Privacy preserving authentication

Using advanced cryptographic protocols

Trung Dinh

Supervisor: Dr. Ron Steinfeld

Supervisor: Dr. Nandita Bhattacharjee

Faculty of Information Technology

Monash University

This dissertation is submitted for the degree of

Doctor of Philosophy

January 2018

I would like to dedicate this thesis to my loving family.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Trung Dinh
January 2018

Acknowledgements

And I would like to acknowledge ...

Abstract

In recent years, lattice-based cryptography has been becoming one of the most active research topic in cryptography. Many security mechanisms are adapting the technique to improve and provide themselves with extra usability and security features, this research focus on user authentication aspect, especially privacy preserving biometrics authentication. Many known techniques use the homomorphic operations of some lattice-based cryptosystems to provide privacy preserving feature to the authentication protocol. However, most of them still need to rely on a trusted third party to decrypt the authentication result. Many protocols assume Honest-But-Curious security model for the communicating parties while the client should be modeled as malicicous in authentication scenarios. It is desirable to have a solution that provides all of such security features while still keeping the protocol working under practical performance.

In this thesis, we construct a series of protocols with strong security guarantees and reasonable computation time and communication size, that can be applied in any Hamming Distance based biometric authentication system. Our constructions rely on different state of the art cryptographic techniques designed to work with lattice-based cryptosystems.

Table of contents

List of figures **xv**

List of tables **xvii**

1 Introduction **1**

1.1 User Authentication and Biometrics 1

1.2 Privacy-preserving Biometric Authentication 2

1.3 Contributions of the thesis 5

1.4 Organization of the thesis 6

2 Definitions and Preliminaries **7**

2.1 General Notations 7

2.2 Lattices 8

2.2.1 The motivations 8

2.2.2 Lattice Preliminaries 8

2.2.3 q-ary lattices and SIS problem 11

2.2.4 A cryptographic example 12

2.2.5 Security of lattice-based cryptography 13

2.2.6 An example of parameters choice 18

2.3 Learning With Error 20

2.3.1 Definitions 21

2.3.2 Security of LWE 22

2.3.3 Symmetric cryptosystem from LWE 22

2.3.4 Assymmetric Cryptosystem from LWE 25

2.3.5 Efficiency of Lattice-based cryptosystems 29

2.3.6 The Ring Variant Systems - RLWE 33

2.4 Homomorphic Cryptosystems 36

2.4.1 Homomorphic Encryption 36

2.4.2 Ciphertext packing . . . 37
2.5 Zero Knowledge Proof Systems . . . 38
2.5.1 Zero Knowledge Proofs and ISIS problem . . . 38
2.5.2 SternExt Protocol . . . 39
2.6 Garbled Circuit and Oblivious Transfer . . . 40
2.6.1 Garbled Circuit . . . 40
2.6.2 Oblivious Transfer . . . 41
2.7 The Syntax and Security model . . . 43
2.8 Conclusion . . . 48

3 The First Protocol - Computing HD Homomorphically 49
3.1 Introduction . . . 49
3.2 Related Work . . . 50
3.3 The BGV Cryptosystem and Yasuda Packing Method . . . 51
3.3.1 Feature codes and HD Computation homomorphically . . . 51
3.4 Zero-knowledge proof system . . . 52
3.5 Stern-based variant ZKP . . . 52
3.5.1 Stern-based ZKP . . . 52
3.6 Our Protocol . . . 56
3.7 Security Proofs . . . 58
3.7.1 Security Proof for Theorem 15: Privacy against Server . . . 58
3.7.2 Security Proof for Theorem 16: Type I Impersionation attack . . . 59
3.7.3 Security Proof for Theorem 16: Type II Impersionation attack . . . 61
3.8 Result Evaluation . . . 62
3.8.1 Parameters . . . 62
3.8.2 Limitations and open problems . . . 63
3.9 Conclusion . . . 63

4 The Second Protocol - Covering Circuit Privacy 65
4.1 Introduction . . . 65
4.2 Previous Works . . . 65
4.3 Renyi Divergence Analysis technique . . . 65
4.3.1 Correctness Analysis . . . 69
4.4 Results . . . 70

5 The third protocol - Computing and Comparing HD Homomorphically 71
5.1 Introduction . . . 71

5.2 The Homomorphic tools 72
5.2.1 Extracting the Most Significant Bit homomorphically 72
5.2.2 Proof of lemma 9 73
5.2.3 Converting from binary-encoded to unary-encoded plaintext 73
5.3 The Zero Knowledge Tools 74
5.3.1 ZKPoPK of Regev Cryptosystem 74
5.3.2 ZKPoPK of BV cryptosystem 76
5.3.3 ZKP of plaintext with zero coefficients 77
5.3.4 ZKP of re-encryption correctness 79
5.3.5 Schnorr-based ZKP technique 80
5.3.6 ZKP and $\sum$-Protocol 83
5.3.7 Combination of ZKP 84
5.4 Proposed Scheme 86
5.4.1 The protocol 87
5.5 Security Proofs 90
5.5.1 Security Proof for Theorem 16: Type I Impersionation attack 90
5.5.2 Security Proof for Theorem 16: Type II Impersionation attack . . . 92
5.5.3 Security Proof for Theorem 15: Privacy against Server 93
5.6 The algorithms 94
5.7 Evaluation and Results 94

6 The fourth protocol - Removing The Overhead 97
6.1 Introduction 97
6.2 The Challenge-Response Technique 98
6.3 The BGV Cryptosystem and CRT packing method 99
6.3.1 The rotate and permute function 101
6.3.2 The flattening gadget 103
6.3.3 The keyswitch and modswitch operation 103
6.4 HD computation 103
6.5 Garbled Circuit and Oblivious Transfer 103
6.5.1 Oblivious Transfer 103
6.5.2 Yao Garbled Circuit 103
6.6 The details 104
6.6.1 The protocol description 104
6.6.2 The Garble Circuit and Oblivious Transfer 104
6.6.3 HD Computation Homomorphically - BGV 109
6.7 Results 109

References **111**

Appendix A How to install LaTeX **119**

Appendix B Installing the CUED class file **123**

List of figures

2.1 Example of a 2-dimensional lattice 9
2.2 A paralellogram of 2-dimentional lattice 10
2.3 Example of GSO in 2-dimensions 14
2.4 BKZ parameters 18
2.5 Choosing best m' for SIS attack for parameters $\delta = 1.01, q = 4416857, n = 100$ 19
2.6 Garbled Circuit example 41
2.7 Minion 42
2.8 OT protocol based on DH 43

5.1 ZKP for BV cryptosystem relation 85
5.2 Sigma Protocol 86
5.3 AND-Composition for ZKP-BV 87
5.4 88

6.1 The fourth protocol 105
6.2 Substractor Circuit 106
6.3 Half-substractor and Full-substractor 106
6.4 Comparator Circuit 107

List of tables

Chapter 1

Introduction

1.1 User Authentication and Biometrics

User authentication is the process of verifying the claimed identity of a user, which is an important aspect in the big picture of information and network security. Generally, there are three factors that can be used for this purpose: Something You Know (SYK), such as password; Something You Have (SYH), such as smartcard and Something You Are (SYA), such as iris or fingerprint. The last factor, also known as biometrics, can be considered the most usable factor as one does not have to carry or remember anything during authentication. Biometric authentication is lately deployed widely in many platforms thanks to the reduced cost of hardware.

Various traits from body have been proposed and used for person recognition and verification. The three most popular traits deployed are fingerprint, face and iris modalities. Traits such as palmprint, hand geometry, voice are deployed in some commercial applications. There are many other traits proposed by researchers (such as gait, ear, keystroke dynamics, etc) but yet to attain sufficient level of technological maturity for deployment. Regardless of the traits used, in biometrics verification systems, there are generally 2 stages: enrollment and authentication. In the enrollment stage, a user registers his biometric template with a server, later in the authentication stage, he extracts his template again and submit to the server to authenticate. Unlike password authentication where a user always enters the same password he used when registered, in biometrics system, the query template of a genuine user is only very similar to the registered one due to many factors (for instance, the user might touch the fingerprint sensor in different corners when authenticating). The server has to compute some kind of distance between the registered and the query templates and compares the result with some threshold values to decide the authentication results. Compared to SYK and SYH, the SYA factor has these 2 parameters, namely False Acceptance Rate (FAR) and False Rejection

Rate(FRR): FAR is the rate that the system wrongly accepts an impersonating user, FRR is the rate that the system rejects a genuine user. Different biometrics authentication systems use different extraction methods and distance measures to reduce FAR and FRR, this is also an active research area. In this thesis, we do not focus on template extraction techniques or improving FAR and FRR but we are interested on methods to provide extra privacy and security features to any methods that are based on Hamming Distance of biometrics represented by bitstrings.

1.2 Privacy-preserving Biometric Authentication

Many biometrics authentication systems do not have protection for the data stored in server. A server breach incident would result a catastrophic consequence (in 2015, nearly 6 millions of plaintext fingerprint data of US goverment employees were leaked due to a security attack [OPM]). We see that the main drawback of SYA authentication is lack of biometric privacy against server exposure. If biometric data is ever revealed or stolen, the victims may be vulnerable to impersonation attacks for the rest of their life, as it is nearly impossible to change one's fingerprints or iris, unlike password or smartcard approaches. Hence there is a strong motivation to develop authentication systems that protect the privacy of biometric data. There are four main approaches to privacy preserving biometrics authentication at present (details avalable in surveys of [43] and [42])

- Transforming biometric data using a key-less many-to-one transformation, also known as 'Non-Invertible' transforms (e.g. [70]): Since the function is key-less, biometric impersonation security under server exposure is vulnerable to off-line brute-force attacks, also known as FAR attacks [84, 74], that are the biometric analogue of off-line dictionary attacks on weak password hashing. Moreover, the many-to-one mapping trades-off the biometric accuracy (FRR) performance of the biometric scheme. A solution to overcome such issue is to introduce a cryptographic secret key authentication factor, i.e transforming the registered biometric template stored on the server by encrypting it with a secret key stored at the client's side. With this second authentication factor, verification will require the secret key, which prevents brute-force attacks. Generally speaking, two-factor security can be obtained if the system is secure when either the secret key is exposed (e.g., client's device is stolen), or if the encrypted biometric stored in the server is leaked (e.g., server breach). All other approaches we consider below are based on this two-factor method.

- Transforming biometric data using a keyed 'distance-preserving' encryption scheme: Here, the distance between the queried and registered biometric templates is compared by the server using the encrypted query and registered templates. Typical techniques in this category include cancelable biometrics and biohashing ([83], [44], 2P-MCC ([15]).). However, the security of such encryption schemes is unclear as they are based on heuristic and non-standard cryptographic assumptions, and several have been found to be insecure (e.g. [51, 49]).

- Biometric cryptosystems / Fuzzy hashing: This approach is based on error correcting codes techniques ([85], [62]) to extract a noiseless cryptographic key from a noisy biometric. However, depending on parameter settings, these techniques may leak significant information, and the tradeoff between biometric accuracy (False Acceptance Rate and False Rejection Rate) and security are controversial or not well understood, and in practice the technique may needs strong restriction of accuracy. In particular, the underlying error correcting codes can only handle a specific range of the threshold depending on the usability parameters.

- Secure Computation / Homomorphic encryption techniques ([87], [78], [37]): In this approach, the biometrics data is encrypted in the server using a homomorphic encryption scheme. The operations on encrypted data would measure the similarity between the request and the stored templates. This approach has the potential to overcome the heuristic security issues of the above approaches (by using an encryption scheme based on standard cryptographic assumptions), as well as the biometric accuracy issues (by implementing homomorphically the same verification check as the underlying biometric scheme). However, existing protocols following this approach suffer from other significant drawbacks. In particular, earlier approaches [78] were based on the Paillier *additive* homomorphic scheme and offer a limited efficiency due to both the inefficiency of homomorphic operations of the Pailler scheme, and the extra protocol overhead required to handle the 'addition-only' homomorphism limitation of the Paillier scheme. Besides, Paillier system does not provide *long term* security (it is not quantum resistent). The efficiency is significantly improved by a recent protocol of Yasuda et al. [87] based on lattice-based SomeWhat Homomorphic encryption (SWHE) supports both homomorphic additions and multiplications. The cryptosystem used is believed to be quantum-resistant as well. However, this protocol requires the use of a trusted-third party server to verify the final authentication result: The main server stores encrypted biometric and compute HD homomorphically while the other server stores the decryption key and decrypts/verifies the HD. As a result, it is not secure under server exposure attack (as keys from trusted server can decrypt data on the main one).

We focus our work on the fourth approach with different improvements, the goal is to design authentication protocol that satisfies more complex security and usability requirements according to the current trends. The requirements include:

Privacy against server exposure. The biometric data should be securely encrypted prior to uploading to the server, using a key stored only to the biometric owner. Ideally, the server storing the encrypted biometric should not be able to comprehend the data, in other words, it does not have the encryption key.

Quantum resistant privacy. Considering quantum computing is developing fast and the biometric data will persist over the lifetime of a user, the encrypted biometric data should have long term security against quantum computing attacks.

No reliance on trusted third parties. With the wide exposure of cloud computing to potential attacks, the protocol should not rely on one or more cloud-based trusted third parties to help with the authentication process.

Security against malicious client. An attacker attempting to impersonate the real client to the server should not be able to authenticate without a genuine biometric, even if the attacker is malicious and does not follow the authentication protocol, i.e. the impersonation attacker cannot be assumed to be 'honest but curious'(HBC).

Two factor security. If the client is responsible for decrypting biometric-related data, the protocol should be multi-factor secure: an attacker with a compromised key should not be able to authenticate without a genuine biometric.

Practical performance. The computation time and communication size of the whole protocol should be within practical time frame.

Previous protocols in the literature do not meet one or more of our requirements. In particular, many previous protocols ([**?**], [**?**], [**?**]) assume honest-but-curious clients and are insecure in the authentication context that involves malicious clients. The few protocols involving malicious clients ([78], [77]) are not quantum-resistant. Previous practical quantum-resistant protocols ([87], [58]) are not secure against malicious client and involve the use of a trusted third-party verification server and therefore client privacy is not completely achieved.

1.3 Contributions of the thesis

We propose different protocols to support above requirements. Our technique combines state-of-the-art cryptographic tools such as Homomorphic Encryption (HE) and Zero-Knowledge-Proof (ZKP) to balance security and usability of the system. The techniques are all lattice-based, which is currently the best candidate for long term security against quantum attacks. Our protocol also achieves privacy against server exposure without relying on a trusted third party. The contributions include:

- Quantum resistant and provable-secure biometric authentication protocols that does not rely on trusted third-parties. (Section **??**). The server stores the encrypted data and does homomorphic operations to compute the distance (HD) to decide the authentication result. It does not need a third party to decrypt the encrypted HD but sending it to the client for decryption. The client uses ZKP to convince the server that the ciphertext was decrypted correctly.

- The protocols provide security under malicious client model, this is done by new ZKP techniques that we design specifically for different ciphertext packing methods . It is also applicable to do proof of plaintext knowledge for the BGV Homomorphic Encryption scheme ([11]) that we adapted. We have different ZKP variants are based on [82] and [75], which are the two main approaches for Zero Knowledge Proof.

- Due to the noise inherent inside lattice-based homomorphic encryption and its correlation with the evaluated ciphertext, we observe that there can be information leakage about the original plaintexts used in the homomorphic computations to a two-factor attacker that exposed the client's secret key. We propose an approach to cover such leakage without significantly reducing the efficiency of the protocol: the approach is a new application of Renyi Divergence (RD) based analysis to show the security of the protocol with a small 'imperfect' one-time pad. (Section **??**). This correlation of Homomorphic Encryption noise with the original plaintext before homomorphic evaluation was observed as a problem of "circuit privacy" in theoretical HE literature ([**?**], [**?**]), but the proposed solutions ([26], [**?**], [**?**]) involves 'smudging' (imperfect masking) or bootstrapping techniques with an exponentially large noise (in the security parameter) that reduces efficiency. In contrast, our Renyi-based method can use much smaller imperfect masks leading to better efficiency. This is the first application of Renyi divergence techniques to circuit privacy of HE to our knowledge.

- We propose a new Oblivious Transfer (OT) approach to obtain inputs to Garbled Circuits. Unlike traditional OT approaches where the receiver obtains keys as inputs to a garbled circuit, our approach is compatible with the undernearth homomorphic cryptosystem and obtain nicely the encryptions of the keys to the circuit. The approach is very effective with only one level of homomorphic multiplication operation and can be applied to other application scenarios where secure multy-parties computation is required.

1.4 Organization of the thesis

The thesis is organized as follows:

In Chapter 2, we describe the general notations and recall the basics of Lattice-based cryptography, Authentication, Zero Knowledge Proof and Secure Multiparties Protocols that are used throughout the thesis.

In Chapter 3, we present our first variant of the protocol that removes the role of trusted third parties in a privacy preserving authentication system.

In Chapter 4, we present the new technique to cover circuit privacy while still keeping the parameters of the homomorphic cryptosystems being in practical thresholds.

In Chapter 5, we construct the third variant of the protocol to do both of the tasks computing and comparing the Hamming Distance homomorphically (The previous protocols only compute HD on ciphertexts).

In Chapter 6, we present the last variant of the protocol, which removes the computation and communication overheads with the extra cost during the initialization stage (one time only cost).

In the last Chapter, we conclude the thesis and stat the open problems for future research.

Chapter 2

Definitions and Preliminaries

2.1 General Notations

We use the following notations throughout the thesis:

Vectors, Matrices We denote column vectors by standard vector notation (e.g., $\vec{x}, \vec{y}$) and matrices by bold upper-case letters (e.g., $\mathbf{A}, \mathbf{B}$). We write $\vec{b}^T$ to convert a vector $\vec{b}$ to a row vector. The length of a vector $\vec{b} = [b_0, b_1, \ldots, b_{n-1}]$ (*norms*) is denoted by $||\vec{b}|| = \sqrt{b_0^2 + b_1^2 + \cdots + b_{n-1}^2}$ for Euclidean norm and $||\vec{b}||_\infty = max(|b_i|)$ for infinity norm. The i component of a vector is denoted interchangebly as b_i or $b[i]$.

Groups and Rings Sets of numbers are denoted as standard (e.g., $\mathbb{R}$ for set of rational numbers, $\mathbb{Z}$ for integers), we denote $\mathbb{Z}_q$ to be the set of integers modulo q (in the range $(-q/2, q/2]).\mathbb{Z}[n]$ denotes polynomials of order n with elements in $\mathbb{Z}$, R_q denotes the ring of polynomial modulo $x^n + 1$: $R_q = \frac{\mathbb{Z}_q[x]}{x^n+1}$. We use bold lower-case to denote an element of the ring R_q (e.g., $\mathbf{a} \in R_q$).

Others We use notation $x \overset{r}{\hookleftarrow} D$ when x is sampled uniformly random from the distribution D. If x is the output of an algorithm D, we write $x \leftarrow D$. Algorithms are $efficient$ if they run in probabilistic polynomial time (PPT). A function is said *negligible* in n if it vanishes faster than the inverse of any polynomial.

2.2 Lattices

2.2.1 The motivations

We choose lattice-based cryptography as the main techniques used in the thesis for several reasons:

Provable Security. Lattice-based cryptosystems' security can be related to known computationally hard problems such as Shortest Vector Problem (SVP) or Closest Vector Problem (CVP) and therefore we can demonstrate the security property in terms of mathematical proofs. For classical system such as RSA, there is no proof showing that breaking RSA is as hard as factoring an integer even though in many cases the best attack we know is to do such operation.

Quantum Resistance. In 1994, Shor [79] showed that we can write a program with quantum computers to factor a large integer into prime factors.The latest D-Wave systems initial deployments show that the idea is feasible in near future. If this happens, it has significant implications to the currently used public key cryptosystems such as RSA: they will be broken. Lattice based cryptography is currently one of the best candidates that resists even quantum attacks.

Homomorphic Operations. Finally, lattice-based cryptography allows us to do operations that we could not do before: computing on encrypted data. This special type of operatation can provide privacy features in online communications.

Research Activities. Lattice-based systems have been researched actively in the last decade and the development has come to the stage that it can be implemented and can be used in the foreseable future.

Performance. Lattice-based cryptography has the potential to give us very fast operations and it is a good option for high performance required cryptographic application

We will recall some of the mathematics behind lattices (they are actually mathematical objects). We will look at the definitions and some of the hard problems that we will base the security on and how to use lattice to build cryptographic schemes.

2.2.2 Lattice Preliminaries

This is an area of mathematics that combines both matrices, vector algebra and integer variables. The definitions include:

Definition 1 (Lattice)**.** *An n-dimensional (full-rank)* lattice $L(B)$ *is the set of all integer linear combinations of some* basis *set of linearly independent vectors* $\vec{b_1}, \dots, \vec{b_n} \in \mathbb{R}^n$*:*

$$L(B) = \left\{ c_1\vec{b_1} + c_2\vec{b_2} + \dots + c_n\vec{b_n} : c_i \in \mathbb{Z}, i = 1, \dots, n \right\}$$

The $n \times n$ *matrix* $B = (\vec{b_1}, \dots, \vec{b_n})$ *is called a basis for* $L(B)$*. There are infinitely many different bases for a lattice.*

Example. Consider a 2-dimensional lattice generated by the basis $(\vec{b}1, \vec{b}2) = \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} \right)$ To generate a lattice from a basis, we compute all the integer multiples of each vector of the

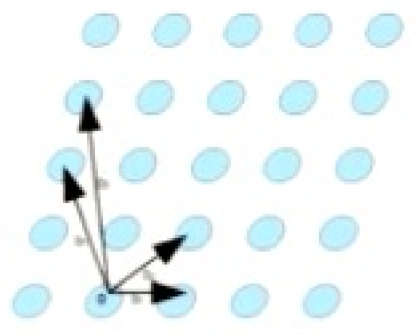


Fig. 2.1 Example of a 2-dimensional lattice

basis, then we add them together to generate new vectors. The result is an infinite grid of vectors, this grid is a lattice. We can illustrate two or three dimensional lattice easily, for higher dimensions, it is not possible to draw the lattice, but the intuition stays the same and all the algebraic operations can be done easily on the coordinates of the vectors. In lattice-based cryptography, we work with high dimension lattices ($n > 100$) for security properties. In fact, we can also say that a lattice is an infinite group (a group is a mathematical structure, a set with one operation on it.) with the main operation is addition.

Definition 2 (Fundamental Paralellepiped)**.** *For an n-dimensional lattice basis* $B = \left(\vec{b_1}, \dots, \vec{b_n}\right) \in \mathbb{R}^{n \times n}$*, the* fundamental paralellepiped *(FP), denoted* $P(B)$*, is the set of all real valued* $[0,1)$*-linear combinations of some basis set of linearly independent vectors* $\vec{b_1}, \dots, \vec{b_n} \in \mathbb{R}^n$*.*

In order to determine if a set of vectors is a basis of a lattice, there are geometric and algebraic approaches.

Lemma 1. *There is exactly one* L *point contained in P(B') (the* $\vec{0}$ *vector) if and only if B' is a basis of L.*

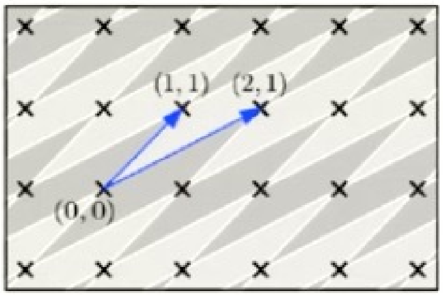


Fig. 2.2 A paralellogram of 2-dimentional lattice

Lemma 2. *B' is a basis of L(B) if and only if B'=B.U, for some $n \times n$ integer matrix U with $det(U) = \pm 1$, such U is called unimodular matrix.*

Where $det(U)$ denotes the determinant of the matrix U. Geometrically, the determinant computes the area (or volume) of the paralellogram of a lattice. We can associate each point of a lattice with a paralellogram, consequently, for a large n-dimensional ball *S*, the number of lattice points in *S* is $\approx vol(S)/det(L)$.

Definition 3 (Determinant)**.** *For an n-dimensional latice $L(B)$, the determinant of $L(B)$, denoted $det(L(B))$ is the n-dimensional volume of the FP P(B).*

Lemma 3. *For an n-dimensional lattice $L(B)$, we have $det(L(B)) = |det(B)|$.*

In cryptography, we are interested in computationally hard problems, there are some problems related to geometry of lattices that are hard, including Shortest Vector Problem (SVP) and Closest Vector Problem (CVP).

Definition 4. *For an n-dimensional lattice L, its* minimum $\lambda(L)$ *is the length of the shortest non-zero vector of L:*

$$\lambda(L) = min(\|\vec{b}\| : \vec{b} \in L\,0$$

Theorem 1 (Minkowski's First Theorem)**.** *For any n-dimensional lattice L, we have $\lambda(L) \leq \sqrt(n).det(L)^{1/n}$.*

As mentioned, the problem of finding SVP of a lattice is hard as the dimension grows, in theoretical computer science, this problem is categorized as NP-hard. In cryptography, we do not use SVP as it is, we use a variant of the problem, the definition of approximate SVP problem $\gamma - SVP$ follows.

Definition 5 (γ-SVP problem)**.** *Given a basis B for an n-dimensional lattice L, find $\vec{b} \in L$ with $0 \leq \|\vec{b}\| \leq \lambda(L)$.*

As γ increases, the problem becomes easier. We have good algorithms to find γ-SVP as follows

- For $\gamma \geq 2^{O(n)}$: LLL algorithm solves in Poly(n) time.
- For $\gamma \leq O(1)$: NP-hard, it is not likely that there exists Poly(n) algorithm to solve the problem.

In cryptography, many systems depend on this γ-SVP problem, where γ is in between the two extremes $\gamma \approx O(n^c)$. Even with this factor, the best algorithm (including quantum ones) still take $2^{O(n)}$ time to solve γ-SVP. (In classical cryptosystems for example RSA, the best algorithm to solve the integer factorization problem take $\approx O(2^{n^{1/2}})$ in time, this is why we need 4096 bits key to obtain about 64 bits of security for lattices).

2.2.3 q-ary lattices and SIS problem

There are many useful subclasses of special lattices, in cryptography, we often use one of them called *q-ary* lattices, which relate to a problem called Short Integer Solution - SIS (This is a hard problem used in many cryptographic design, our ZKP technique is also based on a variant of this problem). Recall that our motivation for lattice-based cryptosystem is the hardness of $\gamma - SVP$ problem given a basis B. It is critical to initially chose the basis B to make sure that solutions to such hard problem is hard to find. (As there are easy instances of $\gamma - SVP$, even for $\gamma = 1$. We need a way to generate *random* lattices' bases for which $\gamma - SVP$ is hard to solve 'on-average'. One possible answer is from [2]: a random *q-ary* lattice, based on this, we can prove how hard it is to find short vectors in it.

Definition 6 (Ajtai's perp lattices)**.** *Given an integer* q *and a uniformly random matrix* $A \in \mathbb{Z}_q^{n\times m}$*, the* q-ary *perp lattice* $L_q^{\perp}(A)$ *is defined by:*

$$L_q^{\perp}(A) = \left\{ \vec{v} \in \mathbb{Z}^m : A.\vec{v} = \vec{0} \mod q \right\}$$

It might not be clear from the definition of lattices, but the set of all vector $\vec{v}$ actually forms a lattice. They can be generated by some basis, and there is a fast algorithm to compute such basis. The parameters for these lattices are A and q, we can choose a lattice to work on by selecting random values for the parameters. This particular set of *q-ary* lattices is nice because Ajtai mathematically proved that: If there is an algorithm to break the $\gamma - SVP$ problem for these lattice, then we could use the algorithm to break $\gamma - SVP$ for any lattices. This is the *worst-case to average-case* security reduction and gives us confidence that the problem is hard on average.

Definition 7 (Small Integer Solution (SIS) problem). *$SIS_{q,m,n,\gamma}$ Given n and matrix A sampled uniformly in $\mathbb{Z}_{||}^{\ltimes\times\gtrdot}$, find $\vec{z} \in \mathbb{Z}^{m\times n}$ such that $A\vec{v} = \vec{0}$ and $\|v\|_\infty \leq \beta$.*

Informally, SIS is just the name given to the $\gamma - SVP$ problem for the particular *q-ary* lattices. β is the approximation parameter, it tells us how short the lattice vector should be. Some relations between SIS and $\gamma - SVP$

- $det(L_q^{\perp}(A)) = q^n$
- By Minkowski's Theorem, $\lambda(L_q^{\perp}(A)) \leq \sqrt{m}$, for $m \geq n \log q$. So to solve SIS, we need to find vectors not much longer than $\sqrt{m}$.
- We can relate γ of SVP to β of SIS: $\gamma \approx \beta / \sqrt{m} q^{n/m}$.

2.2.4 A cryptographic example

Cryptographic hash functions have many applications in many security contexts to provide authenticity and integrity. One of the main properties of cryptographic hash functions is collision resistant: it is hard to find 2 message inputs that produce same hash value output. The current constructions being used in practice are not lattice-based but base on non-linear boolean functions. We discuss how to construct one variant from lattice and demonstrate the relevant techniques that will be used during in the security proofs and parameter choices of the thesis. The security of this function is directly related to the hardness of the SIS problem that we saw before

Definition 8 (Ajtai's Hash Function.). *Pick $A = (\mathbf{a_{i,j}}) \overset{r}{\hookleftarrow} \mathbb{Z}_q$ (A is the function 'public key'). Given $\vec{x} \in \mathbb{Z}^{mn}$ having 'small' coordinates ($\|\vec{x}\|_\infty \leq d$), the hash function out put is defined as*

$$g_{q,m,n,d,A}(\vec{x}) = A.\vec{x} \mod q$$

We see that from a long input vector, we obtain a short output vector, that is our hash value. We can choose the parameters to make sure that constraint is satisfied. The important question is how do we determine the security of this function, or how can we know that this function is collision resistant. We introduce this concept of security reduction, which is used frequently in lattice-based cryptography's security proofs. We show that if there was an algorithm to find a collision in this hash function, then we could use such an algorithm to solve a hard lattice problem. In our case, we know that SIS was analyzed by Ajtai and shown to be hard based on good foundations. What we are going to show here is, if we can find a collision efficiently in this hash function, we could also efficiently find a way to break the SIS problem, that would contradict the hardness of SIS. Based on this contradiction we can conclude that there must not exist any efficient algorithm to find collision for the hash function.

Theorem 2. *Collision-Resistance of g is at least as hard as $SIS_{q,m,n,\beta}$ with $\beta = 2d$.*

Proof. Suppose there was an efficient collision-finder attack algorithm CF for function g: Given a random instance (A,q), CF outputs $\vec{x_1} \neq \vec{x_2}$ such that $A\vec{x_1} = A\vec{x_2}$.

We then can use CF to build another algorithm S that solves SIS problem for any instance (A,q):

- Runs CF on (A,q) to get $\vec{x_1}, \vec{x_2}$.
- S outputs solution for SIS problem: $\vec{v} = \vec{x_1} - \vec{x_2}$.

