLPN}(k_e, n_e, \mathbf{B}(n_e, t_e))$ problem asks to determine whether a pair $(\mathbf{A}, \mathbf{s}) \in \mathbb{Z}_2^{n_e \times k_e} \times \mathbb{Z}_2^n$ is uniformly chosen from $\mathbb{Z}_2^{n_e \times k_e} \times \mathbb{Z}_2^n$ or is obtained by choosing $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n_e \times k_e}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_2^k$, $\mathbf{e} \xleftarrow{\$} \mathbf{B}(n_e, t_e)$ and outputting $(\mathbf{A}, \mathbf{A} \cdot \mathbf{u} \oplus \mathbf{e})$.

When $k_e = k_e(\lambda)$, $n_e = n_e(\lambda)$, $t_e = t_e(\lambda)$, we say that the $\text{DLPN}(k_e, n_e, \mathbf{B}(n_e, t_e))$ problem is hard, if the success probability of any PPT distinguisher is at most $1/2 + \text{negl}(\lambda)$.

6.4 Our Logarithmic-Size Group Signature

Scheme

GKeygen $(1^\lambda, 1^N)$: On input the parameters $1^\lambda, 1^N$, the algorithm samples a uniformly random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times q}$, where $q = 2 \cdot 2^c \cdot n/c$. Then it performs as follows to get the group public key, group manager's secret key and secret signing key for each group user.

1. For each $j \in [0, N-1]$, sample a random binary vector $\mathbf{x}_j = \begin{pmatrix} \mathbf{x}_j^{(0)} \\ \mathbf{x}_j^{(1)} \end{pmatrix} \xleftarrow{\$} \{0, 1\}^{2n}$ and compute $\mathbf{d}_j = h_{\mathbf{B}}(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)}) \in \{0, 1\}^n$. $\{\mathbf{d}_j\}_{j=0}^{N-1}$ should be pairwise distinct, otherwise restart the process. Then define the set $R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$.
2. Run algorithm $\text{Accu}_{\mathbf{B}}(R)$ to build the Merkle tree based on R and the hash function $h_{\mathbf{B}}$, and obtain the accumulated value $\mathbf{u} \in \{0, 1\}^n$.
3. Let $\ell = \lceil \log N \rceil$. For each $j \in [0, N-1]$, let $(j_1, \dots, j_\ell)$ be the binary representation of j . Run algorithm $\text{WitGen}_{\mathbf{B}}(R, \mathbf{d}_j)$ to outputs a witness $w^{(j)}$ of $\mathbf{d}_j$ such that $\mathbf{d}_j$ is correctly accumulated in $\mathbf{u}$.

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

Then define $\text{gsk}[j] = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)})$.

4. Run $\text{ME.KeyGen}(n_e, k_e, t_e)$ twice to obtain two key-pairs $(\text{pk}_{\text{ME}}^{(1)} = \mathbf{G}_1 \in \mathbb{Z}_2^{n_e \times k_e}, \text{sk}_{\text{ME}}^{(1)})$ and $(\text{pk}_{\text{ME}}^{(2)} = \mathbf{G}_2 \in \mathbb{Z}_2^{n_e \times k_e}, \text{sk}_{\text{ME}}^{(2)})$.

5. Output

$$\text{gpk} := \{\mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2\}; \quad \text{gmsk} := \text{sk}_{\text{ME}}^{(1)}; \quad \text{gsk} := (\text{gsk}[0], \dots, \text{gsk}[N-1]).$$

GSign(gpk, gsk[j], M): On input $M \in \{0, 1\}^*$ and the user's secret signing key $\text{gsk}[j] = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)})$, where $w^{(j)} = ((j_1, \dots, j_\ell), (\mathbf{w}_\ell^{(j)}, \dots, \mathbf{w}_1^{(j)}))$, the user performs as follows :

1. Encrypt the identity $\text{bin}(j) = (j_1, \dots, j_\ell) \in \{0, 1\}^\ell$ twice using the randomized McEliece encryption scheme. More precisely, for each $i \in \{1, 2\}$, sample $\mathbf{r}_i \xleftarrow{\$} \mathbb{Z}_2^{k_e - \ell}$, $\mathbf{e}_i \xleftarrow{\$} \mathbf{B}(n_e, t_e)$ and compute

$$\mathbf{c}_i = \mathbf{G}_i \cdot \begin{pmatrix} \mathbf{r}_i \\ \text{bin}(j) \end{pmatrix} \oplus \mathbf{e}_i \in \mathbb{Z}_2^{n_e}.$$

2. Generate a NIZKAoK Π_{group} to show the possession of a valid secret tuple $\tau = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)}, \mathbf{r}_1, \mathbf{e}_1, \mathbf{r}_2, \mathbf{e}_2)$, where

$$\mathbf{x}_j = \begin{pmatrix} \mathbf{x}_j^{(0)} \\ \mathbf{x}_j^{(1)} \end{pmatrix} \text{ and } w^{(j)} = ((j_1, \dots, j_\ell), (\mathbf{w}_\ell^{(j)}, \dots, \mathbf{w}_1^{(j)})),$$

such that:

- (a) $h_{\mathbf{B}}(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)}) = \mathbf{d}_j$ and $\text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}_j, w^{(j)}) = 1$.
- (b) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\text{bin}(j) = (j_1, \dots, j_\ell)^\top$ with randomness $(\mathbf{r}_1, \mathbf{e}_1) \in \mathbb{Z}_2^{k_e - \ell} \times \mathbf{B}(n_e, t_e)$ and $(\mathbf{r}_2, \mathbf{e}_2) \in \mathbb{Z}_2^{k_e - \ell} \times \mathbf{B}(n_e, t_e)$, respectively.

This is done by running the interactive argument system of Section 6.5 on public

input $(\mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2)$ and prover's witness τ defined above. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as

$$\Pi_{\text{group}} = (\{\text{CMT}_i\}_{i=1}^{\kappa}, \text{CH}, \{\text{RSP}\}_{i=1}^{\kappa}),$$

where

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

3. Outputs the group signature $\Sigma = (\Pi_{\text{group}}, \mathbf{c}_1, \mathbf{c}_2)$.

GVerify(gpk, M , Σ) : On input gpk, M , Σ , the verification algorithm proceeds as follows:

1. 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 $(Ch_1, \dots, Ch_{\kappa}) \neq \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^{\kappa}, \mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2)$, then return 0.
2. For each $i = 1$ to κ , run the verification phase of the protocol in Section 6.5 with public input $(\mathbf{A}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2)$ to check the validity of RSP_i w.r.t. CMT_i and Ch_i . If any of the conditions does not hold, then return 0.
3. Return 1.

GOpen(gpk, gmsk, Σ , M): On input $\text{gmsk} = \text{sk}_{\text{ME}}^{(1)}$ and a group signature $\Sigma = (\Pi_{\text{group}}, \mathbf{c}_1, \mathbf{c}_2)$ on message M , this algorithm proceeds as follows:

1. Run $\text{ME.Dec}(\text{sk}_{\text{ME}}^{(1)}, \mathbf{c}_1)$ to decrypt $\mathbf{c}_1$. If decryption fails, then return $\perp$.
Otherwise, let $\mathbf{p} = (j'_1, \dots, j'_\ell)^{\top} \in \{0, 1\}^{\ell}$ be the result of decryption.
2. Outputs index $j \in [0, N - 1]$ that has binary representation $(j'_1, \dots, j'_\ell)$.

6.5 The Supporting Zero-Knowledge Argument System

In this section, we present the zero-knowledge protocol that is exploited by signers when generating signatures in Section 6.4. This protocol is extended from the one in Section 5.4, for which an encryption layer is added. Specifically, the prover additionally proves possession of secret values $\mathbf{r}_1, \mathbf{r}_2 \in \{0, 1\}^{k_e - \ell}$, $\mathbf{e}_1, \mathbf{e}_2 \in \mathcal{B}(n_e, t_e)$ such that

$$\mathbf{c}_1 = \mathbf{G}_1 \cdot \begin{pmatrix} \mathbf{r}_1 \\ \text{bin}(j) \end{pmatrix} \oplus \mathbf{e}_1, \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2 \cdot \begin{pmatrix} \mathbf{r}_2 \\ \text{bin}(j) \end{pmatrix} \oplus \mathbf{e}_2, \quad (6.1)$$

where $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{Z}_2^{n_e \times k_e}$ and $\mathbf{c}_1, \mathbf{c}_2 \in \{0, 1\}^{n_e}$. The associated relation R_{group} is given below.

$$R_{\text{group}} = \left\{ \left((\mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1, \mathbf{c}_2); (\mathbf{d}, w, \mathbf{x}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{e}_1, \mathbf{e}_2) \right) : \right. \\ \left. \text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \wedge \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}^{(1)}) = \mathbf{d} \wedge \right. \\ \left. \mathbf{c}_1 = \mathbf{G}_1 \cdot \begin{pmatrix} \mathbf{r}_1 \\ \text{bin}(j) \end{pmatrix} \oplus \mathbf{e}_1 \wedge \mathbf{c}_2 = \mathbf{G}_2 \cdot \begin{pmatrix} \mathbf{r}_2 \\ \text{bin}(j) \end{pmatrix} \oplus \mathbf{e}_2 \right\}.$$

We first recall the permuting technique to prove in ZK the knowledge of $\mathbf{e} \in \mathcal{B}(n_e, t_e)$ suggested by Stern [122]. To this end, the prover samples a uniformly random permutation $\sigma \in \mathcal{S}_{n_e}$ and shows the verifier that $\sigma(\mathbf{e}) \in \mathcal{B}(n_e, t_e)$. Due to the following equivalence

$$\mathbf{e} \in \mathcal{B}(n_e, t_e) \iff \sigma(\mathbf{e}) \in \mathcal{B}(n_e, t_e), \quad (6.2)$$

the verifier should be convinced that $\mathbf{e} \in \mathcal{B}(n_e, t_e)$. Furthermore, $\sigma(\mathbf{e})$ is uniform in $\mathcal{B}(n_e, t_e)$ since σ is uniform in $\mathcal{S}_{n_e}$.

Now let us look at relation R_{group} . As the transformation for the ring signature layer has been demonstrated in Section 5.4, we consider the newly appeared relations in (6.1).

Denote $\text{bin}(j) = (j_1, \dots, j_\ell)^\top \in \{0, 1\}^\ell$, $\mathbf{f} = \text{Encode}(\text{bin}(j)) \in \{0, 1\}^{2\ell}$, and $\mathbf{h}_i = \text{Encode}(\mathbf{r}_i) \in \{0, 1\}^{2k_e - 2\ell}$ for $i \in \{1, 2\}$. Let $\mathbf{G}_i^* \in \mathbb{Z}_2^{n_e \times (k_e + \ell)}$ be the matrix obtained by adding a zero-column $\mathbf{0}^{n_e}$ right before each of the columns of $\mathbf{G}_i$, for $i \in \{1, 2\}$. It is then verifiable that equations in (6.1) is equivalent to

$$\mathbf{c}_1 = \mathbf{G}_1^* \cdot \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{f} \end{pmatrix} \oplus \mathbf{e}_1, \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2^* \cdot \begin{pmatrix} \mathbf{h}_2 \\ \mathbf{f} \end{pmatrix} \oplus \mathbf{e}_2. \quad (6.3)$$

Let $\mathbf{w}_g = (\mathbf{f} \parallel \mathbf{h}_1 \parallel \mathbf{h}_2 \parallel \mathbf{e}_1 \parallel \mathbf{e}_2) \in \{0, 1\}^{2n_e + 4k_e - 2\ell}$ be the secret vectors appeared in (6.3). Before proceeding further, let us recall that for the ring signature layer, we obtain $\mathbf{M}_R \cdot \mathbf{w}_R = \mathbf{v}_R$, where $\mathbf{w}_R$ is of the form (5.5). Combing the ring signature layer and the new encryption layer, we are able to obtain a unifying equation $\mathbf{M}_G \cdot \mathbf{w}_G = \mathbf{v}_G$, where $\mathbf{w}_G = (\mathbf{w}_R \parallel \mathbf{w}_g) \in \{0, 1\}^{L_G}$ with $L_G = L_R + 2n_e + 4k_e - 2\ell$.

Up to this point, we have transformed the considered statement into a unified form. We are now ready to define the set VALID_G that consists of our secret vector $\mathbf{w}_G$, the set $\mathcal{S}_G$ and the associated permutation $\{\Gamma_{\phi_G} : \phi_G \in \mathcal{S}_G\}$ such that the conditions in (2.2) hold.

Let VALID_G be the set of all vectors of the form $\mathbf{w}'_G = (\mathbf{w}'_R \parallel \mathbf{f}' \parallel \mathbf{h}'_1 \parallel \mathbf{h}'_2 \parallel \mathbf{e}'_1 \parallel \mathbf{e}'_2) \in \{0, 1\}^{L_G}$, satisfying the following conditions:

- $\mathbf{w}'_R$ satisfies the conditions specified in (5.6).
- $\mathbf{f}' = \text{Encode}(\text{bin}(j'))$ with $\text{bin}(j') = (j'_1, \dots, j'_\ell)^\top$.
- There exists $\mathbf{r}'_i \in \{0, 1\}^{k_e - \ell}$ such that $\mathbf{h}'_i = \text{Encode}(\mathbf{r}'_i)$ for $i \in \{1, 2\}$.
- $\mathbf{e}'_1, \mathbf{e}'_2 \in \mathcal{B}(n_e, t_e)$.

Define the set $\mathcal{S}_G = \mathcal{S}_R \times \left(\{0, 1\}^{k_e - \ell}\right)^2 \times \left(\mathcal{S}_{n_e}\right)^2$, where $\mathcal{S}_R$ is defined in Section 5.4. Then for each $\phi_G = (\phi_R, \mathbf{a}_1, \mathbf{a}_2, \sigma_1, \sigma_2) \in \mathcal{S}_G$, the associated permutation Γ_{ϕ_G} performs as follows. It transforms vector of the form $\mathbf{w}_G^* = (\mathbf{w}_R^* \| \mathbf{f}^* \| \mathbf{h}_1^* \| \mathbf{h}_2^* \| \mathbf{e}_1^* \| \mathbf{e}_2^*)$ into

$$\Gamma_{\phi_G}(\mathbf{w}_G^*) = (\Gamma_{\phi_R}(\mathbf{w}_R^*) \| F'_{\mathbf{g}}(\mathbf{f}^*) \| F'_{\mathbf{a}_1}(\mathbf{h}_1^*) \| F'_{\mathbf{a}_2}(\mathbf{h}_2^*) \| \sigma_1(\mathbf{e}_1^*) \| \sigma_2(\mathbf{e}_2^*)),$$

where $\mathbf{g} = (g_1, \dots, g_\ell)^\top$, g_i is the one appearing in ϕ_R for all $i \in [\ell]$, and Γ_{ϕ_R} is defined as in Section 5.4.

Based on the equivalences observed in (4.2), (4.1), (3.6) and (6.2), it can be checked that the conditions in (2.2) are satisfied. We thus have reduced the considered relation into an instance of $\mathbf{R}_{\text{abstract}}$ and the desired statistical ZKAoK protocol can be obtained by running the protocol in Figure 2.1. The protocol has communication cost $\mathcal{O}(L_G) = \mathcal{O}((2\ell + 1)q + 2\ell n + 2n_e + 4k_e - 2\ell)$ bits.

6.6 Analysis of the Scheme

We summarize the properties of the given group signature scheme in the following theorem.

Theorem 6.6.1. The group signature scheme described in Section 6.4 is correct, and produces signatures of bit-size $\mathcal{O}((2\ell + 1)q + 2\ell n + 2n_e + 4k_e - 2\ell)$ with $\ell = \lceil \log N \rceil$.

Theorem 6.6.2. The scheme provides full traceability in the random oracle model if the $2\text{-RNSD}_{n, 2n, c}$ problem is hard.

Proof. The proof is done by contraposition. Assuming that an adversary $\mathcal{A}$ succeeds with non-negligible advantage ϵ , then we construct a PPT algorithm $\mathcal{B}$ that solves a $2\text{-RNSD}_{n, 2n, c}$ problem with non-negligible probability.

Given a matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$, $\mathcal{B}$ faithfully runs the **GKeygen** algorithm and then provides the adversary with **gpk** and **gmsk**. At the same time, $\mathcal{B}$ keeps all the users' private

keys $\text{gsk}[j] = (\mathbf{x}_j, \mathbf{d}_j, w^{(j)})$, where $\mathbf{u} = \text{Accu}_{\mathbf{B}}(\mathbf{d}_0, \dots, \mathbf{d}_1)$ and $\mathbf{d}_j = h_{\mathbf{B}}(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)}) \in \{0, 1\}^n$ for each $j \in [0, N-1]$. This allows $\mathcal{B}$ to answer all the corruption and signing queries. Note that the oracle $\mathcal{H}_{\text{FS}}(\cdot)$ outputs uniformly random elements in $\{1, 2, 3\}^\kappa$ and the same answer is output when a hash query is made more than once.

When $\mathcal{A}$ halts and outputs (M^*, Σ^*) . Consider the case that $\mathcal{A}$ succeeds in breaking the traceability. It then follows that Σ^* is a valid signature on message M^* . Let (M^*, Σ^*) open to some uncorrupted user j^* , for which (j^*, M^*) is not queried to the signing oracle. Parse Σ^* as $(\Pi_{\text{group}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$ and Π_{group}^* as $(\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}^*\}_{i=1}^\kappa)$.

We claim that $\mathcal{A}$ had queried $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1^*, \mathbf{c}_2^*)$ with overwhelming probability. Otherwise, the probability of guessing the value of $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1^*, \mathbf{c}_2^*)$ would be at most $3^{-\kappa}$, which is negligible. Therefore, with probability at least $\epsilon' := \epsilon - 3^{-\kappa}$, $\mathcal{A}$ had queried the hash oracle $\mathcal{H}_{\text{FS}}$ and denote by $t^* \in \{1, \dots, Q_H\}$ 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 $32 \cdot Q_H / (\epsilon - 3^{-\kappa})$ extra executions of the adversary $\mathcal{A}$. In each new run, all queries receive exactly the same answers as in the original run until the point of t^* -th query to the hash oracle. From the t^* -th query on, $\mathcal{B}$ replies $\mathcal{A}$ with uniformly random and independent values for each new run. As a result, the input of t^* -th query is the same as in the initial run while the output is uniformly random and independent from the original run. By the Forking Lemma of Brickell et al. [30], with probability at least $1/2$, $\mathcal{B}$ can obtain a 3-fork involving the same tuple $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{u}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{c}_1^*, \mathbf{c}_2^*)$ with pairwise distinct hash values $\text{CH}_{t^*}^{(1)}, \text{CH}_{t^*}^{(2)}, \text{CH}_{t^*}^{(3)} \in \{1, 2, 3\}^\kappa$ and corresponding valid responses $\text{RSP}_{t^*}^{(1)}, \text{RSP}_{t^*}^{(2)}, \text{RSP}_{t^*}^{(3)}$. With probability $1 - (7/9)^\kappa$, the results of [30] imply that there exists some $j \in \{1, \dots, \kappa\}$ such that $\{\text{CH}_{t^*,j}^{(1)}, \text{CH}_{t^*,j}^{(2)}, \text{CH}_{t^*,j}^{(3)}\} = \{1, 2, 3\}$.