The algorithm works because $A\vec{v} = A\vec{x_1} - A\vec{x_2} = \vec{0}$, i.e., $\vec{v}$ is in the lattice $L_q^{\perp}(A)$ and $\|\vec{v}\|_\infty \leq \beta$, where $\beta = 2d$ (When we add/substract vectors ,the result vector's length is upper bounded by the sum of the lengths of the vectors). Moreover, S is efficient as CF is efficient (run-time $T_S \approx T_{CF}$). $\square$

This is the first application of lattices in cryptography and it actually started the other modern techniques. The next important questions are

- How should we choose the parameters to get a given security level.
- How secure is SIS problem related to $\gamma - SVP$ problem.

2.2.5 Security of lattice-based cryptography

Lenstra–Lenstra–Lovász (LLL) and various improvements are the main ways to access the security of lattice-based systems (by access the security of underlying problems such as SVP .or $\gamma - SVP$). From there, it allows us to choose the size of the parameters for the problem (such as dimensions) in order to give guarantee of security against the best known attack.

We first recall this important concept related to lattices that is used in LLL: Gram-Schmidt Orthogonization (GSO)

Definition 9 (GSO). *For a lattice basis $B = \left(\vec{b_1}, \vec{b_2}, \ldots, \vec{b_n}\right)$, its GSO is the matrix of vectors $B^* = \left(\vec{b_1^*}, \vec{b_2^*}, \ldots, \vec{b_n^*}\right)$ defined by $\vec{b_1^*} = \vec{b_1}$ and for $i \geq 2$, $\vec{b_i^*} = \vec{b_i} - \sum_{j=1}^{i-1} \mu_{i,j}.\vec{b_j^*}$, where $\mu_{i,j} = \frac{\langle \vec{b_i}, \vec{b_j^*} \rangle}{\langle \vec{b_j^*}, \vec{b_j^*} \rangle}$.*

GSO is basically a way to convert a basis that we have (that are not orthogonal) into another set of vectors are orthogonal. Figure 2.3 shows an example of GSO in 2-dimension GSO is used as part of the LLL algorithm to find short vector of a lattice. Note that the

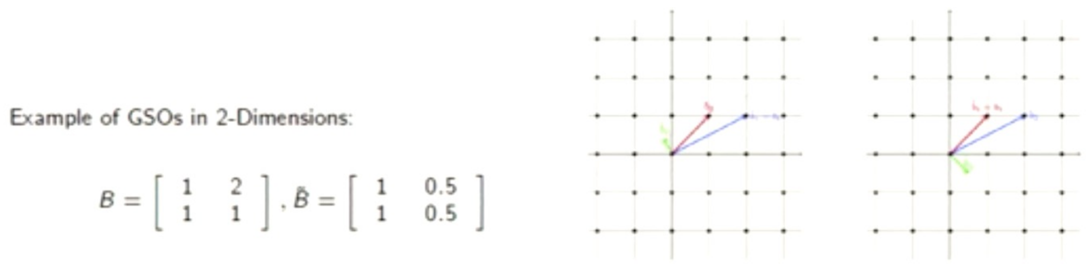

Example of GSOs in 2-Dimensions:
$B = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}, \tilde{B} = \begin{bmatrix} 1 & 0.5 \\ 1 & 0.5 \end{bmatrix}$

Fig. 2.3 Example of GSO in 2-dimensions

$GSO(B)$ is not another basis for the original lattice, it is just a related matrix, it has the same determinant as the original one.

LLL algorithm

Lenstra–Lenstra–Lovász (LLL) algorithmnot only finds short vectors but also manipulate the basis of the lattice into another basis of the lattice. Given some basis as a input to LLL (whose vectors are long), LLL makes sequential modifications to the basis, every modification keeps it still a basis of the lattice and improve it a little bit. When the improvement is repeated, the sizes of the vectors of the basis are also reduced until it ends with a matrix that cannot be improved anymore. That is the output of LLL: a basis of the lattice that is much better than the original input, in the sense that it has much shorter vectors.

In more details, LLL looks at the property of the GSO of the basis to make the basis vectors 'approximately' orthogonal. The size of the GSO basis tells us the length of the projections of one vector along the earlier vectors. The larger these are, the less orthogonal the basis vectors are because they have bigger components along the earlier vectors. LLL tries to reduce those coefficient (projection coefficients), to make them more orthogonal, that will also make them shorter. There are 2 tests that LLL does to see how orthogonal the basis vectors are

LLL property 1. Take the current basis of the lattice and computes the GS coefficients $\mu_{i,j}$. It checks $|\mu_{i,j}| \leq 1/2$, that means the length of the projections on the previous vectors are at most half of the actual full vector. If this condition is not satisfied, LLL modifies the basis until the condition passes.

LLL property 2. The algorithm looks at the large remaining component $||\vec{b^*_{i+1}}||$ of $\vec{b_{i+1}}$ after removing components along $\vec{b^*_j}$'s $(j < i+1)$ and check

$$||\vec{b^*_{i+1}} + \mu_{i+1}\vec{b^*_i}||^2 \geq \delta.||\vec{b^*_i}||^2$$

for all $i = 1,\ldots,n-1$ and for some constant $\delta(1/4 \leq \delta \leq 1)$.

Informally, the second check is a bit more complex: LLL looks at the last 2 components of each vector. If we take a vector $\vec{b_i}$ and decompose it into all the projections along the previous $\vec{b^*_j}$ and check the the sum of the orthogonal component and the one previous to that: For example, if we look at $\vec{b_{i+1}}$, then we check $||\vec{b^*_{i+1}} + \mu_{i+1}\vec{b^*_i}||^2$. If this quantity is big enough, it is an indication that we have a lot left over after we remove the components along the previous one. The bigger this length is, the more orthogonal the basis becomes. The algorithm tries to make it big enough. The parameter δ of the algorithm decides this magnitude (the larger δ is, the better result returned from the algorithm).

This check is also done for all the vectors of the basis. If it is not satisfied, then again LLL does some changes to make sure the test passes.

When the algorithm runs, the 2 checks keep fighting between each other. Eventually, it hopes to get to a stage where both of the checks satisfy simultaneously and the algorithm stops and outputs the basis. (Algorithm 1)

Definition 10. *A basis B for lattice L is $\delta - LLL$ reduced if both LLL properties 1 and 2 are satisfied.*

Algorithm 1 LLL algorithm

```
1: procedure LLL($B$)
2:     Compute GSO $B*$ for B.
3:     for $(i = 2,\ldots,n)$ do
4:         for j = i - 1 to 1 do
5:             $c_{i,j} \leftarrow \lceil \frac{\langle \vec{b_i}, \vec{b^*_j} \rangle}{\langle \vec{b^*_j}, \vec{b^*_j} \rangle} \rfloor$
6:             $\vec{b_i} \leftarrow \vec{b_i} - c_{i,j}\vec{b_j}$
7:     if $||\vec{b^*_{i+1}} + \mu_{i+1}\vec{b^*_i}||^2 \geq \delta.||\vec{b^*_i}||^2$ then
8:         Swap $\vec{b_i}$ and $\vec{b_{i+1}}$.
9:         Go back to step 2.
10:    else
11:        Return B
```

Note that in the algorithm we take the rounded multiple of $\vec{b}_j$, this important operation helps the algorithm maintaining the vectors to be basis of the lattice. As the lattice consists of only integer multiples of all the basis vectors: if we add an integer multiple of $\vec{b}_j$ to $\vec{b}_i$, we still end up with another lattice vector, we can also show that the new matrix is also a basis of the lattice. In other words, if we add fraction of $\vec{b}_j$, it is not lattice point anymore. Also, due to this rounding error, instead of having a precisely orthogonal vectors in the result matrix, we only obtain an approximate orthogonal matrix. For more intuition of the algorithm, we refer reader to [52]. Last but not least, the loop of the algorithm was shown to always terminate in polynomial time!

Theorem 3 (LLL run time)**.** *The number of iterations of LLL on an input basis B before termination is at most*

$$n^2 . \log(\max \|\vec{b}_i\|) / \log(1/\sqrt{\delta}).$$

For any constant $1/4 < \delta < 1$*, this is polynomial in bit length of the algorithm input. Moreover, the run-time for each iteration is also polynomial in the input bit length. Overall, run-time is polynomial in input length, or LLL is efficient!*

LLL solution to $\gamma - SVP$

The question is how short are the vectors of LLL's output, or what kind of approximate SVP can LLL solve: If we run LLL on some basis, we get back a reduced basis, and we take the shortest vector in that result basis. How long is that vector compare to the shortest vector in the lattice.

Theorem 4 (Short vector from LLL)**.** *. The LLL algorithm solves in polynomial time* $\gamma - SVP$ *for* $n - dim$ *lattices, with* $\gamma < 2^{(n-1)/2}$*.*

In summary, the first vector of LLL output is bounded by

$$\|\vec{b}_1^*\| \leq (1/(\delta - 1/4))^{(n-1)/2} . \lambda(L)$$

We can see that the approximation factor is exponential in the dimention n of the lattice. Another way to measure how short of output vector of LLL is Hermite Factor (HF), which is the ratio of the algorithm's output with the n^{th} root of the determinant of the lattice. Sometimes it is more convinient to use this measure because the determinant of the lattice sometimes is easy to compute, whereas the length of the shortest vector of the lattice is not, in general. In practice we usually use this as a main measure. Again, LLL gives HF that is exponential in the dimension n. The conclusion is, although LLL runs in polynomial time, but the approximation factor increases exponentially fast with the dimension. When the dimension is large, the approximation factor will grow so fast that LLL is not useful to break the security of underlying cryptosystems.

Theorem 5. *The LLL algorithm solves in polynomial time $\gamma - SVP$ for $n - dim$ lattices, with $\gamma \leq 2^{(n-1)/2}$. Can also be shown that Hermite Factor $\gamma_{HF} = \frac{\|\vec{b_1}\|}{det(L)^{1/n}} \leq 2^{(n-1)/4}$*

LLL in practice

This section discuss the trade-off between the running time and the length of the output of the attack. This helps us in selecting parameters for state-of-the-art lattice attack

In practice we try to improve this bound so LLL can get shorter vector. One way to do that is to use the fact that, in our analysis, we do worst-case analysis on the output that LLL can give. In practice, they tend to be shorter than that. When we run it experimentally ([64]) on random lattice we will find that it perform much better than the bound proved. Specifically, HF is close to 1.02^{n-1}. Although this factor is still exponential in dimension n, this slowth growth allows LLL to be used with large dimension lattices. In order to reduce this HF further, we can further modify LLL. There are several ways to do that. The basic idea is to combine LLL with exhaustive search (enumeration algorithm). The best lattice exhaustive search algorithm (Fincke-Phost/Kannan) does it in time $2^{O(n \log(n)}$, it is even worst than exponential. There are variants of such method but they all take time at least exponential with the dimension n (with the expense of using $2^{O(n)}$ memory). In general, all enumeration algorithms are very inefficient. Therefore, combining LLL with bruteforcing is a more practical approach: BKZ is the typical algorithm in this category. BKZ allows trades off between run-time and approximation factor γ, LLL is modified by changing the blocksize k of vectors in the swapping step. As we increase k: We search in the lattice of dimension k using brute force search and then run LLL with smaller dimension k. We get the 'interpolation' between the two extremes of LLL (corresponding to $k = 2, \gamma = 2^{O(n)}, T = n^c$) and the enumeration (corresponding to $k = n, \gamma = 1, T = 2^{O(n \log(n))}$). In between, BKZ can achieve $\gamma(k) \leq k^{(n-1)/(k-1)}$ with running time $n^c.k^{O(k)}$. [36]. From this trade-off setting, an attacker then can choose the optimal value of k for particular context: A lattice-based cryptographic scheme can only be broken if the adversary can find short enought vectors. This also tells how small γ needs to be in order to defend against such attack:

In summary, given a security level λ for a system (i.e., it would take time 2^{λ} for the best attack to work even with the best choice of the parameter k for BKZ) . The best approximation factor γ that the algorithm can achieve is related to λ as:

$$\log(\gamma_{HF}) = \Omega(\frac{n\log^2\lambda}{\lambda})$$

where Ω represent asymptotic "lower than" condition when the parameters are large. Overall, we can conclude the *lattice 'rule of thumb'* for $\gamma - SVP$ using BKZ:

$$n = \Omega(\frac{\lambda}{\log^2\lambda}.\log\gamma_{HF}) \approx \lambda.\log\gamma_{HF}$$

Remark 1. *The dimension n of a lattice-based cryptosystem needs to be proportional to the product of bit-security level λ and the* log *(base 2) of the approximation factor γ_{HF}. This factor is a reason behind the long keys of such cryptosystems.*

For concrete values of parameters, Chen and Nguyen [17] gave numerial estimates for Hermite Factor and time for random lattices versus block size for optimized BKZ variant:

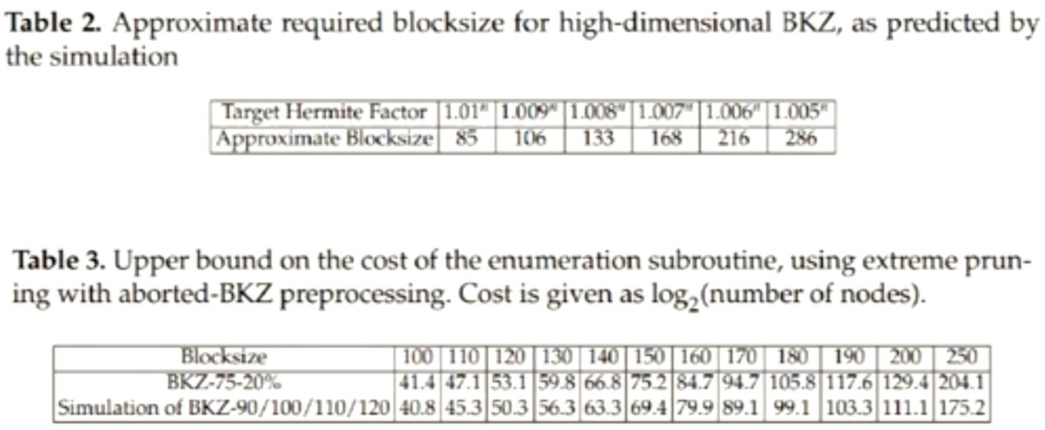


Table 2. Approximate required blocksize for high-dimensional BKZ, as predicted by the simulation

Target Hermite Factor	1.01^n	1.009^n	1.008^n	1.007^n	1.006^n	1.005^n
Approximate Blocksize	85	106	133	168	216	286

Table 3. Upper bound on the cost of the enumeration subroutine, using extreme pruning with aborted-BKZ preprocessing. Cost is given as $\log_2$(number of nodes).

Blocksize	100	110	120	130	140	150	160	170	180	190	200	250
BKZ-75-20%	41.4	47.1	53.1	59.8	66.8	75.2	84.7	94.7	105.8	117.6	129.4	204.1
Simulation of BKZ-90/100/110/120	40.8	45.3	50.3	56.3	63.3	69.4	79.9	89.1	99.1	103.3	111.1	175.2

Fig. 2.4 BKZ parameters

From table 2, for a given HF that we want BKZ to achieve, it gives us the block size parameter k that need to be used. Note that the HF is much smaller than what we had for LLL (whose $HF \approx 1.02^{n-1}$ in practice). Once we have the blocksize, we can look up in table 3 to know the running time of the algorithm.

2.2.6 An example of parameters choice

We discuss Ajtai's hash function as an example of how to choose parameters for some security level. We showed that if there is an algorithm that can break the collision-resistance property of the hash function, we can use it to find a short vector of the SIS problem with length $\beta = 2d\sqrt{m}$. The question is how do we choose the parameters q,n,m,d for the hash function to get a given security level λ based on the hardness of SIS? According to the BKZ state-of-the-art attack, to get running time of 2^{λ}, we can use the table 3 to derive the blocksize and corresponding Hermite Factor γ_{HF}. Then we know the best attacker can compute a non-zero vector $\vec{v}$ in SIS lattice $L_q(A)$ of norm $\leq l = min(q, \gamma_{HF}.det(L_q(A))^{1/m})$. Then we can compare l with β (or $2d\sqrt{m}$), if $l < \beta$, the attack will work and otherwise.

There are often ways to optimize such attacks where attacker choose parameters to minimize the his effort. For example in the attack of SIS problem of [60], the attacker can look at a subset $m' \leq m$ of columns of A, suppose that he can find a short vector $\vec{v'}$ such that $A'\vec{v'} = \vec{0} \mod q$. He can choose $\vec{v''} = \vec{0}$ and set $\vec{v} = \vec{v'} + \vec{v''}$. It turns out that there are optimal values that an attacker can choose for m' that minimize the length $l(m')$ for $\vec{v'}$. Specifically, $l(m') = min(q, \gamma'_{HF}.det(L_q(A))^{1/m'})$ is a function of m', when we graph this function, it is as illustrated in Figure 2.5. We can see that at some point m' there is some minimum value

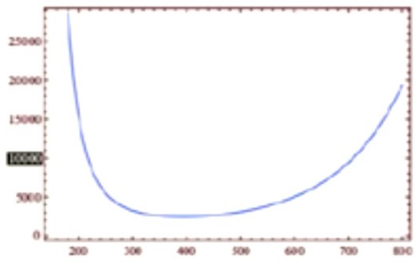


Fig. 2.5 Choosing best m' for SIS attack for parameters $\delta = 1.01, q = 4416857, n = 100$

for the length of the vector we can get with BKZ, this is the value that an attacker want to choose. Denote m^* to be this optimal selection, we have $l(m^*) = min(q, 2^{2.sqrtn \log q \log \delta})$. The condition that the attack fails is $l(m^*) > \beta$, so

$$q \geq \beta = 2d\sqrt{m} \text{ and } n \geq \frac{\log^2(\beta)}{\log q \log(\delta)}$$

We have looked at the best practical algorithms to attack lattice problems (LLL and its variants). In theory, there is also Ajtai's proof where it said there is a relationship between how hard it is to solve the SIS problem for random matrices. The connection between this kind of average case and the worst-case complexity of lattice problem in general (lattices that are completely unrelated to the *q-ary* ones) is the theoretical base behind lattice-based cryptography. We present Ajtai's average-case to worst-case connection Theorem (1996, improved by [28]).

Theorem 6 (Ajtai's hardness proof for SIS.)**.** *If there is an algorithm A that solves $SIS_{m,n,q,\beta}$ in poly-time, for some non-negligible fraction of input matrices $G \in \mathbb{Z}_{q}^{n\times m}$, then there is an algorithm B that solves $\gamma - SVP$ in polynomial time for all input lattices L of dimension n with:*

$$\gamma = O(\beta\sqrt{n}), q = \omega(\gamma\sqrt{\log n})$$

Although the theorem does not directly gives us concrete information to choose parameters to secure a cryptosystem like we discussed previously, it does give us the qualitative confidence in the security of the problem.

2.3 Learning With Error

In the last section, we discussed how to use lattice-based cryptography to construct a hash function, this section looks at another main tool in cryptography , that is encryption, or how to provide confidentiality. We need to use a slightly different type of lattice problem instead of the SIS problem: we introduce Learning With Error (LWE) problem, which is much more suitable than SIS for building the encryptions based on it. We will discuss how to use LWE to build symmetric key encryption as well as how to modify it to get public key encryption (Regev cryptosystem). This cryptosystem is a very important foundation, it is the basis for a lot of further developments in the area as well as the techniques used in the project.

In the SIS problem, given a matrix A, it is asked to find a short vector $\vec{v}$ such that $A\vec{v} = \vec{0}$. The cryptographic hash function that is based on SIS is constructed such that it can take arbitrary inputs and produces collision-resistance outputs. When we do that, we always have a many-to-one function nature in the construction: given an output, it is not possible to derive the original input. This many-to-one property is not useful in encryption context: we cannot decrypt the ciphertext to the original plaintext using such method. In encryption contexts, we usually have the reverse situation: the set of possible inputs might be a lot smaller than the set of possible outputs. The function serving encryption should be at least a one-to-one function that map a plaintext to a unique ciphertext. Or we can even have one-to-many functions that map a plaintext to many possible ciphertexts, as long as there is no intersection between the ciphertext output sets, we are still able to decrypt correctly. We see such situation in non-deterministic cryptosystems where the encryption function takes the message as well as a randomness factor as inputs. Actually, almost all encryptions being used in practice are somehow randomized, the main reason is to stop the leakage of information given many ciphertexts. A deterministic cryptographic scheme can be exploited if the plaintext space is small, for example, consider a situation where Alice sends deterministic encrypted messages to her agent Bob everyday to instruct him to sell or not to sell some of her companies' shares. An attacker capturing the ciphertexts of several days can easily distinguish such decisions without having to decrypt the messages at all. This is one of the important property that we want to have in cryptosystems: indistinguishability, also known as IND-CPA. This type of security requires that the adversary cannot distinguish the ciphertexts even when he know all the posibilities of the plaintexts.

We introduce LWE problem [71] that allows us to construct one-to-one or one-to-many functions. The main idea behind LWE is, given a secret vector and a random lattice point with some 'small noises' added to the point, it is infeasible for an attacker to learn anything from the output but it is easy for the data owner with the secret to revert back the original point. The description of LWE follows.

2.3.1 Definitions

Fix integers q, m, n and initialize the matrix $A^{m\times n}$. Note that the matrix A is a bit different from SIS regarding its dimension, this is just for convenient representation purpose. q is still the modulus of $a_{i,j}$, where $a_{i,j} \overset{r}{\hookleftarrow} \mathbb{Z}_q$. Let $\vec{s}^T = [s_1, s_2, \ldots, s_n]$ be a vector of independent uniformly random elements of $\mathbb{Z}_q$, this is the secret element that corresponds to the secret key of the cryptosystem. Let $\vec{e}^T = [e_1, \ldots, e_n, \ldots, e_m]$ be a vector of independent 'small' integers, each sampled from a probability distribution $\chi_{\alpha q}$. By 'small', we mean $\|e_i\|_\infty \leq \alpha.q$ for some parameter $0 < \alpha < 1$. Typically, $\chi_{\alpha q}$ is chosen from a Normal (Gaussian) distribution with standard deviation $\approx \alpha.q$. (From the implementation point of view, one way to do such sampling efficiently is to take the real samples from a normal continuous distribution and round it to the neareast integers, the mean is set to 0 to make sure the samples are small).

We discuss two variants of the LWE problems to be used in cryptosystems: Search-LWE and Decision-LWE problems.

Definition 11 (Search-LWE Problem.). *Given q, m, n, α and a matrix $A \overset{r}{\hookleftarrow} \mathbb{Z}_q^{m\times n}$ and $\vec{y} = A.\vec{s} + \vec{e} \mod q$ (with $\vec{e} \overset{r}{\hookleftarrow} \chi_{\alpha q}^m$ and $\vec{s} \overset{r}{\hookleftarrow} \mathbb{Z}_q^n$), find $\vec{s}$.*

Definition 12 (Decision-LWE Problem.). *Given q, m, n, α and $A \overset{r}{\hookleftarrow} \mathbb{Z}_q^{m\times n}$, $\vec{y}$, distinguish between the following two scenarios*

- Real Scenario: $\vec{y} = A.\vec{s} + \vec{e} \mod q$ *(with $\vec{e} \overset{r}{\hookleftarrow} \chi_{\alpha q}^m$ and $\vec{s} \overset{r}{\hookleftarrow} \mathbb{Z}_q^n$)*
- Random Scenario: $\vec{y} \overset{r}{\hookleftarrow} \mathbb{Z}_q^m$.

2.3.2 Security of LWE

An average-case to worst-case connection similar to Theorem (6) for SIS problem can also be established for LWE ([71])

Theorem 7. *If there is an algorithm A that solves $DLWE_{q,m,n,\alpha}$ in poly-time, with non-negligible distinguishing advantage, for $\alpha.q > 2\sqrt{n}$. Then there is a quantum algorithm B that solves $\gamma - GapSVP$ in polynomial time for all input lattices L of dimension n with:*

$$\gamma = O(n/\alpha)$$

Theoretically we have strong reason to believe that LWE is hard. The theorem even says that it links the security of LWE to the quantum security of the shortest vector problem. It therefore gives us more confidence to use LWE as the basis for quantum resistant cryptographic technique. In practice, the best known attack against LWE is to reduce the problem to the SIS problem: Given a LWE instance $(A \in \mathbb{Z}_q^{m\times n}, \vec{y} \in \mathbb{Z}_q^m)$:

- Find a short non-zero vector $\vec{v}$ in the SIS lattice $L_q^{\perp}(A^T)$ with $\|v\| \leq \beta$. That is, $A^T.\vec{v} = \vec{0} \mod q$.
- Compute $e' = \vec{v}^T.\vec{y} \mod q$ and check:
  - In 'Real DLWE Scenario', e' is small
  - In 'Random Scenario', e' is not small

In other words, solving $DLWE_{q,m,n,\alpha}$ reduces to solving $SIS_{q,m,n,\beta=1/\alpha}$, therefore to secure the system based on DLWE, we must choose parameters so that $SIS_{q,m,n,\beta}$ is hard. (The smaller the noise we choose for DLWE, the larger the β becomes, or $\gamma - SVP$ is easier, which is expected). Also, the condition of $\alpha q > 2\sqrt{n}$ is also important as when the noise become small enough, it turns out that there are efficient algebraic attacks against LWE.

2.3.3 Symmetric cryptosystem from LWE

We present a randomized cryptosystem based on LWE

Key Generation - KeyGen. Fix integers q, n. Pick a secret key $\vec{s} \overset{r}{\hookleftarrow} \mathbb{Z}_q^n$.

Encryption - Enc. Fix integers t, l. Given a message $\vec{m} \in \mathbb{Z}_t^l$:

- Pick $A \overset{r}{\hookleftarrow} \mathbb{Z}_q^{l\times n}$ and 'small' noise $\vec{e} \overset{r}{\hookleftarrow} \chi_{\alpha q}^l$.

- Compute $\vec{c} = A.\vec{s} + \vec{e} + \lceil q/t \rfloor .\vec{m} \mod q$.
- Return ciphertext $(A, \vec{c}$.

Decryption - Dec. Given a ciphertext $(A, \vec{c})$ and a secret key $\vec{s}$:

- Compute $\vec{c'} = \vec{c} - A.\vec{s} \mod q$.
- Compute $\vec{c''}$ by rounding coordinates of $\vec{c'}$ to the neareast multiple of $\lceil q/t \rfloor \mod q$. This step exploits the fact that $\vec{e}$ is a 'small' vector rather than a random one.
- Return plaintext $\vec{m} = \frac{\vec{c''}}{\lceil q/t \rfloor}$

Correctness. Decryption is correct if the rounding step succeeds, or if all the noise coordinates e_i of $\vec{e}$ is sufficiently small:

$$e_i < \frac{1}{2}.\lceil q/t \rfloor \approx \frac{q}{2t}$$

If the noise distribution $\chi_{\alpha q}$ is a normal distribution with standard deviation αq, the probability that the noise exceeds $\frac{1}{2t\alpha}$ in magnitude is $p_e \approx 2.\left(1 - \Phi\left(\frac{1}{2t\alpha}\right)\right)$, where Φ is the cumulative distribution function of the standard normal distribution (mean 0 and standard deviation 1). Therefore, to assure correctness, we need this probability to be sufficiently small, the following *correctness condition* must holds:

$$t << \frac{1}{2\alpha}$$

We can see that the larger the noise is, the smaller our message space becomes. However, when the noise is large, LWE is harder as well: security goes up when noise increase. Generally, in lattice-based cryptosystems, there is this trade-off between security and the size of messages to encrypt.

Security. This section discusses how to analyze the security of lattice-based cryptosystems. Our goal is to show that if an adversary can break the encryption scheme's security, then he can also break DLWE problem, which will be the hard problem that relate to lattice problems that we known that are hard (SIS or $\gamma - SVP$). The standard security model that is used in our analysis would be indistinguishability security against Chosen Plaintext Attack (IND-CPA). The formal 'security game' is defined between a challenger Ch and the attacker B against the encryption scheme

CPA Security Game. The game is defined as follows.

- The challenger *Ch* runs *Keygen* algorithm and obtains a secret key $\vec{s}$.
- The attacker B is given access to an 'encryption oracle': B can submit a plaintext $\vec{m}$ and get back from *Ch* a ciphertext $(A,C) = Enc(\vec{,}\vec{m})$. B can query the oracle as many times as he wants. When B finishes the query phrase, he submits a pair of 'challenge messages' $\vec{m_0^*}, \vec{m_1^*}$.
- Challenger picks a random bit $b \overset{r}{\hookleftarrow} U(0,1)$, computes a 'challenge ciphertext' $(A^*,C^*) = Enc(\vec{s},\vec{m_b^*})$ for the challenge message selected by b, and sends (A^*,C^*) to B.
- B can continue querying the 'encryption oracle' if he would like. Finally, he outputs a guess b' for the bit b chosen by the challenger. The attacker wins the game if $b' = b$.

Definition 13 (IND-CPA security.). *A cryptosystem is IND-CPA secure with a security level λ if any attack algorithm B with run-time $T(B) \leq 2^\lambda$ can only win the security game with probability $\leq \frac{1}{2} + \frac{1}{2^\lambda}$. (Note that we have to restrict the run-time of the attacker as we know that for any cryptosystem, an attacker with unlimited time can always win the game with probability 1 using brute force search.)*

Security Reduction from IND-CPA to DLWE. Given the formal security game, we need to prove that the cryptosystem defined satisfies such model. By applying the security reduction technique, we demonstrate that if an attacker could somehow break the cryptosystem, then he could also solve DLWE, which was proved (Regev, 2005) to be a lattice-related hard problem:

Suppose there was an efficient IND-CPA attack algorithm B, breaking 2^λ security of the LWE encryption scheme:

- B runs in time T_B and wins IND-CPA game with probability 1/2 + ε_b (with $T_B < 2^\lambda$ and $\varepsilon_B > 1/2^\lambda$).
- B makes Q encryption queries overall, including the challenge ciphertext.

Then, given a DLWE instance $(q,n,A,\vec{y})$, we build a distinguisher algorithm D that runs as follows:

- D runs attacker B against the encryption scheme, D acts as the challenger. When B makes its i^{th} encryption oracle query $\vec{m}_i$, D uses the i^{th} block $A_i \in \mathbb{Z}_q^{l\times n}$ of l consecutive rows of A and corresponding i^{th} block $\vec{y}_i \in \mathbb{Z}_q^l$ of l consecutive rows of $\vec{y}$ to answer the oracle query with $(A_i, \vec{c}_i$, where:

$$\vec{c}_i = \vec{y}_i + \lceil q/t \rfloor . \vec{m}_i \mod q.$$

- Similarly, when B makes its challenge query $(\vec{m}_0^*, \vec{m}_1^*)$. D chooses a random bit b and uses the next (not yet used) blocks $A_{i^*}, \vec{y}_{i^*}$ of A and $\vec{y}$ to respond with $(A^* = A_{i^*}, \vec{c}^* = \vec{y}_{i^*} + \lceil q/t \rfloor . \vec{m}_b^* \mod q)$.
- The rest of encryption oracle queries of B answered as above.
- When B returns a guess b' for b, D returns 'Real' if $b' = b$, and 'Random' if $b' \neq b$.