From 3 valid responses $(\text{RSP}_{t^*,j}^{(1)}, \text{RSP}_{t^*,j}^{(2)}, \text{RSP}_{t^*,j}^{(3)})$ w.r.t. the same commitment CMT_j^* and all challenges 1, 2, 3, Theorem 2.3.13 ensures that $\mathcal{B}$ is able to extract

witnesses

$$(\mathbf{x}^*, \mathbf{d}^*, w^*, \mathbf{r}_1^*, \mathbf{r}_2^*, \mathbf{e}_1^*, \mathbf{e}_2^*),$$

where $\mathbf{x}^* = \begin{pmatrix} \mathbf{x}^{*(0)} \\ \mathbf{x}^{*(1)} \end{pmatrix} \in \{0, 1\}^{2n}$, $w^* = ((j_1^*, \dots, j_\ell^*), (\mathbf{w}_\ell^*, \dots, \mathbf{w}_1^*))$ such that $(j_1^*, \dots, j_\ell^*) \in \{0, 1\}^\ell$ is the binary expansion of some integer $j^* \in [0, N - 1]$ and

$$h_{\mathbf{B}}(\mathbf{x}_0^*, \mathbf{x}_1^*) = \mathbf{d}^*, \quad (6.4)$$

$$\text{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*) = 1. \quad (6.5)$$

Since $\mathcal{A}$ wins the game, either we have (i) $\mathbf{d}^* \notin R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$ or (ii) $\mathbf{d}^* \in R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$ and $\mathbf{d}^* = \mathbf{d}_{j^*}$. Note that $\{\mathbf{d}_j\}_{j=0}^{N-1}$ are pairwise distinct, as ensured by the **GKey** algorithm.

Case (i) implies a violation of the security of the underlying accumulator.

Case (ii) implies that, $\mathbf{c}_1^*$ decrypts to $(j_1^*, \dots, j_\ell^*) \in \{0, 1\}^\ell$ from the soundness of the argument system. Therefore, $\mathcal{A}$ did not obtain the private key $\text{gsk}[j^*]$, which contains a vector $\mathbf{x}_{j^*} = \begin{pmatrix} \mathbf{x}_{j^*}^{(0)} \\ \mathbf{x}_{j^*}^{(1)} \end{pmatrix} \in \{0, 1\}^{2n}$ that satisfies

$$\mathbf{d}_{j^*} = \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}_{j^*}^{(0)}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}_{j^*}^{(1)}). \quad (6.6)$$

Since j^* is uncorrupted, $\mathbf{x}_{j^*} \neq \mathbf{x}^*$ with probability at least $1/2$. Combining equation (7.8) and equation (6.6), it follows that

$$\mathbf{B}_0 \cdot (\text{RE}(\mathbf{x}_{j^*}^{(0)}) \oplus \text{RE}(\mathbf{x}^{*(0)})) \oplus \mathbf{B}_1 \cdot (\text{RE}(\mathbf{x}_{j^*}^{(1)}) \oplus \text{RE}(\mathbf{x}^{*(1)})) = \mathbf{0},$$

which yields a valid $2\text{-RNSD}_{n, 2n, c}$ solution $\mathbf{w} = \text{RE}(\mathbf{x}_{j^*}) \oplus \text{RE}(\mathbf{x}^*)$. To argue $\mathbf{x}_{j^*} \neq \mathbf{x}^*$ with probability at least $1/2$, we observe that $\mathcal{A}$ may learn $\mathbf{d}_{j^*} = \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}_{j^*}^{(0)}) \oplus \mathbf{B}_1 \cdot$

$\text{RE}(\mathbf{x}_{j^*}^{(1)})$ by corrupting $\text{gsk}[j^* + 1]$ or $\text{gsk}[j^* - 1]$. However, there exists at least one another vector $\mathbf{x}^* \neq \mathbf{x}_{j^*}$. What is more, the statistical WI property implies that a negligible amount of information regarding which witness among $\mathbf{x}^*$ and $\mathbf{x}_{j^*}$ is leaked when replying the signing queries $(j^*, \cdot)$.

Therefore, with probability at least $1/2 \cdot (\epsilon - 3^{-\kappa}) \cdot (1 - (7/9)^\kappa) \cdot 1/2$, which is non-negligible, $\mathcal{B}$ solves an instance of the $2\text{-RNSD}_{n,2n,c}$ problem.

□

Theorem 6.6.3. Assume that the $\text{DMcE}(n_e, k_e, t_e)$ and the $\text{DLPN}(k_e, n_e, \mathcal{B}(n_e, t_e))$ problems are hard and the argument system is simulation-sound, then the group signature scheme is fully anonymous.

Proof. We prove the theorem using a sequence of indistinguishable games. The first game is experiment $\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-0}}(\lambda, N)$ whereas the last game is experiment $\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-1}}(\lambda, N)$. It then follows that our group signature scheme is fully anonymous. For each i , denote by W_i the event that the adversary outputs 1 in Game i .

Game 0: In this game, the challenger $\mathcal{B}$ runs experiment $\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-0}}(\lambda, N)$, in which the adversary $\mathcal{A}$ receives signature $\Sigma^* \leftarrow \text{GSign}(\text{gpk}, \text{gsk}[j_0], M^*)$ in the challenge phase. Let $\Sigma^* = (\Pi_{\text{group}}^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$. Then we have $\Pr[W_0] = \Pr[\mathbf{Exp}_{\mathcal{GS}, \mathcal{A}}^{\text{anon-0}}(\lambda, N) = 1]$.

Game 1: This game is the same as Game 0 except that $\mathcal{B}$ keeps both decryption keys of McEliece encryption scheme instead of wiping out the second one $\text{sk}_{\text{ME}}^{(2)}$. We can see that this change does not affect $\mathcal{A}$'s view. Therefore $\Pr[W_1] = \Pr[W_0]$.

Game 2: This game modifies Game 1 as follows: The opening oracle employs the second McEliece decryption key $\text{sk}_{\text{ME}}^{(2)}$ instead of the first one $\text{sk}_{\text{ME}}^{(1)}$. The view of $\mathcal{A}$ is the same as in Game 1 until the event F_1 , that $\mathcal{A}$ queries the opening oracle a signature $\Sigma = (\Pi_{\text{group}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1$ and $\mathbf{c}_2$ encrypting distinct ℓ -bit strings, happens. Since event F_1 breaks the soundness of the underlying

argument system, we have $|\Pr[W_2] - \Pr[W_1]| \leq \Pr[F_1] \leq \mathbf{Adv}_B^{\text{sound}}(\lambda) \in \text{negl}(\lambda)$.

Game 3: This game changes Game 2 by generating simulated proof instead of real proof in the challenged phase. In other words, Π_{group^*} is now a simulated proof while $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ encrypt the same ℓ -bit string. Due to the statistical zero-knowledge property of the underlying argument system, the view of the adversary in Game 2 and Game 3 are statistically indistinguishable, implying that $|\Pr[W_3] - \Pr[W_2]| \in \text{negl}(\lambda)$.

Game 4: This game modifies game 3 by changing the distribution of the challenged signature Σ^* . Here, we compute $\mathbf{c}_1^*$ by encrypting the ℓ -bit binary representation of j_1 instead of j_0 and keep $\mathbf{c}_2^*$ unchanged. By the semantic security of McEliece's encryption scheme for the public key $\mathbf{G}_1$, we have $|\Pr[W_4] - \Pr[W_3]| \in \text{negl}(\lambda)$. We remark that the semantic security can be relied on since we only utilize $\text{sk}_{\text{ME}}^{(2)}$ in the opening oracle queries.

Game 5: This game is identical to Game 4 except that we switch back to the key $\text{sk}_{\text{ME}}^{(1)}$ in the opening oracle queries. Note that $\mathcal{A}$'s view remains unchanged unless the event F_2 , that $\mathcal{A}$ queries the opening oracle a signature $\Sigma = (\Pi_{\text{group}}, \mathbf{c}_1, \mathbf{c}_2)$ with $\mathbf{c}_1$ and $\mathbf{c}_2$ encrypting different ℓ -bit strings, occurs. However, event F_2 would violate the simulation-soundness of the underlying argument system, we thus have $|\Pr[W_5] - \Pr[W_4]| \leq \Pr[F_2] \leq \mathbf{Adv}_B^{\text{ss-sound}}(\lambda)$.

Game 6: In this game, we modify again the distribution of the challenged signature Σ^* . It changes $\mathbf{c}_2^*$ to be encryption of the binary representation of j_1 instead of j_0 . By the semantic security of McEliece's encryption scheme with respect to $\mathbf{G}_2$, we have $|\Pr[W_6] - \Pr[W_5]| \in \text{negl}(\lambda)$. Note that now both $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ encrypt the same message, therefore Π_{group}^* is a simulated proof for a true statement.

Game 7: This game modifies Game 6 in one aspect: it generates a real proof Π_{group}^* in the challenged phase. Since the underlying argument system is statistically zero-knowledge, we have $|\Pr[W_7] - \Pr[W_6]| \in \text{negl}(\lambda)$. This is in fact experiment $\mathbf{Exp}_{\mathcal{G}, \mathcal{A}}^{\text{anon-1}}(\lambda, N)$, hence $\Pr[W_7] = \Pr[\mathbf{Exp}_{\mathcal{G}, \mathcal{A}}^{\text{anon-1}}(\lambda, N) = 1]$.

As a result, we find that

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

this concludes the proof. □

Chapter 7

Verifier-Local Revocation Group Signature with Backward Unlinkability and Code-Based Decentralized E-cash System

This chapter presents our construction of the verifier-local revocation (VLR) group signature with backward unlinkability, for which the syntax and security requirements are given in Section 7.2. Our group signature construction is then described in Section 7.3. Its underlying ZKAoK is put in Section 7.4 and its correctness and security proofs are provided in Section 7.5.

7.1 Overview of Our Techniques

To design a VLR group signature scheme, we make some modifications to the ordinary group signature scheme in Chapter 6, which is in the static model. Our scheme employs on a $2\text{-RNSD}_{n,2n,c}$ -based Merkle tree accumulator that is also described in Chapter 4 and a statistically hiding commitment scheme. Consider a

group with $N = 2^l$ group members, the manager chooses N uniformly random vectors $\mathbf{x}_0, \dots, \mathbf{x}_{N-1} \in \{0, 1\}^n$ and commits them with uniformly random openings $\mathbf{s}_0, \dots, \mathbf{s}_{N-1} \in \{0, 1\}^k$ to $\mathbf{d}_0, \dots, \mathbf{d}_{N-1}$, respectively. Here we utilize the code-based statistically hiding commitment in Chapter 3. Then build a Merkle-tree accumulator on top of these N commitments, iteratively hashes them from level ℓ to level 1 and outputs the accumulated root value $\mathbf{u}$. The user i 's signing secret key is $\mathbf{p}_i = (\mathbf{x}_i, \mathbf{s}_i)$ and the witnesses consist of all the information needed to recompute a path along $\mathbf{d}_i$ to the accumulated root $\mathbf{u}$. The user needs to prove in ZK of $(\mathbf{p}_i, \mathbf{d}_i)$ such that $\mathbf{d}_i$ is a committed value of $\mathbf{p}_i$ and there is a valid hash chain from $\mathbf{d}_i$ to $\mathbf{u}$. Then the user performs a ZKAoK to prove that he is a certified group member who possess a corresponding secret signing key.

To equip our scheme with feature of backward unlinkability, we divide time into T intervals, signatures achieve the nice property when they are tied to the time interval in which they were issued. In our construction, a deterministic function which is mapping time interval j into matrix $\mathbf{D}_j$ works as a random oracle. For each user i at period j , the user's revocation key $\mathbf{u}_{ij}$ is the hash value of $h_{\mathbf{D}_j}$ on $(\mathbf{x}_i, \mathbf{s}_i)$. Since the hash function $h_{\mathbf{D}_j}$ follows the design of our commitment which is also statistical hiding, all revocation tokens $h_{\mathbf{D}_j}(\mathbf{x}_i, \mathbf{s}_i)$ are statistically close to uniformly random vectors, for all $i \in [N - 1]$, $j \in [T - 1]$. and one can check that user's revocation tokens at different period are unlinkable due to the different j and associated matrices $\mathbf{D}_j$. It keeps the former signatures also unlinkable, which is the property of backward unlinkability. When a VLR group signature is issued at a certain time interval, the user outputs a LPN instance:

$$\mathbf{b} = \mathbf{A} \cdot \text{token} + \mathbf{e},$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}$ and a vector $\mathbf{e}$ from Bernoulli distribution. Then he signs a signature by extending the ZKAoK of proving the accumulator relation and commitment

relation to further prove this LPN relation, in which $\mathbf{b}$ is honestly generated and $\mathbf{token}$ is correctly computed from $\mathbf{p}_j$. In the end, the user outputs the signature, together with vector $\mathbf{b}$ and a revocation list RL_j containing all $\mathbf{t}_{ij}$ at period j , the verifier computes vector $\mathbf{e}'_{ij} = \mathbf{b} - \mathbf{A} \cdot \mathbf{t}_{ij}$. For each $\mathbf{t}_{ij}$ in RL_j , if an index i exists such that $\mathbf{e}'_{ij} \in \text{Ber}(m, t)$, it means the user i has been revoked from the group and the signature should be rejected.

The underlying ZKAoK for our VLR group signature with backward unlinkability has a constant soundness error of $2/3$ and it can be achieved negligibly small by running the protocol for a sufficient number times and achieves a NIZK via the Fiat-Shamir Paradigm [50].

The design of our code-based decentralized e-cash scheme follows the framework of Zerocoin [95]. The scheme is quite simple to apply our VLR group signature without the revocation mechanism and backward unlinkability property and adapt it to the decentralized e-cash system. A user first generates a random vector $\mathbf{x}$ as his coin serial number and then commits it using the same code-based commitment scheme to get a coin $\mathbf{d}$ and an opening $\mathbf{s}$. To spend the coin in the transaction, the user first obtains a set of coins $(\mathbf{d}_1, \dots, \mathbf{d}_N)$ that includes his coin, then builds a code-based accumulator on top of these coins and produces a non-interactive ZKAoK to show that he owns a coin in the set and possesses $(\mathbf{x}, \mathbf{s})$ for the commitment. Any users can verify the proof to check that his serial number has not appeared in other spend transaction. In our scheme, the coin is linked to the user serial number and the user cannot double spend his coins without reusing his serial number due to the zero-knowledge proof for the commitment and the membership in the accumulator.

In summary, we solve open questions of achieving VLR and backward unlinkability in code-based group signature and achieving a code-based decentralized e-cash. Our solution is simple but efficient.

7.2 VLR Group Signatures with Backward Unlinkability

7.2.1 Syntax of VLR Group Signature

First, we review the model and the security requirements that specified in [98]. The scheme requires revocation of the users, and all revocation token list can only be downloaded by the verifier. The lifetime of the scheme is divided into T periods and the revocation token for user i at time period j is given by $\text{grt}[i][j]$. To achieve backward unlinkability, all the revocation token need to be statistical close.

Definition 7.2.1. A VLR group signature scheme with backward unlinkability consists of the following 3 polynomial-time algorithms (KGen_{vtr} , Sign_{vtr} , Verify_{vtr}):

- KGen_{vtr} : Take $\lambda \in \mathbb{N}$, $N, T \in \mathbb{N}$, where λ is the security parameter, N is the group size and T is the number of the time interval, this key generation algorithm outputs the group public key gpk , user secret keys $\text{gsk}[i]$ and user revocation tokens $\text{grt}[i][j]$, for group user $i \in \{0, \dots, N-1\}$ at time interval $j \in \{0, \dots, T-1\}$.
- Sign_{vtr} : On input gpk , $\text{gsk}[i]$, and message $M \in \{0, 1\}^*$ and the current period j , outputs a signature Σ .
- Verify_{vtr} : Take gpk , j , a revocation token list $RL_j \subseteq \{\text{grt}[0][j], \dots, \text{grt}[N-1][j]\}$ at period j , M , and the signature Σ . It outputs *valid* iff Σ is generated by an unrevoked user or *invalid* otherwise.

7.2.2 Notions of Correctness and Security Requirements

There are three requirements for a VLR group signature scheme that achieve BU as mentioned in [98]: *correctness*, *BU-anonymity* and *traceability*.

Correctness For all $\lambda, N, T \in \mathbb{N}$, for all $(\text{gpk}, \text{gsk}, \text{grt}) \leftarrow \text{KGen}_{vtr}(\lambda, N, T)$, $i \in [0, N-1]$, $j \in [0, T-1]$, all RL_j and $M \in \{0, 1\}^*$. A sufficient and necessary condition hold

that a user i has not revoked at time period j if and only if the signature issued by him is always valid:

$$\text{Verify}_{vtr}(\text{gpk}, j, M, RL_j, \text{Sign}_{vtr}(\text{gpk}, j, \text{gsk}[i], M)) = \text{valid} \Leftrightarrow \text{grt}[i][j] \notin RL_j.$$

In the group signature, there is always a tracing mechanism to retrieve the identity of the signer who issued a signature. Interestingly, the definition of VLR group signature implicit implies an inherent tracing algorithm for revealing the actual signer of a signature. One can use the revocation token grt as the tracing key [24], then the verification algorithm will turns into a tracing algorithm by testing all tokens in RL_j , that is, processing $\text{Verify}_{vtr}(\text{gpk}, M, RL_j = \{\text{grt}[i][j]\}_{i \in [0, N-1]}, \Sigma)$ and outputting the first one for which the verification algorithm indicates invalid, it should be the retrieved signer. In other words, if a signer is in the revocation list and issues a signature, the latter should be invalid.