The distinguisher D works because:

- If $\vec{y}$ comes from the real scenario, then we can rewrite each of the block $\vec{y}_i = A_i\vec{s} + \vec{e}_i$, for $i = 1, \ldots, q$. When replacing this onto $\vec{c}_i$, we see that it behaves exactly the same as in the encryption algorithm: $\vec{c}_i = A_i\vec{s} + \lceil q/t \rfloor \vec{m}_i + \vec{e}_i$, in which the attacker B wins the game with good probability $1/2 + \varepsilon_B$. Hence D also returns 'Real' with good probability $1/2 + \varepsilon_B$.
- If $\vec{y}$ comes from the 'Random' scenario, $\vec{y}$ is independent and uniformly random in $\mathbb{Z}_q^{l.Q}$, therefore, in challenge ciphertext, $\vec{c}_i$ is uniformly random in $\mathbb{Z}_q^l$, independent of bit b - Hence, B gets no information on b and wins the game with probability exactly $1/2$. So, D returns 'Real' with probability $1/2$

In conclusion, the distinguishing advantage of D is $\varepsilon_B > 1/2^\lambda$. Also, the run-time of D is approximately the run-time of B, which is 2^λ.

Theorem 8. *IND-CPA security of LWE cryptosystem with Q queries to the encryption oracle is at least as hard as DLWE*

2.3.4 Assymmetric Cryptosystem from LWE

This section discuss a technique to modify Symmetric-Key to Public-key Encryption (Regev's cryptosystem, 2005). This cryptosystem is the basis for a lot of other lattice-based cryptographic techniques and the ones that are used later in our project. The idea is, suppose that we use symmetric key sytem to encrypt a message $\vec{m} = \vec{0}$, we observe that $Enc(\vec{s}, m) = Enc(\vec{s}, 0) + [\vec{0}, m] \mod q$ (recall that $[\vec{a}, \vec{a}\vec{s} + \vec{e} + m] = [\vec{a}, \vec{a}\vec{s} + \vec{e}] + [\vec{0}, m]$). This implies that we can encrypt a message by adding itself to the encryption of zero $Enc(0)$, and therefore can use that $Enc(0)$ as the public key! However, the ciphertext encrypted this way can be simply broken by just one substraction. Our next attempt can be publishing several $\vec{p}_i = Enc(\vec{s}, 0)$, they are all different as our scheme is non-deterministic. During encryption, we can generate a random linear combination of such encryptions to have a 'fresh' $Enc(0)$ and used it for encryption.

Keygen. Fix integers q, m, n. Select a secret key $\vec{s} \overset{r}{\hookleftarrow} \mathbb{Z}_q^n$ and publish the public key $(A, \vec{p})$, where $A \overset{r}{\hookleftarrow} \mathbb{Z}_q^{m\times n}$ and $\vec{p} = A.\vec{s} + \vec{e} \mod q$ with $\vec{e} \overset{r}{\hookleftarrow} \chi_{\alpha q}^m$.

Encryption - Enc. Fix integers t, B_r. Given a message $m \in \mathbb{Z}_t$ and the public key $(A, \vec{p})$, select coefficients vector $\vec{r} \overset{r}{\hookleftarrow} \{-B_r, \ldots, B_r\}^m$ and compute and return:

$$(\vec{a}^T, c) = (\vec{r}^T.A, \vec{r}^T.\vec{p} + \lceil q/t \rfloor.m \mod q)$$

Decryption - Dec. Given a ciphertext $(\vec{a}^T, c)$ and the secret key $\vec{s}$:

- Compute $c' = c - \vec{a}^T.\vec{s} \mod q$
- Compute $c'' \in \mathbb{Z}_q$ by rounding c' to the neareast multiple of $\lceil q/t \rfloor \mod q$.
- Return plaintext $m = \frac{c''}{\lceil q/t \rfloor}$.

Observation. For small r_i: $r_1.Enc(\vec{s}, 0) + r_2.Enc(\vec{s}, 0) + \cdots + r_i.Enc(\vec{s}, 0) = Enc(\vec{s}, 0)$.

Correctness. The condition for correct decryption is similar to the sysmmetric setting where the new noise $e < \frac{1}{2}.\frac{q}{t}$. The noise used in the public key randomizing process growths $(|e| > |e_1|, |e_2|, \ldots, |e_i|)$, but it is still small as long as r_i are small. Generally, if the original noise distribution is normal with standard deviation αq, then the distribution of the new noise $e = \vec{r}\vec{e}$ is also normal distributed with standard deviation $\alpha q \|\vec{r}\|$. Also, the expected value of $\|\vec{r}\| \approx \sqrt{B_r(B_r+1)m/3}$. At the end, to ensure the correctness of the system, we need to set up the parameters such that the probability that the noise grow big is small. The error probability per coordinates p_e is the probability that a standard normal random variable (mean 0, standard deviation 1) exceeds $\frac{1}{2t\alpha}$ in magnitude:

$$p_e \approx 2.\left(1 - \Phi\left(\frac{1}{2t\alpha}.\sqrt{\frac{3}{B_r(B_r+1)m}}\right)\right)$$

Where Φ is the cumulative distribution function of the standard normal distribution. We can rewrite the following correctness condition in terms of t:

$$t << \frac{1}{2\alpha}.\sqrt{\frac{3}{B_r(B_r+1)m}}$$

Again, we see that if we use larger noise, we have better level of security but our message space becomes smaller. In the Public-key scheme, we lose another factor of $O(\sqrt{m}$ in t when doing randomization, compared with the symmetric-key system.

Security. Similar to the symmetric key system, we first define the IND-CPA attack model for an attacker B and a challenger Ch:

- Ch runs KeyGen algorithm to obtain a secret key $\vec{s}$ and a public key $(A, \vec{p})$. The public key is given to the attacker B.
- B does not need to access an encryption oracle, it can simulate the operation itself, this is the main difference compared to the symmetric security model. B just send to Ch a pair of 'challenge messages' $\vec{m_0^*}, \vec{m_1^*}$.
- Ch selects a random bit $b \overset{r}{\hookleftarrow} 0, 1$ and compute the 'challenge ciphertext' $(\vec{a^*}, c^*) = Enc((A, \vec{p}), \vec{m_b^*})$ for the challenge message selected by b, and gives the result to B.
- Attacker B outputs a guess b' for the bit b chosen by the challenger. B wins the game if $b' = b$.

Definition 14 (Regev IND-CPA security.). *The public key cryptosystem is secure at 2^λ level if any attacker B with run-time $T(B) \leq 2^\lambda$ wins the game with probability $\leq 1/2 + 1/2^\lambda$.*

We introduce another technique to analyze the security of a cryptosystem. Unlike the symmetric key scenario, where we built a distinguisher using an assumed attacker and proved the security by contradiction, in this scenario, we measure the closeness of probability distribution of the ciphertexts from the real attack and the reduction. The idea is, if the distance is small enough, the attack algorithm cannot distinguish the ciphertexts. In cryptography, statistical distance usually used to measure such distance

Definition 15 (Statistical Distance.). *For two probability distributions D_1 and D_2 on a discrete set S, the statistical distance $\Delta(D_1, D_2)$ is defined as*

$$\Delta(D_1, D_2) = \frac{1}{2} \cdot \sum_{s \in S} |D_1(x) - D_2(x)|$$

Lemma 4. *Let D_1, D_2 be any two distributions, and A be any algorithm, then:*

$$|Pr_{x \overset{r}{\hookleftarrow} D_1}[A(x) = 1] - Pr_{x \overset{r}{\hookleftarrow} D_2}[A(x) = 1]| \leq \Delta(D_1, D_2)$$

This lemma is very useful because it says, if we make the distance small enough (negligible), then no matter what algorithm an attacker uses, he is not able to distinguish the distributions by non-negligible advantage. We will show that the distribution that we generate in the security reduction that the distinguisher shows to the attacker is within a negligible statistical distance to what the attacker sees in the real attack.

Security Reduction from DLWE. Suppose there was an efficient IND-CPA attack algorithm B that breaks 2^{λ} security of Regev's public key system. (B runs in time T_B and wins IND-CPA game with probability $1/2+\varepsilon_B$ with $T_B < 2^{\lambda}$ and $\varepsilon_B > 1/2^{\lambda}$).

Then, given a DLWE instance $(q,n,A,\vec{y})$, DLWE algorithm D works as follows:

- D runs attacker B on input public key $(A,\vec{p}=\vec{y})$.
- When B makes its challenge query $(\vec{m}_0^*,\vec{m}_1^*)$, D behaves like the real challenger: chooses a random bit b and selects coefficient vector $\vec{r} \overset{r}{\hookleftarrow} \{-B_r,\ldots,B_r\}^m$, computes and sends back:

$$(\vec{a}^*,\vec{c}^*) = (\vec{r}.A,\vec{r}.\vec{y}+\lceil q/t\rfloor.m_b \mod q$$

- When B returns a guess b' for b, D returns 'Real' if $b'=b$ and 'Random' if $b'\neq b$.

Why does D work. Consider two LWE scenarios for $\vec{y}$:

- 'Real' LWE scenario, $\vec{y}=A.\vec{s}+\vec{e}$. The public key and challenge ciphertext returned by D to B are computed exactly as in the real IND-CPA game, so B wins the game with good probability $1/2+\varepsilon_B$, hence D returns 'Real' with probability $1/2+\varepsilon_B$.
- 'Random' LWE scenario, $\vec{p}=\vec{y}$ is independent and uniformly random in $\mathbb{Z}_q^m$. We use the following 'Leftover Hash Lemma'(LHL), this is a property related to the way we generate the fresh encryptions of 0 from the public one: The new matrix $A'=\vec{r}A$ is within statistically distance negligible to uniformly random when the vector $\vec{r}$ is big enough.

  Lemma 5 (Leftover Hash Lemma (LHL).). *Let $C \overset{r}{\hookleftarrow} \mathbb{Z}_q^{m\times(n+1)}$ and $\vec{r}\in\{-B_r,\ldots,B_r\}^m$. If the following condition holds:*

$$(2B_r+1)^m >> q^{n+1}$$

  then the probability distribution P of the pair $(C,\vec{r}.C \mod q)$ is statistically indistinguishable from the uniform distribution $\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^{n+1}$. More precisely, *the statistical distance $\Delta(P,U)$ between the probability distributions P,U is at most*

$$\frac{1}{2}.\sqrt{\frac{q^{n+1}}{(2B_r+1)^m}}$$

The lemma also gives us indication about how large parameters should be. The proof continues as follows:

- If the distribution P of $(A,\vec{y},\vec{a^*}=\vec{r}.A,\vec{r}.\vec{y})$ was exactly $U(\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^{n+1})$, then similar to the symmetric key case, the ciphertext $(\vec{a},c^*=\vec{r}\vec{y}+\lceil q/t\rfloor.m_b)$ is independent of b and the public key $\vec{y}$, that contains no information on b, and hence D returns 'Real' with probability 1/2.
- By LHL, $\Delta(P,U)\le\frac{1}{2}.\sqrt{\frac{q^{n+1}}{(2B_r+1)^m}}=\delta$. and $\delta\le\frac{1}{2^{\lambda+1}}$ is negligible. From the property of statistical distance, D returns 'Real' with probability $\le 1/2+\delta\le 1/2+1/2^{\lambda+1}$. The distinguishing advantage of D

$$Adv(D)\ge\varepsilon_B-\frac{1}{2^{\lambda+1}}\ge\frac{1}{2^{\lambda}}-\frac{1}{2^{\lambda+1}}\ge\frac{1}{2^{\lambda+1}}.$$

Also, the run-time of D is approximately the run-time of B, which is $O\left(2^{\lambda}\right)$. This contradicts with $2^{\lambda+1}$ security of DLWE

Theorem 9. *If LHL condition holds, IND-CPA security of Regev's encryption scheme is at least as hard as $DLWE_{q,m,n,\alpha}$*

- For future reference of parameter choices, we consider this example of choosing parameters for Regev's cryptosystem. We can rearrange the requirements for the LHL condition hold, it tells us how large m should be:

$$(2B_r+1)^m\ge 2^{2(\lambda+1)}.q^{n+1}$$

or

$$m\ge\frac{(n+1).\log q+2.(\lambda+1)}{\log 2B_r+1}$$

2.3.5 Efficiency of Lattice-based cryptosystems

For any cryptosystem, we are interested in some aspects in terms of efficiency:

- Key length is important as the key might be stored in the memory or the public key will be sent over the network (communication size).

- Ciphertext size, we would like to reduce this as much as possible. The expansion ratio of the ciphertext should not be too big.
- Computation time for doing encryption and decryption.

For the Regev's Public-key encryption scheme, the public key is $pk = (A \overset{r}{\hookleftarrow} \mathbf{Z}_q^{m\times n}, \vec{p} = A.\vec{s} + \vec{e})$, hence, $length(pk) = m.(n+1)\log q \geq n^2 \log q$. According to the 'lattice rule of thumb', we can see that the key length is at least quadratic in the security parameter λ: $O(\lambda^2)$, this is a very large factor compared to ones of classical public key systems. Similarly, we will get the ciphertext expansion ratio is at least $O(\lambda)$, which is also large (in practice we would want something close to 1). In terms of the encryption time, the main cost is in the matrix multiplication operation, and generally it is also at least quadratic in λ: $O(\lambda^2)$. We will discuss approaches to improve all of the aspects in this section.

Reducing Ciphertext Expansion. The first idea was proposed by [67] to improve Regev's public key scheme by squeezing more messages to a ciphertext. The observation is the $\vec{a^T} = \vec{r^T}.A \mod q$ part of the ciphertext is independent of the message m and takes lots of space. In stead of using a new randomness for each message m, we can 'reuse' a randomness with new secret keys $\vec{s_i}$, to encrypt many integer messages , encoded in an integer vector. The modified scheme is presented as follows, with l being the number of secret key vectors.

- Secret-key $S = (\vec{s}_1, \ldots, \vec{s}_l) \in \mathbb{Z}_q^{n\times l} \log q$.
- Public-key $pk = (A \overset{r}{\hookleftarrow} \mathbb{Z}_q^{m\times n}, P = (\vec{p}_1, \ldots, \vec{p}_l))$ where $\vec{p}_i = A.\vec{s}_i + \vec{e}_i \mod q$, with $\vec{e}_i \overset{r}{\hookleftarrow} \chi_{\alpha q}^m$.
- Encryption - Enc $(\vec{m} \in \mathbb{Z}_t^l)$: Return the ciphertext $C = (\vec{a}^T = \vec{r}^T.A \mod q, \vec{c}^T = \vec{r}\cdot P + \lceil q/t \rfloor.\vec{m} \mod q)$. We see that the ciphertext size increases from $n\log q$ to $(n+l)\log q$, the message length also increases from one integer to l integers. The expansion ratio $\frac{(n+l)\log q}{l \log t}$ is therefore can be reduced by using larger l, for example, if $l = n$, the ratio is as good as 2. Note that IND-CPA security still holds as the scheme does not reuse $\vec{p}_i$ during encryptions. Although $\vec{p}_i$ do make the public key size a bit bigger, but A's size still dominate.
- Decryption - $Dec(C = (\vec{a}^T, \vec{c}^T))$: Compute $\vec{c'}^T = \vec{c}^T - \vec{a}^T.S \mod q$ and round to the neareast multiple of $\lceil q/t \rfloor \mod q$ to get $\vec{c''}$. Return plaintext $\vec{m} = \frac{\vec{c''}}{\lceil q/t \rfloor}$

Reducing Storage and Computation. The approach toward this problem has an interesting history, it evolved even before the introduction of LWE. The idea is to put some structure to the matrix A: So far all the elements of A are chosen uniformly at random, recall that A is a random $m \times n$ matrix with $m \geq n$, the total number of elements in the matrix is $m.n \geq n^2$:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

If we do not choose the elements independently but having some corelations between the rows or columns, then we do not have to specify n^2 elements anymore to describe A. In other words, we can derive elements from existing one. The question is how to do that securely? The most common approach was introduced by [39, 59], it is a special structured type of matrix called negacyclic matrix, denoted by $rot(\vec{a})$, where $\vec{a}$ is an n-dimensional vector. Once $\vec{a}$ is specified, which is the first column of the rot() matrix, all the other columns can then be derived from the first one: the rule is quite simple, the next column is generated by rotate the previous column by one position and change the sign of the first element:

$$\begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix}$$

The output of $rot(\vec{a})$ is a square $n \times n$ matrix. In order to construct the matrix A, we can do

$$A = \begin{bmatrix} rot(\vec{a}_1) \\ rot(\vec{a}_2) \\ \vdots \\ rot(\vec{a}_{m/n}) \end{bmatrix}$$

Therefore, the storage for the matrix A reduces from $m \times n \log q$ to $m \log q$ as we only need to store the first column of the matrix. Another nice property of this rotational structure is its correspondence with the ring multiplication operation of the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. In other words, the expensive operation of one matrix multiplied by a vector can be done in one polynomial multiplication operation: For two n-dimensional vectors $\vec{a}, \vec{x}$ that represented by two polynomials $a(x), s(x) \in \mathbb{Z}_q[x]$ of degree less than $n-1$, let $c(x) = a(x).s(x) \mod x^n+1$, or $c(x) = \sum_{i<n} s_i x^i a(x) \mod x^n+1$. We observe that $x(a_0+a_1x+a_2x^2+\cdots+a_{n-1}x^{n-1}) \mod x^n+1 = -a_{n-1}+a_0x+a_1x^2+\cdots+a_{n-2}x^{n-1}$.

Hence, $rot(\vec{a}).\vec{s} \mod q = a(x).s(x) \mod x^n+1$ can be presented as:

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \dots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \dots & -a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_0 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s1 \\ \vdots \\ s_{n-1} \end{bmatrix}$$

After reinterpreting the operation this way, we can apply fast algorithm to do polynomial multiplication to speed up the computation time. Specifically, we can go from $O(n^3)$ cost of matrix-vector multiplication down to $O(n\log n)$ Fast Fourier Transform (FFT) polynomial multiplication.

Reducing Computation with FFT. There are several variants of Fourier Transform (FT), one of the classical usage of FT in engineering is to convert a time function of periodic signal to the frequency domain of that function. The version of FT that we will use is similar but it works on discrete structure: the input to the FT is a n-dimensional vector (which can also be represented in terms of a polynomial), the output is another polynomial whose coefficients are the evaluations of the input polynomial at the n roots of unity. If the FFT is done in $\mathbb{Z}_q$, it is called the Number Theoric Transform (NTT) (The original FT usually works with complex numbers). Recall that the NTT's roots of unity are ζ_i such that $\zeta_i^n = -1 \mod q$ and in order for ζ_i to exist, the condition on q is $q-1$ is divisible $2n$. The polynomial multiplication speed up of $\mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}).\mathbf{s}(\mathbf{x}) \mod x^n+1$, where $\mathbf{a}(\mathbf{x}), \mathbf{s}(\mathbf{x}) \in \mathbb{Z}_q[x]$ can be done as follows.

- Choose q such that $2n$ divides $q-1$, then x^n+1 has n zeros in $\mathbb{Z}_q$ of the form ζ^{2i+1} for $i = 0, \dots, n-1$, where $\zeta \in \mathbb{Z}_q$ is a primitive $2n^{th}$ root of 1 in $\mathbb{Z}_q$.
- Evaluate $\mathbf{a}(\mathbf{x})$ and $\mathbf{s}(\mathbf{x})$ at the n points ζ^{2i+1} in $\mathbb{Z}_q$ to compute the evaluation vectors $(\mathbf{a}(\zeta), \dots, \mathbf{a}(\zeta^{\mathbf{2n-1}}))$ and $(\mathbf{s}(\zeta), \dots, \mathbf{s}(\zeta^{\mathbf{2n-1}}))$. This operation corresponds to multiplication by an FFT-like matrix and takes $O(n\log n)$ multiplication/addition over the ring $\mathbb{Z}_q$.
- Multiply the evaluations at each point $\mathbf{c}(\zeta^{\mathbf{2i+1}}) = \mathbf{a}(\zeta^{\mathbf{2i+1}})\mathbf{s}(\zeta^{\mathbf{2i+1}})$ for $i = 0, \dots, n-1$.
- Interpolate (inverse NTT) $(\mathbf{c}(\zeta), \dots, \mathbf{c}(\zeta^{\mathbf{2n-1}}))$ to reconstruct $\mathbf{c}(\mathbf{x})$. This operation again takes $O(n\log n)$ multiplications/additions over $\mathbb{Z}_q$.

The details of NTT algorithms are not discussed in this project, we remark that the time to compute polynomial multiplication with NTT is $O(n\log n)$ in stead of $O(n^2)$ if we use classical arithmetic method.

In summary, by using this structured rot matrix, not only does we save storage of the key from $O(n^2)$ to $O(n)$, but also the time of the most expensive operation from $O(n^2)$ to $O(n\log n)$.

2.3.6 The Ring Variant Systems - RLWE

We summarize the two optimizations and derive the following *Ring* variant of Regev's encryption scheme over the ring $\mathbb{R}_q = \mathbb{Z}_q[x]/(x^n+1)$, with $m' = m/n$ and $l = n$:

KeyGen. Secret key $sk = \mathbf{s} \overset{r}{\hookleftarrow} \mathbb{R}_q$, public key $pk = (\mathbf{A} \overset{r}{\hookleftarrow} \mathbb{R}_q^{m'\times 1}, \vec{\mathbf{p}} = \mathbf{As} + \vec{\mathbf{e}} \mod q)$, with $\vec{\mathbf{e}} = [\mathbf{e_1}, \ldots, \mathbf{e_{m'}}] \overset{r}{\hookleftarrow} \chi^n_{\alpha q}$. Note that the length of the public key is now $O(n\log^2 q) = O(\lambda \log^2 \lambda)$ bits, or 'quasi-linear' in security λ.

Encryption. For a plaintext $\mathbf{m} \in \mathbb{R}_t$, return ciphertext $C = (\mathbf{a} = \mathbf{rA}, \mathbf{c} = \mathbf{rp} + \lceil \mathbf{q}/\mathbf{t} \rfloor.\mathbf{m} \mod \mathbf{q})$. Note that the ciphertext expansion ratio is now $O(\log \lambda)$ and the encryption time (with NTT method) also reduces to 'quasi-linear': $O(\lambda \log^2 \lambda)$.

Decryption. Given a ciphertext $C = (\mathbf{a}, \mathbf{c})$, compute $\mathbf{c}' = \mathbf{c} - \mathbf{a}.\mathbf{s}$ and round the result to the neareast multiple of $\lceil q/t \rfloor \mod q$ to get $\mathbf{c}'' \in \mathbb{R}_q$. Return the plaintext $\mathbf{m} = \frac{\mathbf{c}''}{\lceil q/t \rfloor} \in \mathbb{R}_t$.

The improvements discussed above based on the polynomial ring $\mathbb{R}_q$ can also be applied to improve the efficiency of other cryptographic schemes such as the Ajtai's hash function that was detailed in Definition 8. The idea is again replacing the matrix A with the structured matrix $rot(\vec{a})$.

Definition 16 (Ring variant of Ajtai's Hash Function.). *Given an input* $\mathbf{x} \in \mathbb{R}^{m'}$ *having 'small' coordinates* ($\|\mathbf{x}\| \le d$), *selects a matrix over* $\mathbb{R}_q$ $A = (\mathbf{a_1}, \ldots, \mathbf{a_{m'}})$ *uniformly random to be the hash function's public key. The output of the function is defined as*

$$g_{q,m,n,d,A}(\mathbf{x}) = A.\vec{\mathbf{x}} = \mathbf{a_1}.\mathbf{x_1} + \cdots + \mathbf{a_{m'}}.\mathbf{x_{m'}} \in \mathbb{R}_q$$

This has function has improve the efficiency to $O(n\log n)$ for the key A, the computation is $O(n\log^2 n)$. A practical implementation was done [56] with some further optimizations for a specific set of parameters: $n = 64, m = 16, q = 257$. The hash compresses 1024-bit input to 512-bit output with the key length 8kbits. The performance is competitive to other hash functions, which is about 60 CPU cycles/bytes.

SecurityImpact

We want to learn how is security impacted when switching from using completely random matrix A to the structured polynomials to improve the efficiency of lattice-based cryptographic schemes. We have to discuss the hardness of Ring-SIS or Ring-LWE compared to the original problems. Many works have been done on this, we will see that when choosing the parameters properly, we can obtain the same level of security. The definition of the problems follows.

Definition 17 (Decision Ring Learning with Errors (Decision-RLWE)). *Given $q, m, n, \alpha, A \overset{r}{\hookleftarrow} R_q^{m' \times n}$ and $\vec{y}$, distinguish between the following two scenarios:*

- *'Real' Scenario: $\vec{y} = A.\vec{s} + \vec{e} \mod q$ (with $\vec{e} \overset{r}{\hookleftarrow} \chi_{\alpha q}^{m'}$ and $\vec{s} \overset{r}{\hookleftarrow} \mathbb{Z}_q^n$)*
- *'Random' Scenario: $\vec{y} \overset{r}{\hookleftarrow} \mathbb{Z}_q^m$*

There is a similar theoretical result to LWE with regards to average-case to worst-case lattice reduction for Ring-SIS/Ring-LWE. [57]. The work proved that if we can find an efficient algorithm that breaks Ring-LWE for random instance over the ring, then we can use it to break $\gamma - SVP$ on some structured sets of lattices called ideal lattices. Furthermore, we also have practical reason to believe that RLWE is hard: The best known attack on RLWE is to reduce it to RSIS, and the hardness of RSIS is assessed similarly to SIS.

It is important to mention that all the worst-case to average-case reduction results need the polynomial ring R to satisfy some conditions for the connection to hold. The choice of polynomial ring is very important for security: there was choice such as $\frac{\mathbb{Z}[x]}{x^n - 1}$ which was proved to be insecure in some systems such as the original NTRU.

Further Optimizations

Lately, there were some works that improved the Ring-Regev encryption scheme further. Firstly, the public key size of the scheme is still a vector of polynomials: $(A, \mathbf{p}) \in R_q^{m' \times 2}$. The security requirement specifies the lower limit of m'. Also, the ciphertext includes two ring element $(\mathbf{a}, \mathbf{c})$. The first problem solved was how to reduce these factors even further, to just 2 or even 1 ring element. There were two solutions proposed:

ElGamal analogue of Ring-Regev [57] This scheme is similar to discrete-log based scheme of ElGamal system and could reduce the public key and ciphertext to 2 elements of the ring R_q. Recall the Diffie-Hellman/ElGamal encryption scheme in a group G of order q with a generator g:

Public Key. $(g, p_b = g^b) \in G^2$, **Secret key:** $b \overset{r}{\hookleftarrow} G$.

Encryption. Given $m \in G$, sample $a \overset{r}{\hookleftarrow} G$ and compute the ciphertext $(p_a = g^a \in G, c = p_b^a.m = g^{ab}.m \in G)$.

Decryption. Given $(p_a, c) \in G^2$, compute $c/p_a^b = c/g^{ab} = m$

The Ring-based ElGamal system in R_q immitates the classical scheme as follows.

Public key. $(\mathbf{g} \overset{r}{\hookleftarrow} R_q, \mathbf{p_b} = \mathbf{g}.\mathbf{b} + \mathbf{e_b})$ where $\mathbf{b}, \mathbf{e_b} \overset{r}{\hookleftarrow} \chi_{\alpha q}$.

Secret key. $\mathbf{b} \overset{r}{\hookleftarrow} R_q$

Encryption. For $\mathbf{m} \in R_t$, sample $\mathbf{a}, \mathbf{e_a}, \mathbf{e_c} \overset{r}{\hookleftarrow} \chi_{\alpha q}$ and compute

$$(\mathbf{p_a} = \mathbf{g}.\mathbf{a} + \mathbf{e_a}, \mathbf{c} = \mathbf{p_b}.\mathbf{a} + \mathbf{e_c} + \lceil \mathbf{q}/\mathbf{t} \rfloor.\mathbf{m}) \in R_q^2$$

Decryption. Given a ciphertext $(\mathbf{p_a}, \mathbf{c}) \in G^2$:

$$\mathbf{c} - \mathbf{p_a}.\mathbf{b} = \mathbf{c} - (\mathbf{g}.\mathbf{a}.\mathbf{b} + \mathbf{e_a}.\mathbf{b}) = \lceil q/t \rfloor.\mathbf{m} + \mathbf{e_c}.\mathbf{a} + \mathbf{e_a}.\mathbf{b} \approx \lceil q/t \rfloor.\mathbf{m}$$

NTRUEEncrypt [39] Both of the public-key and ciphertext could be shrunk to just a single ring element in R_q, which is as small as we can hope for ideally. We discuss some variants and their properties. We saw that one of the issue of the Elgamal analogue scheme discussed above is the small secrets generated from the encryption process. The natural question to ask is what is the security effect of changing from something completely random to something small? It turns out that this variant of Ring-LWE with secret sampled from error distribution (SSRing-LWE) is as hard as the original Ring-LWE [54]. This helps gaining more efficiency without trading of security.

Lemma 6 (SSRing-LWE). *Ring-LWE with parameters m', n, α, q and secret sampled from the error distribution is as hard as standard Ring-LWE with parameter $m' + 1, n, \alpha, q$*

The original NTRU scheme was first introduced in 1996, working with the ring $R^- = \mathbb{Z}[x]/(x^n - 1\ )$ instead. We briefly describe the scheme as follows.

Setup Ring parameters: a prime n, $q \approx n$ a power of 2, a small p, and the ring $R^-\mathbb{Z}[x]/(x^n - 1)$

KeyGen Secret key $skf, g \overset{r}{\hookleftarrow} R^-$ sampled independently from a distribution χ_σ with f

The main issue with the original NTRU was its security: it relied on another problem called 'NTRU key-cracking' (which was not well understood) besides Ring-LWE. There was later a variant that only relied on Ring-LWE [81],

2.4 Homomorphic Cryptosystems

2.4.1 Homomorphic Encryption

Homomorphic Encryption (HE) is a family of cryptosystems that allows operations on encrypted data. The idea was introduced since late 1970s [73] and has been actively researched lately ([80], [?], [?], [29], etc) since the break through work of Gentry [26]. Although the idea of Fully Homomorphic Encryption (arbitrary number of operations) is feasible, its performance has not been considered practical enough. We only present a Somewhat Homomorphic Encryption system, the BV system by [11], it allows additions and some levels of multiplications on the ciphertexts, and it serves well our purpose. The security of this cryptosystem is based on the hardness of the Ring-Learning With Error (RLWE) problem [57], we informally describe the concept.

The Ring Learning With Errors Problem.

Definition 18 (RWLE). *Given parameters q,n define the ring $R_q = \frac{\mathbb{Z}_q[x]}{x^n+1}$ and a distribution $\chi_{\alpha q}$ defines a small noise distribution, the $RWLE_{q,n,\chi}$ problem asks to distinguish two distributions. In the first distribution, one samples uniformly $(\mathbf{a_i}, \mathbf{b_i})$ from R_q^2. In the second distribution, one first samples $\mathbf{s} \overset{r}{\hookleftarrow} R_q$, $\mathbf{e_i} \overset{r}{\hookleftarrow} \chi$ and generates $(\mathbf{a_i}, \mathbf{b_i})$ by sampling $\mathbf{a_i} \overset{r}{\hookleftarrow} R_q$ and compute $\mathbf{b_i} = \mathbf{a_i}.\mathbf{s} + \mathbf{e_i}$.*

SHE Scheme construction The BV cryptosystem is as follows.

Setup. Initiate (n,m,q,t,χ) to define the ciphertext space R_q, the plaintext space $R_t = \frac{\mathbb{Z}_t[x]}{x^n+1}$, and the error distribution, note that $t << q$.

KeyGen. The secret key sk can be chosen by select a small element $\mathbf{s} \in R_q$, one can do $\mathbf{s} \overset{r}{\hookleftarrow} \chi^n$. The public key pk is a pair of ring element $(\mathbf{p_0}, \mathbf{p_1})$ where $\mathbf{p_1} \overset{r}{\hookleftarrow} R_q$ and $\mathbf{p_0} = -(\mathbf{p_1}\mathbf{s} + t\mathbf{e})$ with $\mathbf{e} \overset{r}{\hookleftarrow} \chi^n$.

Encryption. Given a plaintext $\mathbf{m} \in R_t$ and a public key $pk = (\mathbf{p_0}, \mathbf{p_1})$, the encryption first does $\mathbf{u}, \mathbf{f}, \mathbf{g} \overset{r}{\hookleftarrow} \chi$ and compute a fresh ciphertext by

$$Enc_{pk}(\mathbf{m}) = (\mathbf{c_0}, \mathbf{c_1}) = (\mathbf{p_0}\mathbf{u} + t\mathbf{g} + \mathbf{m}, \mathbf{p_1}\mathbf{u} + t\mathbf{f})$$

Conventionally, we use $[\![P]\!]$ to denote the encryption of a plaintext P under BV scheme with the public key and we do not take into account the randomness. When we want to specify also the noise used in the encryption, we write $[\![(P,e)]\!]$, where e is the noise.

Decryption. Although the above encryption generates ciphertexts of 2 elements only in R_q, the homomorphic operations (discussed next) will make the ciphertext longer. We can write the decryption for ciphertext $c = (\mathbf{c_0}, \mathbf{c_1}, \ldots, \mathbf{c_L})$ with secret key $sk = (1, \mathbf{s}, \mathbf{s^2}, \ldots, \mathbf{s^L})$ as $Dec(\mathbf{c}, sk) = \left[[\langle \mathbf{c}, \mathbf{sk} \rangle]_Q\right]_t$.

Homomorphic Operations. Given 2 ciphertext $c = (\mathbf{c_0}, \mathbf{c_1}, \ldots, \mathbf{c_L})$ and $c' = (\mathbf{c'_0}, \mathbf{c'_1}, \ldots, \mathbf{c'_K})$, before any operations, the ciphertexts are padded with zeros first if necessary (to make $K = L$, assume that $K < L$ initially). The homomorphic addition $add(c,c')$ is computed by component wise addition $add(c,c') = (\mathbf{c_0} + \mathbf{c'_0}, \ldots, \mathbf{c_L} + \mathbf{c'_L})$. The homomorphic multiplication $mult(c,c')$ is computed by $mult(c,c') = (\hat{\mathbf{c}}_\mathbf{0}, \hat{\mathbf{c}}_\mathbf{1}, \ldots, \hat{\mathbf{c}}_\mathbf{2L-2})$ with $\sum_{i=0}^{2L-2} \hat{\mathbf{c}}_\mathbf{i} z^i = \sum_{i=0}^{L-1} \mathbf{c_i} z^i \times \sum_{j=0}^{L-1} \mathbf{c'_j} z^j$, where z denotes a symbolic variable.

2.4.2 Ciphertext packing

Given a bit string plaintext $m \in \{0,1\}^*$, there are several ways that one can encode it to a polynomial, or a ring element $\mathbf{m} \in \mathrm{R}_t$ before encryption. A recent popular approach for BV cryptosystem is called CRT packing method, which is based on Chinese Remainder Theorem [80]. The method allows Single Instruction, Multiple Data (SIMD) operations on encrypted data. However, we do not use this packing technique as there is not yet known efficient method to compute HD based on it. We instead apply the method from [87], which is an extension of [**?**], the technique allow HD computation in just one level of multiplication. The definition follows.

Definition 19. *For* $\mathbf{T} = (t_0, \ldots, t_{n-1})$ *and* $\mathbf{Q} = (q_0, \ldots, q_{n-1})$, *we define two types of polynomials in the ring* R_Q *of the SHE scheme:* $pm_1(\mathbf{T}) = \sum_{i=0}^{n-1} t_i x^i$ and $pm_2(\mathbf{Q}) = -\sum_{j=0}^{n-1} q_j x^{n-j}$. *The two types of packed ciphertexts are defined as* $[\![pm_1(\mathbf{T})]\!]$ and $[\![pm_2(\mathbf{Q})]\!]$

In the ring R_Q we have $x^n = -1$, then when we do multiplication between $pm_1(\mathbf{T})$ and $pm_2(\mathbf{Q})$, the constant term of the result would be the inner product $\langle \mathbf{T}, \mathbf{Q} \rangle$. We can also do homomorphic multiplication on the ciphertexts and get the ciphertext of the inner product similarly. Furthermore, we can use this result to compute HD as follows, this operation costs one level of multiplication with 3 additions and 3 multiplications on ciphertexts.

Theorem 10 ([87])**.** *Let* $C_1 = -\sum_{i=0}^{n-1} x^{n-i}$ *and* $C_2 = 2 - C_1 = \sum_{i=0}^{n-1} x^i$. *Let* $Enc(HD)$ *be a ciphertext given by*

$$[\![pm_1(\mathbf{T})]\!] * [\![C_1]\!] + [\![pm_2(\mathbf{Q})]\!] * [\![C_2]\!] - 2 * [\![pm_1(\mathbf{T})]\!] * [\![pm_2(\mathbf{Q})]\!]$$

Then, the constant term of $Dec(Enc(HD))$ *gives the Hamming Distance of* $\mathbf{T}$ *and* $\mathbf{Q}$.

2.5 Zero Knowledge Proof Systems

2.5.1 Zero Knowledge Proofs and ISIS problem

Zero Knowledge Proofs (ZKP), first introduced by [33], is a strong cryptographic tool , a beautiful notion that goes beyond the limits of traditional proofs: In a ZKP system, a *Prover* P convinces a *Verifier* P that some statement is true without leaking any thing but the validity of the assertion. There are several types of ZKP, which are the building blocks in many cryptographic protocols (anonymous credential systems, identification schemes, group signatures, etc). In this work, we focus on ZKP of knowledge (ZKPoK) ([7], [33]), where P needs to also convince V that he knows a "witness" for the given statement, we then apply such proof to enforce the user to follow the authentication protocol transcript and therefore claim that the protocol is secure against malicious clients. ZKPoK has been actively studied in the last 30 years ([21], [69], [61], [53]), we focus our work on techniques to do ZKPoK for an important hard-on-average problem in lattice-based cryptography: the Inhomogeneous Small Integer Solution (ISIS) problem. The proof relation is

$$R_{ISIS_{n,m,Q,\beta}} = \{((\mathbf{A},\vec{y}),\vec{x}) \in \mathbb{Z}_Q^{n\times m} \times \mathbb{Z}_Q^n \times \mathbb{Z}^m : (\|\vec{x}\|_\infty \leq \beta) \wedge (\mathbf{A}\vec{x} = \vec{y} \mod Q)\}$$

The secret witness of P is $\vec{x}$ and the public parameters for V are $(\mathbf{A},\vec{y})$. One of the main research directions was initiated by Stern ([82]), he proposed the solution for a simpler problem (Syndrome Decoding Problem). Ling et. al. ([53]) developed a scheme to fully support ISIS proofs. The proof is a 3-move interactive protocol : P starts the protocol by computing and sends to V three commitments; V then sends to P a random challenge; P reveals two of the three commitments according to the challenge. The *Prover*'s witness is the secret vector $\vec{x}$, the public inputs are $\mathbf{A}$ and $\vec{y}$. The protocol is detailed in Appendix 2.5.2. We refer readers to [53] for correctness and statistical zero-knowledge proofs. We note that from their result, each round of communication costs $\log\beta\tilde{O}(n\log Q)$ bits and we denote **SternExt(A,x,y)** for the whole run

2.5.2 SternExt Protocol

The protocol includes 2 phases

Setup. Let COM be a statistical hiding and computational biding commitment scheme ([46] shown that such scheme can be constructed based on the hardness of ISIS problem). Before the interaction, P and V create matrix $\mathbf{A}' \in \mathbb{Z}_Q^{n\times 3m}$ by extending $\mathbf{A}$ (padding $\mathbf{A}$ with $2m$ zero-columns). P also does some extra preparation steps:

- Decomposition: Represent vector $\vec{x} = (x_0, \dots, x_{m-1})$ by k vectors $\vec{u'}_j \in \{-1,0,1\}^m$. The algorithm first breaks each x_i to its binary representation: $x_i = b_{i,0}2^0 + b_{i,1}2^1 + \cdots + b_{i,k-1}2^{k-1}$, where $b_{i,j} \in \{-1,0,1\}$. Then $\vec{u'}_j$ is constructed by $\vec{u'}_j = (b_{0,j}, b_{1,j}, \dots, b_{m-1,j})$. Note that $\vec{x}$ can be reconstruct by $\vec{x} = \sum_{j=0}^{k-1} 2^j \vec{u'}_j$.
- Extension: This step masks the number of -1s,1s and 0s in each $\vec{u'}_j$ by transforming each $\vec{u'}_j \in \{-1,0,1\}^m$ to another $\vec{u}_j \in \{-1,0,1\}^{3m}$. The mask is done by padding to each $\vec{u}_j$ a random vector $\vec{t} \in \{-1,0,1\}^{2m}$ such that after the masking is done, the number of -1s, 0s, and 1s in $\vec{u}_j$ are equal to each other and equal m. We observe that

$$\mathbf{A}' \sum_{j=0}^{k-1} 2^j \vec{u}_j = \vec{y} \mod Q \iff \mathbf{A}\vec{x} = \vec{y} \mod Q$$

The interactive proof system. The *Prover* P ad the *Verifier* V interact as follows

1. **Commitment.** P does sampling $\vec{r_0}, \vec{r_1}, \dots, \vec{r_{k-1}} \overset{r}{\hookleftarrow} \mathbb{Z}_Q^{3m}$, $\pi_0, \dots, \pi_{k-1} \overset{r}{\hookleftarrow} \mathcal{S}_{3m}$ and sends the commitments to V:

$$\begin{cases} \mathbf{c_1} = COM(\pi_0, \dots, \pi_{k-1}, \mathbf{A}' \sum_{j=0}^{k-1} 2^j \vec{r}_j) \\ \mathbf{c_2} = COM(\pi_0(\vec{r}_0), \dots, \pi_{k-1}(\vec{r}_{k-1})) \\ \mathbf{c_3} = COM(\pi_0(\vec{u}_0 + \vec{r}_0), \dots, \pi_{k-1}(\vec{u}_{k-1} + \vec{r}_{k-1})) \end{cases}$$

2. **Challenge.** After receiving the commitments $\mathbf{c_i}$, V send a challenge $Ch \overset{r}{\hookleftarrow} \{1,2,3\}$ to the *Prover* P.
3. **Response.** P replies as follows:
   - If $Ch = 1$, P reveals $\mathbf{c_2}$ and $\mathbf{c_3}$: For each j, let $v_j = \pi_j(\vec{u}_j)$, and $w_j = \pi_j(\vec{r}_j)$ and send back $RSP = (v_0, \dots, v_{k-1}, w_0, \dots, w_{k-1})$.
   - If $Ch = 2$, P reveals $\mathbf{c_1}$ and $\mathbf{c_3}$: For each j, let $\phi_j = \pi_j$, $\vec{z}_j = \vec{u}_j + \vec{r}_j$ and send back $RSP = (\phi_0, \dots, \phi_{k-1}, \vec{z}_0, \dots, \vec{z}_{k-1})$.

- If $Ch = 3$, P reveals $\mathbf{c_1}$ and $\mathbf{c_2}$: For each j, let $\psi_i = \pi_j$, $\vec{s}_j = \vec{r}_j$ and send back $RSP = (\psi_0, \ldots, \psi_{k-1}, \vec{s}_0, \ldots, \vec{s}_{k-1})$.

4. **Verification.** Receiving the response RSP, V can do the following checks
   - If $Ch = 1$, check that $\vec{v}_j \in \{-1, 0, 1\}^{3m}$ and

$$\begin{cases} \mathbf{c_2} = COM(w_0, \ldots, w_{k-1}) \\ \mathbf{c_3} = COM(v_0 + w_0, \ldots, v_{k-1} + w_{k-1}) \end{cases}$$

   - If $Ch = 2$, check that

$$\begin{cases} \mathbf{c_1} = COM(\phi_0, \ldots, \phi_{k-1}, \mathbf{A}' \sum_{j=0}^{k-1} 2^j \vec{z}_j - y) \\ \mathbf{c_3} = COM(\phi_0(\vec{z}_0, \ldots, \phi_{k-1}(\vec{z}_{k-1})) \end{cases}$$

   - If $Ch = 3$, Check that

$$\begin{cases} \mathbf{c_1} = COM(\psi_0, \ldots, \psi_{k-1}, \mathbf{A}' \sum_{j=0}^{k-1} 2^j \vec{s}_j) \\ \mathbf{c_2} = COM(\psi_0(\vec{s}_0), \ldots, \psi_{k-1}(\vec{s}_{k-1})) \end{cases}$$

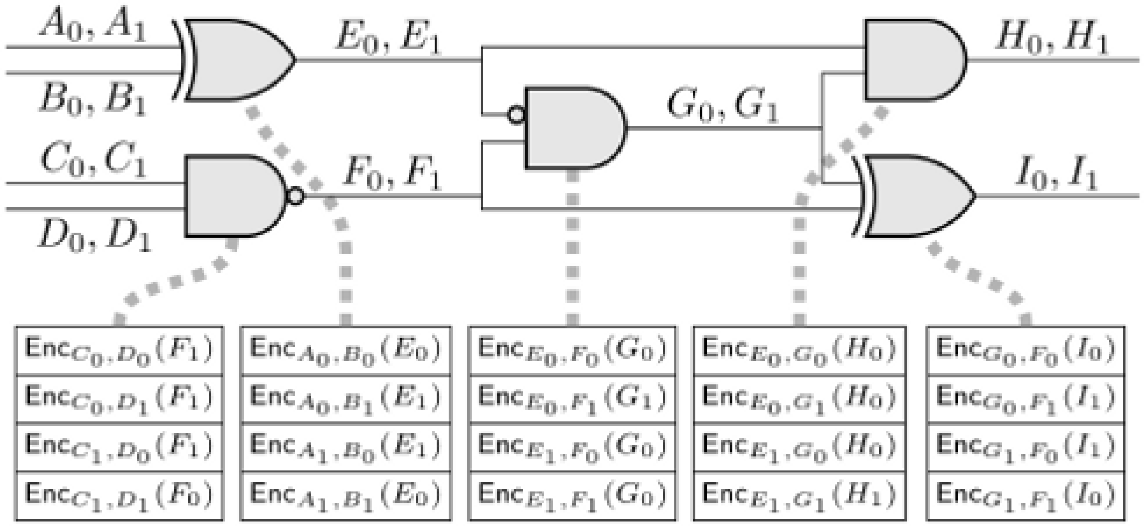
In each case, V outputs *Accept* if and only if all the conditions hold. Otherwise he outputs *Reject*. Also note that all the computations are done in $\mathbb{Z}_Q$.

2.6 Garbled Circuit and Oblivious Transfer

2.6.1 Garbled Circuit

The garbled circuit concept was originally proposed in [86] to allow two parties to evaluate securely any functions (represented by logic circuit). It is secure in the sense that the communicating parties do not learn any information about the others' inputs but only the output of the function. The idea is that given a circuit (composed of gates connected by wires), the server "garbles" the circuit by randomly assigning two encryption keys $\omega_{j,0}$ and $\omega_{j,1}$ to each wire ω_j. The pair of keys represent respectively values 0 and 1, which are possible values of the logic gates' wires. The server then encrypts a truth table corresponding to each gate using nested encryption. Figure 2.6 illustrates how this can be done: Computing the output key of a gate requires knowning two of its input's keys.

After garbling the gates, the server sends the ciphertext tables to the client and also its' keys corresponding to the server's input values. The client uses *Oblivious Transfer* technique (discussed below) to obtain the keys corresponding to the client's input values. After having the keys for each wire, the client can in turn decrypt the gates until it learn the final output keys (which can be encoded initially in a special way to represent value 0 or 1). The wires' keys are also referred to as *label* in literature.



$\mathsf{Enc}_{C_0,D_0}(F_1)$	$\mathsf{Enc}_{A_0,B_0}(E_0)$	$\mathsf{Enc}_{E_0,F_0}(G_0)$	$\mathsf{Enc}_{E_0,G_0}(H_0)$	$\mathsf{Enc}_{G_0,F_0}(I_0)$
$\mathsf{Enc}_{C_0,D_1}(F_1)$	$\mathsf{Enc}_{A_0,B_1}(E_1)$	$\mathsf{Enc}_{E_0,F_1}(G_1)$	$\mathsf{Enc}_{E_0,G_1}(H_0)$	$\mathsf{Enc}_{G_0,F_1}(I_1)$
$\mathsf{Enc}_{C_1,D_0}(F_1)$	$\mathsf{Enc}_{A_1,B_0}(E_1)$	$\mathsf{Enc}_{E_1,F_0}(G_0)$	$\mathsf{Enc}_{E_1,G_0}(H_0)$	$\mathsf{Enc}_{G_1,F_0}(I_1)$
$\mathsf{Enc}_{C_1,D_1}(F_0)$	$\mathsf{Enc}_{A_1,B_1}(E_0)$	$\mathsf{Enc}_{E_1,F_1}(G_0)$	$\mathsf{Enc}_{E_1,G_1}(H_1)$	$\mathsf{Enc}_{G_1,F_1}(I_0)$

Fig. 2.6 Garbled Circuit example

Recent works provides optimizations to improve computation and communication overhead associated with encrypt/decrypt operations and ciphertext tables' sizes. In [48], the authors proposed a modification to allow XOR gates to be evaluated *for free*: the labels of XOR gates are not chosen independently but by $\omega_{i,0} = \omega_{i,1} \oplus r$, for some random value of r. Pinkas et al. [68] introduced a way to reduce communication size of binary gates by 25%: each gate can be specified by three ciphertexts instead of all four. Finally, [47] improve some commonly used circuits such as addition, comparision, etc. by reducing the number of non-XOR gates.

2.6.2 Oblivious Transfer

"You take the blue pill, the story ends. You wake up in your bed and believe whatever you want to believe. You take the red pill, you stay in wonderland, and I show you how deep the rabbit hole goes." - Morpheus to Neo, the Matrix.

What if, in such situation, there is a protocol to give privacy to *Neo*: *Morpheus* should not learn about *Neo*'s selection. Moreover, the protocol can also provide privacy to *Morpheus*: *Neo* should not learn anything about the unchosen pill. In cryptography, Oblivious Transfer

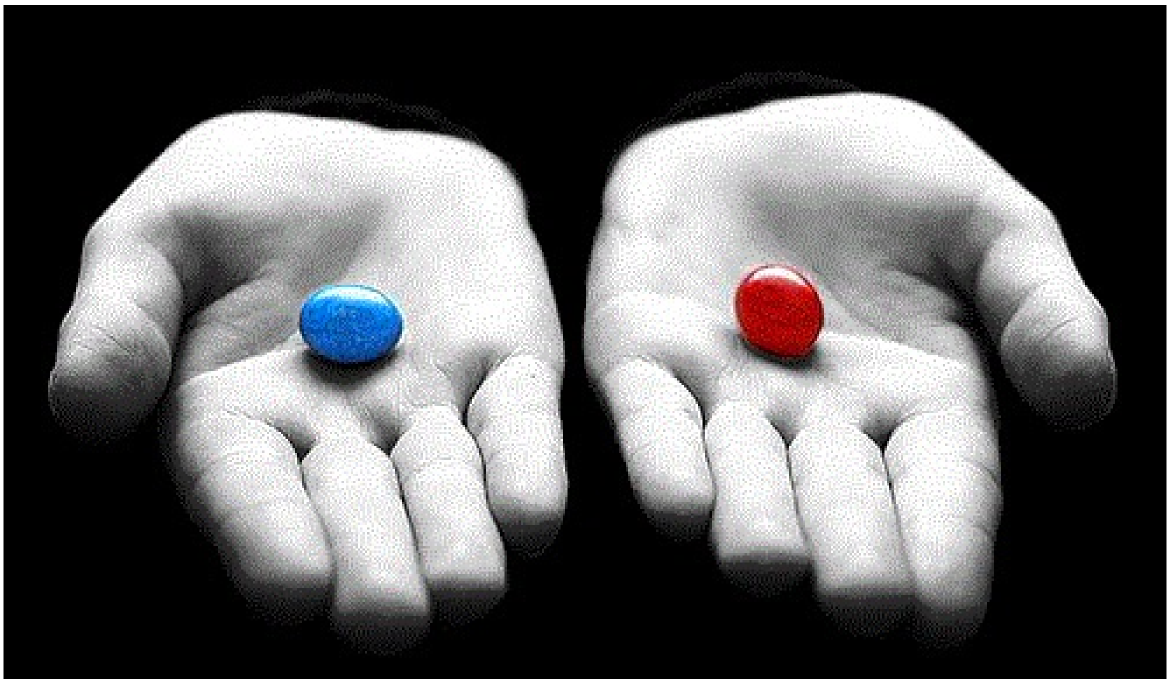


Fig. 2.7 Red Pill - Blue Pill

(OT) is a protocol that can support such scenario. In the most basic form, it is a two-parties protocol between a *Sender* and a *Receiver*, denoted by $\binom{2}{1}$-OT. The *Sender* uses two private inputs x_0, x_1 and the *Receiver* uses one input bit s. At the completion of the protocol, the *Receiver* gets the bit x_s without letting the *Sender* know any information about the value of s: $\binom{2}{1}$-OT$(x_0, x_1; s) = x_s$.

The general idea is, when the receiver requests an item, the sender sends all the items to the receiver and therefore it does not know which item is the one requested. However, the response is encrypted in such a way that the receiver can only decrypt the one he requested. A concrete implementation example of $\binom{2}{1}$-OT protocol based on discrete log DH is illustrated in figure 2.8. The receiver picks h_0, h_1 such that $h_0 h_1 = h$, he cannot know both $\log_g h_0$ and $\log_g h_1$. Given h_0, h_1, the sender returns ElGamal encryptions of bits x_0, x_1 using h_0, h_1 as public keys. The receiver then decrypts one of the encryption to recover either $x_0 or x_1$

Efficient implementations of Oblivious Transfer can be found from [63]. The techniques from [40] can reduce a large number of OT protocol executions to λ, where λ is the security parameter. In this thesis, we propose a new OT technique to be used with lattice-based cryptosystem.

$$\begin{array}{ccc}
\textbf{Sender} & & \textbf{Receiver} \\
(x_0, x_1 \in \{0,1\}) & & (s \in \{0,1\}) \\
 & & u \overset{r}{\hookleftarrow} \mathbb{Z}_n \\
 & & h_s \leftarrow g^u \\
 & & h_{1-s} \leftarrow h/g^u \\
 & \xleftarrow{h_0,h_1} & \\
u_0,u_1 \overset{r}{\hookleftarrow} \mathbb{Z}_n & & \\
(A_0,B_0) \leftarrow (g^{u_0}, h_0^{u_0} g^{x_0}) & & \\
(A_1,B_1) \leftarrow (g^{u_1}, h_1^{u_1} g^{x_1}) & & \\
 & \xrightarrow{(A_0,B_0),(A_1,B_1)} & \\
 & & x_s \leftarrow \log_g(B_s/A_s^u)
\end{array}$$

Fig. 2.8 OT protocol based on DH

2.7 The Syntax and Security model

We first describe the protocol and its security model in generic form. We then can use them as a framework to apply and analyze in our specific proposal.

The threats

There are common security threats to many authentication systems such as Trojan horse, replay, man-in-the-middle (MITM) attack, etc. Biometrics authentication systems are also vulnerable to such attacks. We can borrow ideas from secure password-based schemes to address such issues. However, there are two categories of vulnerabilities that are specific to biometric systems. The first one is impersonation, or spoofing, the attack happens at the client side where the adversary tries to cheat the system with counterfeit or invalid inputs. The other issue is at the server side with the privacy concern.

We will discuss the formal models that capture all the known threats of biometric authentication:

- Confidential data leakage due to attacks on the template database: Server breaches of biometric data always have catastrophic consequences ([OPM]), as we cannot change our fingerprints as easily as changing our passwords.
- Hill-Climbing attack from the server side: This is also a privacy threat where the server trying to compute good inputs X from the distance information between X and Y ([84],

[37]). Note that Hill-Climbing attack by the client is very limited due to the limitation of the number of false authentication attempts in almost every biometric authentication system.

- Impersonation by malicious client when the secret key of the user is known (replay attack): This happens when an attacker has access to the user's device but not his biometric, for example, in stolen device scenarios.([88])
- Impersonation by malicious client when the biometric template of the user is known (spoofing attack): This happens when an attacker collects biometric data and tries to reconstruct the template for authentication, for example, rebuilding the fingerprint from captured photos ([88],[22]).
- Cross matching of biometric data among databases: This threat uses information of the same user from different compromised databases to reconstruct the biometric template.

The Generic Two-party model

Entities: There can be 2 or 3 typical entities involved in a secure biometric authentication system. The user $\mathscr{U}$, an authentication server $\mathscr{S}$, and a decryptor, who is a third party trusted by both of the users and the server. The decryptor presents in some systems ([58], [38], [37]), with the assumption that there is no collusion between this entity and $\mathscr{U}$ or $\mathscr{S}$. In our work, we aim to avoid the assumption of trusted decryptor party, so we only have two parties, $\mathscr{U}$ and $\mathscr{S}$.

Biometrics Features in non-private setting In biometric authentication systems (e.g., fingerprint authentication system), a user $\mathscr{U}$ first enrolls his fingerprint template X with the server $\mathscr{S}$. $\mathscr{U}$ later authenticates with $\mathscr{S}$ using the same finger with a template Y, $\mathscr{S}$ uses an algorithm $Verify(X,Y)$ to obtain the result of the authentication: **Accept** or **Reject**. Different fingerprint system might use different features of fingers such as minutia or fingercode to compute this distance Δ between X and Y in the algorithm $Verify$. The distance Δ is compared to some predifined threshold value τ to determine the result of the authentication. We refer the reader to [41] for biometric feature extraction and comparison techniques.

Unlike password based system where $\mathscr{U}$ always uses one same query for many authentication, all biometric systems have the concept of False Acceptance Rate (FAR), where the system **Accept** an incorrect template; and False Rejection Rate (FRR), where the system **Reject** a genuine one. Balancing these 2 rates while keeping good performances is one of the main challenges that fingerprint verification algorithms [FVC] are trying to solve. We also reflect these two rates in our models.

Algorithms and Procedures in privacy-preserving setting: We describe the high level constructions of the protocol as follows

Enroll: This procedure inserts records into the server's database.

- Input: Client: identity k, a registered template X_k; Server: Parameters of the cryptographic tools used.
- Output: A public-private key pair (sk_k, pk_k) for the user $\mathscr{U}_k$. The server learns the protected template of T_k of X_k.

Auth: This procedure allows a user to authenticate with the system.

- Input: Client: identity k, a query template Y_k and the secret key sk_k; Server: record (k, T_k, pk_k)
- Output: The server learns the authentication result $res = \{\textbf{Accept,Reject}\}$

Correctness Requirement: A genuine user $\mathscr{U}_k$ does $(sk_k, T_k) \leftarrow \textbf{Enroll}(k, X_k)$ using a $X_k \in Supp(D_k)$ and later uses his biometric template $Y_k \overset{r}{\hookleftarrow} D_k$ to do

$$res \leftarrow \textbf{Auth}((k, Y_k, sk_k), (k, T_k, pk_k))$$

The privacy-preserving protocol works correctly if FRR under this system is exactly equal to FRR of the non-privacy preserving system:

$$Pr[res = verify(X_k, Y_k)] = 1$$

The security model

Privacy against an Honest But Curious server: The security model is defined in terms of following security games.

The real game $\textbf{Real}_{\mathscr{A}}(D_k, X_k)$: This is the game for a privacy attack against the privacy-preserving protocol for the underlying biometric system, between an attacker $\mathscr{A}$ and a challenger $\mathscr{C}$. The input the game is an attacked $\mathscr{U}_k$ biometric distribution $D_k \in D_{bio}$ and a user template $X_k \in Supp(D_k)$.

1. $\mathscr{C}$ runs $(T_k, sk_k) \leftarrow \textbf{Enrol}(k, X_k)$ and sends T_k to $\mathscr{A}$.
2. For $i = 1 \ldots q$:

- $\mathscr{C}$ samples $Y_i \overset{r}{\hookleftarrow} D_k$
- $\mathscr{C}$ simulates the **Auth** protocol, playing the roles of both the client and the server:

$$res \leftarrow \mathbf{Auth_i}((k, Y_i, sk_k), (k, T_k, pk_k))$$

- Let V_i denotes the i^{th} view of $\mathscr{S}$ when $\mathscr{C}$ runs $\mathbf{Auth_i}$. $\mathscr{C}$ sends the view V_i to $\mathscr{A}$.

3. $\mathscr{A}$ outputs a bit β, representing some information that $\mathscr{A}$ has learned about (D_k, X_k). The game output is $\mathbf{Real}_{\mathscr{A}}(D_k, X_k) = \beta$.

The ideal game $\mathbf{Ideal}_{\mathscr{A}'}(D_k, X_k)$: This is the game for a privacy attack against an ideal privacy scenario for the underlying biometric authentication system, where the attacker $\mathscr{A}'$ interacts with a challenger $\mathscr{C}'$. The input the game is an attacked $\mathscr{U}_k$ biometric distribution $D_k \in D_{bio}$ and a user template $X_k \in Supp(D_k)$. In this ideal game, the information $\mathscr{A}'$ can learn about (D_k, X_k) is the value of HD_{X_k, Y_k} and the bit $Verify(X_k, Y_i)$.

1. For $i = 1 \dots q$:
   - $\mathscr{A}'$ chooses query template $Y_i \in \{0,1\}^n$ and send it to $\mathscr{C}'$.
   - $\mathscr{C}'$ sends HD_{X, Y_i} to $\mathscr{A}'$.
2. $\mathscr{A}'$ output a bit β', representing some information that $\mathscr{A}'$ has learned about (D_k, X_k). The game output is $\mathbf{Ideal}_{\mathscr{A}'}(D_k, X_k) = \beta'$.