BU-anonymity It is infeasible for For any PPT $\mathcal{A}$ to tell apart which key is used to generate a given signature among two adaptively chosen keys.

1. **Setup** The challenger $\mathcal{C}$ runs KGen_{vtr} and obtains $(\text{gpk}, \text{gsk}, \text{grt})$. The adversary $\mathcal{A}$ is only provided with gpk .

2. **Query** At period j , the $\mathcal{A}$ inform the start of j to $\mathcal{C}$ so that they can increment j simultaneously. During j , $\mathcal{A}$ is accessible to the signing oracle, corruption oracle and revocation oracle.

- $\mathcal{O}_{\text{Signing}}$: It returns a signature Σ on input message M , an element $i \in \{0, \dots, N-1\}$ and current interval j .
- $\mathcal{O}_{\text{Corruption}}$: It outputs group secret key $\text{gsk}[i]$ when it takes input of $i \in \{a, \dots, N-1\}$.
- $\mathcal{O}_{\text{Revocation}}$: Outputs group revocation token $\text{grt}[i][j]$ when it takes input of $i \in \{a, \dots, N-1\}$ at time period j .

3. **Challenge** The adversary $\mathcal{A}$ outputs two distinct users i_1, i_2 such that neither i_1 nor i_2 has been revoked before or during j^* , for some j^* . The challenger $\mathcal{C}$ then selects a challenge bit $c \xleftarrow{\$} \{0, 1\}$ and outputs a signature Σ^* on user i_c , sends back to $\mathcal{A}$.

4. **RestrictQueries** $\mathcal{A}$ can make further queries as in phase 2 for time intervals after j^* under restrictions of not accessible to the secret keys for user i_0 or i_1 and the revocation tokens at period j^* .

5. **Output** $\mathcal{A}$ outputs a bit value b and wins if the output is the same as the one chosen by $\mathcal{C}$, namely, $b = c$.

The advantage of an adversary succeed in breaking BU-anonymity is

$$\mathbf{Adv}_{\mathcal{A}}^{bu-ano} = |\Pr[b = c] - 1/2|.$$

The scheme is *BU-anonymous* if the above advantage $\mathbf{Adv}_{\mathcal{A}}^{bu-ano}$ is negligible.

Traceability demands that for an PPT adversary, the probability of forging a valid group signature that can be traced to a user who is not in the coalition set is negligible.

1. **Setup** The challenger $\mathcal{C}$ obtains $(\mathbf{gpk}, \mathbf{gsk}, \mathbf{grt})$ by running the key generation algorithm $\mathbf{KGen}_{vtr}$, and denotes an initial empty corruption list CL . $\mathcal{A}$ is given $\mathbf{gpk}$ and $\mathbf{grt}$.

2. **Query** The adversary $\mathcal{A}$ is accessible to two oracles listed below.

- $\mathcal{O}_{\text{Signing}}$: Outputs Σ on input $M, i \in \{0, \dots, N-1\}$ and j .
- $\mathcal{O}_{\text{Corruption}}$: Outputs group secret key $\mathbf{gsk}[i]$ when it takes input of $i \in \{a, \dots, N-1\}$ and recodes it into the corruption list CL .

3. **Forgery** $\mathcal{A}$ outputs a signature Σ^* on M^* , an time interval j^* and a set $RL_{j^*}^*$ of the revocation tokens at j^* , under the following conditions.

- $\text{Verify}_{vtr}(\mathbf{gpk}, j^*, M^*, RL_{j^*}^*, \Sigma^*) = \text{valid}$.
- No signing queries on M^* are made by the adversary.

- The implicit tracing algorithm fails to output valid for all tokens or it traces to a user in CL/RL_{j*}^* .

Denote the advantage of $\mathcal{A}$ against traceability as

$$\mathbf{Adv}_{\mathcal{A}}^{\text{trace}} = \Pr[\mathcal{A} \text{ is successful}].$$

A scheme is *traceable* if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\text{trace}}$ is negligible.

7.3 Our VLR Group Signature Scheme

KGen_{vlr}($1^\lambda, 1^N$) : This algorithm takes the parameters $1^\lambda, 1^N$. This algorithm samples a uniformly random hash matrix $\mathbf{B} = (\mathbf{B}_0, \mathbf{B}_1) \xleftarrow{\$} \mathbb{Z}_2^{n \times q}$, where $q = 2 \cdot 2^c \cdot n/c$ and $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_2^{n \times 2^c \cdot n/c}$, as well as sampling commitment matrix $\mathbf{C} = (\mathbf{C}_0, \mathbf{C}_1) \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$, where $m = 2^c \cdot (n + k)/c$, $\mathbf{C}_0 \in \mathbb{Z}_2^{n \times 2^c \cdot n/c}$ and $\mathbf{C}_1 \in \mathbb{Z}_2^{n \times 2^c \cdot k/c}$. Then it performs as follows.

1. Sample N random binary vector $\mathbf{x}_i \in \{0, 1\}^n$ and binary opening $\mathbf{s}_i \in \{0, 1\}^k$. Denote $\mathbf{p}_i = \begin{pmatrix} \mathbf{x}_i \\ \mathbf{s}_i \end{pmatrix} \xleftarrow{\$} \{0, 1\}^{n+k}$, $\forall i \in [0, N-1]$. Then compute $\mathbf{d}_i = \text{Com}(\mathbf{x}_i, \mathbf{s}_i) = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}_i) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}_i) \in \{0, 1\}^n$. Here $\{\mathbf{d}_i\}_{i=0}^{N-1}$ are all pairwise distinct. Set $R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$.
2. Run algorithm $\text{Accu}_{\mathbf{B}}(R)$ on input of R and iteratively hash them using the hash function $h_{\mathbf{B}}$ and publish the accumulated root value $\mathbf{u} \in \{0, 1\}^n$.
3. Let $\ell = \lceil \log N \rceil$. For each $i \in [0, N-1]$, let $(i_1, \dots, i_\ell)$ be the binary representation of i . Run algorithm $\text{WitGen}_{\mathbf{B}}(R, \mathbf{d}_j)$ in Section 4.5 and outputs a witness $w^{(i)}$ of $\mathbf{d}_i$ such that $\mathbf{d}_i$ is correctly computed along a path into the root $\mathbf{u}$.

$$w^{(i)} = ((i_1, \dots, i_\ell) \in \{0, 1\}^\ell, (\mathbf{w}_\ell^{(i)}, \dots, \mathbf{w}_1^{(i)}), \mathbf{w}_j^{(i)} \in \{0, 1\}^n),$$

where $\mathbf{w}_j^{(i)} \in \{0, 1\}^n$, for $j \in [\ell]$. Then define $\mathbf{gsk}[i] = (\mathbf{p}_i, \mathbf{d}_i, w^{(i)})$.

4. Define $\mathbf{grt}[i][j]$ as the revocation token of user i for time interval j , for all $i \in \{0, 1, \dots, N-1\}$, $j \in \{0, 1, \dots, T-1\}$. If a group user i revoked during interval j , then the group manager includes $\mathbf{grt}[i][j]$ to RL_j . Let $n = n(\lambda)$, $m = (n + k)/c \cdot 2^c$, $\epsilon > 0$ and $n \leq k - 2 \log(1/\epsilon) - \mathcal{O}(1)$.

(1) Let a deterministic function $H: \{0, 1, \dots, T-1\} \rightarrow \mathbb{Z}_2^{n \times m}; j \mapsto \mathbf{D}_j$, where $\mathbf{D}_j = [\mathbf{D}_{j0} \mid \mathbf{D}_{j1}] \in \mathbb{Z}_2^{n \times m}$ with $\mathbf{D}_{j0} \in \mathbb{Z}_2^{n \times n/c \cdot 2^c}$, $\mathbf{D}_{j1} \in \mathbb{Z}_2^{n \times k/c \cdot 2^c}$ and act as a random oracle on input a time period j and outputs a deterministic matrix $\mathbf{D}_j$.

(2) Define a statistically hiding function $h_{\mathbf{D}_j}: \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$ as

$$h_{\mathbf{D}_j}(\mathbf{x}_i, \mathbf{s}_i) = \mathbf{D}_{j0} \cdot \text{RE}(\mathbf{x}_i) \oplus \mathbf{D}_{j1} \cdot \text{RE}(\mathbf{s}_i) \in \{0, 1\}^n.$$

(3) Set $\mathbf{r}_{ij} = h_{\mathbf{D}_j}(\mathbf{x}_i, \mathbf{s}_i)$, and let the revocation token $\mathbf{grt}[i][j] = \mathbf{r}_{ij}$.

5. Outputs $\mathbf{gpk} := \{\mathbf{B}, \mathbf{C}, \mathbf{u}\}$; $\mathbf{gsk} := (\mathbf{gsk}[0], \dots, \mathbf{gsk}[N-1])$; and

$$\mathbf{grt} := (\mathbf{grt}[0][0], \dots, \mathbf{grt}[0][T-1], \dots, \mathbf{grt}[N-1][T-1]).$$

Sign_{vlr}($\mathbf{gpk}, \mathbf{gsk}[i], j, M$): On input a message $M \in \{0, 1\}^*$, j , group secret key $\mathbf{gsk}[i] = (\mathbf{p}_i, \mathbf{d}_i, w^{(i)})$, where $w^{(i)} = ((i_1, \dots, i_\ell), (\mathbf{w}_\ell^{(i)}, \dots, \mathbf{w}_1^{(i)}))$. Let $\mathcal{H}_{\text{FS}}: \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ and function $\mathcal{G}: \{0, 1\}^* \rightarrow \mathbb{Z}_2^{m \times n}$ be random oracles. The user performs as follows :

1. Sample a string $\rho \xleftarrow{\$} \{0, 1\}^n$ and run the random oracle $\mathcal{G}$ on inputs of $\mathbf{u}, \mathbf{B}, \mathbf{C}, M, \rho$ to obtain a matrix $\mathbf{A} = \mathcal{G}(\mathbf{u}, \mathbf{B}, \mathbf{C}, M, \rho)$.
2. Sample $\mathbf{e} \xleftarrow{\$} \text{Ber}(m, t)$, then compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{grt}[i][j] + \mathbf{e} = \mathbf{A} \cdot \mathbf{r}_{ij} + \mathbf{e}$.
3. Generate a ZKAoK Π to show a valid secret signing key $\tau = (\mathbf{p}_i, \mathbf{d}_i, w^{(i)}, \mathbf{e})$ satisfy the following commitment relation, accumulator relation and LPN relation.

(a)

$$\mathbf{p}_i = \begin{pmatrix} \mathbf{x}_i \\ \mathbf{s}_i \end{pmatrix} \text{ and } w^{(i)} = ((j_1, \dots, j_\ell), (\mathbf{w}_\ell^{(j)}, \dots, \mathbf{w}_1^{(j)})),$$

such that: $\text{Com}(\mathbf{x}_i, \mathbf{s}_i) = \mathbf{d}_i$ and $\text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}_i, w^{(i)}) = 1$.

(b) $\mathbf{r}_{ij} = \mathbf{D}_{j0} \cdot \text{RE}(\mathbf{x}_i) \oplus \mathbf{D}_{j1} \cdot \text{RE}(\mathbf{s}_i) \in \{0, 1\}^n$ and $\mathbf{b} = \mathbf{A} \cdot \mathbf{r}_{ij} + \mathbf{e}$, where $\mathbf{e} \xleftarrow{\$} \text{Ber}(m, t)$.

The underlying ZKAoK follows the abstraction of Stern-like protocol on public input $(\mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}, \mathbf{b})$ and prover's witness τ , the soundness error can be negligibly small by repeating the protocol κ times and the protocol become non-interactive by applying Fiat-Shamir paradigm by computing challenge CH via a random oracle. $\Pi = (\{\text{CMT}_i\}_{i=1}^\kappa, \text{CH}, \{\text{RSP}_i\}_{i=1}^\kappa)$, where CH is computed from a random oracle $\mathcal{H}_{\text{FS}}$.

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

4. Outputs Σ to be the ZAKoK together with the LPN value $\mathbf{b}$, $\Sigma = (M, \rho, \Pi, \mathbf{b})$.

Verify_{vlr}($\text{gpk}, j, M, RL_j, \Sigma$) : Take $\text{gpk} = (\mathbf{B}, \mathbf{C}, \mathbf{u})$, time period j , $RL_j = \{\mathbf{t}_{ij}\} \subseteq \{0, 1\}^n$, M and Σ . The verification algorithm is conducted in the following 4 steps:

1. Parse Σ as $(M, \rho, \Pi, \mathbf{b})$ and compute $\mathbf{A} = \mathcal{G}(\mathbf{B}, \mathbf{C}, \mathbf{u}, M, \rho) \in \mathbb{Z}_2^{m \times n}$.

If $(\text{Ch}_1, \dots, \text{Ch}_\kappa) = \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}, \mathbf{b})$, then return *invalid*.

2. For each iteration $i = 1$ to κ , check RSP_i with public input $(\mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}, \mathbf{b})$. Returns *invalid* if any of the conditions does not hold.

3. For each $\mathbf{t}_{ij} \in RL_j$ compute vector $\mathbf{e}'_{ij} = \mathbf{b} - \mathbf{A} \cdot \mathbf{t}_{ij}$. Returns *invalid* if there is an index value $i \in \{0, \dots, N-1\}$ such that $\mathbf{e}'_{ij} \in \text{Ber}(m, t)$.

4. Return *valid*.

Remarks We have some observations on the above algorithms:

1. Signature are tied to the revocation token which is related to the time period that they were issued. By leftover hash lemma, one can easily check that the distribution of $\mathbf{r}_{ij}$ is statistically close to uniform over $\mathbb{Z}_2^n$. At time period $j_1 < j_2$, we have $\mathbf{r}_{ij_1}$ is not linkable to $\mathbf{r}_{ij_2}$ since $\mathbf{u}_{ij_1}$ is generated from $h_{\mathbf{D}_{j_1}}$ and $\mathbf{u}_{ij_2}$ is generated from $h_{\mathbf{D}_{j_2}}$. In other words, it keep the older signatures unlinkable and does not affect the anonymity of the signature signed before the user revoked.
2. For a given time interval j' , the revocation tokens for any two users are distinct. Otherwise there exist $i_1, i_2 \in \{0, 1, \dots, N-1\}$ such that $i_2 > i_1$ and $\text{grt}[i_1][j'] = \text{grt}[i_2][j']$, so $\mathbf{p}_1 = \mathbf{p}_2$. In this case, the algorithm re-samples $\mathbf{p}_i$.

7.4 The Supporting Zero-Knowledge Argument System

We now proceed to analyse the ZKAoK for our scheme constructed in Section 7.3 and the associated relation R_{vtr} is given below.

$$R_{vtr} = \left\{ \left((\mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}, \mathbf{b}); (\mathbf{d}, w, \mathbf{p}, \mathbf{e}) \right) : \text{TVerify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \wedge \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \right. \\ \left. \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{d} \wedge \mathbf{D}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{D}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{r} \wedge \mathbf{b} = \mathbf{A} \cdot \mathbf{r} + \mathbf{e} \right\}$$

Now let us look at relation R_{vtr} . Given $\mathbf{B}$ uniformly random chosen in $\mathbb{Z}_2^{n \times q}$, $\mathbf{C} \leftarrow \mathbb{Z}_2^{n \times m}$, $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$, $\mathbf{u} \in \{0, 1\}^n$, $\mathbf{b} \in \mathbb{Z}_2^m$, a matrix $(\mathbf{D}_0 \parallel \mathbf{D}_1) = \mathbf{D} \in \mathbb{Z}_2^{n \times m}$ from a deterministic function H on input the time interval, the prover $\mathcal{P}$ is aiming to convince $\mathcal{V}$ about his possession of all secrets $(\mathbf{d}, w, \mathbf{p}, \mathbf{e})$ such that it is valid in the accumulator verification algorithm Verify and the signer's revocation token $\mathbf{u}$ is correctly computed through his secret and committed via the hash function. We

define R_{vtr} as follows:

$$R_{vtr} = \left\{ \left((\mathbf{B}, \mathbf{C}, \mathbf{u}) \in \mathbb{Z}_2^{n \times q} \times \mathbb{Z}_2^{n \times m} \times \{0, 1\}^n; w \in \{0, 1\}^\ell \times (\{0, 1\}^n)^\ell, \mathbf{d} \in \{0, 1\}^n : \right. \right. \\ \left. \text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \wedge \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{d} \wedge \mathbf{D}_0 \cdot \text{RE}(\mathbf{x}) \right. \\ \left. \oplus \mathbf{D}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{r} \wedge \mathbf{b} = \mathbf{A} \cdot \mathbf{r} + \mathbf{e} \right\}.$$

We follow the extension and permutation techniques in Section 4.6 and Section 6.5. Now let us look the the relation R_{vtr} , which contain the relation in accumulator of proving a path with nodes $\mathbf{v}_k$ computed as follow.

$$\mathbf{v}_k = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{k+1}, \mathbf{w}_{k+1}), & \text{if } i_{k+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{k+1}, \mathbf{v}_{k+1}), & \text{if } i_{k+1} = 1. \end{cases} \quad (7.1)$$

Rewrite equation (7.1) into the following equivalent form.

$$\begin{aligned} \mathbf{v}_k &= \bar{i}_{k+1} \cdot h_{\mathbf{B}}(\mathbf{v}_{k+1}, \mathbf{w}_{k+1}) \oplus i_{k+1} \cdot h_{\mathbf{B}}(\mathbf{w}_{k+1}, \mathbf{v}_{k+1}) \\ &= \bar{i}_{k+1} \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{v}_{k+1}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{w}_{k+1})) \oplus i_{k+1} \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{w}_{k+1}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{v}_{k+1})) \\ &= \mathbf{B} \cdot \begin{pmatrix} \bar{i}_{k+1} \cdot \text{RE}(\mathbf{v}_{k+1}) \\ i_{k+1} \cdot \text{RE}(\mathbf{v}_{k+1}) \end{pmatrix} \oplus \mathbf{B} \cdot \begin{pmatrix} i_{k+1} \cdot \text{RE}(\mathbf{w}_{k+1}) \\ \bar{i}_{k+1} \cdot \text{RE}(\mathbf{w}_{k+1}) \end{pmatrix} \\ &= \mathbf{B} \cdot \text{Ext}(i_{k+1}, \text{RE}(\mathbf{v}_{k+1})) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_{k+1}, \text{RE}(\mathbf{w}_{k+1})) \end{aligned}$$

Therefore, $\mathcal{P}$ is able to prove in ZKAoK to make the verifier $\mathcal{V}$ be convinced about his possession of all information $i_1, \dots, i_\ell \in \{0, 1\}^\ell$, $\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{w}_1, \dots, \mathbf{w}_\ell \in \{0, 1\}^n$ that can be used to rebuild a hash chain from the tree leave and the revocation token $\mathbf{r}$ was correctly computed and committed. The equation is as follows:

$$\left\{ \begin{array}{l} \mathbf{B} \cdot \text{Ext}(i_1, \text{RE}(\mathbf{v}_1)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_1, \text{RE}(\mathbf{w}_1)) = \mathbf{u}; \\ \mathbf{B} \cdot \text{Ext}(i_2, \text{RE}(\mathbf{v}_2)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_2, \text{RE}(\mathbf{w}_2)) \oplus \mathbf{v}_1 = \mathbf{0}. \\ \dots\dots\dots \\ \mathbf{B} \cdot \text{Ext}(i_\ell, \text{RE}(\mathbf{v}_\ell)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_\ell, \text{RE}(\mathbf{w}_\ell)) \oplus \mathbf{v}_{\ell-1} = \mathbf{0} \\ \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) \oplus \mathbf{v}_\ell = \mathbf{0} \\ \mathbf{D}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{D}_1 \cdot \text{RE}(\mathbf{s}) \oplus \mathbf{r} = \mathbf{0} \\ \mathbf{b} = \mathbf{A} \cdot \mathbf{r} + \mathbf{e} \end{array} \right. \quad (7.2)$$

Next, we apply extension technique for vectors **Encode** which is defined in Section 4.6 to vectors $\mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1$ to obtain $\mathbf{f}_k = \text{Encode}(\mathbf{v}_k) \in \{0, 1\}^{2n}$ for $k \in [\ell - 1]$, and it can be rewritten as $\mathbf{v}_k = \mathbf{I}_n^* \cdot \mathbf{f}_k$ for $k \in [\ell - 1]$.