Definition 20 (Privacy Security against Server)**.** *We say that a biometric authentication protocol is q-private in the sense of biometric template privacy against an honest but curious server $\mathscr{S}$ if for every efficient real-game attacker $\mathscr{A}$, there exists an efficient ideal-game attacker $\mathscr{A}'$ such that, for all (D_k, X_k) we have:*

$$|\Pr[\mathbf{Real}_{\mathscr{A}}(D_k, X_k) = 1] - \Pr[\mathbf{Ideal}_{\mathscr{A}'}(D_k, X_k) = 1]| \le negl(\lambda).$$

Security against the malicious client: In this work, we aim for security against active client, where an attacker $\mathscr{A}$ is assumed not to follow the protocol transcript.

Biometric Impersonation: FAR is the usual biometric impersonation probability, it is inherent to the biometrics themselves without any cryptographic protocols. We first discuss this security game (which will be refered to as *biometric impersonation*), then we elaborate the discussion to the models with privacy-preserving requirements.

Setup: $\mathscr{C}$ samples $X_k \overset{r}{\hookleftarrow} D_k$ from a random $\mathscr{U}_k$ ($D_k \overset{r}{\hookleftarrow} D_{bio}$).

Query: $\mathscr{A}$ is given access to the authentication oracle $verify(X_k, Y)$ that returns the authentication result of $\mathscr{U}_k$ with a query template Y. $\mathscr{A}$ has q attempts to make queries, in each attempt, $\mathscr{A}$ chooses a Y_q by himself and does $verify(X_k, Y_q)$.

Guess: $\mathscr{A}$ outputs $Y_{q'}$ such that $verify(X_k, Y_{q'}) = \mathbf{Accept}$.

The advantage of $\mathscr{A}$ in the game is defined as

$$\mathbf{Adv}^{bio}_{\mathscr{A}}(\lambda) = Pr[verify(X_k, Y_{q'}) = \mathbf{Accept}]$$

In this basic model, when $q = 1$, the advantage of $\mathscr{A}$ is FAR. In other words, we can say the advantage of $\mathscr{A}$ is $\mathbf{Adv}^{bio}_{\mathscr{A}}(\lambda) \leq q \times FAR$.

Privacy-Preserving Protocol: This model extend the above protocol and captures the client side attacks mentioned in section 2.7.

Setup: The setup phase includes 2 steps:

- $\mathscr{C}$ samples $X_k \overset{r}{\hookleftarrow} D_k$ from a random $\mathscr{U}_k$ ($D_k \overset{r}{\hookleftarrow} D_{bio}$).
- $\mathscr{C}$ runs $\mathbf{Enroll}(k, X_k)$ that returns (sk_k, T_k).

Query: In the query phase:

- $\mathscr{A}$ is given access to the authentication oracle $\mathbf{Auth}(Y)$ that returns the authentication result of $\mathscr{U}_k$ with a query template Y.
- $\mathscr{A}$ chooses the attack type $t \in \{I, II\}$ which specifies the scenario of key exposed or template exposed.
- $\mathscr{C}$ gives sk_k if $t = I$ or T_k if $t = II$ to $\mathscr{A}$. Note that, this model reflects the 2-factors authentication (the secret key and the biometric template), if $\mathscr{A}$ requests both factors, he loses the game.
- $\mathscr{C}$ and $\mathscr{A}$ runs $\mathbf{Auth}()$ q times, For the i^{th} run, $\mathscr{A}$ plays the client's role that chooses and sends Y_i to $\mathscr{C}$, $\mathscr{C}$ plays the server's role that replies with $res_i = \mathbf{Auth}(Y_i)$.

Guess: $\mathscr{A}$ wins the game if it outputs Y such that $\mathbf{Auth}(Y) = \mathbf{Accepted}$.

The advantage of $\mathscr{A}$ in this game is defined as

$$\mathbf{Adv}^{Imp}_{\mathscr{A}}(\lambda) = Pr[\mathscr{A}\ wins]$$

We would want this advantage value not to be too large compared to the non-privacy-preserving biometric impersonation model's advantage $\mathbf{Adv}^{bio}_{\mathscr{A}}(\lambda)$, which was bounded by $q \times FAR$.

Definition 21 (Impersonation Security)**.** *We say that a biometric authentication protocol is c-secure in the sense of template protection against the malicious user $\mathscr{U}$ if $\boldsymbol{Adv}_{\mathscr{A}}^{Imp}(\lambda)$ is not greater than $\boldsymbol{Adv}_{\mathscr{A}}^{bio}(\lambda)$ in some factor c (if $c = 1$ we would have perfectly the same security level as the non-privacy-preserving system):*

$$\boldsymbol{Adv}_{\mathscr{A}}^{Imp}(\lambda) \leq c \times \boldsymbol{Adv}_{\mathscr{A}}^{bio}(\lambda)$$

2.8 Conclusion

Chapter 3

The First Protocol - Computing HD Homomorphically

3.1 Introduction

Many biometrics privacy preserving authentication protocols rely on a trusted third party to keep the client's secret key to decrypt the authentication result ([REF]!). Our first attempt to improve the security of such schemes is to remove the role of such party. We cannot let the server keeping the key due to privacy requirement, in order to rely on the client only to decrypt the distance plaintext, we would need to solve the following challenges:

Multifactor secure A malicious client with a compromised secret key should not learn any information about the templates distance (and the registered template) when he decrypts the homomorphic ciphertext result sent by the server.

Malicious Client Model It is important for the protocol to be secure against the *active client* model, where an adversary does not simply follow the protocol transcript and tries to learn information (this model is also known as *Semi Honest*, *Honest-But-Curious*, or *Passive* Client Model). In the

The first issue was resolved by having the server masking the distance with some random value before sending it to the client ([58]). In this chapter, we discuss the solution to the second issue, where a new Zero-Knowledge-Proof (ZKP) technique for lattice-based cryptosystem is introduced to enforce the client to prove that he is actually following the protocol honestly. The technique not only provides a proof of plaintext knowledge, it has another important aspect to prove the binary format of the query template as well. Therefore, we can use the ZKP at two stages: When the client first sends the query when he start authenticating, and at the final step where he proves that he decrypts the Hamming Distance (HD) correctly.

3.2 Related Work

In a biometric authentication system, a user $\mathscr{U}$ first enrols his fingerprint template X with the server $\mathscr{S}$. $\mathscr{U}$ later authenticates with $\mathscr{S}$ using the same finger with a template Y, $\mathscr{S}$ uses an algorithm $Verify(X,Y)$ to obtain the result of the authentication: **Accept** or **Reject**. Different biometric system might use different methods to compute the distance Δ between X and Y in the algorithm $verify$. The distance Δ is compared to some predifined threshold value τ to determine the result of the authentication. We refer the reader to [41] for biometric feature extraction and comparison techniques. Unlike password based system where $\mathscr{U}$ always uses one same query for many authentications, all biometric systems have the concept of False Acceptance Rate (FAR), where the systems **Accept** an incorrect template; and False Rejection Rate (FRR), where the systems **Reject** a genuine one. Balancing these 2 rates while keeping good performances is one of the main challenges that fingerprint verification algorithms [FVC] are trying to solve. In this work, we consider the biometric data are represented as binary codes and HD is used to measure the similarity between two of them. We refer the readers to [19] or [Fuj] for examples of 2048-bit iris codes generation and HD comparison.

There are three main approaches for privacy-preserving biometric authentication ([43], [6], [42]). In the *Feature transformation approach* (cancelable biometrics or biohashing, such as [83], [15]), the template data are encrypted using a client's key, it is single factor and not secure if the key is leaked. The *Biometric cryptosystem approach* (fuzzy vault and fuzzy commitment, [85], [62]) is based on error correcting codes and it is not well understood the tradeoff between biometric accuracy and security. We focus our work on the last approach, *Homomorphic Encryption*, which seems to be the best candidate to provide all of the system design requirements mentioned.

The idea was first proposed in 2006 ([**?**]) using addictive homomorphic system Paillier ([**?**]). In 2010, Osadchy et al. ([**?**]) providing privacy-preserving feature by combining Paillier system with oblivious transfer protocol. SCiFI uses 900-bit vector to represent face image data and Hamming Distance (HD) to compare two vectors. [**?**] developed a similar system for iris and fingerprints but using DGK cryptosystem [18] and garble circuit technique instead, they represented biometric data as 2048-bit vectors and also used HD for threshold comparison. [87] proposed an approach based on Somewhat Homomorphic Encryption (SHE) [11], they introduced a ciphertext packing technique to speed up the HD computation operation. There have been variations and improvements over time ([78], [58], etc). However, most of the protocols are only secure against a semi-honest client, many relied on one or more trusted third parties with the client's secret key to decrypt the HD.

3.3 The BGV Cryptosystem and Yasuda Packing Method

3.3.1 Feature codes and HD Computation homomorphically

Biometrics are used in authentication as they are unique for each person. There are many algorithms that extract the data used in the process ([FVC]) (we refer to this data as biometric templates). Many works (such as [19] and [Fuj]) use Hamming Distance (HD) as the metric to measure the similarity of the stored and query templates, we also use HD to evaluate our idea (3). In the work of [87], the authors proposed a clever approach that can compute HD of two n-bits templates X and Y in ciphertext domain. Specifically, the method pack all the bits of a template into a single ciphertext, and a linear combination of homomorphic operations would give the result of HD. We quickly present their method below.

Definition 22. *For $\mathbf{T} = (t_0, \dots, t_{n-1}$ and $\mathbf{Q} = (q_0, \dots, q_{n-1})$, we define two types of polynomials in the ring R_q of the SHE schme:*

$$pm_1(\mathbf{T}) = \sum_{i=0}^{n-1} t_i x^i \text{ and } pm_2(\mathbf{Q}) = -\sum_{j=0}^{n-1} q_j x^{n-j}$$

The two types of packed ciphertexts are defined as

$$ct_1(\mathbf{T}) = Enc(pm_1(\mathbf{T})) \text{ and } ct_2(\mathbf{Q}) = Enc(pm_2(\mathbf{Q}))$$

The main idea of [87] is that, in the ring R_q we have $x^n = -1$, then when we do multiplication between $pm_1(\mathbf{T})$ and $pm_2(\mathbf{Q})$, the constant term of the result would be the inner product $\langle \mathbf{T}, \mathbf{Q} \rangle$. We can also do homomorphic multiplication on the ciphertexts and get the ciphertext of the inner product similarly. Furthermore, we can use this result to compute HD as follows, this operation costs one level of multiplication with 3 additions and 3 multiplications on ciphertexts.

Theorem 11. *Let $C_1 = -\sum_{i=0}^{n-1} x^{n-i}$ and $C_2 = 2 - C_1 = \sum_{i=0}^{n-1} x^i$. Let $Enc(HD)$ be a ciphertext given by*

$$ct_1(\mathbf{T}) * Enc(C_1) + ct_2(\mathbf{Q}) * Enc(C_2) - 2 * ct_1(\mathbf{T}) * ct_2(\mathbf{Q})$$

Then, the constant term of $Dec(Enc(HD))$ gives the Hamming Distance of $\mathbf{T}$ and $\mathbf{Q}$.

Algorithm 2 HD Computation Homomorphically

```
1: procedure EvalDistance($\mathbf{T}, \mathbf{Q}$)
2:     $C_1 \leftarrow -\sum_{i=0}^{n-1} x^{n-i}$
3:     $C_2 \leftarrow 2 - C_1$
4:     $C_{HD} \leftarrow ct_1(\mathbf{T}) * Enc(C_1) + ct_2(\mathbf{Q}) * Enc(C_2) - 2 * ct_1(\mathbf{T}) * ct_2(\mathbf{Q})$
5:     return $C_{HD}$
```

3.4 Zero-knowledge proof system

Suppose that *Paul* has found a solution to a hard problem and he wants to convince *Vicky* that it has been solved. *Paul* can simply write out the solution and gives to *Vicky*. However it would not be nice if *Vicky* is a dishonest person that brings the solution to show others and claims that it was her who solved the problem. Zero Knowledge Proof (ZKP) is a beautiful cryptographic concept that allows *Paul* to do the proof in such a way that *Vicky* does not learn any information more than the fact that the problem has been solved. Since introduced in the 80s [33], ZKP has been extensively studied and currently being the important building blocks in many cryptographic protocols: identification schemes ([23], [21], [35], [76], [82], [46],...), group signatures([14], [3], [9], [10], [34]...), anonymous credential systems ([12], [13], [5], [16],...), interactive encryption protocol([25], [31], [32], [45],...), and much more.

3.5 Stern-based variant ZKP

3.5.1 Stern-based ZKP

Recall that we need to construct a proof for the ISIS relation:

$$R_{ISIS_{n,mq,\beta}} = \{((\mathbf{A}, \vec{y}), \vec{x}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n \times \mathbb{Z}^m : (\|\vec{x}\|_\infty \leq \beta) \wedge (\mathbf{A}\vec{x} = \vec{y} \mod q)\}$$

There are several approaches to construct such proof (e.g. [54], [61], [82]). We discuss an approach based on [82] in this work. We refer our readers to the original paper for the detailed steps of the protocol and we denote $\mathbf{Stern}_{A,x,y}$ for the whole proof. Provided such a proof, it can be applied to do a Zero Knowledge Proof of Plaintext Knowledge (ZKPoPK) for latticed-based cryptosystems. For example, in the work of [53] with an extenstion (denoted by **SternExt**), plaintext knowledge proof was done by proving the encryption relation of Regev's cryptosystem [72]:

$$R_{Regev}^{q,m,n,t,\chi} = \{((p_0,p_1),(c_0,c_1),\vec{r}||M) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m) \times (\mathbb{Z}_q^n \times \mathbb{Z}_q) \times \mathbb{Z}_q^{n+1} : \\ (c_0 = p_0\vec{r}) \wedge (c_1 = p_1\vec{r} + M)\}$$

The proof worked by letting $\mathbf{A}' = \begin{bmatrix} p_1, 1 \\ p0, 0 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} c_1 \\ c_0 \end{bmatrix}$ being the public parameters and let $\mathbf{x} = \begin{bmatrix} \vec{r} \\ M \end{bmatrix}$ be the *Prover*'s witness. We observe that $\mathbf{A}'\mathbf{x} = \mathbf{y} \mod q$, that is, $\mathbf{x}$ is a solution to the ISIS problem, provided that $||\vec{r}||_\infty \approx |M|$ AND the *Prover* must know $\vec{r}$. This solution only works in the symmetric key setting, in many other contexts, the client does not know $\vec{r}$ as encryption is done by other parties using public key. For such situation, we can look at the decryption equation:

$$\begin{aligned} c_1 - c_0\vec{s} &= p_1\vec{r} + M - p_0\vec{r}\vec{s} \\ &= \mathbf{A}\vec{s}\vec{r} + t\vec{e}\vec{r} + M - \mathbf{A}\vec{r}\vec{s} \\ &= t\tilde{e} + M \end{aligned}$$

Therefore, we can write the decryption relation as:

$$R_{Regev,dec}^{q,m,n,t\chi} = \{((p_0,p_1),(c_0,c_1),\vec{s},\vec{e},\tilde{e},M) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m) \times (\mathbb{Z}_q^n \times \mathbb{Z}_q) \times \chi^n \times \chi^n \times \chi \times \mathbb{Z}_Q : \\ (p_1 = p_0\vec{s} + t\vec{e}) \wedge (c_1 = c_0\vec{s} + t\tilde{e} + M)\}$$

Similarly, we can let $\mathbf{A}_{Stern} = \begin{bmatrix} c_0, t, 0, 1 \\ p_0, 0, t, 0 \end{bmatrix}$ to be the public parameters and $\mathbf{X}_{Stern} = [\vec{s}, \tilde{e}, \vec{e}, M]^T$ and try applying **SternExt** to obtain the ZKPoPK. However, it will not work because the original solution proves that $||\mathbf{X}_{Stern}||_\infty < \beta$. In this situation, we want to prove the bound of separate components differently: in our above example, $||\tilde{e}||_\infty > ||\vec{e}||_\infty$. In Regev cryptosystem, the problem might not be clear enough as the norm of each vector in $\mathbf{X}_{Stern}$ is quite close to each other, for other latticed-based system, we can see a big difference. For example, if we look at the BV system's decryption relation:

$$R_{BV}^{q,n,t,\chi} = \{((\mathbf{c_0},\mathbf{c_1}),(\mathbf{p_0},\mathbf{p_1}),\mathbf{s},\mathbf{e}',\mathbf{e},\mathbf{m} \in (R_Q \times R_Q) \times (R_Q \times R_Q) \times \chi^n \times \chi^n \times \chi^n \times R_t : \\ (\mathbf{p_1}\mathbf{s} + t\mathbf{e} = -\mathbf{p_0}) \wedge (\mathbf{c_1}\mathbf{s} - t\mathbf{e}' - \mathbf{m} = -\mathbf{c_0})\}$$

Our *Prover*'s witness in this situation is $\mathbf{X}_{Stern} = [\mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m}]^T$ with $||s||_\infty \approx ||e||_\infty < ||e'||_\infty << ||m||_\infty$. Therefore, instead of proving $||\mathbf{X}_{Stern}||_\infty < \beta$, we need a proof with different constraints on the witness's components. We present a solution for this problem.

Our construction

The idea. Our first observation is, in stead of proving $||x_i||_\infty < \beta_i$, or all the coefficients of x_i is in the range $\{-\beta_i, \ldots, \beta_i\}$, we can also prove $x_i + \beta_i.f(x)$ is in the range $\{0, \ldots, 2\beta_i\}$, where $f(x) = 1 + x + x^2 + \cdots + x^{n-1}$. Secondly, if we decompose $x_i + \beta_i.f(x)$ to their binary representation and applying the Stern's variant of [46] to prove the relation

$$R_{KTX} = \{((\mathbf{A}, \mathbf{y}), \mathbf{x}) \in \mathscr{Z}_q^{n\times m} \times \mathscr{Z}_q^n \times \{0,1\}^m : wt(\mathbf{x}) \wedge \mathbf{A}.\mathbf{x} = \mathbf{y} \mod q\}$$

Then we can obtain the prove for the original relation $R_{BV}^{q,n,t,\chi}$. Note that at this point the *Prover*'s witness is a binary vector, that is, if we need to prove some part of the message is binary, we obtain that goal at this point as well. It is important if we use such proof for latticed-based cryptosystem where the message space is $\mathbf{R}_2$: a proof for ISIS relation is not useful in this situation because it only proves that the infinity norm of the whole witness is less than some β.

Protocol description. The protocol **SternBV(A,y,x)** works as follows. Let A be a matrix of $m \times l$ ring element ($A \in R_q^{m\times l}$), $\mathbf{x}$ be a vector of l ring elements $\mathbf{x} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_l}\}$ and similarly $\mathbf{y} = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\}$. The protocol includes the following steps.

Step 1. Normalizing the bound of each component x_i of $\mathbf{x}$ from $\{-\beta_i, \ldots, \beta_i\}$ to $\{0, \ldots, 2^{l_i}\}$, where l_i is the smallest integer satisfying $2^{l_i} > (2\beta_i - 1)$. This step is done by one ring multiplication for each $\mathbf{x_i}$, let $x_i' = x_i + \beta_i.\mathbf{f}(x)$, where $f(x) = 1 + x + x^2 + \cdots + x^{n-1}$. After this normalization step, instead of proving the relation $\mathbf{a_i x_i} = \mathbf{y_i}$ with $||x_i||_\infty \in \{-\beta_i, \ldots, \beta_i\}$, we prove $\mathbf{a_i x_i'} = \mathbf{y_i'}$ with $||x_i'||_\infty \in \{0, \ldots, 2^{l_i}\}$, where $\mathbf{y_i'} = \mathbf{y_i} + \mathbf{a_i}\beta_\mathbf{i}\mathbf{f}(\mathbf{x})$.

Step 2. Decompose $x_i' = x_i + \beta_i$ into their binary representation

$$\mathbf{x_i''} = \sum_{j=0}^{l_i - 1} 2_j b_j$$

Let $\mathbf{x}''$ be the result ring element that concatenates all $\mathbf{x_i''}$ and has $L = \sum l_i$ coefficients. In this step we need to hide the Hamming Weight of the secret vector $\mathbf{x_i'}$. This hiding task is done by padding:

1. Let ζ_0 and ζ_1 be the number of coefficients of $\mathbf{x}''$ that equal to 0 or 1, respectively.
2. Sample a random vector $\zeta \in \{0,1\}^L$ that has $(L - \zeta_0)$ coefficients 0 and $(L - \zeta_1)$ coefficients 1.
3. Output $\mathbf{x_{Stern}} = \mathbf{x}''||\zeta$.

the result binary vector $\mathbf{x_{Stern}}$ has length $2L$ and the total number of 0s and 1s in the $\mathbf{x_{Stern}}$ are the same.

Step 3. We denote $rot(\mathbf{c}) \in \mathbb{Z}_Q^{n\times n}$ to be an anti-circulant square matrix, whose first column is $\mathbf{c}$ and the other columns are the cyclic rotations of $\mathbf{c}$ with the cycled entries negated

$$rot(\mathbf{c}) = \begin{bmatrix} c_0 & -c_{n-1} & -c_{n-2} & \cdots \\ c_1 & c_0 & -c_{n-1} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots \end{bmatrix}$$

and reconstruct the matrix A with the rot matrices:

$$\forall \mathbf{a_{i,j}} \in \mathbf{A} : \mathbf{a'_{i,j}} = rot(\mathbf{a_{i,j}})$$

The result expanded matrix is denoted $\mathbf{A'}$. We also need to pad the resulting matrix with corresponding number of 0s to make sure $\mathbf{A'}$ complying with all x'_i. Let $\mathbf{A_{Stern}}$ be the padded result

Step 4. Modify $\mathbf{y_i}$: Let $\mathbf{y'_i} = \mathbf{y_i} + \mathbf{a_i}\beta_\mathbf{i}\mathbf{f}(\mathbf{x})$ and let $\mathbf{y_{Stern}}$ be the concatenation of all $\mathbf{y'_i}$.

Step 5. Run the Stern protocol (Section 2.5.2) for the proof of $\mathbf{A_{Stern}x_{Stern}} = \mathbf{y_{Stern}}$.

Result. Our protocol has the following properties:

- The knowledge extractor produces different x_i with $||x_i||_\infty \leq \beta_i$. Inheriting from the original Stern protocol, the extraction gap is $\gamma = 1$.
- The communication cost is $2(n \log q)\sum l_i + commitmentSize$ for each round.

Algorithm 3 ZKPoPK Improved for BV

```
1: procedure ZKPBV($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e'})$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let Z be the $n \times n$ zero matrix
6:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I), (rot_{p_1}, Z, tI, Z))$
7:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e'}, \mathbf{e}, \mathbf{m})$
8:     $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
9:     Return SternExt($\mathbf{A}, \mathbf{x}, \mathbf{y}$)
```

3.6 Our Protocol

We denote $\mathscr{U}$ to be the client and $\mathscr{S}$ to be the server. There are 3 main submodules in the protocol: Setup, Enrol, and Authenticate

Setup. $\mathscr{U}$ and $\mathscr{S}$ initialize the parameters, there are several categories:

Biometric Authentication System Parameters. These parameters are standard ones used by non privacy preserving biometric authentication systems:

- False Acceptance Rate (FAR) and False Rejection Rate (FRR)
- a: The maximum number of incorrect authentication attempts allowed by $\mathscr{S}$.
- τ: Threshold to compare the Hamming Distance to decide the authentication result.
- n': The bit-length of the encoded biometric data.

Ring-LWE based techniques parameters. These parameters are used in the lattice-based cryptosystem which provide client privacy against long term quantum attacks.

- λ: General security parameter of the cryptosystem (the adversary's winning chance in the CPA security game is $1/2^{\lambda}$)
- n: Integer n defining the plaintext and ciphertext spaces rings. This will be refered to as the degree of the polynomial objects or dimension of the underlying lattice during correctness and security proofs.
- t: Integer t defining the plaintext space ring $R_t = \mathbb{Z}_t[x]/x^n + 1$.
- q: Integer q defining the ciphertext space ring $R_q = \mathbb{Z}_q[x]/x^n + 1$
- $\chi_{\alpha q}$: A distribution which is used to sample noises for LWE-based techniques. Typically, χ is a Gaussian distribution with standard deviation αq.
- δ: Renyi Divergence parameter for the security of noise masking.

Keygen. Keys are generated for $\mathscr{U}$:

- Secret key: $\mathbf{s} \overset{r}{\hookleftarrow} \chi_{\alpha q}^n$, $sk = (1, \mathbf{s}, \mathbf{s^2}, ...)$
- Public key: $pk = (\mathbf{p_0}, \mathbf{p_1})$, with $\mathbf{p_1} \overset{r}{\hookleftarrow} R_q$ and $\mathbf{p_0} = -\mathbf{p_1}\mathbf{s} - t\mathbf{e}$, with $\mathbf{e} \overset{r}{\hookleftarrow} \chi_{\alpha q}^n$.

Enrolment. $\mathscr{U}$ extracts the biometric template $\mathbf{x}$, note that the bit string $\mathbf{x}$ can be represented as a ring element of R_t. The encryption is done by $[\![\mathbf{x}]\!] = (\mathbf{c_0}, \mathbf{c_1})$ and sends to $\mathscr{S}$.

Authentication. This is done in following steps:

1. $\mathscr{U}$ extracts his biometric again $\mathbf{y}$ to use as the query. $\mathscr{U}$ sends $[\![\mathbf{y}]\!] = (\mathbf{c}'_0, \mathbf{c}'_1)$ to $\mathscr{S}$.
2. ZKP for the first relation: $\mathscr{U}$ has to prove that $[\![\mathbf{y}]\!]$ is a valid encryption, that is, it encrypts a bit string under the BV cryptosystem using the corresponding secret key. This is done by module **SternBV(A, y, x)** described in Sect. 3.5.1, where $\mathbf{A} = \begin{bmatrix} c_{y0}, t, 0, 1 \\ p_{y0}, 0, t, 0 \end{bmatrix}$, $\mathbf{X} = [\vec{s}, \tilde{e_y}, \vec{e_0}, y]^T$ and $\mathbf{Y} = [\mathbf{c_{y1}}, \mathbf{p_{y1}}]^T$.
3. HD Computation: $\mathscr{S}$ computes $[\![HD_{\mathbf{x},\mathbf{y}}]\!]$ using procedure 2.4.2. We note that this noise term $\mathbf{e_{HD}}$ of $[\![\mathbf{HD}, \mathbf{e_{HD}}]\!]$ can leak information about $\mathbf{x}$ when $\mathbf{HD}$ is decrypted. Therefore, we need to do an extra step to secure this operation.
   - Sample $\mathbf{e_r} \overset{r}{\hookleftarrow} \chi_r^n$ such that $||\mathbf{e_r}||_\infty$ is big enough compared to $||\mathbf{e_{HD}}||_\infty$.(Section **??**)
   - Compute $[\![\mathbf{r}, \mathbf{e_r}]\!]$ and do one homomorphic addition operation to mask both the values of $\mathbf{HD}$ and the noise $\mathbf{e_{HD}}$: $[\![\mathbf{HD}', \mathbf{e}'_{\mathbf{HD}}]\!] = [\![\mathbf{HD}, \mathbf{e_{HD}}]\!] + [\![\mathbf{r}, \mathbf{e_r}]\!]$

   The result $[\![\mathbf{HD}']\!]$ is then sent to $\mathscr{U}$.
4. $\mathscr{U}$ decrypts $[\![\mathbf{HD}']\!]$ and derive the actual value HD' from the first coefficient of the plaintext:$dec\,[\![\mathbf{HD}']\!] = HD' + r_1 + r_2 + \cdots + r_{n-1}$. $\mathscr{U}$ sends HD' to $\mathscr{S}$.
5. $\mathscr{U}$ proves that it does the decryption honestly, this is done similarly to step 2.
6. $\mathscr{S}$ unmasks HD' and output the authentication result $HD \overset{?}{<} \tau$

Correctness

In the enrolment step, recall that in order to encrypt $\mathbf{x}$, $\mathscr{U}$ samples $\mathbf{u}, \mathbf{f}, \mathbf{g} \overset{r}{\hookleftarrow} \chi_{\alpha q}^n$ and does $\mathbf{c_0} = \mathbf{p_0}\mathbf{u} + t\mathbf{f} + \mathbf{x}$ and $\mathbf{c_1} = \mathbf{p_1}\mathbf{u} + t\mathbf{g}$. The correctness condition for decryption correctness is $[\langle \mathbf{c_0} + \mathbf{c_1}\mathbf{s}\rangle]_q < q/2$, or, $-t\mathbf{eu} + t\mathbf{f} + t\mathbf{gs} < q/2$.

In the authentication step, $[\![HD]\!]$ is decryptable if $||\langle [\![HD]\!], \mathbf{s}\rangle||_\infty < q/2$, if we let U to be the upper bound of $||\langle [\![HD]\!], \mathbf{s}\rangle||_\infty$, and using the fact that $||a+b||_\infty \leq ||a||_\infty + ||b||_\infty$ and $||a.b||_\infty \leq n.||a||_\infty.||b||_\infty$. From theorem 10 we can derive $||\langle [\![HD]\!], \mathbf{s}\rangle||_\infty \leq 2nU + 2nU^2$. As in the work of [**?**], we can take U to be $2t\sigma^2\sqrt{n}$, which is an experimental estimation. Therefore, the final correctness condition for authentication is $16n^2t^2\sigma^4 < q$

Lemma 7 (Condition for Correct Decryption of HD)**.** *For the BV encrypted Hamming Distance $[\![HD]\!]$, the decryption recovers the correct result if $\langle [\![HD]\!], \mathbf{s}\rangle$ does not wrap around mod q, namely, if $16n^2t^2\sigma^4 < q$*

Security

The proposed scheme satisfies the security notions defined in Section 2.7, proofs are provided in Appendix 5.5

Theorem 12 (Server side security). *Under the IND-CPA security of BV cryptosystem, and the zero-knowledge property of the Stern protocol, the proposed scheme satisfies (Honest But Curious) Server Privacy Security.*

Theorem 13 (Client side security). *Under the IND-CPA security of BV cryptosystem and the soundness property of the underlying Stern protocol, the proposed scheme satisfies Impersonation Security. Concretely, for $\delta > 0$, the protocol is (q, c)-secure against impersonation with $c \le c(\delta) + 3 \cdot c_1$, assuming the underlying non-private biometric protocol has imperonsation probability ε_{bio} and the underlying Stern ZK protocols have knowledge error $\varepsilon_{ZK1}, \varepsilon_{ZK2}$ such that $q(\varepsilon_{ZK1} + \varepsilon_{ZK2}) + \delta \le c_1 \cdot \varepsilon_{bio}$, $c(\delta) = 2e^{1+2\delta}$, and the condition $\sigma/r_0 \ge 4\pi knq$ holds, with $k = 1 + \sqrt{1/\pi \ln(2nq/\delta)}$ and r_0 an upper bound on the size of the noise in C_{HD}.*

3.7 Security Proofs

3.7.1 Security Proof for Theorem 15: Privacy against Server

The proof of Theorem 16 can be done using a sequence of games between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. We present a sequence of games. The idea is to proceed to remove sk_k and X_k from being used to compute the view of $\mathscr{A}$, except for $Verify(X_k, Y_k^j)$ queries, as in the ideal game, relying on the correctness of the protocol and the IND-CPA security of the BV encryption scheme.