For $\mathbf{b} = \mathbf{A} \cdot \mathbf{u} + \mathbf{e} = \mathbf{b} = \mathbf{A} \cdot (\mathbf{D}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{D}_1 \cdot \text{RE}(\mathbf{s})) + \mathbf{e}$. Let $\mathbf{A}_0^* = \mathbf{A} \cdot \mathbf{D}_0 \in \mathbb{Z}_2^{m \times (2^c \cdot n/c)}$ and $\mathbf{A}_1^* = \mathbf{A} \cdot \mathbf{D}_1 \in \mathbb{Z}_2^{m \times (2^c \cdot k/c)}$. Then $\mathbf{b} = \mathbf{A}_0^* \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{A}_1^* \cdot \text{RE}(\mathbf{s}) + \mathbf{e}$. For ease of notation, $q = 2^c \cdot 2n/c$, for $k \in [\ell - 1]$, denote

$$\left\{ \begin{array}{l} \mathbf{y}_k = \text{Ext}(i_k, \text{RE}(\mathbf{v}_k)) \in \{0, 1\}^q, k \in [\ell] \\ \mathbf{z}_k = \text{Ext}(\bar{i}_k, \text{RE}(\mathbf{w}_k)) \in \{0, 1\}^q, k \in [\ell] \\ \mathbf{f}_k = \text{Encode}(\mathbf{v}_k) \in \{0, 1\}^{2n}, k \in [\ell - 1] \end{array} \right. \quad (7.3)$$

From the equation 7.3, we simply denote $\mathbf{y}_k$ as $\text{Ext}(i_k, \text{RE}(\mathbf{v}_k))$, denote $\mathbf{z}_k$ to be $\text{Ext}(\bar{i}_k, \text{RE}(\mathbf{w}_k))$ and $\mathbf{f}_k$ to be the extension of the vector $\mathbf{v}_k$.

Therefore, the equations in (7.2) is converted into:

$$\left\{ \begin{array}{l} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u}; \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{f}_1 = \mathbf{0}. \\ \dots\dots\dots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{f}_{\ell-1} = \mathbf{0}. \\ \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) \oplus \mathbf{I}_n^* \cdot \mathbf{f}_\ell = \mathbf{0} \\ \mathbf{b} = \mathbf{A}_0^* \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{A}_1^* \cdot \text{RE}(\mathbf{s}) + \mathbf{e} \end{array} \right. \quad (7.4)$$

Equations in (7.4) can be unified into the form of $\mathbf{M}_{vlr} \cdot \mathbf{w}_{vlr} = \mathbf{v}$ applying the basic algebra, where $\mathbf{M}_{vlr} \in \mathbb{Z}_2^{\binom{(\ell+1)n+m}{\times L}}$, $\mathbf{v} \in \mathbb{Z}_2^{(\ell+1)n+m}$ are public and $\mathbf{w}_{vlr} \in \{0, 1\}^L$ is the secret with length $L = 2\ell q + 2\ell n + 2m$ and in the form of

$$\mathbf{w}_{vlr} = (\mathbf{y}_1 \| \dots \| \mathbf{y}_\ell \| \mathbf{z}_1 \| \dots \| \mathbf{z}_\ell \| \mathbf{f}_1 \| \dots \| \mathbf{f}_\ell \| \text{RE}(\mathbf{x}) \| \text{RE}(\mathbf{s}) \| \mathbf{e})$$

We define a set **VALID** that contains all vectors in the same form as $\mathbf{w}_{vlr}$, a set $\mathcal{S}$ and the permutations $\{\Gamma_\phi : \phi \in \mathcal{S}\}$ that satisfy the conditions in (2.2). Let **VALID** be a set in the form of

$$\mathbf{w}' = (\mathbf{y}'_1 \| \dots \| \mathbf{y}'_\ell \| \mathbf{z}'_1 \| \dots \| \mathbf{z}'_\ell \| \mathbf{f}'_1 \| \dots \| \mathbf{f}'_\ell \| \mathbf{q}'_0 \| \mathbf{q}'_1 \| \mathbf{e}')$$

and it satisfy:

- For $k \in [\ell]$, there exists $\mathbf{v}'_k, \mathbf{w}'_k \in \{0, 1\}^n$, $i'_k \in \{0, 1\}$ such that

$$\mathbf{y}'_k = \text{Ext}(i'_k, \text{RE}(\mathbf{v}'_k)) \in \{0, 1\}^q, \quad \text{and} \quad \mathbf{z}'_k = \text{Ext}(\bar{i}'_k, \text{RE}(\mathbf{w}'_k)) \in \{0, 1\}^q.$$

- For $k \in [\ell]$, $\mathbf{f}'_k = \text{Encode}(\mathbf{v}'_k) \in \{0, 1\}^{2n}$.

- For $\mathbf{x}' \in \{0, 1\}^n$, $\mathbf{q}'_0 = \text{RE}(\mathbf{x}') \in \{0, 1\}^{2^{c \cdot n/c}}$;
 $\mathbf{s}' \in \{0, 1\}^k$, $\mathbf{q}'_1 = \text{RE}(\mathbf{s}') \in \{0, 1\}^{2^{c \cdot k/c}}$.
- $\mathbf{e}' \in \text{Ber}(m, t)$.

Let $\mathcal{S}$ be of the following form:

$$\mathcal{S} = \left(\{0, 1\}^n\right)^\ell \times \left(\{0, 1\}^n\right)^\ell \times \{0, 1\}^\ell \times \{0, 1\}^n \times \{0, 1\}^k \times S_m$$

Then, for each

$\phi = (\mathbf{b}_1, \dots, \mathbf{b}_\ell, \mathbf{e}_1, \dots, \mathbf{e}_\ell, g_1, \dots, g_\ell, \mathbf{k}_0, \mathbf{k}_1, \sigma) \in \mathcal{S}$, define the permutation Γ_ϕ that transforms

$$\mathbf{w}^* = (\mathbf{y}_1^* \parallel \dots \parallel \mathbf{y}_\ell^* \parallel \mathbf{z}_1^* \parallel \dots \parallel \mathbf{z}_\ell^* \parallel \mathbf{f}_1^* \parallel \dots \parallel \mathbf{f}_\ell^* \parallel \mathbf{q}_0^* \parallel \mathbf{q}_1^* \parallel \mathbf{e}^*) \in \{0, 1\}^L$$

with $\mathbf{y}_k^*, \mathbf{z}_k^* \in \{0, 1\}^m, \mathbf{f}_k^* \in \{0, 1\}^{2n}$ for $k \in [\ell]$, $\mathbf{q}_i^* \in \{0, 1\}^{m/2}$ for $i \in \mathbb{Z}_2$, and $\mathbf{e}^* \in \text{Ber}(m, t)$ to

$$\begin{aligned} \Gamma_\phi(\mathbf{w}^*) = & \left(\Psi_{g_1, \mathbf{b}_1}(\mathbf{y}_1^*) \parallel \dots \parallel \Psi_{g_\ell, \mathbf{b}_\ell}(\mathbf{y}_\ell^*) \parallel \Psi_{\bar{g}_1, \mathbf{e}_1}(\mathbf{z}_1^*) \parallel \dots \parallel \Psi_{\bar{g}_\ell, \mathbf{e}_\ell}(\mathbf{z}_\ell^*) \parallel F'_{\mathbf{b}_1}(\mathbf{f}_1^*) \parallel \right. \\ & \left. \dots \parallel F'_{\mathbf{b}_\ell}(\mathbf{f}_\ell^*) \parallel E'_{\mathbf{k}_0}(\mathbf{q}_0^*) \parallel E'_{\mathbf{k}_1}(\mathbf{q}_1^*) \parallel \sigma(\mathbf{e}^*) \right). \end{aligned}$$

Equations (7.2) and (7.4) show that the conditions in (2.2) are all satisfied. Thus our relation R_{vlr} can be reduced in an form of abstraction relation R and we can obtain a ZKAoK follow the protocol in Figure 2.1. The communication cost for this protocol is $\mathcal{O}(L) = \mathcal{O}(2\ell q + 2\ell n + 2m)$ bits.

Our scheme satisfies two security requirements and the detailed proofs for these two requirements, BU anonymity and traceability, are given in section below.

7.5 Analysis of the Scheme

We summarize the properties of the our scheme in the following theorems.

First, we give a lemma that will be applied in our later proof.

Lemma 7.5.1. Let $m = m(\lambda), n = n(\lambda), t = t(\lambda), \mathbf{A} \in \mathbb{Z}_2^{m \times n}$. Given a matrix $\mathbf{A}$, there is a negligible probability to find a vector $\mathbf{v}$ such that $\mathbf{A} \cdot \mathbf{v}$ is in Bernoulli distribution.

$$\Pr[\exists \text{ non-zero } \mathbf{v} \in \{0, 1\}^n : \mathbf{A} \cdot \mathbf{v} \in \text{Ber}(m, t)] \leq \text{negl}(\lambda).$$

Proof. Given $\mathbf{A} \cdot \mathbf{v} \in \text{Ber}(m, t)$, the probability of finding a non-zero vector $\mathbf{v} \in \{0, 1\}^n$ such that $\mathbf{A} \cdot \mathbf{v} \in \text{Ber}(m, t)$ is $\binom{m}{t} (t/m)^t [(m-t)/m]^{(m-t)}$, which is negligible. $\square$

Theorem 7.5.2. The VLR code-based group signature with backward unlinkability is correct with overwhelming probability.

Proof. Correctness requires that the signature signed by a unrevoked user should be verified to be valid with overwhelming probability.

Let λ be the security parameter, for all $\lambda, N, T \in \mathbb{N}$, all $(\text{gpk}, \text{gsk}, \text{grt})$ generated by $\text{KGen}_{\text{vtr}}(\lambda, N, T)$, for any $i \in [0, N-1]$, any $j \in [0, T-1]$ and message $M \in \{0, 1\}^*$. By the definition of correctness, we need to prove the following hold :

$$\text{Verify}_{\text{vtr}}(\text{gpk}, j, M, RL_j, \text{Sign}_{\text{vtr}}(\text{gpk}, \text{gsk}[i], j, M)) = \text{valid} \Leftrightarrow \text{grt}[i][j] \notin RL_j. \quad (7.5)$$

From the verification algorithm, we have for each $\mathbf{t}_{ij} \in RL_j$ compute vector $\mathbf{e}'_{ij} = \mathbf{b} - \mathbf{A} \cdot \mathbf{t}_{ij}$. and $\mathbf{b}$ is computed in the signing algorithm as $\mathbf{b} = \mathbf{A} \cdot \text{grt}[i][j] + \mathbf{e}$. Then we can easily compute the following.

$$\mathbf{e}'_{ij} = \mathbf{b} - \mathbf{A} \cdot \mathbf{t}_{ij} = \mathbf{A} \cdot \text{grt}[i][j] + \mathbf{e} - \mathbf{A} \cdot \mathbf{t}_{ij} = \mathbf{A} \cdot (\text{grt}[i][j] - \mathbf{t}_{ij}) + \mathbf{e}$$

The equation (7.5) can be transformed into:

$$\mathbf{e}'_{ij} \notin \text{Ber}(m, t) \Leftrightarrow \text{grt}[i][j] \notin RL_j. \quad (7.6)$$

We will consider in two directions for proving the sufficient condition and necessary condition. On one hand, we will show that

$$\mathbf{e}'_{ij} \notin \text{Ber}(m, t) \Rightarrow \text{grt}[i][j] \notin RL_j.$$

We prove it by contrapositive. Suppose that $\text{grt}[i][j] \in RL_j$, then there exists a $\mathbf{t}_{ij} \in RL_j$ such that $\text{grt}[i][j] = \mathbf{t}_{ij}$, we deduce that $\mathbf{e}'_{ij} = \mathbf{A} \cdot (\text{grt}[i][j] - \mathbf{t}_{ij}) + \mathbf{e} = \mathbf{e} \in \text{Ber}(m, t)$.

On the other hand, we need to prove the other direction.

$$\mathbf{e}'_{ij} \notin \text{Ber}(m, t) \Leftarrow \text{grt}[i][j] \notin RL_j.$$

Since $\text{grt}[i][j] \notin RL_j$, we have $\text{grt}[i][j] - \mathbf{t}_{ij}$ is non-zero. Then $\mathbf{A} \cdot (\text{grt}[i][j] - \mathbf{t}_{ij}) \notin \text{Ber}(m, t)$ with overwhelming probability from Lemma 7.5.1. Hence, $\mathbf{e}'_{ij} = \mathbf{A} \cdot (\text{grt}[i][j] - \mathbf{t}_{ij}) + \mathbf{e} \notin \text{Ber}(m, t)$

□

Theorem 7.5.3. The scheme achieves traceability in the random oracle model, under the hardness of $2\text{-RNSD}_{n,n+k,c}$ problem.

Proof. The proof is done by contraposition. Assuming that an adversary $\mathcal{A}$ succeeds in breaking traceability with non-negligible probability, then there exists a PPT algorithm $\mathcal{B}$ that is capable of solving a $2\text{-RNSD}_{n,n+k,c}$ problem with non-negligible probability. In other words, breaking traceability leads to obtain an solution to the hard $2\text{-RNSD}_{n,n+k,c}$ problem.

$\mathcal{B}$ faithfully runs the KGen_{vtr} algorithm to obtain gpk , gsk and grt , and defines an

initially empty corruption list set CL and revocation token list RL . Then it provides $\mathcal{A}$ with $\mathbf{gpk}$ and $\mathbf{grt}$, while $\mathcal{B}$ keeps all the users' secret keys $\mathbf{gsk}$ to itself, it enables him to act as the corruption and signing oracles to answer all the queries made from $\mathcal{A}$. At period j , $\mathcal{B}$ outputs the secret key $\mathbf{gsk}[i]$ when $\mathcal{A}$ request the corruption oracle for user i . $\mathbf{gsk}[i]$ is added to the set CL and the revocation token for user i for the onwards period j' are added to the $RL_{j'}$. $\mathcal{B}$ computes the signature Σ for a given user i , a period j and M using **Sign** algorithm and returns it to $\mathcal{A}$.

When $\mathcal{A}$ halts and eventually outputs $(M^*, \Sigma^*, j^*, RL_{j^*}^*)$ such that the verification algorithm $\text{Verify}_{\text{vtr}}(\mathbf{gpk}, M^*, j^*, RL^*, \Sigma^*)$ outputs valid and the implicit tracing algorithm output valid for all users or traces to an uncorrupted user during period j^* . Parse Σ^* as $(M^*, \rho^*, \Pi^*, \mathbf{b}^*)$ and Π^* as $(\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}^*\}_{i=1}^\kappa)$. Let matrix $\mathbf{A}^* = \mathcal{G}(\mathbf{B}, \mathbf{C}, \mathbf{u}, M^*, \rho^*) \in \mathbb{Z}_2^{m \times n}$. Note that the oracle $\mathcal{H}_{\text{FS}}(\cdot)$ maps any elements to an element uniformly random chosen in set $\{1, 2, 3\}^\kappa$ and it always output the same value even though making many queries on same input.

Next, we want to prove that $\mathcal{A}$ made queries to the random oracle $\mathcal{H}_{\text{FS}}$ on input $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}^*, \mathbf{b}^*)$ with overwhelming probability in order to obtain a challenge

$$\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}^*, \mathbf{b}^*) \in \{1, 2, 3\}.$$

Prove by contradiction in this case on if he tries to guess the value, then the probability guessing it exactly the same as the output of the random oracle would be at most $3^{-\kappa}$. Therefore, $\mathcal{A}$ had queried $\mathcal{H}_{\text{FS}}$ at his t^* -th query among the in total Q_H queries he made.

Algorithm $\mathcal{B}$ would run extra executions of the adversary $\mathcal{A}$ for at most $32 \cdot Q_H / (\epsilon - 3^{-\kappa})$ times, since he will not be able to receive different answers until his t^* -th query made. From the t^* -th query on, $\mathcal{B}$ output uniformly random values for each new query and send them to $\mathcal{A}$, besides, all the output values are independent. As we can see, the output for t^* -th query will be uniformly random

and independent while the input is still the same as in the initial run. By following the Forking Lemma of Brickell et al. [30], $\mathcal{B}$ can obtain a 3-fork with the same input of $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}, \mathbf{A}^*, \mathbf{b}^*)$ and output three pairwise distinct hash values $\text{CH}_{t^*}^{(1)}, \text{CH}_{t^*}^{(2)}, \text{CH}_{t^*}^{(3)} \in \{1, 2, 3\}^\kappa$, as well as their corresponding responses $\text{RSP}_{t^*}^{(1)}, \text{RSP}_{t^*}^{(2)}, \text{RSP}_{t^*}^{(3)}$ with probability at least $1/2$. Moreover, the results of [30] imply that it can be achieved to get $\{\text{CH}_{t^*,k}^{(1)}, \text{CH}_{t^*,k}^{(2)}, \text{CH}_{t^*,k}^{(3)}\} = \{1, 2, 3\}$ with probability $1 - (7/9)^\kappa$ at the $k \in \{1, \dots, \kappa\}$ run.

From 3 valid responses $(\text{RSP}_{t^*,k}^{(1)}, \text{RSP}_{t^*,k}^{(2)}, \text{RSP}_{t^*,k}^{(3)})$ w.r.t. the same commitment CMT_k^* and all challenges 1, 2, 3, $\mathcal{B}$ is able to extract witnesses during period j^* since the conditions in Theorem 2.3.13 are fulfilled. The witnesses he extract is as follows

$$(\mathbf{p}^*, \mathbf{d}^*, w^*, \mathbf{e}_{j^*}^*),$$

where $\mathbf{p}$ in the form of $\mathbf{p}^* = \begin{pmatrix} \mathbf{x}^* \\ \mathbf{s}^* \end{pmatrix} \in \{0, 1\}^{n+k}$, $w^* = ((i_1^*, \dots, i_\ell^*), (\mathbf{w}_\ell^*, \dots, \mathbf{w}_1^*))$ with $(i_1^*, \dots, i_\ell^*) \in \{0, 1\}^\ell$ be the binary representation of an integer $i^* \in [0, N - 1]$ and vector $\mathbf{e}^*$ in $\text{Ber}(m, t)$ distribution. We have:

$$\text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}^*, w^*) = \text{valid}, \quad (7.7)$$

$$\mathbf{C}_0 \cdot \text{RE}(\mathbf{x}^*) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}^*) = \mathbf{d}^*, \quad (7.8)$$

$$\mathbf{D}_{j^*0} \cdot \text{RE}(\mathbf{x}^*) \oplus \mathbf{D}_{j^*1} \cdot \text{RE}(\mathbf{s}^*) = \mathbf{r}_{j^*}^*, \quad (7.9)$$

$$\mathbf{b}^* = \mathbf{A}^* \cdot \mathbf{r}_{j^*}^* + \mathbf{e}_{j^*}^*. \quad (7.10)$$

Since $\mathcal{A}$ wins the game during period j^* , we can derive that two cases of either (i)

$\mathbf{d}^* \notin R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$ or (ii) $\mathbf{d}^* \in R = (\mathbf{d}_0, \dots, \mathbf{d}_{N-1})$ and $\mathbf{d}^* = \mathbf{d}_{i^*}$. Note that the KGen_{vtr} algorithm ensure that $\{\mathbf{d}_i\}_{i=0}^{N-1}$ are pairwise distinct.

Case (i) implies a violation of the security of the underlying accumulator.

Case (ii) implies a violation of the computationally binding property of commitment. We have $\mathbf{d}^* = \mathbf{d}_{i^*}$, thus,

$$\mathbf{C}_0 \cdot \text{RE}(\mathbf{x}^*) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}^*) = \mathbf{d}^* = \mathbf{d}_{i^*} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}_{i^*}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}_{i^*}). \quad (7.11)$$

We will prove that $(\mathbf{x}^*, \mathbf{s}^*) \neq (\mathbf{x}_{i^*}, \mathbf{s}_{i^*})$ with overwhelming probability. Thus we obtain the solution to the 2-RNSD problem. Since Σ^* is a valid signature such that the implicit tracing algorithm fails or traces to a user y^* in set $CL_{j^*}/RL_{j^*}^*$ during period j^* .

- If the implicit tracing algorithm fails, then the verification algorithm output valid for every revocation token, $\text{Verify}_{vtr}(\text{gpk}, M^*, j^*, \text{grt}[i^*][j^*], \Sigma^*) = \text{valid}$ which means it fails to identity one of the group users $\{0, 1, \dots, N-1\}$. So $\mathbf{r}_{j^*}^* \neq \text{grt}[i^*][j^*] = \mathbf{r}_{i^*j^*}^*$. Since we will get a contradiction if achieves equality, vector $\mathbf{e}_{j^*}' = \mathbf{b}^* - \mathbf{A}^* \cdot \text{grt}[i^*][j^*] = \mathbf{b}^* - \mathbf{A}^* \cdot \mathbf{r}_{j^*}^* = \mathbf{e}_{j^*}^* \in \text{Ber}(m, t)$ and thus the algorithm $\text{Verify}_{vtr}(\text{gpk}, M^*, j^*, \text{grt}[i^*][j^*], \Sigma^*)$ would output *invalid*, which leads to an contradiction. Thus, we deduce that $\mathbf{r}_{j^*}^* \neq \mathbf{r}_{i^*j^*}^*$. As a result, we have $\mathbf{p}^* \neq \mathbf{p}_{i^*}$, where $\mathbf{p}^* = (\mathbf{x}^*, \mathbf{s}^*)$ and $\mathbf{p}_{i^*} = (\mathbf{x}_{i^*}, \mathbf{s}_{i^*})$.
- If the tracing algorithm traces to a user y^* in $CL_{j^*}/RL_{j^*}^*$, that is, $y^* \notin CL_{j^*}/RL_{j^*}^*$, which means that the uncorrupted and revoked user y^* did not sign M^* at period j^* . Thus we have $\text{Verify}_{vtr}(\text{gpk}, M^*, j^*, \text{grt}[y^*][j^*], \Sigma^*) = \text{invalid}$ and the implicit tracing algorithm $\text{Verify}_{vtr}(\text{gpk}, M^*, j^*, RL_{j^*}^*, \Sigma^*) = \text{valid}$ and $\text{grt}[y^*][j^*] \notin RL_{j^*}^*$ and $y^* \notin CL_{j^*}$. Then we have $\mathbf{b}^* - \mathbf{A}^* \cdot \text{grt}[y^*][j^*] = \mathbf{A}^* \cdot (\mathbf{r}_{j^*}^* - \text{grt}[y^*][j^*]) + \mathbf{e}_{j^*}^* \in \text{Ber}(m, t)$, with the extra condition $\mathbf{e}_{j^*}^* \in \text{Ber}(m, t)$, we deduce that $\mathbf{r}_{j^*}^* = \text{grt}[y^*][j^*]$ by applying Lemma 7.5.1.

Then we need to consider that if $\mathcal{A}$ made a corruption query to $\text{gsk}[i^*]$:

(1) If $\mathcal{A}$ has never made queries to obtain $\mathbf{gsk}[i^*]$, then we have $\mathbf{p}^* \neq \mathbf{p}_{i^*}$ with overwhelming probability, here $\mathbf{p}_{i^*} = (\mathbf{x}_{i^*}, \mathbf{s}_{i^*})^T$ and is statistically close to uniform vector

(2) If $\mathcal{A}$ has queried and obtain $\mathbf{gsk}[i^*]$ during period j^* in the Query phase, then $i^* \in CL_{j^*}$. In this case, $y^* \neq i^*$ since $y^* \notin CL_{j^*}$. Then $\mathbf{grt}[y^*][j^*] \neq \mathbf{grt}[i^*][j^*]$, that is, $\mathbf{r}_{j^*}^* \neq \mathbf{r}_{i^*j^*}$. Thus, we have $\mathbf{p}^* \neq \mathbf{p}_{i^*}$, where $\mathbf{p}^* = (\mathbf{x}^*, \mathbf{s}^*)$ and $\mathbf{p}_{i^*} = (\mathbf{x}_{i^*}, \mathbf{s}_{i^*})$.

To summarize, we obtain $(\mathbf{x}^*, \mathbf{s}^*) \neq (\mathbf{x}_{i^*}, \mathbf{s}_{i^*})$ such that:

$$\mathbf{C}_0 \cdot \mathbf{RE}(\mathbf{x}^*) \oplus \mathbf{C}_1 \cdot \mathbf{RE}(\mathbf{s}^*) = \mathbf{d}^* = \mathbf{d}_{i^*} = \mathbf{C}_0 \cdot \mathbf{RE}(\mathbf{x}_{i^*}) \oplus \mathbf{C}_1 \cdot \mathbf{RE}(\mathbf{s}_{i^*}). \quad (7.12)$$

$$\mathbf{z} = (\mathbf{z}_1 \| \mathbf{z}_2) = (\mathbf{RE}(\mathbf{x}^*) \oplus \mathbf{RE}(\mathbf{x}_{i^*}), \mathbf{RE}(\mathbf{s}^*) \oplus \mathbf{RE}(\mathbf{s}_{i^*})) \neq \mathbf{0} \quad (7.13)$$

Then $\mathbf{C}_0 \cdot \mathbf{z}_1 \oplus \mathbf{C}_1 \cdot \mathbf{z}_2 = \mathbf{0}$. we have $\mathbf{C} \cdot \mathbf{z} = \mathbf{0}$, which means $\mathbf{z}$ is a solution to $2\text{-RNSD}_{n,n+k,c}$ problem,

Therefore, $\mathcal{B}$ is able to solve an instance of the $2\text{-RNSD}_{n,n+k,c}$ problem with a non-negligible probability to be at least $1/2 \cdot (\epsilon - 3^{-\kappa}) \cdot (1 - (7/9)^\kappa)$.

□

Theorem 7.5.4. Assume that the $\text{DLPN}(m, n, \text{Ber}(m, t))$ problems is hard and the argument system is simulation-sound, then the scheme is BU-anonymous.

Proof. This theorem is proved from a sequence of indistinguishable games. We start from a experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{anon-0}}(\lambda, N, T)$ to the indistinguishable last experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda, N, T)$. It then follows that our group signature scheme follows BU-anonymous. For each i , denote W_i as the event when $\mathcal{A}$ outputs 1 in Game i .

Here are all the games.

Game 0: 1. *Setup* The challenger $\mathcal{C}$ runs KGen_{vtr} to produce $(\mathbf{gpk}, \mathbf{gsk}, \mathbf{grt})$. $\mathcal{A}$ is only provided with $\mathbf{gpk}$. Set $CL = \emptyset$ and $RL = \emptyset$.

2. **Query** At the beginning of time period j , the $\mathcal{A}$ informs $\mathcal{C}$ to increment j simultaneously. During time interval j , $\mathcal{A}$ is accessible to the following oracles.

- $\mathcal{O}_{\text{Signing}}$: It returns $\text{Sign}_{v_{lr}}(\text{gpk}, \text{gsk}[i], j, M)$ if $\mathcal{A}$ takes any message M and an index $i \in \{0, \dots, N-1\}$ as input.
- $\mathcal{O}_{\text{Corruption}}$: If $\mathcal{A}$ on input $i \in \{0, \dots, N-1\}$, it returns $\text{gsk}[i]$ and $CL_j = CL \cup \{i\}$.
- $\mathcal{O}_{\text{Revocation}}$: It returns $\text{grt}[i][j]$ if the adversary $\mathcal{A}$ made queries on input $i \in \{0, \dots, N-1\}$ at period j . Set $RL_j = RL \cup \{\text{grt}[i][j]\}$.

3. **Challenge** At time interval j^* , $\mathcal{A}$ outputs M and two unrevoked and uncorrupted users $i_0, i_1 \in \{0, \dots, N-1\}$. $\mathcal{C}$ computes a signature Σ^* on behalf of user i_0 and returns to $\mathcal{A}$.

$$\Sigma = \text{Sign}_{v_{lr}}(\text{gpk}, \text{gsk}[i_0], j, M) = (M, \rho, \Pi, \mathbf{b}),$$

where $\mathbf{b} = \mathbf{A} \cdot \text{grt}[i_0][j^*] + \mathbf{e}_{i_0 j^*}$.

4. **RestrictQueries** $\mathcal{A}$ can make further queries as in phase 2 for time intervals after j^* except making queries on $\text{gsk}[i_0]$ and $\text{gsk}[i_1]$ and $\text{grt}[i_0][j^*]$, $\text{grt}[i_1][j^*]$.

5. **Output** Eventually, $\mathcal{A}$ outputs a bit c' .

Here $\Pr[W_0] = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{anon-0}}(\lambda, N, T) = 1]$.

Game 1: We change Game 0 by generating simulated NIZK argument Π^* in the third step of the signing algorithm in 7.3 instead of a real proof in the challenged phase. In other words, the signature changes to $\Sigma^* = (M, \rho, \Pi^*, \mathbf{b})$. Game 1 and Game 0 are statistically indistinguishable in view of the adversary, since the underlying ZKAoK is statistical zero-knowledge. Then we have $|\Pr[W_1] - \Pr[W_0]| \in \text{negl}(\lambda)$.

Game 2: In Game 1, we have $\mathbf{b} = \mathbf{A} \cdot \text{grt}[i_0][j^*] + \mathbf{e}_{i_0 j^*}$, where $\text{grt}[i_0][j^*] = \mathbf{D}_{j^* 0} \cdot \text{RE}(\mathbf{x}_{i_0}) \oplus \mathbf{D}_{j^* 1} \cdot \text{RE}(\mathbf{s}_{i_0})$. By Leftover Hash Lemma 3.4.2, $\text{grt}[i_0][j^*]$ is statistically

close to uniform over $\mathbb{Z}_2^n$. Then we modify $\mathbf{b}$ as $\mathbf{b} = \mathbf{A}^* \cdot \mathbf{c} + \mathbf{e}_{i_0 j^*}$, where $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_2^n$. It easily implies that $|\Pr[W_2] - \Pr[W_1]| \in \text{negl}(\lambda)$, since Game 1 and Game 2 are statistically close to each other.

Game 3: We introduce a small modification on $\mathbf{b}$ by sampling $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_2^m$ and set $\mathbf{b} = \mathbf{a}$. In Game 2, $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_2^n$; matrix $\mathbf{A}$ is uniformly random over $\mathbb{Z}_2^{m \times n}$; vector $\mathbf{e}_{i_0 j^*} \leftarrow \text{Ber}(m, t)$. We have $(\mathbf{A}, \mathbf{b})$ is a proper $\text{DLPN}(m, n, \text{Ber}(m, t))$ instance, thus the distribution is computationally close to the uniform distribution over $\mathbb{Z}_2^{m \times n} \times \mathbb{Z}_2^m$. With computationally indistinguishability of Game 2 and Game 3, we have $|\Pr[W_3] - \Pr[W_2]| \in \text{negl}(\lambda)$.

Game 4: We compute $\mathbf{b} = \mathbf{A} \cdot \text{grt}[i_1][j^*] + \mathbf{e}_{i_1 j^*}$ since $\text{grt}[i_1][j^*]$ is statistically close to uniform over $\mathbb{Z}_2^n$ and $\mathbf{e}_{i_1 j^*} \leftarrow \text{Ber}(m, t)$. Game 4 and Game 3 are computationally indistinguishable, under the hardness of problem $\text{DLPN}(m, n, \text{Ber}(m, t))$. Thus $|\Pr[W_4] - \Pr[W_3]| \in \text{negl}(\lambda)$.

Game 5: We modify Game 4 by generating a real NIZK Π in the challenged phase instead a simulated one Π^* . Thus obtain a real signature $\Sigma = \text{Sign}_{\text{vtr}}(\text{gpk}, \text{gsk}[i_1], j^*, M) = (M, \rho, \Pi, \mathbf{b})$. The distribution of Π is statistically close to Π^* since its a statistically zero-knowledge argument. We have $|\Pr[W_5] - \Pr[W_4]| \in \text{negl}(\lambda)$. This is in fact experiment $\text{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda, N, T)$, hence $\Pr[W_5] = \Pr[\text{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda, N, T) = 1]$.

In conclusion, we have

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

this concludes the proof. □

7.6 Decentralized E-cash Scheme

In this section, we present the syntax of the decentralized e-cash scheme in Section 7.6.1, then we present our code-based decentralized e-cash scheme in Section 7.6.2 by employing on the code-based accumulator in Section 4.5 and zero-knowledge proof. We adapt the idea in Section 7.3 to the e-cash syntax to obtain a code-based decentralized e-cash scheme.

7.6.1 Syntax of the Decentralized E-cash Scheme

We present the definition and security requirements for a decentralized e-cash scheme of Zerocoin in [95].

Setup(λ) : Take security parameter λ , output public parameters pp and a set $\mathcal{C}$.

Mint(pp) : Take pp , the randomized algorithm outputs a coin $c \in \mathcal{C}$ and a trapdoor sk .

Spend(pp, c, sk, M, R) : On input pp , a coin c and its trapdoor sk , some transaction string M and an coin set R , output a coin spend transaction (Π, x) , where Π is a proof and x is a serial number, if the coin $c \in R \subseteq \mathcal{C}$. Otherwise output 0.

E-Verify(pp, Π, x, M, R) : On input pp , a proof Π , a serial number x , some transaction string M and R , output 1 if $R \subseteq \mathcal{C}$ and (Π, x, M) is valid. Otherwise 0.

The serial number x is generated in the **Spend** period and is a unique value in case users spend the same coin twice. We now demonstrate three requirements for the decentralized e-cash scheme: *Correctness*, *Anonymity* and *Balance*.

Correctness requires that a spend transaction generated by a honest user is always deemed valid. Formally, for all n , pp generated by **Setup**(n), (c, sk) generated by **Mint**(pp). Let $R \in \mathcal{C}$ be a coin set and the spend algorithm **Spend**(pp, c, sk, M, R)

outputs a pair (Π, x) . For all R, M , the following equation hold:

$$\text{E-Verify}(pp, \Pi, x, M, R \cup \{c\}) = 1.$$

Anonymity

For any PPT adversary, it is infeasible to tell which one was associated with a given coin spend transaction, under the condition that the adversary is given access to many of the coins used in the spend transactions .

1. **Setup** $\mathcal{C}$ runs **Setup** to produce pp and **Mint** to obtain coins (c_0, sk_0) and (c_1, sk_1) . The adversary $\mathcal{A}$ is provided with (pp, c_0, c_1) .