Game 0. Game 0 is the original real privacy game, in which $(T_k = (pk_k, C_k = Enc^{(1)}_{pk_k}(X_k)))$ is given by $\mathscr{C}$ to $\mathscr{A}$ at the beginning of the game. Then, for $j = 1 \dots, q$, the attacker sends $Y_k^{(j)} \in \{0,1\}^n$ to $\mathscr{C}$, and the latter simulates a run of the authentication protocol between an honest client with input $(k, Y_k^{(j)}, sk_k)$ and an honest server with input (k, T_k), returning to $\mathscr{A}$ the protocol view $V_S^{(j)}$ of the server. Finally, $\mathscr{A}$ outputs a bit β. In the following Game i, we let S_i denote the event that $\beta = 1$.

Game 1. We change the computation of the authentication result bit $res^{(j)}$ sent by the server to the client from the decryption of the ciphertext $[\![HD']\!]$(its value in Game 0) to the result returned by $Verify(X_k, Y^{(j)})$. By correctness of the protocol, this does not change the value of $res^{(j)}$, so $\Pr[S_1] = \Pr[S_0]$.

Game 2. We change the computation of the zero-knowledge protocol transcripts. Instead of computing those transcripts using the secret witnesses, we simulate them using the statistical zero-knowledge simulator algorithms for the zero-knowledge proofs. By the zero-knowledge property, this is a perfect simulation, and we have $\Pr[S_2] = \Pr[S_1]$.

Game 3. We change the computation of the ciphertexts $C_i^{(j)}$ for $i = 0, \ldots, l-1$ and $Q_k^{(j)}$ for $j = 1, \ldots, q$ and C_k to encrypt zero messages, instead of encrypting the secret-related messages in the previous game. Since now sk_k is not used anywhere in generating the view of $\mathscr{A}$, it follows by a hybrid argument that $|\Pr[S_3] - \Pr[S_2]|((l+1) \cdot q + 1) \cdot \varepsilon_{BV}$, where ε_{BV} denotes the maximal advantage of an attacker against IND-CPA of BV scheme against attacks with run-time $T + \text{poly}(n, \log Q)$ where T is the run-time of $\mathscr{A}$. In this game, since the only information on X_k comes via the $Verify(X_k, Y_k^j)$ queries, the challenger together with $\mathscr{A}$ constitute an efficient attacker against the ideal privacy game, which outputs 1 with probability different by at most $(l+1) \cdot q + 1) \cdot \varepsilon_{BV}$ than the probability of outputting 1 in the real privacy game, as required.

3.7.2 Security Proof for Theorem 16: Type I Impersionation attack

The proof of theorem 15 and 16 can be done using a sequence of games between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. We present a sequence of games as well as the relations among them to demonstrate type I security model proof.

Game 0. Game 0 is the original impersonation game for type I attack.

Setup. $\mathscr{C}$ intiates $D_k \overset{r}{\hookleftarrow} D_{bio}$ and $X_k \overset{r}{\hookleftarrow} D_k$. $\mathscr{C}$ sets up (sk_k, pk_k) and does $Enrol(k, X_k)$ to get $(sk_k, T_k = (pk_k, C_k = Enc_{pk_k}^{(1)}(X_k)))$. $\mathscr{A}$ submits the attack query type I and receives sk_k from $\mathscr{C}$.

Query. $\mathscr{A}$ runs q authentication sessions. In each session $j = 1, \ldots, q$, $\mathscr{A}$ sends $(Q_k^{(j)} = Enc^{(1)}(Y^{(j)}))$. $\mathscr{A}$ and $\mathscr{C}$ runs $\mathbf{ZKPValidEnc}(Q_k^{(j)}, Y^{(j)})$. $\mathscr{C}$ evaluates $C_{HD} = \mathbf{EvalDistance}_{pk_k}(C_k, Q_k^{(j}$ then computes $C_{HD'} = (-1)C_{HD} + Enc_{pk_k}^{(1)}(r_{HD'}, e_{HD'})$ for $r_{HD'} \overset{r}{\hookleftarrow} \mathscr{P}$ and $e_{HD'} \overset{r}{\hookleftarrow} \chi_{HD}$. The result ciphertext is sent to $\mathscr{A}$, $\mathscr{A}$ decrypts $C_{HD'}$ and decomposes it to bits and sends back the ciphertexts $C_0^{(j)}, \ldots, C_{l-1}^{(j)} (= Enc^{(1)}(b_i), i = 0, \ldots, l-1)$ as well as the re-encryption $C_{HD'_0}$. $\mathscr{A}$ and $\mathscr{C}$ does $\mathbf{ZKPUnpack}(C_{HD}, C_{HD'_0})$ and $\mathbf{ZKPBinDecomp}(C_{HD'_0}, C_i^{(j)})$. $\mathscr{C}$ evaluates $C''_{HD} = Enc^{(2)}(2^l + t) + \mathbf{ToUnary}(C_i^{(j)}) + Enc^{(2)}(-r)$ to get $Enc^{(2)}(2^l + t - HD)$. $\mathscr{C}$ does $Enc^{(3)}(res) \leftarrow \mathbf{MSBExtract}(C''_{HD})$ and sends to $\mathscr{A}$. $\mathscr{A}$ decrypts and sends the authentication result bit back. $\mathscr{C}$ and $\mathscr{A}$ does $\mathbf{ZKPCorrectDec}(res)$ to convince the authentication result. At the end, the server outputs **Accept** if all the proofs pass and $res =$ **Accept**.

Next, we discuss following games, the plan is to proceed towards the final game where we can simulate everything that $\mathscr{A}$ receives that related to X_k (without any knowledge about X_k,

the only thing that known about X_k is the function $Verify(X_k, Y)$). Let $res_s = Verify(X_k, Y^{(j)})$ and S_i be the event in the game i such that $res_s = Accept$.
Game 1. In this game, we abort **ZKPValidEnc** if $Q_k^{(j)}$ is not a valid encryption of the query, but the *Prover* manages to pass the proof. Let $(*)_1$ be this event.

$$(*_1) res_s = \begin{cases} \text{Reject if } \textbf{ZKPValidEnc} \text{ fails} \\ \text{res else} \end{cases}$$

Let bad_0 be the event in game 0 : $\exists j \leq q\ s.t\ (*_1)$ happens. We want to show that when we modify *game 0*, the probability of a successful forgery S_1 in this *game 1* is not much lower than what it was. Due to the modification, we observe that $Pr[S_1] \geq Pr[S_0] - Pr[bad_0]$ .For any j in the q authentication attemps, by the $\varepsilon - soundness$ property of **ZKPValidEnc** as a proof of membership in (*), we have $Pr[(*_1)\ occurs\ for\ some\ j] \leq \varepsilon_{ZK1}$. So the probability of bad_0 would be the union of these events which is bounded by $Pr[bad_0] \leq q\varepsilon_{ZK1}$. In other words, the advantage of $\mathscr{A}$ in *game 1* is

$$Pr[S_1] \geq Pr[S_0] - Pr[bad_0] \geq \varepsilon_{imp} - q\varepsilon_{ZK1}$$

Game 2. In this game, we abort **ZKPUnpack** if in one of the j^{th} runs, the *Verifier* accepts but the ciphertext did not satisfy the relation. Let bad_1 be this event, by the same type of argument, we can derive $Pr[bad_1] \leq q\varepsilon_{ZK2}$ and therefore

$$Pr[S_2] \geq Pr[S_1] - Pr[bad_1] \geq \varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2})$$

Game 3 and Game 4. Similarly, we abort **ZKPBinDecomp** and **ZKPCorrectDec** if in any j^{th} runs, the *Verifier* accepts even when the correctness of the ZKP is broken. We have

$$Pr[S_3] \geq \varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3})$$

and

$$Pr[S_4] \geq \varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4})$$

By the end of *Game 4*, if in any authentication attempt j, there is no abort in any of the 4 games, then by the correctness property of ZKP, the server output is equal to the output of $Verify(X_k, Y^{(j)})$. In other words, we have shown what we could get by just querying the oracle $Verify()$. Next, we want to simulate everything related to X_k that the attacker can see using just that oracle.

Game 5. At the end of *Game 4*, $\mathscr{C}$ has the bit $b = Verify(Y^{(i)}, X_k)$. In this game, we can change $C_{res}^{(j)}$ from **MSBExtract**$(C''_{HD}) = Enc(b; e_{res})$ to $Enc(verify(Y^{(i)}, X_k); e_{res})$, given that the challenger can use the secret key to extract e_{res}. By doing this change, $\mathscr{A}$ still sees the same ciphertext, so the probability of winning of $\mathscr{A}$ in this game is the same as in *Game 4*.
Game 6. In this game, we change the way $\mathscr{C}$ computes C''_{HD}: In the orignal game 0, $Enc(-r)$ was added to remove the mask, we want to remove this r from being used anywhere in the game, so we replace this with $Enc(0)$. This change does not affect $\mathscr{A}$'s success probability: r only affects the plaintext inside C''_{HD}, since we do not use this plaintext anymore (it was replaced in *Game 5*), so this change does not affect what the attacker sees. Again, $Pr[S_6] = Pr[S_5] = Pr[S_4]$.
Game 7. We modify the way C'_{HD} is computed in this game. Instead of doing $C'_{HD} \leftarrow (-1)C_{HD} + Enc(r; e_{HD'})$, the challenger chooses a random $HD' \overset{r}{\hookleftarrow} \mathbb{Z}_p$ and encrypt it with the noise was used before: $C'_{HD} \leftarrow Enc(HD'; -e_{HD} + e_{HD'})$. In this game, the plaintext has changed from being $r + HD$ to a uniform $HD' \in \mathbb{Z}_p$. Since r is also uniform in $\mathbb{Z}_p$, the attacker sees a uniform plaintext in both cases. Therefore, $Pr[S_7] = Pr[S_6]$.
Game 8. Finally, we set $C_{HD'} = Enc(HD', e_{HD'})$ for $e_{HD'} \overset{r}{\hookleftarrow} \chi_{mask}$ instead of $-e_{HD} + e_{HD'}$. We replace the sum of the Gaussian noise with the random noise. In Section **??**, we showed that $Pr[S_8] \geq \frac{1}{c(\delta)}(Pr[S_7] - q \cdot \delta)$, where $c(\delta) = RD(-e_{HD} + e_{HD'}, e_{HD}) \leq 2 \cdot e^{1+2\delta}$ by Lemma 8 and the assumption on the parameters. After we finish *Game 8*, we can see that all the messages that the attacker sees, can be simulated with only the verified bit $b = Verify(Y^{(i)}, X_k)$. We now have an attacker A' against the biometric impersonation with advantage:

$$\varepsilon_{bio} = Adv(A') = Pr[S_8] \geq \frac{1}{c(\delta)}(\varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta),$$

which gives the claimed bound $\varepsilon_{imp} \leq c(\delta) \cdot \varepsilon_{bio} + q \cdot (\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta \leq (c(\delta) + c_1) \cdot \varepsilon_{bio}$ if $q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta \leq c_1 \cdot \varepsilon_{bio}$.

3.7.3 Security Proof for Theorem 16: Type II Impersionation attack

The proof of Theorem 16 can be done using a sequence of games between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. We present a sequence of games as well as the relations among them to demonstrate type II security model proof. The idea is that in this type of attack, sk_k is not used to compute the view of $\mathscr{A}$. On the other hand, the soundess of the zero-knowledge proof of knowledge **ZKPValidEnc** implies the existence of an efficient witness extractor algorithm, that can be used to extract the witness (i.e. the secret key sk_k) from a cheating prover that succeeds with probability non-negligibly higher than the knowledge error of the zero-knowledge proof, thus contradicting the IND-CPA security of the BV encryption scheme.

Game 0. Game 0 is the original impersonation game for type II attack, i.e. the same as Game 0 in the proof of security against Type I attacks, except that $(T_k = (pk_k, C_k = Enc^{(1)}_{pk_k}(X_k)))$ is given by $\mathscr{C}$ to $\mathscr{A}$ at the beginning of the game, rather than sk_k. For $j \in \{1,\ldots,q\}$, let res^j denote the result of the jth authentication protocol run between $\mathscr{A}$ and $\mathscr{C}$, and S_0 be the event in the game 0 such that $res^j = Accept$ for some $j = 1,\ldots,q$. We have $\Pr[S_0] = \varepsilon_{imp,II}$, the type II success probability of $\mathscr{A}$.

Game 1. From the definition of event S_0 in Game 0, it follows that there exists some $j^* \in \{1,\ldots,q\}$ such that $\Pr[res^{j^*} = Accept] \geq \varepsilon_{imp,II}/q$. Furthermore, by an averaging argument, there must exist a set G of (D_k, X_k, pk_k) such that $\Pr[(D_k, X_k, pk_k) \in G] \geq \varepsilon_{imp,II}/(2\cdot q)$, and for each $(D'_k, X'_k, pk'_k) \in G$, we have $\Pr[res^{j^*} = Accept | (D_k, X_k, pk_k) = (D'_k, X'_k, pk'_k)] \geq \varepsilon_{imp,II}/(2\cdot q)$. By ε_{ZK1}-soundness of the zero knowledge proof of knowledge **ZKPValidEnc** (cite Golderich's 'Foundations of Cryptography' book, volume 1, Prop. 4.7.5), there exists an witness extractor algorithm that runs in expected time $T' = O(\text{poly}(n \log Q) \cdot T/(\varepsilon_{imp,II}/(2*q) - \varepsilon_{ZK1}))$, where T denotes the run-time of $\mathscr{A}$ and outputs a witness containing sk_k for *ZKPValidEnc*. Therefore, we obtain a (secret key recovery) attack algorithm against the IND-CPA security of the BV encryption scheme with expected run-time T' and advantage $\varepsilon' \geq \varepsilon_{imp,II}/(2\cdot q)$. Hence, if $\varepsilon_{imp,II}/(2*q) - \varepsilon_{ZK1} > \varepsilon_{imp,II}/(4*q)$, or equivalently, if $\varepsilon_{imp,II} > 2\cdot q\varepsilon_{ZK1}$, then we obtain a contradiction with the assumption the BV encryption scheme with parameters Q, n, σ is IND-CPA against attacks with expected time $O(\text{poly}(n \log Q) \cdot T/\varepsilon_{ZK1})$ and advantage $\geq \varepsilon_{ZK1}$. It follows under the latter assumption that $\varepsilon_{imp,II} \leq 2q\varepsilon_{ZK1} \leq 2c_1 \cdot \varepsilon_{bio}$, under the assumption that $q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4} + \delta) \leq c_1 \cdot \varepsilon_{bio}$.

3.8 Result Evaluation

3.8.1 Parameters

We consider how to set concrete parameters. From Lemma 7 and 8 and [**?**], we can take $\sigma = 50$ to make the protocol secure against the lattice attacks ([60]) and also satisfy the circuit privacy requirement. We can take $t = 2048$ to define the plaintext space ring R_t, this is enough to cover Hamming Distance of bit string up to 2048 bits. We need $n \geq 2048$ for the packing ciphertext. With these parameter, we need q to be approximately 70 bits to satisfy $16n^2t^2\sigma^4 < q$. The proposed scheme works with only 1 level of homomorphic multiplication for the proposed set of parameters $(n = 2048, q \approx 2^{70}, t = 2048, \sigma = 50)$, proof of concept implementation can be found in github [github], communication size for ZKP is approximately 8MB to provide security against malicious client model.

3.8.2 Limitations and open problems

We expose the HD to the server in the last step of the protocol and let the server do the threshold comparison operation in the plaintext domain. We believe that given such value, the server should not be able to learn any information about the original bit strings template (assuming the provable CPA-secure ciphertexts of the undernearth cryptosystem, and the HD noise is likely independent of templates). Doing the comparision of HD with threshold homomorphhically is much less efficient and removing this assumption is left as an open problem.

Also, we assume honest but curious server, that is reasonable against passive exposure attacks. Active attacks are harder to do undetected and slower provided that the server can be audited regularly. We emphasize that previous quantum resistant protocols also made this assumption, and did not even defend against passive honest but curious trusted party. Defending against malicoius server privacy is left as an open problem.

Finally, the communication size of Stern-based ZKP protocol is large due to the round soundness error $2/3$ (many communication rounds will be needed for security). We leave the problem of how to reduce such overhead for future works.

3.9 Conclusion

Chapter 4

The Second Protocol - Covering Circuit Privacy

4.1 Introduction

4.2 Previous Works

4.3 Renyi Divergence Analysis technique

Consider the a product ciphertexts result in BV cryptosystem (section 2.4.1).

$$mult(c,c') = (\mathbf{c_0 c'_0}, \mathbf{c_0 c'_1} + \mathbf{c_1 c'_0}, \mathbf{c_1 c'_1})$$

The noise term of this result ciphertext correlates to both **s** and **m**. In many contexts, this leakage might be fine for a genuine user with a secret key because s/he is supposed to know the key and decrypt the message. However, in many other contexts, especially in multi-factor authentication scenarios, this is not the case. For instance, in our system, we do not want the client to know the Hamming Distance result, so that an attacker with a stolen device and a secret key cannot derive information about the data stored in the server. We propose to mask this leakage by homomorphically adding the ciphertext $Enc(HD)$ with $Enc(r)$ to refresh the noise terms, together with masking the Hamming Distance. The question is how much can we shift the original noise distribution to preserve correctness while providing the new security measure.

Let D_1 and D_2 be the probability distributions of the original noise in the ciphertext $Enc(HD)$ and the new shifted noise of $Enc(HD+r)$. We observe that the 2 distributions are identical Gaussian distribution with the same standard deviation σ and different means. Let r_0 be the shifted in means of D_1 and D_2. Our goal is to set up parameter r_0 such that D_1 and D_2 are computational indistinguishable while keeping other parameters of the cryptosystem within practical performance thresholds. Statistical Distance (SD) is normally used to measure the difference of distributions and is generally bounded by

$$SD(D_1, D_2) = \frac{1}{2} \sum_{x \in X} |D_1(x) - D_2(x)| \leq K \times \frac{r_0}{\sigma}$$

Where K is a constant. We quickly see that in order for SD to be indistinguishable ($SD < \varepsilon \approx \frac{1}{2^\lambda}$, where λ is the security parameter), the standard deviation of the initial noise needs to be really large, which is $\sigma \geq K r_0 2^\lambda$.

In the work of [4], the authors proposed Renyi Divergence (RD) as an alternative to measure distributions closeness and its applications to security proofs. $R_a(D_1 \| D_2)$ of order a between D_1 and D_2 is defined as the expected value of $(D_1(x)/D_2(x))^{a-1}$ over the randomness of x sampled from D_1.

$$R_a(D_1 \| D_2) = \left(\sum_{x \in D_1} \frac{D_1(x)^a}{D_2(x)^{a-1}} \right)^{\frac{1}{a-1}}$$

Similar to SD, RD is useful in our context with its *Probability Preservation* property (we refer readers to [4] for detailed formal descriptions): Given D_1 and D_2 as described, for any event E, for instance, we want to look at $P(E)$ is the winning probability of the attacker in the distinguishing game, the probability of the event with respect to D_2 is bounded by

$$D_2(E) \geq D_1(E)^{\frac{a}{a-1}} / RD_a(D1 \| D_2) \tag{4.1}$$

Particularly, if we look at the second order ($a = 2$) of RD like previous works, we would have $D_2(E) \geq D_1(E)^2 / RD_2(D1 \| D_2)$. Provided that the distribution functions of D_1 and D_2 are discrete Gaussian on lattices, which is of the form $\rho_\sigma(x) = \frac{1}{\sigma} e^{-\pi \frac{x^2}{\sigma^2}}$, we have $RD_2(D_1 \| D_2) = e^{2\pi \frac{r_0^2}{\sigma^2}} \approx e^{2\pi}$, when r_0 is much smaller than σ. That means when switching from D_1 to D_2, the success probability of E will be at least the old probability to the power of 2 divided by some constant. This squaring factor brings a big trade-off, for example, in our protocol, we would need to use $FAR = 2^{-20}$ in the non-privacy biometric settings to get $FAR = 2^{-10}$ in our scheme.

We aim at a solution to remove this factor. The idea is to look at RD_∞ in stead of RD_2: from equation (4.1), we can see that when a is large, $\frac{a}{a-1}$ becomes 1. However, for usual

Gaussian distributions, RD_∞ is also infinity (not a constant $e^{2\pi}$ like in RD_2 when $a=2$). This is due to the ratios $\frac{D_1(x)}{D_2(x)}$ become large when samplings are done in the extreme tails of the distributions. Our idea is to truncate the distribution when doing noise sampling: if we get a noise value that is too far in the tail, we reject and sample again. As a result, the truncated distribution grows slightly, that means, the small noises have a bit higher probability when sampling, which does not have a big impact in security.

For the following analysis, let D_1 to be a discrete Gaussian on $\mathbb{Z}$ with deviation parameter σ shifted by the constant $r_0 \in \mathbb{Z}$, while D_2 is a discrete Gaussian on $\mathbb{Z}$ with dev. par. σ centered on zero, i.e. $D_1 = D_{\mathbb{Z},\sigma} + r_0$ and $D_2 = D_{\mathbb{Z},\sigma}$, where $D_{\mathbb{Z},\sigma}(x) = e^{-\pi \cdot x^2/\sigma^2} / \sum_{z\in\mathbb{Z}} e^{-\pi\cdot z^2/\sigma^2}$ for $x \in \mathbb{Z}$. To allow us to use RD_∞ we use tail-cut variants $D_1^{(cut)}$ and $D_2^{(cut)}$ of D_1 and D_2, respectively with parameter k. The parameter k defines where D_1 and D_2 are cut at, for example, we can set $k=3$ to cut the distributions at 3 deviation parameters from the mean. So, we let $D_{\mathbb{Z},\sigma}^{(cut)}$ denote distribution $D_{\mathbb{Z},\sigma}$ tail-cutted to the interval $[-k\cdot\sigma, k\cdot\sigma]$ by rejection sampling. We let $D_1^{(cut)} = D_{\mathbb{Z},\sigma}^{(cut)} + r_0$ and $D_2^{(cut)} = D_{\mathbb{Z},\sigma}^{(cut)}$. Notice that the supports of $D_1^{(cut)}$ and $D_2^{(cut)}$ are different, namely $Supp(D_1^{(cut)}) = [-k\sigma + r_0, k\sigma + r_0]$ while $Supp(D_2^{(cut)}) = [-k\sigma, k\sigma]$. We assume, without loss of generality, that $r_0 > 0$. We would like to switch from distribution $D_1^{(cut)}$ to $D_2^{(cut)}$, but unfortunately $R_\infty(\overline{D}_1^{(cut)} \| D_2^{(cut)})$ is not finite since $Supp(D_1^{(cut)})$ is not a subset of $Supp(D_2^{(cut)})$. To satisfy the latter condition, we first switch from $D_1^{(cut)}$ to $\overline{D}_1^{(cut)}$ by further cutting (by rejection sampling) the positive tail of $D_1^{(cut)}$ to ensure it does not go beyond the $k\sigma$ upper bound on tail of $D_2^{(cut)}$, and use a (mild condition) statistical distance step to lower bound $\overline{D}_1^{(cut)}(E)$. Then, in a second step using $Supp(\overline{D}_1^{(cut)}) = [-k\sigma + r_0, k\sigma] \subseteq [-k\sigma, k\sigma] = Supp(D_2^{(cut)})$, we derive a finite upper bound on $R_\infty(\overline{D}_1^{(cut)} \| D_2^{(cut)})$ to lower bound $D_2^{(cut)}(E)$. Details follow.

First SD Step. Since $Supp(D_1^{(cut)})$ is transformed into $\overline{D}_1^{(cut)}$ by rejection and re-sampling if a sample of $Supp(D_1^{(cut)})$ falls in $(k\sigma, k\sigma + r_0]$, we have $SD(D_1^{(cut)}, \overline{D}_1^{(cut)}) \leq D_1^{(cut)}((k\sigma, k\sigma + r_0]) = D_2^{(cut)}((k\sigma - r_0, k\sigma]) = D_{\mathbb{Z},\sigma}((k\sigma - r_0, k\sigma])/C_2$, where $C_2 = D_{\mathbb{Z},\sigma}([-k\sigma, k\sigma])$. Now, we have

$$\Delta \overset{\text{def}}{=} \frac{D_{\mathbb{Z},\sigma}((k\sigma - r_0, k\sigma])}{C_2} = \frac{\sum_{z\in(k\sigma - r_0, k\sigma])} \mathrm{e}^{-\pi z^2/\sigma^2}}{\sum_{z\in[-k\sigma, k\sigma])} \mathrm{e}^{-\pi z^2/\sigma^2}}.$$

For the numerator, we have an upper bound $\sum_{z\in(k\sigma - r_0, k\sigma])} \mathrm{e}^{-\pi z^2/\sigma^2} \leq \int_{k\sigma - r_0}^{\infty} \mathrm{e}^{-\pi z^2/\sigma^2} dz \leq \sigma \cdot \mathrm{e}^{-\pi(k\sigma - r_0)^2/\sigma^2}$, using the standard normal distribution upper bound $\int_\gamma^\infty \frac{1}{\sqrt{2\pi}\sigma} \cdot \mathrm{e}^{-z^2/\sigma^2} dz \leq \mathrm{e}^{-\gamma^2/(2\sigma^2)}$ for $\gamma \geq 0$. For the denominator, we have a lower bound $\sum_{z\in[-k\sigma, k\sigma]} \mathrm{e}^{-\pi z^2/\sigma^2} \geq 2 \cdot \sum_{z\in[0,k\sigma])} \mathrm{e}^{-\pi z^2/\sigma^2} \geq 2 \cdot (\int_0^\infty \mathrm{e}^{-\pi z^2/\sigma^2} dz - \int_{k\sigma}^\infty \mathrm{e}^{-\pi z^2/\sigma^2} dz) \geq \sigma \cdot (1 - 2 \cdot \mathrm{e}^{-\pi(k\sigma - r_0)^2/\sigma^2})$. Therefore, $SD(D_1^{(cut)}, \overline{D}_1^{(cut)}) \leq \Delta \leq \delta'/(1-2\delta') \leq 2\delta'$ if $\delta' \leq 1/4$, where $\delta' =$

$e^{-\pi(k\sigma-r_0)^2/\sigma^2}$. Defining $\delta = 2\delta'$, we have $\Delta \leq \delta$ if $\delta \leq 1/8$ and the conditions $r_0 \leq \sigma$ and $k \geq 1+\sqrt{1/\pi \cdot \ln(2/\delta)}$ hold. Therefore, for any event E we have $\overline{D}_1^{(cut)}(E) \geq D_1^{(cut)}(E) - \delta$.

Second RD step. The desired RD of order ∞ is defined by

$$R_\infty(\overline{D}_1^{(cut)} \| D_2^{(cut)})) = \max_{x\in[-k\sigma+r_0,k\sigma]} \frac{\overline{D}_1^{(cut)}}{D_2^{(cut)}(x)}.$$

Observe that, for each $x \in [-k\sigma + r_0, k\sigma]$, we have $\overline{D}_1^{(cut)}(x) = C \cdot D_1^{(cut)}(x)$, where the normalization constant $C = \frac{1}{1-D_1^{(cut)}((k\sigma,k\sigma+r_0])} = \frac{1}{1-D_2^{(cut)}((-k\sigma+r_0,k\sigma])} = \frac{1}{1-\Delta}$, where Δ is defined and upper bounded by δ above under the assumed conditions on k and δ. Since the $D_1^{(cut)}(x)$ and $D_2^{(cut)}(x)$ are shifts of each other, they have the same rejection sampling normalization constant with respect to D_1 (resp. D_2). Therefore, $D_1^{(cut)}(x)/D_2^{(cut)}(x) = D_1(x)/D_2(x)$ for each x in the support of both $D_1^{(cut)}$ and $D_2^{(cut)}$, and we have

$$\begin{aligned} R_\infty(\overline{D}_1^{(cut)} \| D_2^{(cut)}) &\leq \frac{1}{1-\delta} \cdot \max_{x\in[-k\sigma+r_0,k\sigma]} \frac{D_1(x)}{D_2(x)} \\ &= \max_{x\in[-k\sigma,k\sigma]} \frac{e^{\frac{-\pi(x-r_0)^2}{\sigma^2}}}{e^{-\pi\frac{x^2}{\sigma^2}}} \\ &= e^{\pi\cdot r_0^2/\sigma^2} \cdot \max_{x\in[-k\sigma+r_0,k\sigma]} e^{\frac{2\pi r_0}{\sigma^2}x} \end{aligned}$$

This is an exponential function and we get the max value at $x = k\sigma$:

$$R_\infty(\overline{D}_1^{(cut)} \| D_2^{(cut)}) = e^{1/(1-\delta)} \cdot e^{\pi\cdot r_0^2/\sigma^2 + 2\pi k\cdot r_0/\sigma}.$$

Since $0 < \delta \leq 1/8$, the first factor above is $\leq 1+2\delta \leq e^{2\delta}$. Also, a simple computation shows that the second factor is $\leq e$ if the condition $\sigma/r_0 \geq 4\pi \cdot k$ is satisfied using $k \geq 1$. We conclude, under the assumed parameter conditions that $R_\infty(\overline{D}_1^{(cut)} \| D_2^{(cut)})) \leq e^{1+2\delta} = c'(\delta)$ is constant for constant $\delta > 0$, so that, by the RD probability preservation property $D_2^{(cut)}(E) \geq \frac{1}{c(\delta)} \cdot \overline{D}_1^{(cut)}(E) \geq \frac{1}{c(\delta)} \cdot (D_1^{(cut)}(E) - \delta)$. Note that if $D_1^{(cut)}(E) = \varepsilon$, then by choosing $\delta = \varepsilon/2$, we get $D_2^{(cut)}(E) \geq \frac{1}{2c(\delta)} \cdot \varepsilon$, and we only need $k \geq 1+\sqrt{1/\pi \cdot \ln(2/\delta)}$ and σ/r_0 logarithmic in $1/\delta$, much smaller than σ/r_0 linear in $1/\delta$, which we would need if we were to use the 'SD only' analysis approach.

The above discussion immediately generalizes from the one-dimensional case of discrete Gaussian samples over $\mathbb{Z}$ to the m-dimensional case of discrete Gaussian samples over $\mathbb{Z}^m$ due to the independence of the m coordinates. The only changes to the above argument is that the statistical distance in the 'SD step' can multiply by at most a factor m, whereas the

RD in the 'RD step' above gets raised to the m'th power, where we replace r_0 by $\|\vec{r}_0\|_\infty$. We compensate for this by replacing the bound δ on Δ in the above analysis by the bound δ/m. We have therefore proved the following result used in our impersonation security proof, which improved upon the R_2-based analogue result for shifted Gaussians stated in [50].