2. **Challenge** $\mathcal{A}$ outputs a transaction string M and a coins set R . Then $\mathcal{C}$ chooses a bit $b \xleftarrow{\$} \{0, 1\}$ and computes a spend transaction $(\Pi, x) \leftarrow \text{Spend}(pp, c_b, sk_b, M, R \cup \{c_0, c_1\})$ on behalf of coin c_b .

3. **Output** At the end, $\mathcal{A}$ outputs a bit b' . If $b' = b$, $\mathcal{A}$ wins.

Define the advantage of an adversary $\mathcal{A}$ wins if

$$\text{Adv}_{\mathcal{A}}^{e-ano} = |\Pr[b' = b] - 1/2|.$$

A decentralized e-cash scheme achieves *anonymity* if advantage $\text{Adv}_{\mathcal{A}}^{e-ano}$ is negligible.

Balance demands that for any PPT adversary spending more coins than he mints, the probability is negligible.

1. **Setup** $\mathcal{C}$ runs **Setup** to produce pp and the mint algorithm **Mint** to obtain coins (c_i, sk_i) for all $i \in [N]$. $\mathcal{A}$ is provided with (pp, c_0, c_1) .

2. **Query** $\mathcal{A}$ made queries to the following oracles.

$\mathcal{O}_{\text{Spend}}$: Take a coin c_j , a set of coins R_j and a transaction message M_j , it performs $\text{Spend}(pp, c_j, sk_j, M_j, R_j)$ to output (Π_j, x_j) and records (x_j, M_j) in the set Corr .

3. **Output** Eventually, $\mathcal{A}$ outputs m coins $c'_1, \dots, c'_m$ and $m + 1$ valid spend transactions $X_1, \dots, X_{m+1}$, where $X_j = (\Pi_j, x_j, M_j, R_j)$.

The adversary $\mathcal{A}$ is successful if transaction $X \in \{X_1, \dots, X_{m+1}\}$ where $X = (\Pi^*, x^*, M^*, R^*)$ satisfies:

- $\text{E-Verify}(pp, \Pi^*, x^*, M^*, R^*) = 1$.
- (x^*, M^*) is not in the set Corr and is not queried to the oracle.
- $R^* \subseteq \{c_1, \dots, c_N, c'_1, \dots, c'_m\}$.
- $\mathbf{x}^*$ only appears once in the $m + 1$ spend transaction.

The advantage of an adversary $\mathcal{A}$ against balance is

$$\mathbf{Adv}_{\mathcal{A}}^{\text{balance}} = \Pr[\mathcal{A} \text{ is successful}].$$

A decentralized e-cash scheme satisfies *balance* if advantage $\mathbf{Adv}_{\mathcal{A}}^{\text{balance}}$ is negligible.

7.6.2 Our Code-Based Decentralized E-cash Scheme

We will give our construction in this section.

Setup(λ): Take security parameter λ . Let $k > n + 2\log(1/\epsilon) + \mathcal{O}(1)$, $m = 2^c \cdot (n + k)/c$, $q = 2^c \cdot 2n/c$. This algorithm samples matrix $\mathbf{B} = (\mathbf{B}_0, \mathbf{B}_1) \xleftarrow{\$} \mathbb{Z}_2^{n \times q}$, where $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_2^{n \times 2^c \cdot n/c}$, for the accumulator construction and $\mathbf{C} = (\mathbf{C}_0, \mathbf{C}_1) \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$, where $\mathbf{C}_0 \in \mathbb{Z}_2^{n \times 2^c \cdot n/c}$ and $\mathbf{C}_1 \in \mathbb{Z}_2^{n \times 2^c \cdot k/c}$, for the commitment construction. It outputs $pp = (\mathbf{B}, \mathbf{C}, n, k, c)$.

Mint(pp) Sample a random binary vector $\mathbf{x} \in \mathbb{Z}_2^n$ and commit to $\mathbf{x}$ with a uniformly random opening $\mathbf{s} \in \mathbb{Z}_2^k$ to obtain a coin $\mathbf{d} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) \in \{0, 1\}^n$. Set secret value $sk = (\mathbf{x}, \mathbf{s})$ and output $(\mathbf{d}, sk)$.

- Spend**($pp, \mathbf{d}, sk, M, R$) 1. Take pp, sk , transaction string M , a coin $\mathbf{d}$ and an arbitrary set of coins R .
2. Run algorithm $\text{Accu}_{\mathbf{B}}(R)$ to obtain the accumulated value $\mathbf{u} \in \{0, 1\}^n$ based on the hash function $h_{\mathbf{B}}$.
 3. Run $\text{WitGen}_{\mathbf{B}}(R, \mathbf{d})$ to obtain witness w of $\mathbf{d}$ such that $\mathbf{d}$ is correctly accumulated in $\mathbf{u}$.
 4. Generate a NIZKAoK to show that a coin $\mathbf{d}$ is a correct commitment of $\mathbf{x}$ and the coin is indeed one coin of the coin set R such that we can compute a hash chain from $\mathbf{d}$ along to the Merkle-tree root $\mathbf{u}$: $\text{Com}(\mathbf{x}, \mathbf{s}) = \mathbf{d}$ and $\text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1$.

On public input $(\mathbf{B}, \mathbf{C}, \mathbf{u})$ and prover's witness $(\mathbf{x}, \mathbf{d}, w)$, it can be done by running the interactive protocol of Section 4.6 and achieving negligible soundness error by repeating $\kappa = \omega(\log \lambda)$ times and converted into a NIZK via the Fiat-Shamir heuristic by computing

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

in $\Pi = (\{\text{CMT}_i\}_{i=1}^{\kappa}, \text{CH}, \{\text{RSP}\}_{i=1}^{\kappa})$.

Outputs a pair of a proof Π and serial number $\mathbf{x}$, that is, $(\Pi, \mathbf{x})$.

E-Verify($pp, \Pi, \mathbf{x}, M, R$) : On input $(\mathbf{B}, \mathbf{C}, \mathbf{u})$, Π , a serial number $\mathbf{x}$, M and R . The verification algorithm works as follows:

1. Run $\text{Accu}_{\mathbf{B}}(R)$ and output the root value $\mathbf{u} \in \{0, 1\}^n$.
2. If $(\text{Ch}_1, \dots, \text{Ch}_{\kappa}) \neq \mathcal{H}_{\text{FS}}(M, \{\text{CMT}_i\}_{i=1}^{\kappa}, \mathbf{B}, \mathbf{C}, \mathbf{u})$, then return *invalid*.
3. For each $i = 1$ to κ , check the validity of RSP_i w.r.t. CMT_i and Ch_i . Returns *invalid* if any of the conditions does not hold.

4. If $\mathbf{x}$ appears more than once, return *invalid*.
5. Return *valid*.

7.7 The Supporting Zero-Knowledge Argument System

We now present the ZKAoK for our scheme constructed in Section 7.6.2 and the associated relation R_{e-cash} is given below.

$$R_{e-cash} = \left\{ \left((\mathbf{B}, \mathbf{C}, \mathbf{u}); (\mathbf{d}, w, \mathbf{x}, \mathbf{s}) \right) : \text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \wedge \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{d} \right\}$$

Now let us look at relation R_{e-cash} . Given $\mathbf{B}, \mathbf{C}$ uniformly random chosen in $\mathbb{Z}_2^{n \times q}$ and $\mathbb{Z}_2^{n \times m}$, respectively, $\mathbf{u} \in \{0, 1\}^n$, $\mathbf{b} \in \mathbb{Z}_2^m$, a matrix the goal of $\mathcal{P}$ is to convince $\mathcal{V}$ that his possession of the secret $(\mathbf{d}, w, \mathbf{x}, \mathbf{s})$ such that it is valid in the accumulator verification algorithm Verify . We define R_{vtr} as follows:

$$R_{e-cash} = \left\{ \left((\mathbf{B}, \mathbf{C}, \mathbf{u}) \in \mathbb{Z}_2^{n \times q} \times \mathbb{Z}_2^{n \times m} \times \{0, 1\}^n; w \in \{0, 1\}^\ell \times (\{0, 1\}^n)^\ell, \mathbf{d} \in \{0, 1\}^n : \right. \right. \\ \left. \left. \text{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \wedge \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) = \mathbf{d} \right\}.$$

We follow the extension and permutation techniques in Section 4.6 and Section 6.5. Now let us look the the relation R_{ecash} , which contain the relation in accumulator of proving a path with nodes $\mathbf{v}_k$ computed as follow.

$$\mathbf{v}_k = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{k+1}, \mathbf{w}_{k+1}), & \text{if } i_{k+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{k+1}, \mathbf{v}_{k+1}), & \text{if } i_{k+1} = 1. \end{cases} \quad (7.14)$$

Rewrite equation (7.14) into the following equivalent form.

We have

$$\begin{aligned}
\mathbf{v}_k &= \bar{i}_{k+1} \cdot h_{\mathbf{B}}(\mathbf{v}_{k+1}, \mathbf{w}_{k+1}) \oplus i_{k+1} \cdot h_{\mathbf{B}}(\mathbf{w}_{k+1}, \mathbf{v}_{k+1}) \\
&= \bar{i}_{k+1} \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{v}_{k+1}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{w}_{k+1})) \oplus i_{k+1} \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{w}_{k+1}) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{v}_{k+1})) \\
&= \mathbf{B} \cdot \begin{pmatrix} \bar{i}_{k+1} \cdot \text{RE}(\mathbf{v}_{k+1}) \\ i_{k+1} \cdot \text{RE}(\mathbf{v}_{k+1}) \end{pmatrix} \oplus \mathbf{B} \cdot \begin{pmatrix} i_{k+1} \cdot \text{RE}(\mathbf{w}_{k+1}) \\ \bar{i}_{k+1} \cdot \text{RE}(\mathbf{w}_{k+1}) \end{pmatrix} \\
&= \mathbf{B} \cdot \text{Ext}(i_{k+1}, \text{RE}(\mathbf{v}_{k+1})) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_{k+1}, \text{RE}(\mathbf{w}_{k+1}))
\end{aligned}$$

Therefore, $\mathcal{P}$ tries convince the verifier $\mathcal{V}$ that he knows a path in the Merkle tree with all the information $i_1, \dots, i_\ell \in \{0, 1\}^\ell$, $\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{w}_1, \dots, \mathbf{w}_\ell \in \{0, 1\}^n$. The equation is as follows:

$$\begin{cases} \mathbf{B} \cdot \text{Ext}(i_1, \text{RE}(\mathbf{v}_1)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_1, \text{RE}(\mathbf{w}_1)) = \mathbf{u}; \\ \mathbf{B} \cdot \text{Ext}(i_2, \text{RE}(\mathbf{v}_2)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_2, \text{RE}(\mathbf{w}_2)) \oplus \mathbf{v}_1 = \mathbf{0}. \\ \dots\dots\dots \\ \mathbf{B} \cdot \text{Ext}(i_\ell, \text{RE}(\mathbf{v}_\ell)) \oplus \mathbf{B} \cdot \text{Ext}(\bar{i}_\ell, \text{RE}(\mathbf{w}_\ell)) \oplus \mathbf{v}_{\ell-1} = \mathbf{0} \\ \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}) \oplus \mathbf{v}_\ell = \mathbf{0} \end{cases} \quad (7.15)$$

Next, we apply extension technique for vectors **Encode** which is defined in Section 4.6 to vectors $\mathbf{v}_{\ell-1}, \dots, \mathbf{v}_1$ to obtain $\mathbf{f}_k = \text{Encode}(\mathbf{v}_k) \in \{0, 1\}^{2n}$ for $k \in [\ell - 1]$, and it can be rewritten as $\mathbf{v}_k = \mathbf{I}_n^* \cdot \mathbf{f}_k$ for $k \in [\ell - 1]$.

$$\begin{cases} \mathbf{y}_k = \text{Ext}(i_k, \text{RE}(\mathbf{v}_k)) \in \{0, 1\}^q, k \in [\ell] \\ \mathbf{z}_k = \text{Ext}(\bar{i}_k, \text{RE}(\mathbf{w}_k)) \in \{0, 1\}^q, k \in [\ell] \\ \mathbf{f}_k = \text{Encode}(\mathbf{v}_k) \in \{0, 1\}^{2n}, k \in [\ell - 1] \end{cases} \quad (7.16)$$

Therefore, the equations in (7.15) is converted into:

$$\left\{ \begin{array}{l} \mathbf{B} \cdot \mathbf{y}_1 \oplus \mathbf{B} \cdot \mathbf{z}_1 = \mathbf{u}; \\ \mathbf{B} \cdot \mathbf{y}_2 \oplus \mathbf{B} \cdot \mathbf{z}_2 \oplus \mathbf{I}_n^* \cdot \mathbf{f}_1 = \mathbf{0}. \\ \dots\dots\dots \\ \mathbf{B} \cdot \mathbf{y}_\ell \oplus \mathbf{B} \cdot \mathbf{z}_\ell \oplus \mathbf{I}_n^* \cdot \mathbf{f}_{\ell-1} = \mathbf{0}. \end{array} \right. \quad (7.17)$$

Equations in (7.17) can be unified into the form of $\mathbf{M}_e \cdot \mathbf{w}_e = \mathbf{v}$ applying the basic algebra, where $\mathbf{M}_e \in \mathbb{Z}_2^{\binom{(\ell+1)n}{\times L}}$, $\mathbf{v} \in \mathbb{Z}_2^{(\ell+1)n}$ are public and $\mathbf{w}_e \in \{0, 1\}^L$ is the secret with length $L = 2\ell q + 2\ell n + m$ and in the form of

$$\mathbf{w}_e = (\mathbf{y}_1 \| \dots \| \mathbf{y}_\ell \| \mathbf{z}_1 \| \dots \| \mathbf{z}_\ell \| \mathbf{f}_1 \| \dots \| \mathbf{f}_\ell \| \text{RE}(\mathbf{x}) \| \text{RE}(\mathbf{s}))$$

We define a set **VALID** that contains all vectors in the same form as $\mathbf{w}_e$, the set $\mathcal{S}$ and the permutations $\{\Gamma_\phi : \phi \in \mathcal{S}\}$ based on the conditions in (2.2). Let **VALID** be a set of all vectors in the form of

$$\mathbf{w}' = (\mathbf{y}'_1 \| \dots \| \mathbf{y}'_\ell \| \mathbf{z}'_1 \| \dots \| \mathbf{z}'_\ell \| \mathbf{f}'_1 \| \dots \| \mathbf{f}'_\ell \| \mathbf{q}'_0 \| \mathbf{q}'_1) \in \{0, 1\}^L$$

and it satisfy the conditions as Section 7.4:

Let $\mathcal{S}$ be of the following form:

$$\mathcal{S} = (\{0, 1\}^n)^\ell \times (\{0, 1\}^n)^\ell \times \{0, 1\}^\ell \times \{0, 1\}^n \times \{0, 1\}^k$$

Then, for each

$$\phi = (\mathbf{b}_1, \dots, \mathbf{b}_\ell, \mathbf{e}_1, \dots, \mathbf{e}_\ell, g_1, \dots, g_\ell, \mathbf{k}_0, \mathbf{k}_1) \in \mathcal{S}, \text{ define the permutation } \Gamma_\phi$$

that transforms

$$\mathbf{w}^* = (\mathbf{y}_1^* \| \dots \| \mathbf{y}_\ell^* \| \mathbf{z}_1^* \| \dots \| \mathbf{z}_\ell^* \| \mathbf{f}_1^* \| \dots \| \mathbf{f}_\ell^* \| \mathbf{q}_0^* \| \mathbf{q}_1^*) \in \{0, 1\}^L$$

with $\mathbf{y}_k^*, \mathbf{z}_k^* \in \{0, 1\}^m, \mathbf{f}_k^* \in \{0, 1\}^{2n}$ for $k \in [\ell]$, $\mathbf{q}_i^* \in \{0, 1\}^{m/2}$ to

$$\Gamma_\phi(\mathbf{w}^*) = \left(\Psi_{g_1, \mathbf{b}_1}(\mathbf{y}_1^*) \parallel \dots \parallel \Psi_{g_\ell, \mathbf{b}_\ell}(\mathbf{y}_\ell^*) \parallel \Psi_{\bar{g}_1, \mathbf{e}_1}(\mathbf{z}_1^*) \parallel \dots \parallel \Psi_{\bar{g}_\ell, \mathbf{e}_\ell}(\mathbf{z}_\ell^*) \parallel F'_{\mathbf{b}_1}(\mathbf{f}_1^*) \parallel \dots \parallel F'_{\mathbf{b}_\ell}(\mathbf{f}_\ell^*) \parallel E'_{\mathbf{k}_0}(\mathbf{q}_0^*) \parallel E'_{\mathbf{k}_1}(\mathbf{q}_1^*) \right).$$

Equations (7.15) and (7.17) show that the conditions in (2.2) are all satisfied. Thus our relation R_{vlr} can be reduced in an form of abstraction relation R and we can obtain a ZKAoK follow the protocol in Figure 2.1. The communication cost for this protocol is $\mathcal{O}(L) = \mathcal{O}(2\ell q + 2\ell n + m)$ bits.

7.7.1 Security Analysis

We proceed to prove the correctness and the security properties of our scheme proposed in Section 7.6.2. The concrete proof for the Anonymity and Balance are given below.

Theorem 7.7.1. The e-cash scheme is anonymous, assuming that the argument system is statistically zero-knowledge in the random oracle model and simulation-sound

Proof. The anonymity is proved by using game-based method.

Game 0: 1. **Setup** $\mathcal{C}$ runs **Setup** to produce public parameters $pp = \mathbf{C} = (\mathbf{C}_0, \mathbf{C}_1) \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$ where $\mathbf{C}_0 \in \mathbb{Z}_2^{n \times 2^c \cdot n/c}$ and $\mathbf{C}_1 \in \mathbb{Z}_2^{n \times 2^c \cdot k/c}$. Then runs the Mint algorithm, sample random binary vectors $\mathbf{x}_i \in \mathbb{Z}_2^n$, $\mathbf{s}_i \in \mathbb{Z}_2^k$ and compute coin $\mathbf{d}_i = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}_i) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}_i) \in \{0, 1\}^n$, for $i \in 0, 1$. Set $sk_i = (\mathbf{x}_i, \mathbf{s}_i)$ and output $(\mathbf{d}_i, sk_i)$ to obtain coins $(\mathbf{d}_0, sk_0)$ and $(\mathbf{d}_1, sk_1)$, here The adversary $\mathcal{A}$ is provided with $(pp, \mathbf{d}_0, \mathbf{d}_1)$.

2. **Challenge** $\mathcal{A}$ outputs a transaction string M and a coins set R . $\mathcal{C}$ computes a spend transaction $(\Pi_0, \mathbf{x}_0)$ on behalf of coin $\mathbf{d}_0$ and returns it to $\mathcal{A}$.

3. **Output** Eventually, $\mathcal{A}$ outputs b' .

Here we have $\Pr[W_0] = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{e-anon-0}}(\lambda) = 1]$.

Game 1: It changes Game 0 by generating simulated NIZK argument Π^* instead of a real proof in the challenged phase and replacing $\mathbf{x}_0$ with an uniformly random chose vector $\mathbf{y}$, then the challenger $\mathcal{C}$ outputs a spend transaction $(\Pi^*, \mathbf{y})$ in the challenge phase. Note that Π is statistical zero-knowledge, $\mathbf{x}_0$ is uniformly random chose from $\mathbb{Z}_2^n$, Game 1 and Game 0 are statistically indistinguishable in view of the adversary, which implies that $|\Pr[W_1] - \Pr[W_0]| \in \text{negl}(\lambda)$.

Game 2: It makes a small modification on Game 1 by replacing the simulated proof Π^* with the real proof Π_1 and changing $\mathbf{y}$ into an uniformly random vector $\mathbf{x}_1 \in \mathbb{Z}_2^n$. Game 2 and Game 1 would be statistically indistinguishable in adversary' view since the underlying argument system is statistically indistinguishable, which implies that $|\Pr[W_2] - \Pr[W_1]| \in \text{negl}(\lambda)$. This is experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{e-anon-1}}(\lambda)$, thus $\Pr[W_2] = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{e-anon-1}}(\lambda) = 1]$.

In conclusion, we have

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

this concludes the proof. $\square$

Next, we will prove that our e-cash construction satisfies *balance*. First we prove a basic lemma that will be employed in our proof.

Lemma 7.7.2. Let $n \leq k - 2 \log(1/\epsilon) - \mathcal{O}(1)$, a hash function $h : \{0, 1\}^{n+k} \rightarrow \{0, 1\}^n$. Then for $\mathbf{q} \xleftarrow{\$} \{0, 1\}^{n+k}$, the probability for finding a collision $\mathbf{q}' \in \{0, 1\}^{n+k}$ such that $h(\mathbf{q}') = h(\mathbf{q})$ is at least $1 - 2^{-n}$.

Proof. Since $k > n + 2 \log(1/\epsilon) + \mathcal{O}(1)$ from leftover hash lemma, we have $n + k > 2n$. There are in total 2^{n+k} elements $\mathbf{x} \in \{0, 1\}^{n+k}$ and among them, there exist at most $2^n - 1$ elements that do not have $\mathbf{q}' \in \{0, 1\}^{n+k}$ such that $h(\mathbf{q}') = h(\mathbf{q})$. Thus, a

uniformly random element $\mathbf{q}$ has a collision $\mathbf{q}'$ with probability at least $\frac{2^{n+k} - (2^n - 1)}{2^{2n}} > \frac{2^{2n} - (2^n - 1)}{2^{2n}} > 1 - 2^{-n}$. $\square$

For $m = 2^c \cdot (n+k)/c$, $\mathbf{q} \xleftarrow{\$} \{0, 1\}^m$, there exists a distinct vector $\mathbf{q}' \in \{0, 1\}^m \setminus \{\mathbf{q}\}$ such that $\mathbf{C} \cdot \mathbf{q} = \mathbf{C} \cdot \mathbf{q}'$ with overwhelming probability $1 - 2^{-n}$.

Theorem 7.7.3. Assuming that the $2\text{-RNSD}_{n,n+k,c}$ problem is hard, our scheme is balance.

Proof. We prove balance by contraposition. Assuming that an adversary $\mathcal{A}$ succeeds in breaking balance property with non-negligible advantage, then a PPT algorithm $\mathcal{B}$ that either breaks the security of the accumulator, or breaks the computational soundness of the zero-knowledge protocol, or directly solves an $2\text{-RNSD}_{n,n+k,c}$ problem with non-negligible probability.

$\mathcal{B}$ first runs the **Setup** algorithm to produce $pp = (\mathbf{B}, \mathbf{C}, n, k, c)$ and mint coins by running **Mint** algorithm, for $i = 1, \dots, N$, $\mathcal{B}$ obtains $(\mathbf{d}_i, sk_i)$, here $sk = (\mathbf{x}_i, \mathbf{s}_i)$ as in Section 7.6.2. $\mathcal{B}$ retains all the secret values sk_i so that he can answer all the spending queries made from the adversary $\mathcal{A}$. When $\mathcal{A}$ makes queries to the **Spend** oracle with input of a coin $\mathbf{d}_j$, a set of coins R_j and a transaction message M_j , the oracle performs **Spend**($pp, \mathbf{d}_j, sk_j, M_j, R_j$) to output $(\Pi_j, \mathbf{x}_j)$ and adds $(\mathbf{x}_j, M_j)$ to the set $Corr$.

Eventually, $\mathcal{A}$ outputs m coins $\mathbf{d}'_1, \dots, \mathbf{d}'_m$ and $m + 1$ valid spend transactions $X_1, \dots, X_{m+1}$, where $X_j = (\Pi_j, \mathbf{x}_j, M_j, R_j)$. The adversary $\mathcal{A}$ is successful if for any transaction $X \in \{X_1, \dots, X_{m+1}\}$ where $X = (\Pi^*, x^*, M^*, R^*)$. We have the serial number $\mathbf{x}^*$ only be used once in $m + 1$ spend transaction and (x^*, M^*) is not in the set $Corr$, in other words, it has not been queried to the oracle. Furthermore, $\text{E-Verify}(pp, \Pi^*, x^*, M^*, R^*) = \text{valid}$, where $R^* \subseteq \{\mathbf{d}_1, \dots, \mathbf{d}_N, \mathbf{d}'_1, \dots, \mathbf{d}'_m\}$. It means that adversary succeeds when he produces more spend transactions than minted coins.

Since Π^* is a valid spend transaction. Let $\Pi^* = (\{\text{CMT}_i^*\}_{i=1}^\kappa, \text{CH}^*, \{\text{RSP}^*\}_{i=1}^\kappa)$, where $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}, R^*)$. Here $\mathcal{H}_{\text{FS}}(\cdot)$ is an oracle mapping arbitrary length vector to uniformly random elements in $\{1, 2, 3\}^\kappa$ and the same queries

leads to the same output.

Next, we want to prove that $\mathcal{A}$ had queried $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}^*, R^*)$, where $\mathbf{u}^* = \text{Acc}_{\mathbf{B}}(R^*)$, with overwhelming probability in order to obtain a challenge $\text{CH}^* = \mathcal{H}_{\text{FS}}(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}^*, R^*) \in \{1, 2, 3\}$. Prove by contradiction in this case on if he tries to guess the value, then the probability guessing it exactly the same as the output of the random oracle would be at most $3^{-\kappa}$. Therefore, $\mathcal{A}$ had queried $\mathcal{H}_{\text{FS}}$ at his t^* -th query among the in total Q_H queries he made.

Algorithm $\mathcal{B}$ would run extra executions of the adversary $\mathcal{A}$ for at most $32 \cdot Q_H / (\epsilon - 3^{-\kappa})$ times, since he will not be able to receive different answers until his t^* -th query made. From the t^* -th query on, $\mathcal{B}$ output uniformly random and independent values for each new query and send them to $\mathcal{A}$. As we can see, output for t^* -th query will be uniformly random and independent while the input is still the same as in the initial run. By following the Forking Lemma of Brickell et al. [30], $\mathcal{B}$ can obtain a 3-fork with the same input of $(M^*, \{\text{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathbf{C}, \mathbf{u}^*, R^*)$ and output three pairwise distinct hash values $\text{CH}_{t^*}^{(1)}, \text{CH}_{t^*}^{(2)}, \text{CH}_{t^*}^{(3)} \in \{1, 2, 3\}^\kappa$, as well as their corresponding responses $\text{RSP}_{t^*}^{(1)}, \text{RSP}_{t^*}^{(2)}, \text{RSP}_{t^*}^{(3)}$ with probability at least $1/2$. Moreover, the results of [30] imply that it can be achieved to get $\{\text{CH}_{t^*,k}^{(1)}, \text{CH}_{t^*,k}^{(2)}, \text{CH}_{t^*,k}^{(3)}\} = \{1, 2, 3\}$ with probability $1 - (7/9)^\kappa$ at the $k \in \{1, \dots, \kappa\}$ run.

From 3 valid responses $(\text{RSP}_{t^*,k}^{(1)}, \text{RSP}_{t^*,k}^{(2)}, \text{RSP}_{t^*,k}^{(3)})$ w.r.t. the same commitment CMT_k^* and all challenges 1, 2, 3, $\mathcal{B}$ is able to extract witnesses during period j^* since he satisfy the conditions in Theorem 2.3.13. The witnesses he extract is as follows

$$(\mathbf{x}^*, \mathbf{s}^*, \mathbf{d}^*, w^*)$$

, where $w^* = ((j_1^*, \dots, j_\ell^*), (\mathbf{w}_\ell^*, \dots, \mathbf{w}_1^*))$ such that $(j_1^*, \dots, j_\ell^*) \in \{0, 1\}^\ell$ is the binary representation of some $j^* \in \{0, \dots, |R^*| - 1\}$ and

$$h_{\mathbf{B}}(\mathbf{x}^*, \mathbf{s}^*) = \mathbf{d}^*, \tag{7.18}$$

$$\text{Verify}_{\mathbf{B}}(\mathbf{u}^*, \mathbf{d}^*, w^*) = 1. \quad (7.19)$$

Since $\mathcal{A}$ wins the game, either we have (1) coin $\mathbf{d}^*$ does not include in the set R^* , it implies that $\mathbf{d}^* \notin R^* = (\mathbf{d}_0, \dots, \mathbf{d}_{|R^*|-1})$ or (2) coin $\mathbf{d}^*$ contains in the set R^* , $\mathbf{d}^* \in R^* = (\mathbf{d}_0, \dots, \mathbf{d}_{|R^*|-1})$ and $\mathbf{d}^* = \mathbf{d}_{j^*}$.

In case (1), $\mathcal{B}$ is able to break the security of the accumulator.

In case (2), the extracted witnesses $(\mathbf{d}_{j^*}, \mathbf{x}^*, \mathbf{s}^*)$ satisfy equation by the soundness of the argument system. When $\mathcal{A}$ queried the **Spend** oracle, $\mathcal{B}$ chose $sk_{j^*} = \mathbf{x}_{j^*} \in \{0, 1\}^n$ satisfying

$$\mathbf{d}_{j^*} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}_{j^*}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{s}_{j^*}). \quad (7.20)$$

Since $\mathbf{d}^*$ is an uncorrupted coin, then $(\mathbf{x}_{j^*}, \mathbf{s}_{j^*}) = \mathbf{p}_{j^*} \neq \mathbf{p}^* = (\mathbf{x}^*, \mathbf{s}^*)$ with probability at least $1/2$ according to Lemma 7.7.2. Furthermore, the statistical Witness Indistinguishability (WI) of the argument system implies that $\mathbf{p}^*$ and $\mathbf{p}_{j^*}$ would be indistinguishable. Combining equation (7.18) and equation (7.20), we obtain $\mathbf{C}_0 \cdot (\text{RE}(\mathbf{x}_{j^*}) \oplus \text{RE}(\mathbf{x}^*)) \oplus \mathbf{C}_1 \cdot (\text{RE}(\mathbf{s}_{j^*}) \oplus \text{RE}(\mathbf{s}^*)) = \mathbf{0}$, which yields a valid 2-RNSD $_{n, n+k, c}$ solution $\mathbf{w} = (\mathbf{w}_0, \mathbf{w}_1) = (\text{RE}(\mathbf{x}_{j^*}) \oplus \text{RE}(\mathbf{x}^*), \text{RE}(\mathbf{s}_{j^*}) \oplus \text{RE}(\mathbf{s}^*))$.

Therefore, $\mathcal{B}$ is able to solve the 2-RNSD $_{n, n+k, c}$ problem with a non-negligible probability to be at least $1/2 \cdot (\epsilon - 3^{-\kappa}) \cdot (1 - (7/9)^\kappa) \cdot 1/2$. $\square$

Chapter 8

Conclusions and Open Problems

Throughout the thesis, we have filled the gaps in the subfield of code-based privacy-preserving constructions. First, we provided proofs of a hash preimage and thus obtained statistically hiding and computationally binding commitment schemes. Such schemes are often used as an important component in our zero knowledge arguments and are well composed with other primitives or relations throughout this thesis. For example, statistically hiding commitments have been involved in building the Merkle tree accumulator in our code-based VLR group signature scheme, they also have been served as an important component in our decentralized e-cash system in Chapter 7. Second, we proposed proofs of anonymity-oriented applications and set membership proofs in designing code-based logarithmic size accumulators, ring and group signatures. Furthermore, we improved the group signature with decent features to satisfy verifier-local revocation and backward unlinkability to retain anonymity of the previous signatures signed by a revoked user and only the verifier download the up-to-date revocation information. Moreover, we have designed decentralized e-cash systems from Bitcoin by applying a simple combination of some code-based primitives to achieve decentralization property and avoid double spending in transactions.

A basic yet important question about proofs of knowledge of the hash preimage is still an unfilled gap in code-based cryptography. Considering the family of AFS hash

functions, a nontrivial part for solving such a question came out is the preserved non-linearly structure for regular words when we apply permutations on them to show it has been done correctly. Our commitment scheme is constructed by extending the AFS hash functions and proving in zero-knowledge of its valid opening. The statistically hiding and computationally binding properties make it work as one of the important part in our zero-knowledge argument systems. We also made a small modification to the AFS hash functions to obtain collision-resistant hash functions which hashes two same length inputs into one output and proceed to build Merkle-tree accumulators. A non-trivial part is to show a hidden preimage with a hidden image in a hidden path. Next we obtained proofs for anonymity-oriented applications, our ring signatures and group signatures by adding some layers into the accumulator. To equip our group signature with decent feature of achieving revocation and backward unlinkability, we built the Merkle tree upon commitments and obtain the revocation list that can only be downloaded by verifier from LPN samples whose matrix output by a deterministic function as a random oracle.

In our zero knowledge argument systems, we achieved non-interactive by applying Fiat-Shamir paradigm, which also leads to analyse the security proofs in random oracle model. Thus a natural question come to us is whether we can achieve the goal in standard model and open questions arise for our ring signatures in Chapter 5 and group signatures in Chapter 6 are how to construct code-based ring and group signatures in standard model instead of using a random oracle in our settings.

In our Stern-type zero knowledge protocol, the soundness error for each round is $2/3$ and it can be made negligible by standard repetition techniques while causing an not small proof size. However, In 2019, Yang et al [125] introduced efficient zero-knowledge argument of knowledge systems, which has a rather small soundness error to be inverse-polynomial $1/\text{poly}$ and presented some privacy-preserving constructions in lattice-based assumptions. An interesting research direction is to design more efficient ZKAoK either in code-based setting or in other settings and to apply them

for designing more efficient privacy-preserving primitives and constructions.

In Chapter 3, a further direction could be how to construct commitment schemes which are computationally hiding and statistically binding and its hardness based on the 2-RNSD problem. The previous work in designing such schemes in code-based cryptography was proposed in [68] which rely on LPN problem.

In Chapter 4, Camenisch and Lysyanskaya [33] suggested dynamic accumulator with the feature of adding in and remove the element from the set of accumulated value. Later on, some bilinear paring based dynamic accumulator were proposed. An further work could be how to construct a code-based dynamic accumulator.

In Chapter 6 and Chapter 7, we have constructed code-based group signatures, as well as ones with verifier-local revocation and backward unlinkability. In lattice-based group signature constructions, there are various group signature with advanced feature in different applications. In 2015, Kohlweiss and Miers [73] introduced a concept of accountable tracing signature, which is able to limit the ability of the group manager and ensure the accountability of him and suggested a number-theoretic based construction. Later in 2019, Ling et al [86] proposed such scheme in lattice assumptions. It leads to a new direction for us to explore this property in code-based assumptions. Furthermore, we can further aim to explore code-based group signatures with more advanced features.

Last but not least, there are still many interesting privacy-preserving constructions needed to discover and to design in code-based cryptography.

Bibliography

- [1] Tolga Acar and Lan Nguyen. Revocation for delegatable anonymous credentials. In *PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2011.
- [2] Carlos Aguilar, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop*, pages 648–652. IEEE, 2011.
- [3] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A code-based group signature scheme. *Designs, Codes and Cryptography*, 82(1-2):469–493, 2017.
- [4] 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 *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
- [5] Benny Applebaum, Naama Haramaty-Krasne, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In *ITCS 2017*, volume 67 of *LIPICs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

- [6] Giuseppe Ateniese, Dawn Song, and Gene Tsudik. Quasi-efficient revocation of group signatures. In *International Conference on Financial Cryptography*, pages 183–197. Springer, 2002.
- [7] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A fast provably secure cryptographic hash function. *IACR Cryptology ePrint Archive*, 2003:230, 2003.
- [8] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A family of fast syndrome based cryptographic hash functions. In *Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 64–83. Springer, 2005.
- [9] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better—how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 399–414. Springer, 2012.
- [10] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 1997.
- [11] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: how $1+1=0$ improves information set decoding. In *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012.
- [12] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.
- [13] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construc-

- tion based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
- [14] Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer, 1993.
 - [15] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.
 - [16] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, 2009.
 - [17] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
 - [18] Daniel J Bernstein, Tanja Lange, Christiane Peters, and Peter Schwabe. Faster 2-regular information-set decoding. In *International Conference on Coding and Cryptology*, pages 81–98. Springer, 2011.
 - [19] Daniel J Bernstein, Tanja Lange, Christiane Peters, and Peter Schwabe. Really fast syndrome-based hashing. In *AFRICACRYPT 2011*, volume 6737 of *Lecture Notes in Computer Science*, pages 134–152. Springer, 2011.
 - [20] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *STOC 2000*, pages 435–440. ACM, 2000.
 - [21] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

- [22] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
- [23] Dan Boneh, Saba Eskandarian, and Ben Fisch. Post-quantum group signatures from symmetric primitives. *IACR Cryptology ePrint Archive*, 2018:261, 2018.
- [24] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *CCS 2004*, pages 168–177. ACM, 2004.
- [25] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *EUROCRYPT 2018*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564. Springer, 2018.
- [26] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for lpn and cryptographic hashing via code smoothing. In *EUROCRYPT 2019*, volume 11478 of *Lecture Notes in Computer Science*, pages 619–635. Springer, 2019.
- [27] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for lpn and cryptographic hashing via code smoothing. In *EUROCRYPT 2019*, volume 11478 of *Lecture Notes in Computer Science*, pages 619–635. Springer, 2019.
- [28] Pedro Branco and Paulo Mateus. A code-based linkable ring signature scheme. In *ProvSec 2018*, volume 11192 of *Lecture Notes in Computer Science*, pages 203–219. Springer, 2018.
- [29] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988.