Lemma 8. *For integer $m \geq 1$, real $\sigma > 0, k \geq 1$, real $0 < \delta \leq 1/8$ and vector $\vec{r}_0 \in \mathbb{Z}^m$, let $D_1^{(cut)} = D_{\mathbb{Z}^m,\sigma}^{(cut)} + \vec{r}_0$ and $D_2^{(cut)} = D_{\mathbb{Z}^m,\sigma}^{(cut)}$ be relatively shifted tail-cut discrete Gaussian distributions, where $D_{\mathbb{Z}^m,\sigma}^{(cut)}$ is the discrete Gaussian $D_{\mathbb{Z}^m,\sigma}$ with its tails cut to the support $[-k\sigma, k\sigma]^m$ by rejection sampling. If the conditions $k \geq 1 + \sqrt{1/\pi \cdot \ln(2m/\delta)}$ and $\sigma/\|\vec{r}_0\|_\infty \geq 4\pi \cdot k \cdot m$ hold, then, for any event E defined over the support of $D_1^{(cut)}$ we have*

$$D_2^{(cut)}(E) \geq \frac{1}{2e^{1+2\delta}} \cdot \left(D_1^{(cut)}(E) - \delta\right).$$

4.3.1 Correctness Analysis

correctness analysis We start with the noise sampled during key generation:

$$e_0 \overset{r}{\hookleftarrow} \chi_{\alpha q}$$

where χ is typically a Gaussian distribution with standard deviation αq. The noise bound of the first level cihertext $c = (\mathbf{c_0}, \mathbf{c_1})$ would be $B_0 = \|[< \mathbf{c}, \mathbf{s} >]_q\|_\infty$. We have

$$\begin{aligned}[\langle \mathbf{c}, \mathbf{s} \rangle]_q &= \mathbf{p_0 u} + t\mathbf{g} + \mathbf{m} + \mathbf{p_1 us} + t\mathbf{fs} \\ &= m + t(\mathbf{g} + \mathbf{fs} - \mathbf{e_0 u})\end{aligned}$$

So we can approximately bound $B_0 \leq t\|e_0\|_\infty^2$. The noise of the HD ciphertext (unmask) $\|e_{HD}\|_\infty$ can be bounded by $2nB_0 + nB_0^2$, recall that

$$enc(HD) = enc_1(\mathbf{T})C_1 + enc_2(\mathbf{Q})C_2 - 2enc_1(\mathbf{T})enc_2(\mathbf{Q})$$

The final masked ciphertext will have noise

$$\|e_{HDM}\|_\infty \leq (4\pi kn + 1)n(B_0^2 + 2B_0)$$

Where $k = 1 + \sqrt{\frac{1}{\pi \log 4nFAR^{-1}}}$. We can derive the final correctness condition:

$$q > 4\pi n^2 t(B_0^2 + 2B_0)$$

Notations

- αq standard deviation of the original noise distribution, assume Gaussian Distribution χ
- e_0 original noise used in key generation and original encryption $((\mathbf{p_0}, \mathbf{p_1})$ where $\mathbf{p_1} \overset{r}{\hookleftarrow} R_Q$ and $\mathbf{p_0} = -(\mathbf{p_1}\mathbf{s} + t\mathbf{e})$ with $\mathbf{e} \overset{r}{\hookleftarrow} \chi)$
- B_0 noise of the first level ciphertext

$$Enc_{pk}(\mathbf{m}) = (\mathbf{c_0}, \mathbf{c_1}) = (\mathbf{p_0}\mathbf{u} + t\mathbf{g} + \mathbf{m}, \mathbf{p_1}\mathbf{u} + t\mathbf{f})$$

4.4 Results

Chapter 5

The third protocol - Computing and Comparing HD Homomorphically

5.1 Introduction

In the previous chapter, we assumed that given the Hamming Distance (HD) of two binary bitstring templates, it is infeasible to infer or learn any information about the original templates. The assumption is plausible as long as the distributions of the registered and the queried templates are not in any way correlated, studying about those distributions is out of scope of the project. However, in this chapter, we introduce our first attempt to hide this information from the server as well. Such result not only helps the protocol working regardless of such dependence, it is also the first step we can make to improve the authentication protocol to be secured against an *active* server security model. Although the communication size of this variant might not be suitable to apply in practice for current network infrastructures, one can still find it helpful when using some generic techniques of the protocol when applying them to balance the computation time and the communication size of lattice-based cryptographic protocols. The contributions of this variant includes

- A technique to compare Hamming Distance (HD) homomorphically by computing the Most Significant Bit (MSB) of a specific ciphertext.
- A packing methods conversion technique to transform a ciphertext encrypting a binary-encoded to a unary-encoded plaintext.
- Some extensions of the Zero Knowledge Proof (ZKP) technique that we used in previous chapter to prove also the formats of a message in addition to the knowledge of the plaintext itself.

- The combination of ZKP protocols to balance the communication size tradeoffs when used with lattice-based cryptosystem.

5.2 The Homomorphic tools

5.2.1 Extracting the Most Significant Bit homomorphically

We observe that $HD < \tau \iff MSB(2^l + \tau - HD) = 1$, where l is the bit-length of HD and MSB denotes the Most Significant Bit. This is our attempt to compare Hamming Distance homomorphically, the idea is letting the server computing homomorphically the ciphertext of $MSB(2^l + \tau - HD)$, then having the client to decrypt and send back with a zero knowledge proof of the authentication result. This section discuss a variant of the technique from [20] to compute the MSB of the plaintext M given $Enc(M)$. The technique is use in 8. The main idea comes from the way we encode the message M before the encryption.

Given $M \in \mathbb{Z}_{2n}$, we observe that

$$\begin{cases} MSB(M) = 0 \iff M \in [0,n) \\ MSB(M) = 1 \iff M \in [n,2n) \end{cases}$$

Assume that we encrypt M using the SHE scheme (section 2.4.1). With the message space R_t and assume that $2n < t < q$, we can encode M as a ring element $m \in R_t$ as follows

$$m(x) = 0x^0 + 0x^1 + \cdots + 1x^M + \cdots + 0x^{n-1} \text{ if } M \in [0,d)$$

or, due to $x^n = -1$ in R_q, we can also encode M as

$$m(x) = 0x^0 + 0x^1 + \cdots - 1x^M + \cdots + 0x^{n-1} \text{ if } M \in [n,2n)$$

The ciphertext $Enc(M)$ has this form $(\mathbf{c} = \mathbf{p_0}\mathbf{u} + t\mathbf{g} + \mathbf{m}, \mathbf{c}' = \mathbf{p_1}\mathbf{u} + t\mathbf{f})$, where $(\mathbf{p_0}, \mathbf{p_1})$ is the public key and $\mathbf{u}, \mathbf{f}, \mathbf{g} \overset{r}{\hookleftarrow} \chi$. We denote $rot(\mathbf{c}) \in \mathbb{Z}_q^{n\times n}$ to be an anti-circulant square matrix, whose first column is $\mathbf{c}$ and the other columns are the cyclic rotations of $\mathbf{c}$ with the cycled entries negated

$$rot(\mathbf{c}) = \begin{bmatrix} c_0 & -c_{n-1} & -c_{n-2} & \cdots \\ c_1 & c_0 & -c_{n-1} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots \end{bmatrix}$$

It's easy to see that $rot(\mathbf{cu}) = rot(\mathbf{c})\vec{u}$.

Lemma 9. *Given $M \in Z_{2n}$ and $(\mathbf{c}, \mathbf{c}') = Enc(M)$ is the first level ciphertext of M from the BV scheme. Let $\vec{1} = \{1, 1, \dots, 1\} \in \mathbb{Z}^n$. The transformed ciphertext $(l, l') \leftarrow (\vec{1}rot(\mathbf{c})[0], \vec{1}rot(\mathbf{c}')) \in (\mathbb{Z}_Q, \mathbb{Z}_Q^n)$ encrypts the MSB information of M.*

Algorithm 4 Most Significant bit extraction

```
1: procedure MSBEXTRACT($\mathbf{c}$)
2:     $allOne = \{1, \dots, 1\}$
3:     $rot_{c_0} \leftarrow rot(c_0)$
4:     $lwe_0 \leftarrow allOne \times rot_{c_0}[0]$
5:     $rot_{c_1} \leftarrow rot(c_1)$
6:     $lwe_1 \leftarrow allOne \times rot_{c_1}$
7:     return $(lwe_0, lwe_1)$
```

5.2.2 Proof of lemma 9

Proof. We have

$$\begin{aligned}
\vec{1}rot(\mathbf{c})[0] \in \mathbb{Z}_Q &= \vec{1}\overrightarrow{\mathbf{p_0 u} + t\mathbf{g} + \mathbf{m}} \\
&= \vec{1}\overrightarrow{p_0 u} + t\vec{1}\vec{g} + \vec{1}\vec{m} \\
&= -\vec{1}(rot(p_1 s)\vec{u} + t.rot(e)\vec{u}) + t\vec{1}\vec{g} \pm 1 \\
&= -\vec{1}rot(p_1)\vec{u}\vec{s} - t.rot(e)\vec{u}\vec{s} + t\vec{1}\vec{g} \pm 1
\end{aligned}$$

Provided that $\mathbf{c}' = \mathbf{p_1 u} + t\mathbf{f}$, the decryption $\langle (l, l'), (1, \mathbf{s}) \rangle \mod t$ is ± 1. This implies the MSB information of M as discussed. □

5.2.3 Converting from binary-encoded to unary-encoded plaintext

This section discuss a linear transformation to map a message $b \in \{0, 1\}$ to a message x^{jb} for $j \geq 1$. Let $T : cx + d = y$ be the linear transformation. We want T to map $0 \to x^{j0}$ and $1 \to x^j$, or

$$\begin{cases} c.0 + d = 1 \\ c + d = x^j \end{cases} \Leftrightarrow \begin{cases} c = x^j - 1 \\ d = 1 \end{cases}$$

Due to the homomorphism property of BV cryptosystem , we can apply T in the ciphertext domain to obtain $Enc(x^{jb})$ given $Enc(b)$, for $b \in \{0,1\}$:

$$\begin{aligned} Enc(x^{jb}) &= Enc(c)Enc(b) + Enc(d) \\ &= Enc(x^j - 1)Enc(b) + Enc(1) \end{aligned} \tag{5.1}$$

From the implementation point of view, this operation can be done faster by doing $\mathbf{u}, \mathbf{f}, \mathbf{g} \leftarrow 0$ instead of sampling those from χ during $Enc(x^j - 1)$ and $Enc(1)$, which results $Enc(x^j - 1) = (x^j - 1, 0)$ and $Enc(1) = (1, 0)$. These are still valid encryptions and will not affect correctness of Eq. (5.1).

This submodule is used in our protocol at 7. Recall that the server needs to compute $Enc^{(2)}(HD')$ which is the encryption of HD' in the second mode of encoding (message is encoded in the exponent instead of the coefficient of the polynomial). Given $HD' = \sum_{i=0}^{l-1} b_i 2^i$ and $Enc(b_i)$, the server can convert them to $Enc(x^{jb})$ and do

$$Enc^{(2)}(x^{HD'}) = \prod_{i=0}^{l-1} Enc^{(2)}(x^{b_i 2^i})$$

Note that this operation involves $log(l)$ levels of homomorphic multiplication, where l is the bit length of the Hamming Distance.

Algorithm 5 Binary to Unary ciphertext

```
1: procedure ToUnary($\mathbf{c_i}$)
2:     for $j = 0, \ldots, l-1$ do
3:         let $\mathbf{hd_j} \leftarrow (x^j - 1, 0) \times \mathbf{c_j} + (1, 0)$
4:     let $\mathbf{hd} \leftarrow \mathbf{hd_0} \times \mathbf{hd_1} \times \cdots \times \mathbf{hd_{l-1}}$
5:     return hd
```

5.3 The Zero Knowledge Tools

5.3.1 ZKPoPK of Regev Cryptosystem

In this section, we first review one application of [53] technique to do ZKPoPK for Regev Cryptosystem, then we elaborate to our variants to be used in the protocol. Given parameters q, m, n, t, χ of a typical LWE-based cryptosystem, we can describe a variant of Regev's system as follows.

Kengen. A secret key $\vec{s}$ can be chosen from χ^n. The public key then is generated as $pk = (p_0, p_1) = (\mathbf{A}, \mathbf{A}\vec{s} + t\vec{e})$. Where $\mathbf{A} \xleftarrow{r} \mathbb{Z}_q^{m\times n}$ and $\vec{e} \xleftarrow{r} \chi^n$

Encrypt. Given a message $M \in \mathbb{Z}_q$, the ciphertext C is computed by first sampling a random vector $\vec{r} \in \chi^n$ and do $C = (c_0, c_1) = (p_0\vec{r}, p_1\vec{r} + M)$

Decrypt. Given a ciphertext $C = (c_0, c_1)$, the message M can be recovered by compute $M = c_1 - c_0\vec{s} \mod t$

In [53], the ZKPoPK was done by proving the encryption relation

$$R_{Regev}^{q,m,n,t,\chi} = \{((p_0,p_1),(c_0,c_1),\vec{r}||M) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m) \times (\mathbb{Z}_q^n \times \mathbb{Z}_q) \times \mathbb{Z}_q^{n+1} : (c_0 = p_0\vec{r}) \wedge (c_1 = p_1\vec{r} + M)\}$$

Let $\mathbf{A}' = \begin{bmatrix} p_1, 1 \\ p_0, 0 \end{bmatrix}$, and $\mathbf{y} = \begin{bmatrix} c_1 \\ c_0 \end{bmatrix}$ be public parameters in the proof and let $\mathbf{x} = \begin{bmatrix} \vec{r} \\ M \end{bmatrix}$ be the *Prover*'s witness. We observe that $\mathbf{A}'\mathbf{x} = \mathbf{y} \mod q$, that is, $\mathbf{x}$ is a solution to the ISIS problem defined by $(\mathbf{A}', \mathbf{y})$ and we can use the **SternExt** protocol (section 2.5.1) to obtain an efficent ZKPoPK. This works in the symmetric key setting, where the *Prover* knows the random $\vec{r}$ to use as his witness. In our context, the client does not knows $\vec{r}$ as encryption was done by the server. Let's look at the decryption equation:

$$\begin{aligned} c_1 - c_0\vec{s} &= p_1\vec{r} + M - p_0\vec{r}\vec{s} \\ &= \mathbf{A}\vec{s}\vec{r} + t\vec{e}\vec{r} + M - \mathbf{A}\vec{r}\vec{s} \\ &= t\tilde{e} + M \end{aligned}$$

Therefore, we can write the decryption relation as

$$R_{Regev,dec}^{q,m,n,t,\chi} = \{((p_0,p_1),(c_0,c_1),\vec{s},\vec{e},\tilde{e},M) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m) \times (\mathbb{Z}_q^n \times \mathbb{Z}_q) \times \chi^n \times \chi^n \times \chi \times \mathbb{Z}_Q : \tag{5.2}$$

$$(p_1 = p_0\vec{s} + t\vec{e}) \wedge (c_1 = c_0\vec{s} + t\tilde{e} + M)\}$$

In this situation, we can let $\mathbf{A}' = \begin{bmatrix} c_0, t, 0, 1 \\ p_0, 0, t, 0 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} c_1 \\ p_1 \end{bmatrix}$ to be the public parameters and let $\mathbf{x} = \begin{bmatrix} \vec{s} \\ \tilde{e} \\ \vec{e} \\ M \end{bmatrix}$ and apply the **SternExt** to obtain the ZKPoPK . We note that the two separate rows of $\mathbf{A}'$ prove the two separate relations in (5.2): The *Prover* needs to prove that he knows a secret $\vec{s}$ that can decrypt (c_0, c_1) and he also needs to prove that the secret key $\vec{s}$ is also the one corresponding to the public key (p_0, p_1).

5.3.2 ZKPoPK of BV cryptosystem

In this section, we discuss ZKPoPK for a ring variant of Regev scheme discussed above, which is the BV system that we use in our application context (section 2.4.1), we will refer to this proof as **ZKPValidEnc**. Recall that our public key is a pair of ring elements $pk = (\mathbf{p_0}, \mathbf{p_1})$ where $\mathbf{p_0}, \mathbf{p_1} \in R_Q$ and $\mathbf{p_1 s} + t\mathbf{e} = -\mathbf{p_0}$. This is one of the relation that the client needs to prove later on. Next, given a ciphertext $c = (\mathbf{c_0}, \mathbf{c_1})$, the original plaintext $\mathbf{m} \in R_Q$ can be recovered by $\mathbf{m} = \mathbf{c_0} + \mathbf{c_1 s} \mod t$. We can write $\mathbf{c_1 s} + \mathbf{c_0} = \mathbf{m} + t\mathbf{e}'$, or $\mathbf{c_1 s} - t\mathbf{e}' - \mathbf{m} = -\mathbf{c_0}$. In summary, the relation that we need to do the ZKPoPK is:

$$R_{BV}^{Q,n,t,\chi} = \{((\mathbf{c_0}, \mathbf{c_1}), (\mathbf{p_0}, \mathbf{p_1}), \mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m} \in (R_Q \times R_Q) \times (R_Q \times R_Q) \times \chi^n \times \chi^n \times \chi^n \times R_t : \tag{5.3}$$

$$(\mathbf{p_1 s} + t\mathbf{e} = -\mathbf{p_0}) \wedge (\mathbf{c_1 s} - t\mathbf{e}' - \mathbf{m} = -\mathbf{c_0})\}$$

We can do similarly to what was done in section 5.3.1, where we tried to derive the ISIS relation ($\mathbf{Ax} = \mathbf{y} \mod q$) from the above relation and obtain the ZKP accordingly. The matrix $\mathbf{A}$ should be derived from $\mathbf{T} = \begin{bmatrix} \mathbf{c_1}, -t, 0, -1 \\ \mathbf{p_1}, 0, t, 0 \end{bmatrix}$ in order to obtain (5.3). Note that with $\mathbf{c_1}, \mathbf{p_1} \in R_Q$, we can construct $\mathbf{A}$ by replacing from $\mathbf{T}$: $\mathbf{c_1}$ and $\mathbf{p_1}$ are replaced by $rot(\mathbf{c_1})$ and $rot(\mathbf{p_1})$, ;constants are replaced by the product of the constants with the identity matrix $\mathbf{I}$. Recall that $rot(\mathbf{c}) \in \mathbb{Z}_q^{n\times n}$ is defined to be an anti-circulant square matrix, whose first column is $\mathbf{c}$ and the other columns are the cyclic rotations of $\mathbf{c}$ with the cycled entries negated

$$rot(\mathbf{c}) = \begin{bmatrix} c_0 & -c_{n-1} & -c_{n-2} & \cdots \\ c_1 & c_0 & -c_{n-1} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots \end{bmatrix}$$

So, the matrix **A** is of the following form:

$$\left[\begin{array}{|c|c|c|c|}\hline rot(\mathbf{c_1}) & -t\mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \hline rot(\mathbf{p_1}) & \mathbf{0} & t\mathbf{I} & \mathbf{0} \\ \hline \end{array}\right] \tag{5.4}$$

By constructing the matrix **A** this way, we can let $\mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \mathbf{e'} \\ \mathbf{e} \\ \mathbf{m} \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} -\mathbf{c_0} \\ -\mathbf{p_0} \end{bmatrix}$ and come up with the original ISIS relation $\mathbf{Ax} = \mathbf{y} \mod Q$. Again, we can use the **SternExt** protocol (section 2.5.1) to obtain the ZKPoPK with $\mathbf{x}$ being the *Prover*'s witness and $\mathbf{A}, \mathbf{y}$ being the public parameters (Algorithm 15).

Algorithm 6 ZKPoPK for BV

```
1: procedure ZKPBV($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e'})$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let Z be the $n \times n$ zero matrix
6:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I), (rot_{p_1}, Z, tI, Z))$
7:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e'}, \mathbf{e}, \mathbf{m})$
8:     $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
9:     Return SternExt($\mathbf{A}, \mathbf{x}, \mathbf{y}$)
```

5.3.3 ZKP of plaintext with zero coefficients

This section extends the previous proof with several submodules. Firstly, we need a proof to convince the server that our ciphertext encrypts the message of the form $\mathbf{m}(\mathbf{x}) = 0 + m_1x^1 + m_2x^2 + \cdots + m_{n-1}x^{n-1}$. Next, we need another proof to convince about the encryption of $\mathbf{m}(\mathbf{x}) = m_0 + 0x^1 + 0x^2 + \cdots + 0x^{n-1}$. We also use some other proofs for message containing all 1s or 0s. In other words, we want the *Prover* to convince the *Verifier* about the formats of the plaintext in addition to the knowledge of plaintext.

Proving $\mathbf{m} = \sum_{i=0}^{l}(m_i x^i) \wedge m_0 = 0$ (ZKPExt1). If and only if the *Prover* P has the message $\mathbf{m}$ of this form, he can do:

$$rot(\mathbf{c_0})\mathbf{s} - t\mathbf{I}\mathbf{e}' - \begin{bmatrix} 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \mathbf{m_0} = -\mathbf{c_0}$$

$$\iff rot(\mathbf{c_0})\mathbf{s} - t\mathbf{I}\mathbf{e}' - \begin{bmatrix} m_0 \\ 0 \\ \dots \\ 0 \end{bmatrix} = -\mathbf{c_0}$$

In order to do this proof, we can do similarly to what was done in section 5.3.2, only slight modification need to be made is in the matrix represented in (5.4): We completely remove all the last $(n-1)$ columns of the last n columns during constructing the matrix $\mathbf{A}$ (The new $\mathbf{A}$ has dimension $n \times (3n+1)$ instead of $n \times 4n$). The specification is summarized in Algorithm (16).

Algorithm 7 ZKP for zero constant coefficient

```
1: procedure ZKPExt1($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e}')$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let I' be 1 column matrix with all 1s.
6:     let Z be the $n \times n$ zero matrix
7:     let Z' be 1 column matrix with all 0s.
8:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I'), (rot_{p_1}, Z, tI, Z'))$
9:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m})$
10:    $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
11:    Return SternExt$(\mathbf{A}, \mathbf{x}, \mathbf{y})$
```

Proving $\mathbf{m} = \sum_{i=0}^{l}(m_i x^i) \wedge m_j = 0\ for\ j = 1, 2 \dots l-1$. (ZKPExt2) This proof can be done similarly to **ZKPExt1** with the same reasonings. Except that in this proof, in stead of removing $(n-1)$ columns, we remove only the first one column of the last n columns of $\mathbf{A}$. The result is the matrix $\mathbf{A}$ with dimension $n \times (4n-1)$. The specification is summarized in Algorithm (17).

Proving $\mathbf{m} = \sum_{i=0}^{l}(m_i x^i) \wedge m_j = 0 \vee m_j = 1\ for\ j = 0, 1 \dots l-1$ (ZKPExt3 and ZKPExt4). Following the previous extensions, proving a message that contain all zero (or all ones)

Algorithm 8 ZKP for only constant coefficient

```
1: procedure ZKPEXT2($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e}')$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let I' be I with the first column removed.
6:     let Z be the $n \times n$ zero matrix
7:     let Z' be Z with the first column removed.
8:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I'), (rot_{p_1}, Z, tI, Z'))$
9:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m})$
10:    $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
11:    Return SternExt$(\mathbf{A}, \mathbf{x}, \mathbf{y})$
```

coefficients is trivial by modifying the matrix **A** and setting up the bound of **SternExt** accordingly.

5.3.4 ZKP of re-encryption correctness

This section first discusses the module **ZKPUnpack(c, c')**, that proves the correctness of the re-encryption of a single slot unpacked plaintext $\mathbf{m}' = Dec(\mathbf{c}')$ from the coefficients-packed ciphertext $\mathbf{m} = Dec(\mathbf{c})$. Then we details the module **ZKPBinDecomp**$(\mathbf{c}, \mathbf{c_i})$, that proves the correctness of re-encryption of a binary-encoded plaintext as a unary-encoded one.

ZKPUnpack(c,c). The relation of the proof is

$$R_{\mathbf{ZKPUnpack}} = \{\mathbf{c}, \mathbf{c}' \in R_q \times R_q : \mathbf{m} = Dec(\mathbf{c}) \wedge \mathbf{m}' = Dec(\mathbf{c}') \wedge m_0 = m'_0 \wedge (m'_i = 0 \, \forall i \neq 0)\}$$

Given $m_0 = m'_0$, we observe $\mathbf{m}(\mathbf{x}) - \mathbf{m}'(\mathbf{x}) = 0 + m''_1 x + \cdots + m''_{n-1} x^{n-1}$. Therefor, **ZKPUnpack** can be done as specified in Algorithm (18)

Algorithm 9 ZKP of coefficients transform

```
1: procedure ZKPUNPACK($\mathbf{c}, \mathbf{c}'$)
2:     Let $b_1 \leftarrow$ ZKPExt1$((\mathbf{c} - \mathbf{c}', \mathbf{pk}), (\mathbf{Dec}(\mathbf{c} - \mathbf{c}'), \mathbf{s}, \mathbf{e}, \mathbf{e}'))$
3:     Let $b_2 \leftarrow$ ZKPExt2$((\mathbf{c}', \mathbf{pk}), (\mathbf{Dec}(\mathbf{c}'), \mathbf{s}, \mathbf{e}, \mathbf{e}'))$
4:     Return $b_1 \wedge b_2$
```

ZKPBinDecomp$(\mathbf{c}, \mathbf{c_i})$. The relation of the proof is

$$R_{\mathbf{ZKPBinDec}} = \{\mathbf{c}, \mathbf{c_i} \in R_q \times R_q^l : \mathbf{m} = Dec(\mathbf{c}) \wedge \mathbf{m_0} = \sum_{i=0}^{l-1} b_i 2^i \wedge \mathbf{c_i} = Enc(b_i)\}$$

The specification of the proof is detailed in Algorithm (12)

Algorithm 10 ZKP of encoding transform

```
1: procedure ZKPBinDecomp($\mathbf{c}, \mathbf{c_i}$)
2:     Let $\mathbf{c}' \leftarrow \sum_{i=0}^{l-1} \mathbf{c_i} 2^i$
3:     Let $\mathbf{c}'' \leftarrow c - c'$
4:     Return ZKPExt1($(\mathbf{c}'', \mathbf{pk}), (\mathbf{Dec}(\mathbf{c}''), \mathbf{s}, \mathbf{e}, \mathbf{e}')$)
```

Applications in our protocol. In 5, after receiving the ciphertext $C_{HD'}$, which encrypts a plaintext of the form $(HD', g_1, \ldots, g_{n-1})$, $\mathscr{U}_k$ removes the noise terms $g_1, \ldots, g_{n-1}$ and sends back to $\mathscr{S}$ the re-encryption $C_{HD'_0} = Enc(HD', 0, 0, \ldots, 0)$. The client needs to prove that he performs this step correctly, this is done by $\mathbf{ZKPUnpack}(\mathbf{C_{HD}}, \mathbf{C_{HD'_0}})$. In this step, the server also receives $\mathbf{c_i} = Enc(b_i)$ to compute $Enc(x^{HD'})$, as discussed in section 5.2.3. Before doing this operation, the client needs to convince the server about the bits sent are actually the ones decomposed from HD'. This proof is done by $\mathbf{ZKPBinDecomp}(\mathbf{C_{HD'_0}}, \mathbf{c_i})$.

Besides, In 1, $\mathscr{U}_k$ uses **ZKPBV** to convince $\mathscr{S}$ that he is authenticating with a valid template Y. Moreover, in 9, $\mathscr{U}_k$ can use either **ZKPExt3** or **ZKPExt4** to convince $\mathscr{S}$ about the authentication result.

5.3.5 Schnorr-based ZKP technique

Theorem 14. *Figure (5.1) is a $\sum'$-Protocol for the following relations:*

$$\begin{aligned}\mathscr{R} = \{(\mathbf{c_0}, \mathbf{c_1}, \mathbf{p_0}, \mathbf{p1}, t), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e_0}) : \mathbf{c_0} = -\mathbf{c_1}\mathbf{s} + t\mathbf{e} + \mathbf{m} \wedge \mathbf{p_0} = -\mathbf{p_1}\mathbf{s} - t\mathbf{e_0} \\ \wedge \|\mathbf{m}\|_\infty \leq t \wedge \|\mathbf{e}\| \leq \beta \wedge \|\mathbf{s}\|, \|\mathbf{e_0}\| \leq \beta_0\}\end{aligned} \tag{5.5}$$

$$\begin{aligned}\mathscr{R}' = \{(\mathbf{c_0}, \mathbf{c_1}, \mathbf{p_0}, \mathbf{p1}, t), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e_0}) : \mathbf{2c_0} = -\mathbf{2c_1}\mathbf{s} + 2t\mathbf{e} + 2\mathbf{m} \wedge \mathbf{2p_0} = -\mathbf{2p_1}\mathbf{s} - 2t\mathbf{e_0} \\ \wedge \|\mathbf{2m}\|_\infty \leq 2t \wedge \|\mathbf{2e}\| \leq 2\beta \wedge \|\mathbf{2s}\|, \|\mathbf{2e_0}\| \leq 2\beta_0\}\end{aligned} \tag{5.6}$$

The protocol has a knowledge error of $\frac{1}{2n}$ *and a completeness error of* $1 - \frac{1}{M}$.

Proof. We prove the properties specified in Definition 23.

Completeness. If the Prover P sends a response, we have that:

$$\begin{aligned}-\mathbf{c_1}\mathbf{s_s}+t\mathbf{s_e}+\mathbf{s_m} &= -\mathbf{c_1}(\mathbf{r_s}+\mathbf{x^c s})+t(\mathbf{r_e}+\mathbf{x^c e})+\mathbf{r_m}+\mathbf{x^c m}\\ &= \mathbf{x^c}(-\mathbf{c_1 s}+t\mathbf{e}+\mathbf{m})-\mathbf{c_1 r_s}+t\mathbf{r_e}+\mathbf{r_m}\\ &= \mathbf{x^c c_0}+\mathbf{t_c}\end{aligned}$$

Similarly

$$\begin{aligned}-\mathbf{p_1 s_s}-t\mathbf{s_{e_0}} &= -\mathbf{p_1}(\mathbf{r_s}+\mathbf{x^c s})-\mathbf{t}(\mathbf{r_{e_0}}+\mathbf{x^c e_0})\\ &= \mathbf{x^c}(-\mathbf{p_1 s}-t\mathbf{e_0})-\mathbf{p_1 r_s}-t\mathbf{r_{e_0}}\\ &= \mathbf{x^c p_0}+\mathbf{t_k}\end{aligned}$$

Regarding norms, we have $||\mathbf{s_s}|| = ||\mathbf{r_s}+\mathbf{x^c s}|| \leq ||\mathbf{r_s}|| + ||\mathbf{s}|| \leq 2\beta_0$, and similarly for $||\mathbf{s_e}||, ||\mathbf{s_m}||, ||\mathbf{s_{e_0}}||$.