- [30] Ernest Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In *PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 276–292. Springer, 2000.
- [31] Julien Bringer and Alain Patey. Vlr group signatures-how to achieve both backward unlinkability and efficient revocation checks. In *SECRYPT 2012*, pages 215–220, 2012.
- [32] Jan Camenisch. Efficient anonymous fingerprinting with group signatures. In *ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 415–428. Springer, 2000.
- [33] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002.
- [34] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *CCS 2002*, pages 21–30. ACM, 2002.
- [35] Sébastien Canard and Aline Gouget. Multiple denominations in e-cash with compact transaction data. In *International Conference on Financial Cryptography and Data Security*, pages 82–97. Springer, 2010.
- [36] Sébastien Canard and Jacques Traoré. On fair e-cash systems based on group signature schemes. In *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 2003.
- [37] Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In *SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2010.

- [38] David Chaum. Blind signatures for untraceable payments. In *CRYPTO 1982*, pages 199–203. Springer, 1983.
- [39] David Chaum. Blind signature system. In *CRYPTO 1983*, pages 153–153. Springer, 1984.
- [40] David Chaum and Eugène Van Heyst. Group signatures. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 257–265. Springer, 1991.
- [41] Nicolas T Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. In *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2001.
- [42] Léonard Dallot. Towards a concrete security proof of courtois, finiasz and sendrier signature scheme. In *Western European Workshop on Research in Cryptology*, volume 4945 of *Lecture Notes in Computer Science*, pages 65–77. Springer, 2007.
- [43] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs dishonest verifier in public coin zero-knowledge proofs. In *CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 325–338. Springer, 1995.
- [44] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto 2018*, volume 10786 of *Lecture Notes in Computer Science*, pages 419–440. Springer, 2018.
- [45] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

- [46] Nico Döttling, Rafael Dowsley, Jörn Müller-Quade, and Anderson CA Nascimento. A cca2 secure variant of the mceliece cryptosystem. *IEEE Transactions on Information Theory*, 58(10):6672–6680, 2012.
- [47] 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 *Lecture Notes in Computer Science*, pages 260–285. Springer, 2015.
- [48] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426. ACM, 1990.
- [49] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.
- [50] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
- [51] Matthieu Finiasz. Parallel-cfs - strengthening the CFS mceliece-based signature scheme. In *SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2010.
- [52] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009.
- [53] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 245–255. Springer, 1996.

- [54] Ph Gaborit, C Lauderoux, N Sendrier, et al. Synd: a very fast code-based cipher stream with a security reduction. In *IEEE Conference, ISIT*, volume 7, pages 186–190, 2007.
- [55] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from codes with rank metric. In *CRYPTO 2017*, volume 10403 of *Lecture Notes in Computer Science*, pages 194–224. Springer, 2017.
- [56] Henri Gilbert, Matthew JB Robshaw, and Yannick Seurin. How to encrypt with the lpn problem. In *ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 679–690. Springer, 2008.
- [57] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 1986.
- [58] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. pages 230–240, 2010.
- [59] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM, 1982.
- [60] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
- [61] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

- [62] Valerii Denisovich Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.
- [63] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer, 2015.
- [64] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving lpn using covering codes. In *ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2014.
- [65] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, 2007.
- [66] Rong Hu, Kirill Morozov, and Tsuyoshi Takagi. Proof of plaintext knowledge for code-based public-key encryption revisited. In *ASIA CCS 2013*, pages 535–540. ACM, 2013.
- [67] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *Journal of Cryptology*, 10(1):37–49, 1997.
- [68] 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 *Lecture Notes in Computer Science*, pages 663–680. Springer, 2012.
- [69] Jonathan Katz and Ji Sun Shin. Parallel and concurrent security of the hb and hb+ protocols. In *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2006.

- [70] Jonathan Katz and Ji Sun Shin. Parallel and concurrent security of the hb and hb+ protocols. In *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2006.
- [71] 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 *Lecture Notes in Computer Science*, pages 372–389. Springer, 2008.
- [72] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 7–26. Springer, 2011.
- [73] Markulf Kohlweiss and Ian Miers. Accountable metadata-hiding escrow: A group signature case study. *Proceedings on Privacy Enhancing Technologies*, 2015(2):206–221, 2015.
- [74] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer, 2013.
- [75] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2014.
- [76] Pil Joong Lee and Ernest F Brickell. An observation on the security of mceliece’s public-key cryptosystem. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 275–280. Springer, 1988.

- [77] Jeffrey S Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
- [78] Éric Leveil and Pierre-Alain Fouque. An improved lpn algorithm. In *SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006.
- [79] 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 *Lecture Notes in Computer Science*, pages 373–403. Springer, 2016.
- [80] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix–vector relations and lattice-based group encryption. *Theoretical Computer Science*, 759:72–97, 2019.
- [81] 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 *Lecture Notes in Computer Science*, pages 1–31. Springer, 2016.
- [82] Benoît Libert, Thomas Peters, and Moti Yung. Scalable group signatures with revocation. In *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 609–627. Springer, 2012.
- [83] Benoît Libert and Damien Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *CANS 2009*, volume 5888 of *Lecture Notes in Computer Science*, pages 498–517. Springer, 2009.

- [84] Zi Lin and Nicholas Hopper. Jack: Scalable accumulator-based nymble system. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, pages 53–62. ACM, 2010.
- [85] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 427–449. Springer, 2015.
- [86] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Accountable tracing signatures from lattices. In *CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 556–576. Springer, 2019.
- [87] Zeming Lu, Zoe L Jiang, Yulin Wu, Xuan Wang, and Yantao Zhong. A lattice-based anonymous distributed e-cash from bitcoin. In *International Conference on Provable Security*, pages 275–287. Springer, 2019.
- [88] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. *Journal of Cryptology*, 31(3):774–797, 2018.
- [89] Ivanovich Manin. *Vychislimoe i nevychislimoe*. Sov. radio,, 1980.
- [90] K Preetha Mathew, Sachin Vasant, Sridhar Venkatesan, and C Pandu Rangan. An efficient ind-cca2 secure variant of the niederreiter encryption scheme in the standard model. In *ACISP 2012*, volume 7372 of *Lecture Notes in Computer Science*, pages 166–179. Springer, 2012.
- [91] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\mathcal{O}(2^{0.054n})$. In *ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011.
- [92] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

- [93] Carlos Aguilar Melchor, Pierre-Louis Cayrel, Philippe Gaborit, and Fabien Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. *IEEE Transactions on Information Theory*, 57(7):4833–4842, 2011.
- [94] Ralph C Merkle. A certified digital signature. In *CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.
- [95] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.
- [96] Kirill Morozov and Tsuyoshi Takagi. Zero-knowledge protocols for the mceliece encryption. In *ACISP 2012*, volume 7372 of *Lecture Notes in Computer Science*, pages 180–193. Springer, 2012.
- [97] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, <http://bitcoin.org/bitcoin.pdf>. 2008.
- [98] Toru Nakanishi and Nobuo Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 533–548. Springer, 2005.
- [99] Toru Nakanishi and Nobuo Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In *IWSEC 2006*, volume 4266 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2006.
- [100] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437. Citeseer, 1990.

- [101] Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng. New code-based privacy-preserving cryptographic constructions. In *ASIACRYPT 2019*, volume 11922 of *Lecture Notes in Computer Science*, pages 25–55. Springer, 2019.
- [102] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005.
- [103] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Control and Inf. Theory*, 15(2):159–166, 1986.
- [104] Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Designs, Codes and Cryptography*, 49(1-3):289–305, 2008.
- [105] Goldreich Oded. Foundations of cryptography: Volume 2, basic applications. 2009.
- [106] Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In *Post-quantum cryptography*, pages 95–145. Springer, 2009.
- [107] Richard E Overill. Foundations of cryptography: Basic tools. *J. Log. Comput.*, 12(3):543–544, 2002.
- [108] Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi. Streaming authenticated data structures. In *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2013.
- [109] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Authenticated hash tables. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 437–448. ACM, 2008.
- [110] Edoardo Persichetti. On a cca2-secure variant of mceliece in the standard model. *IACR Cryptology ePrint Archive*, 2012:268, 2012.

- [111] Manoj Prabhakaran and Rui Xue. Statistically hiding sets. In *CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2009.
- [112] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
- [113] K Preetha Mathew, Sachin Vasant, and C Pandu Rangan. On provably secure code-based signature and signcryption scheme. Technical report, IACR Cryptology ePrint Archive, 2012.
- [114] Charles Rackoff and Daniel R Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.
- [115] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.
- [116] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.
- [117] Ron M Roth. Introduction to coding theory. 2006.
- [118] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE Computer Society, 1994.
- [119] Dawn Xiaodong Song. Practical forward secure group signature schemes. In *CCS 2001*, pages 225–234. ACM, 2001.
- [120] Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

- [121] Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.
- [122] Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
- [123] Gene Tsudik and Shouhuai Xu. Accumulating composites and improved group signing. In *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 269–286. Springer, 2003.
- [124] Pascal Véron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8(1):57–69, 1997.
- [125] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: construction and applications. In *CRYPTO 2019*, volume 11692 of *Lecture Notes in Computer Science*, pages 147–175. Springer, 2019.
- [126] Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. Collision resistant hashing from learning parity with noise. *IACR Cryptology ePrint Archive*, 2017:1260, 2017.
- [127] Sujing Zhou and Dongdai Lin. Shorter verifier-local revocation group signatures from bilinear maps. In *CANS 2006*, volume 4301 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2006.

Appendices

Appendix A

Proof of Theorem 2.3.13

Proof. By the definition of perfect completeness of a protocol, it requires that for any honest prover who follows the protocol always gets accepted by the verifier. It is also straightforward to check that the communication cost is bounded by $\mathcal{O}(L)$.

We now proceed to prove that the protocol satisfies statistical zero-knowledge by constructing a simulator.

Zero-Knowledge Property. Given the public input, we can construct a PPT simulator Sim which interacts with a probably malicious verifier V^* and outputs a simulated transcript that is statistically close to the output in real interaction with probability close to $2/3$.

The high level idea for proving zero knowledge is as follows.

Sim first guesses a random value $\overline{Ch} \in \{1, 2, 3\}$ as a challenge and compute valid responses for the challenges not chosen. It output $\perp$ if the challenge value from the verifier is the same as $\overline{Ch}$.

For $\overline{Ch} = 1$: SIM computes a vector $\mathbf{z} \in \mathbb{Z}_2^L$ that only satisfy the linear equation $\mathbf{M} \cdot \mathbf{z} = \mathbf{v}$. Then samples random masking $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, random permutation $\phi \xleftarrow{\$} \mathcal{S}$, and randomness η_1, η_2, η_3 that will be used in commitment COM . It computes three

commitments as follow.

$$\begin{aligned} C'_1 &= \text{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w; \eta_1), \\ C'_2 &= \text{COM}(\Gamma_\phi(\mathbf{r}_w); \eta_2), \quad C'_3 = \text{COM}(\Gamma_\phi(\mathbf{z} \oplus \mathbf{r}_w); \eta_3). \end{aligned}$$

Then sends these three commitments to the verifier. After the verifier sends back the challenge Ch , the simulator sends the corresponding respond with respect to the challenge as follows:

- If the verifier sends a challenge value $Ch = 1$: *Sim* outputs $\perp$ and abort the protocol.
- If the verifier sends $Ch = 2$: *Sim* sends $\text{RSP} = (\phi, \mathbf{z} \oplus \mathbf{r}_w, \eta_1, \eta_3)$.
- If the verifier sends $Ch = 3$: *Sim* sends $\text{RSP} = (\phi, \mathbf{r}_w, \eta_1, \eta_2)$.

For $\overline{Ch} = 2$: *SIM* samples a vector $\mathbf{v}$ that only satisfy the constraint of the witnesses $\mathbf{v} \xleftarrow{\$} \text{VALID}$, random masking $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, permutation $\phi \xleftarrow{\$} \mathcal{S}$, and randomness η_1, η_2, η_3 that will be used in commitments. It computes three commitments as follow.

$$\begin{aligned} C'_1 &= \text{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w; \eta_1), \\ C'_2 &= \text{COM}(\Gamma_\phi(\mathbf{r}_w); \eta_2), \quad C'_3 = \text{COM}(\Gamma_\phi(\mathbf{v} \oplus \mathbf{r}_w); \eta_3). \end{aligned}$$

Then sends these three commitments to the verifier. After the verifier sends back the challenge Ch , the simulator sends the corresponding respond with respect to the challenge as follows:

- If the verifier sends $Ch = 1$, *Sim* responds with $\text{RSP} = (\Gamma_\phi(\mathbf{z}), \Gamma_\phi(\mathbf{r}_w), \eta_2, \eta_3)$.
- If the verifier sends $Ch = 2$, *Sim* outputs $\perp$ and abort.
- If the verifier sends $Ch = 3$, *Sim* sends $\text{RSP} = (\phi, \mathbf{r}_w, \eta_1, \eta_2)$.

For $\overline{Ch} = 3$: **SIM** samples $\mathbf{z}$ that only satisfy the constraint $\mathbf{z} \stackrel{\$}{\leftarrow} \text{VALID}$, random masking $\mathbf{r}_w \stackrel{\$}{\leftarrow} \mathbb{Z}_2^L$, permutation $\phi \stackrel{\$}{\leftarrow} \mathcal{S}$, and randomness η_1, η_2, η_3 for computing **COM**, the three commitments are

$$C'_1 = \text{COM}(\phi, \mathbf{M} \cdot (\mathbf{z} \oplus \mathbf{r}_w) \oplus \mathbf{v}; \eta_1).$$

$C'_2 = \text{COM}(\Gamma_\phi(\mathbf{r}_w); \eta_2)$, $C'_3 = \text{COM}(\Gamma_\phi(\mathbf{z} \oplus \mathbf{r}_w); \eta_3)$ Then sends these three commitments to the verifier. After the verifier sends back the challenge Ch , the simulator sends the corresponding respond with respect to the challenge as follows:

- If the verifier sends $Ch = 1$, *Sim* sends RSP computed $\text{RSP} = (\Gamma_\phi(\mathbf{z}), \Gamma_\phi(\mathbf{r}_w), \eta_2, \eta_3)$.
- If the verifier sends $Ch = 2$, *Sim* sends $\text{RSP} = (\phi, \mathbf{z} \oplus \mathbf{r}_w, \eta_1, \eta_3)$.
- If the verifier sends $Ch = 3$, *Sim* outputs $\perp$ and abort.

In the three moves, **COM** is statistically hiding, the distribution of the commitments **CMT** is statistically close to the ones in the real interaction and the verifier uniformly random chosen Ch in $\{1, 2, 3\}$, so it works the same as in the real one. Hence, the probability that the protocol is abort $\perp$ is $1/3$. When the protocol does not abort, the simulator outputs accepted transcript as in the real interaction. Then the simulator succeeds in impersonating the honest prover with probability $2/3$.

Argument of Knowledge. Given 3 valid responses $\text{RSP}_1 = (\mathbf{t}_w, \mathbf{t}_r, \eta_2, \eta_3)$, $\text{RSP}_2 = (\phi_2, \mathbf{w}_2, \eta_1, \eta_3)$, $\text{RSP}_3 = (\phi_3, \mathbf{w}_3, \eta_1, \eta_2)$ to the same commitment $\text{CMT} = (C_1, C_2, C_3)$, for $Ch = 1, 2, 3$, respectively. Then we can derive the following from the six commit-

ments we have:

$$\begin{cases} \mathbf{t}_w \in \text{VALID}; \\ C_1 = \text{COM}(\phi_2, \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v}; \eta_1) = \text{COM}(\phi_3, \mathbf{M} \cdot \mathbf{w}_3; \eta_1); \\ C_2 = \text{COM}(\mathbf{t}_r; \eta_2) = \text{COM}(\Gamma_{\phi_3}(\mathbf{w}_3); \eta_2); \\ C_3 = \text{COM}(\mathbf{t}_w \oplus \mathbf{t}_r; \eta_3) = \text{COM}(\Gamma_{\phi_2}(\mathbf{w}_2); \eta_3). \end{cases}$$

As COM is computationally binding, we obtain

$$\begin{cases} \mathbf{t}_w \in \text{VALID}; \phi_2 = \phi_3; \mathbf{t}_r = \Gamma_{\phi_3}(\mathbf{w}_3); \mathbf{t}_w \oplus \mathbf{t}_r = \Gamma_{\phi_2}(\mathbf{w}_2); \\ \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v} = \mathbf{M} \cdot \mathbf{w}_3. \end{cases} \quad (\text{A.1})$$

Since $\mathbf{t}_w \in \text{VALID}$, if set $\mathbf{z} = [\Gamma_{\phi_2}]^{-1}(\mathbf{t}_w)$, then $\mathbf{z} \in \text{VALID}$. Furthermore, we have

$$\Gamma_{\phi_2}(\mathbf{z}) \oplus \Gamma_{\phi_2}(\mathbf{w}_3) = \Gamma_{\phi_2}(\mathbf{w}_2) \bmod 2,$$

which implies that $\mathbf{z} \oplus \mathbf{w}_3 = \mathbf{w}_2$, and that $\mathbf{M} \cdot \mathbf{z} \oplus \mathbf{M} \cdot \mathbf{w}_3 = \mathbf{M} \cdot \mathbf{w}_2$. which means $\mathbf{M} \cdot \mathbf{z} = \mathbf{v}$, with $\mathbf{z} \in \text{VALID}$. $\square$

Remark A.0.4. In many concrete instances of the protocol, the vector $\mathbf{t}_w = \Gamma_{\phi}(\mathbf{w})$ sent by the prover in the case $Ch = 1$ is fully determined by $d_0 \ll d$ bits. Then, the prover can save the communication cost by sending only the d_0 bits that are sufficient for the verifier to determine $\mathbf{t}_w$.