Special Soundness. Before proving this property, we describe the following technical lemma (the detailed proof can be found in [8])

Lemma 10. *Let n be a power of 2 and let $0 < i, j < 2n-1$. Then $2(x^i - x^j)^{-1}$ mod (x^n+1) only has coefficients in $\{-1,0,1\}$*

Assume that a dishonest prover does not know 'small' $2\mathbf{s}, 2\mathbf{e}, 2\mathbf{m}$ and $2\mathbf{e_0}$ and can output different accepted transcripts of a same commitment: $(c_{aux}, c', (d'_{aux}, \mathbf{t'_c}, \mathbf{t'_k}, \mathbf{s'_s}, \mathbf{s'_e}, \mathbf{s'_m}, \mathbf{s'_{e_0}}))$ and $(c_{aux}, c'', (d''_{aux}, \mathbf{t''_c}, \mathbf{t''_k}, \mathbf{s''_s}, \mathbf{s''_e}, \mathbf{s''_m}, \mathbf{s''_{e_0}}))$. From the binding property of the auxilary commitment scheme [66], we have $\mathbf{t'_c} = \mathbf{t''_c} := \mathbf{t_c}$ and $\mathbf{t'_k} = \mathbf{t''_k} := \mathbf{t_k}$. As the transcripts pass the checks by the verifier, we have:

$$\begin{cases}\mathbf{x^{c'}c_0}+\mathbf{t_c} = -\mathbf{c_1 s'_s}+t\mathbf{s'_e}+\mathbf{s'_m}\\ \mathbf{x^{c'}p_0}+\mathbf{t_k} = -\mathbf{p_1 s'_s}-t\mathbf{s'_{e_0}}\end{cases} \quad \& \quad \begin{cases}\mathbf{x^{c''}c_0}+\mathbf{t_c} = -\mathbf{c_1 s''_s}+t\mathbf{s''_e}+\mathbf{s''_m}\\ \mathbf{x^{c''}p_0}+\mathbf{t_k} = -\mathbf{p_1 s''_s}-t\mathbf{s''_{e_0}}\end{cases}$$

By substracting the equations we get:

$$\begin{cases}\mathbf{c_0}(\mathbf{x^{c'}}-\mathbf{x^{c''}}) = -\mathbf{c_1}(\mathbf{s'_s}-\mathbf{s''_s})+t(\mathbf{s'_e}-\mathbf{s''_e})+(\mathbf{s'_m}-\mathbf{s''_m})\\ \mathbf{p_0}(\mathbf{x^{c'}}-\mathbf{x^{c''}}) = -\mathbf{p_1}(\mathbf{s'_s}-\mathbf{s''s})-t(\mathbf{s'_{e_0}}-\mathbf{s''_{e_0}})\end{cases}$$

Multiplying by $2(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1}$:

$$\begin{cases} \mathbf{2c_0} = -\mathbf{c_1}2(\mathbf{s}'_\mathbf{s} - \mathbf{s}''_\mathbf{s})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1} + t2(\mathbf{s}'_\mathbf{e} - \mathbf{s}''_\mathbf{e})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1} + 2(\mathbf{s}'_\mathbf{m} - \mathbf{s}''_\mathbf{m})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1} \\ \mathbf{p_0} = -\mathbf{p_1}\mathbf{2}(\mathbf{s}'_\mathbf{s} - \mathbf{s}''_\mathbf{s})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1} - t2(\mathbf{s}'_{\mathbf{e_0}} - \mathbf{s}''_{\mathbf{e_0}})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1} \end{cases}$$

Let $2\hat{\mathbf{s}} = 2(\mathbf{s}'_\mathbf{s} - \mathbf{s}''_\mathbf{s})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-1}$, $2\hat{\mathbf{e}} = 2(\mathbf{s}'_\mathbf{e} - \mathbf{s}''_\mathbf{e})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-\mathbf{1}}$, $2\hat{\mathbf{e_0}} = 2(\mathbf{s}'_{\mathbf{e_0}} - \mathbf{s}''_{\mathbf{e_0}})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-}\mathbf{1}$ and $2\hat{\mathbf{m}} = 2(\mathbf{s}'_\mathbf{m} - \mathbf{s}''_\mathbf{m})(\mathbf{x}^{\mathbf{c}'} - \mathbf{x}^{\mathbf{c}''})^{-\mathbf{1}}$ and applying the result from Lemma 10, we see that P knows the 'small' secrets $\|\hat{\mathbf{2s}}\| < 2\beta_0$, $\|\hat{\mathbf{2e}}\| < 2\beta_0$, $\|\hat{\mathbf{2e_0}}\| < 2\beta$ and $\|\hat{\mathbf{2m}}\| < 2t$ that can pass the verifier's checks. This contradicts our assumption. In other words, the prover has to know the secret to pass the proof with probability $\frac{1}{2n}$.

Honest Verifier Zero-Knowledge. The verifier can use a simulator S to reproduce the protocol transcript as follows.

- V samples $c \overset{r}{\hookleftarrow} \mathscr{C}$
- With probability 1/M, S chooses $\mathbf{s_s}, \mathbf{s_{e_0}} \overset{r}{\hookleftarrow} D_{\beta_0}$, $\mathbf{s_e} \overset{r}{\hookleftarrow} D_\beta$ and $\mathbf{s_m} \overset{r}{\hookleftarrow} D_t$.
- S computes $\mathbf{t_c} = -\mathbf{c_1}\mathbf{s_s} + t\mathbf{s_e} + \mathbf{s_m} - \mathbf{x^c}\mathbf{c_0}$ and $\mathbf{t_k} = -\mathbf{p_1}\mathbf{s_s} - t\mathbf{s_{e_0}} - \mathbf{x^c}\mathbf{p_0}$ and $(c_{aux}, d_{aux}) \leftarrow aCommit(\mathbf{t_c}, \mathbf{t_k})$.
- S outputs $(c_{aux}, c, ((\mathbf{t_c}, \mathbf{t_k}), d_{aux}, (\mathbf{s_s}, \mathbf{s_{e_0}}, \mathbf{s_e}, \mathbf{s_m})))$

If no abort occurs, the distribution of $(\mathbf{s_s}, \mathbf{s_{e_0}}, \mathbf{s_e}, \mathbf{s_m})$ does not depend on real secrets hence the simulated transcript is insdistinguishable from the real one. If an abort occurs, the indistinguishability comes from the hiding property of the commitment scheme?

□

Summary of previous works

Lyubashevsky's proof systems ([54], [55]): This system is similar to classical Schnorr's ZKP system, the main advantage is its communication size efficiency. However, this approach is not zero knowledge (it is only proved to be witness indistinguishable). Moreover, there is a completeness error in each round, and lastly, the extraction gap is $O(\sqrt{n})$.

A quick discussion on extraction gap: we refer to extraction gap as a security parameter of ZKP systems for ISIS problem. Generally, we say a proof accepts an extraction gap γ ($\gamma > 1$) if the knowledge extractor of the protocol can extract a vector $\vec{v}$ such that

$\|v\|_\infty = \gamma\|w\|_\infty$, where $\|w\|_\infty$ is a valid witness from the *Prover*. Ideally, when $\gamma = 1$, one can use the extractor to solve the underlying ISIS instance. In other words, ZKP with $\gamma = 1$ is strongly secure as breaking it is at least as hard as solving the underlying problem. When $\gamma > 1$, the soundness of the proof has to rely on stronger hardness assumption of certain potentially easier problem is hard to solve.

Micciancio-Vadhan proof system [61]: This protocol provide ZKP for GapCVP problem, it can be adapted to prove ISIS relation: Let $\mathbf{B}$ be a basis of the perp lattice $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}.\mathbf{x} = \mathbf{0} \mod q\}$. and $\mathbf{t} \in \mathbb{Z}^m : \mathbf{A}.\mathbf{t} = \mathbf{y} \mod q$ ($\mathbf{B}$ and $\mathbf{t}$ can be computed efficiently), then run the ZKP protocol for $GapCVP_\gamma^\infty$ with public parameters $(\mathbf{B}, \mathbf{t}, \beta)$. The *Prover*'s witness is $\mathbf{e} = \mathbf{t} - \mathbf{x}$. Recall that The knowledge extractor of the protocol can output a vector $\mathbf{e}' \in \Lambda_q^\perp(\mathbf{A}$ such that $\|\mathbf{t} - \mathbf{e}'\|_\infty \leq g.\beta$ for some $g \geq O(\sqrt{m})$. This implies a perfect ZKPoPK with extraction gap $O(\sqrt{m})$.

Stern's protocol [] and extension: The original Stern's proof worked on the SD relation . This ZKP does not have any completeness error and there is no extraction gap, i.e., $\gamma = 1$. Extension from this work include KTX [137,85], which associated with the following relation .

citeStern

SD relation

ktxRelation

The work of [53] provide a proof for ISIS relation . Stern's based techniques inherently do not have extraction gap nor completeness error, the communication efficiency is . Our protocol will also be based on this technique, the improvement is, ours can prove different bounds of different elements in the *Prover*'s witness vector. Such improvement is important in ZKPoPK for lattice-based cryptosystem, especially in contexts where public key settings and homomorphic operations are used: The bounds of messages and errors can be largely different: for example, the original message can be binary while the noises' bounds increase according to the ladder of homomorphic operations.

ISIS relation

missing figu

5.3.6 ZKP and $\sum$-Protocol

A Zero-Knowledge Proof of Knowledge (ZKPoK) is a two party protocol between a Prover P and a Verifier V. The protocol allows P to convince V that it knows some secret without revealing anything about the secret. We refer readers to [7] for formal definition of the original ZKPoK. We discuss a definition of sigma protocol from [8] which can be adapted in the techniques we use in the project.

Notations: We denote a language $\mathscr{L} \subseteq \{0,1\}^*$ that has a witness relationship $R \subseteq \{0,1\}^* \times \{0,1\}^*$ if $x \in \mathscr{L} \iff \exists(x,w) \in R$, where w is a witness for $x \in \mathscr{L}$.

Definition 23. *Let (P,V) be a two-party protocol, where V is PPT, and let $\mathscr{L},\mathscr{L}' \subseteq \{0,1\}^*$ be languages with witness relations $\mathscr{R},\mathscr{R}'$ such that $\mathscr{R} \subseteq \mathscr{R}'$. (P,V) is called a $\sum'$ −protocol for $\mathscr{L},\mathscr{L}'$ with completeness error α, challenge set $\mathscr{C}$, public input x and private input w, if and only if it satisfies the following conditions:*

- *Three-move form: The protocol is of the following form: The prover P, on input (x,w), computes a commitment t and sends it to V. The verifier V, on input x, then draws a challenge $c \overset{r}{\hookleftarrow} \mathscr{C}$ and sends it to P. The prover sends a response s to the verifier. Depending on the protocol transcript (t,c,s), V accepts or rejects the proof.*
- *Completeness: Whenever $(x,w) \in \mathscr{R}$, the verifier V accepts with probability at least $1-\alpha$.*
- *Special Soundness: There exists a PPT algorithm E, also known as the knowledge extractor, that takes two accepted transcript $(t,c',s'),(t,c'',s'')$ satisfying $c' \neq c''$ as inputs and output w' such that $(x,w') \in \mathscr{R}'$.*
- *Special honest-verifier zero-knowledge (HVZK): There exists a simulator S, which is a PPT algorithm that takes $x \in \mathscr{L}$ and $c \in \mathscr{C}$ as inputs and outputs (t,s) so that the triple (t,c,s) is indistinguishable from a valid protocol transcript.*

The definition is different from the original sigma protocol in 2 ways. First, it allows a completeness error α in stead of perfect completeness ($\alpha = 0$ in the original protocol). Second, a second language $\mathscr{L}$ is introduced, with witness relation $\mathscr{R} \subseteq \mathscr{R}'$: A prover knows a witness in $\mathscr{R}$ is guaranteed privacy, but the verifier is only ensured that the P only knows a witness for $\mathscr{R}'$. We refer to this as *soundness gap*, if the gap is small enough, it implies security guarantee for higher level application ($\mathscr{R} = \mathscr{R}'$ in the original protocol).

We present an adaption of the ZKP protocol in [8] to prove the plaintext knowledge of BV cryptosystem. Recall that given $(\mathbf{sk},\mathbf{pk}) = (\mathbf{s},(\mathbf{p_0},\mathbf{p_1}))$ where $\mathbf{s} \in \chi^n_{\beta_0}$, $\mathbf{p_0},\mathbf{p_1} \in R_q$, a message $\mathbf{m} \in R_t$ is encrypted by $[\![\mathbf{m}]\!] := (\mathbf{c_0},\mathbf{c_1}) = (\mathbf{p_0}\mathbf{u}+t\mathbf{f}+\mathbf{m},\mathbf{p_1}\mathbf{u}+t\mathbf{g})$. The protocol (Figure 5.1) is an AND composition of 2 ZKPs that prove 2 relations: the secret key $\mathbf{s}$ corresponds to the public key $(\mathbf{p_0},\mathbf{p_1})$, i.e., $\mathbf{p_0} = -\mathbf{p_1}\mathbf{s} - t\mathbf{e_0}$, and the ciphertext $(\mathbf{c_0},\mathbf{c_1})$ is well-formed and encrypt the message $\mathbf{m}$, that is, $\mathbf{c_0}+\mathbf{c_1}\mathbf{s} = \mathbf{m}+t\mathbf{e}$, with $||\mathbf{e}||_\infty \leq \beta$. Note that the protocol guarantees with V that P knows plaintext encrypted in $2\mathbf{c_0}$

5.3.7 Combination of ZKP

$\sum$ −protocol can be composed together to construct more complex relations. There are several forms of decomposition such as AND, OR, NEQ, etc. We describe some forms of composition

$$
\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\[1em]
\mathbf{r_s}, \mathbf{r_{e_0}} \overset{r}{\hookleftarrow} D_{\beta_0} & & \\
\mathbf{r_e} \overset{r}{\hookleftarrow} D_{\beta} & & \\
\mathbf{r_m} \overset{r}{\hookleftarrow} D_{t} & & \\
\mathbf{t_c} = -\mathbf{c_1}\mathbf{r_s} + t\mathbf{r_e} + \mathbf{r_m} & & \\
\mathbf{t_k} = -\mathbf{p_1}\mathbf{r_s} - t\mathbf{r_{e_0}} & & \\
(c_{aux}, d_{aux}) = aCommit(\mathbf{t_c}, \mathbf{t_k}) & \xrightarrow{c_{aux}} & \\
 & & c \overset{r}{\hookleftarrow} \mathscr{C} = \{0, \ldots, 2n-1\} \\
 & \xleftarrow{c} & \\
\mathbf{s_s} = \mathbf{r_s} + \mathbf{x^c s} & & \\
\mathbf{s_e} = \mathbf{r_e} + \mathbf{x^c e} & & \\
\mathbf{s_m} = \mathbf{r_m} + \mathbf{x^c m} & & \\
\mathbf{s_{e_0}} = \mathbf{r_{e_0}} + \mathbf{x^c e_0} & & \\
 & \xrightarrow{d_{aux}, \mathbf{t_c}, \mathbf{t_k}, \mathbf{s_s}, \mathbf{s_e}, \mathbf{s_m}, \mathbf{s_{e_0}}} & \\
 & & \mathbf{x^c c_0} + \mathbf{t_c} \overset{?}{=} -\mathbf{c_1 s_s} + t\mathbf{s_e} + \mathbf{s_m} \\
 & & \mathbf{x^c p_0} + \mathbf{t_k} \overset{?}{=} -\mathbf{p_1 s_s} - t\mathbf{s_{e_0}} \\
 & & aCOpen(\mathbf{t_c}, \mathbf{t_k}, c_{aux}, d_{aux}) \overset{?}{=} accept \\
 & & ||s_s||, ||s_{e_0}|| \leq 2\beta_0 \\
 & & ||s_e|| \leq 2\beta \\
 & & ||s_m|| \leq 2t
\end{array}
$$

Fig. 5.1 ZKP for BV cryptosystem relation

of $\sum -protocol$ that are used in our protocol: AND-composition, OR-composition. We first describe the generic $\sum -$protocol and its combinations. The BV-ZKP (Figure 5.1) protocol is an instance of such protocol and the technique can be applied inherently.

$\sum -$protocol The protocol is a generic 3-moves between 2 parties *Prover* and *Verifier*, it is used to prove (in zero knowledge) a relation $R(x, w)$, where x is the public parameters known by both parties and w is a witness known by the *Prover* only. The protocol's view is a set of 3 messages (Figure 5.2): the commitment c is a function of a random factor ρ, the challenge ch is normally random and the response r is a function computed from x, w, ρ, ch. There are 2 extra algorithms for $\sum -$protocol: a $verify(r, ch, c)$ algorithm used by the $Verifier$ in the last step to check the correctness of the proof result and a simulator sim that takes x and a random parameter ρ' as input and outputs a view $\bar{c}, \bar{ch}, \bar{r}$ that is indistinguishable from the genuine view of the protocol.

$$
\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
\rho \overset{r}{\hookleftarrow} random & & \\
c = comm(\rho) & & \\
& \overset{c}{\rightarrow} & \\
& & ch \overset{r}{\hookleftarrow} random \\
& \overset{ch}{\leftarrow} & \\
r = resp(\rho, w, x, ch) & & \\
& \overset{r}{\rightarrow} & \\
& & verify(r, ch, c)
\end{array}
$$

Fig. 5.2 Sigma Protocol

AND-COMPOSITION Given two relations $R_1 = \{(v_1, w_1)\}$ and $R_2 = \{(v_2, w_2)\}$ with the same challenge space, a $\sum$-Protocol of $R_1 \wedge R_2 = \{(v_1, v_2, w_1, w_2) : (v_1; w_1) \in R_1, (v_2, w_2) \in R_2\}$ can be obtained by running a $\sum$-Protocol for R_1 and a $\sum$-Protocol for R_2 in parallel, using a *common* challenge. (Figure 5.3 is a concrete example combining 2 ZKP for BV)

OR-Compostion In this composition, given the $\sum -protocol$ for many relations relations R_1,R_2,...,R_n the goal is to construct a protocol for the relation $R_{OR}(\vec{x}, w_i)$, where $\exists i \leq n : R(x_i, w_i) = 1$. The main idea to construct this composition is that the verifier can let the prover using the simulator of the $\sum -protocol$ for the relations R_j that the prover does not know the witness. The verifier provides a single challenge c such that the prover can split that into challenges $c_1, c_2, \ldots, c_n$ provided that c_i satisfy a liner constraint in terms of c, for example $c = c_1 \oplus c_2 \oplus \ldots \oplus c_n$. The final OR-combination proof is obtained by composing one run of the $\sum -protocol$ with other runs of the simulators for the $\sum -protocol$. (Figure 5.4)

5.4 Proposed Scheme

We propose a secure fingerprint authentication scheme that combines Somewhat Homomorphic Encryption (SHE) with Zero Knowledge Proof (ZKP) to provide privacy features with low FAR overhead. The scheme is secure under a hybrid model that assumes active client, honest but curious (HBC) server. The server $\mathscr{S}$ is considered to be HBC with the assumption

$$
\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
\mathbf{r_s}, \mathbf{r_{e_0}}, \mathbf{r'_s}, \mathbf{r'_{e_0}} \overset{r}{\hookleftarrow} D_{\beta_0} & & \\
\mathbf{r_e}, \mathbf{r'_e} \overset{r}{\hookleftarrow} D_{\beta} & & \\
\mathbf{r_m}, \mathbf{r'_m} \overset{r}{\hookleftarrow} D_{t} & & \\
\mathbf{t_c} = -\mathbf{c_1}\mathbf{r_s} + t\mathbf{r_e} + \mathbf{r_m} & & \\
\mathbf{t_k} = -\mathbf{p_1}\mathbf{r_s} - t\mathbf{r_{e_0}} & & \\
\mathbf{t'_c} = -\mathbf{c_1}\mathbf{r'_s} + t\mathbf{r'_e} + \mathbf{r'_m} & & \\
\mathbf{t'_k} = -\mathbf{p_1}\mathbf{r'_s} - t\mathbf{r'_{e_0}} & & \\
(c_{aux}, d_{aux}) = aCommit(\mathbf{t_c}, \mathbf{t_k}) & & \\
(c'_{aux}, d'_{aux}) = aCommit(\mathbf{t'_c}, \mathbf{t'_k}) & \xrightarrow{c_{aux}, c'_{aux}} & \\
& & c \overset{r}{\hookleftarrow} \mathscr{C} = \{0, \ldots, 2n-1\} \\
& \xleftarrow{c} & \\
\mathbf{s_s} = \mathbf{r_s} + \mathbf{x^c s}, \mathbf{s'_s} = \mathbf{r'_s} + \mathbf{x^c s} & & \\
\mathbf{s_e} = \mathbf{r_e} + \mathbf{x^c e}, \mathbf{s'_e} = \mathbf{r'_e} + \mathbf{x^c e} & & \\
\mathbf{s_m} = \mathbf{r_m} + \mathbf{x^c m}, \mathbf{s'_m} = \mathbf{r'_m} + \mathbf{x^c m} & & \\
\mathbf{s_{e_0}} = \mathbf{r_{e_0}} + \mathbf{x^c e_0}, \mathbf{s'_{e_0}} = \mathbf{r'_{e_0}} + \mathbf{x^c e_0} & & \\
& \xrightarrow[d'_{aux}, \mathbf{t'_c}, \mathbf{t'_k}, \mathbf{s'_s}, \mathbf{s'_e}, \mathbf{s'_m}, \mathbf{s'_{e_0}}]{d_{aux}, \mathbf{t_c}, \mathbf{t_k}, \mathbf{s_s}, \mathbf{s_e}, \mathbf{s_m}, \mathbf{s_{e_0}}} & \\
& & \mathbf{x^c c_0} + \mathbf{t_c} \overset{?}{=} -\mathbf{c_1}\mathbf{s_s} + t\mathbf{s_e} + \mathbf{s_m} \\
& & \mathbf{x^c c_0} + \mathbf{t'_c} \overset{?}{=} -\mathbf{c_1}\mathbf{s'_s} + t\mathbf{s'_e} + \mathbf{s'_m} \\
& & \mathbf{x^c p_0} + \mathbf{t_k} \overset{?}{=} -\mathbf{p_1}\mathbf{s_s} - t\mathbf{s_{e_0}} \\
& & \mathbf{x^c p_0} + \mathbf{t'_k} \overset{?}{=} -\mathbf{p_1}\mathbf{s'_s} - t\mathbf{s'_{e_0}} \\
& & aCOpen(\mathbf{t_c}, \mathbf{t_k}, c_{aux}, d_{aux}) \overset{?}{=} accept \\
& & aCOpen(\mathbf{t'_c}, \mathbf{t'_k}, c'_{aux}, d'_{aux}) \overset{?}{=} accept \\
& & ||s_s||, ||s_{e_0}||, ||s'_s||, ||s'_{e_0}|| \leq 2\beta_0 \\
& & ||s_e||, ||s'_e|| \leq 2\beta \\
& & ||s_m||, ||s'_m|| \leq 2t
\end{array}
$$

Fig. 5.3 AND-Composition for ZKP-BV

that it can be audited regularly. This scheme uses Hamming Distance (HD) as the main measure unit to determine the difference between two fingerprint template.

5.4.1 The protocol

Setup The server and the user runs the setup process as follows.

$$\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
\rho_i \overset{r}{\hookleftarrow} random & & \\
c_i = comm(\rho_i) & & \\
\text{For } j = 1,\ldots,n \wedge j \neq i: & & \\
\rho_j \overset{r}{\hookleftarrow} random; & & \\
(c_j, ch_j, r_j) \leftarrow ZKPSim(\rho_j, x_j) & & \\
& \xrightarrow[ch_j]{c_1,c_2,\ldots,c_n} & \\
& & ch \overset{r}{\hookleftarrow} \mathbb{Z}_n \\
& \xleftarrow{ch} & \\
ch_i = ch - \sum_{j\neq i} ch_j & & \\
r_i = resp(\rho_i, w_i, x_i, ch_i) & & \\
& \xrightarrow{r_1,r_2,\ldots,r_n} & \\
& & \forall i \in [n], ver(r_i, ch_i, c_i) = 1 \\
& & ch = \sum_{i\in[n]} ch_i
\end{array}$$

Fig. 5.4

- $\mathscr{S}$ invokes $params_{BV} \leftarrow SGen_{BV}(1^{\lambda})$
- We are currently using BV cryptosystem ([11]). We refer the reader to section 2.4.1 for details of $params_{BV}$, the public key pk, the private key sk, as well as the operations of the cryptosystem.
- A user $\mathscr{U}_k$ invokes $(pk_k, sk_k) \leftarrow UGen_{BGV}(params_{BGV})$ and make pk_k publicly available to $\mathscr{S}$

Enrolment The enrolment process runs as follows.

- $\mathscr{U}$ uses a specific sensor and algorithm to extract his biometrics template X. $\mathscr{U}$ encrypts X with his public key pk_k to get $T_k = Enc^{(1)}(X)$.
- $\mathscr{U}$ sends (k, T_k) to $\mathscr{S}$. Noted that k is used as an identity index for $\mathscr{U}$.
- $\mathscr{S}$ stores a tuple (k, T_k) as a record.

Authentication The authentication process for a user $\mathscr{U}_k$ is as follows.

1. $\mathscr{U}_k$ extracts his query template Y. He encrypts Y with his public key pk_k to get $Q_k = Enc^{(1)}(Y)$.

2. $\mathscr{U}_k$ sends (k, Q_k) to $\mathscr{S}$ and run $\mathbf{ZKPValidEnc}((\mathbf{Q_k}, \mathbf{pk_k}), (\mathbf{Y}, \mathbf{sk_k}))$ to prove the validity of Y.
3. $\mathscr{S}$ locates the record (k, T_k) and computes the encrypted Hamming Distance $C_{HD} = Enc^{(1)}(HD)$ of X and Y, using homomorphic operation discussed in Appendix 3.3.1
$$C_{HD} \leftarrow \mathbf{EvalDistance}(T_k, Q_k).$$
4. $\mathscr{S}$ masks the C_{HD} by sampling $r \overset{r}{\hookleftarrow} \mathscr{P}$ and does one homomorphic addition to get $C_{HD'} = Enc^{(1)}(HD + r) \leftarrow Enc^{(1)}(HD) + Enc^{(1)}(r)$. The result ciphertext is sent to $\mathscr{U}_k$.
5. $\mathscr{U}_k$ uses his private key sk_k to decrypts $C_{HD'}$ and sends back to $\mathscr{S}$ the re-encryption $C_{HD'_0} = Enc^{(1)}(HD', 0, 0, \dots, 0)$. $\mathscr{U}_k$ also decomposes the plaintext result $HD' = HD + r$ into its binary representation:
$$HD' = b_0 + b_1 2^1 + \cdots + b_l 2^{l-1}$$
and sends $C_i = Enc^{(1)}(b_i)$ to $\mathscr{S}$ for $i = 0, \dots, l-1$.
6. $\mathscr{U}_k$ and $\mathscr{S}$ run $\mathbf{ZKPUnpack}(C_{HD}, C_{HD'_0})$ and $\mathbf{ZKPBinDecomp}(C_{HD'_0}, C_i)$ protocols to convince the server that $\mathscr{U}_k$ did follow the protocol transcript correctly. This is detailed in section 5.3.4.
7. $\mathscr{S}$ computes $C''_{HD} = Enc^{(2)}(2^l + t - HD) \leftarrow Enc^{(2)}(2^l + t) - Enc^{(2)}(HD + r) + Enc^{(2)}(r)$. Where $Enc^{(2)}(HD + r)$ is computed by $\mathbf{ToUnary}(C_i)$. We note that $Enc^{(2)}$ is the second mode of encryption with the message encoded in the exponent of the polynomial, which allows the Most significat bit (MSB) extraction on ciphertext, this is detailed section 5.2.3
8. $\mathscr{S}$ does $Enc^{(3)}(res) \leftarrow \mathbf{MSBExtract}(C''_{HD})$, this is the ciphertext of the authentication result, which is sent to $\mathscr{U}_k$. This is described in section 5.2.1.
9. $\mathscr{U}_k$ decrypts the result res and sends to $\mathscr{S}$ (which is either **Accepted** or **Rejected**). $\mathscr{U}_k$ also does another proof $\mathbf{ZKPCorrectDec}(res)$ to convince $\mathscr{S}$ that he did follow the protocol honestly (Section 5.3.4).

The proposed scheme satisfies the security notions defined in Sect. 2.7, proofs are provided in Appendix 5.5

Theorem 15. *Under the IND-CPA security of BV cryptosystem, and the zero-knowledge property of the Stern protocol, the proposed scheme satisfies (Honest But Curious) Server Privacy Security.*

Theorem 16. *Under the IND-CPA security of BV cryptosystem and the soundness property of the underlying Stern protocol, the proposed scheme satisfies Impersonation Security. Concretely, for $\delta > 0$, the protocol is (q,c)-secure against impersonation with $c \leq c(\delta) + 3 \cdot c_1$, assuming the underlying non-private biometric protocol has imperonsation probability ε_{bio} and the underlying Stern ZK protocols have knowledge error $\varepsilon_{ZK1}, \dots, \varepsilon_{ZK4}$ such that $q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta \leq c_1 \cdot \varepsilon_{bio}$, $c(\delta) = 2e^{1+2\delta}$, and the condition $\sigma / r_0 \geq 4\pi knq$ holds, with $k = 1 + \sqrt{1/\pi \ln(2nq/\delta)}$ and r_0 an upper bound on the size of the noise in C_{HD}.*

5.5 Security Proofs

5.5.1 Security Proof for Theorem 16: Type I Impersionation attack

The proof of theorem 15 and 16 can be done using a sequence of games between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. We present a sequence of games as well as the relations among them to demonstrate type I security model proof.

Game 0. Game 0 is the original impersonation game for type I attack.

Setup. $\mathscr{C}$ intiates $D_k \overset{r}{\hookleftarrow} D_{bio}$ and $X_k \overset{r}{\hookleftarrow} D_k$. $\mathscr{C}$ sets up (sk_k, pk_k) and does $Enrol(k, X_k)$ to get $(sk_k, T_k = (pk_k, C_k = Enc^{(1)}_{pk_k}(X_k)))$. $\mathscr{A}$ submits the attack query type I and receives sk_k from $\mathscr{C}$.

Query. $\mathscr{A}$ runs q authentication sessions. In each session $j = 1, \dots, q$, $\mathscr{A}$ sends $(Q^{(j)}_k = Enc^{(1)}(Y^{(j)}))$. $\mathscr{A}$ and $\mathscr{C}$ runs **ZKPValidEnc**$(Q^{(j)}_k, Y^{(j)})$. $\mathscr{C}$ evaluates $C_{HD} = \textbf{EvalDistance}_{pk_k}(C_k, Q^{(j)}_k)$ then computes $C_{HD'} = (-1)C_{HD} + Enc^{(1)}_{pk_k}(r_{HD'}, e_{HD'})$ for $r_{HD'} \overset{r}{\hookleftarrow} \mathscr{P}$ and $e_{HD'} \overset{r}{\hookleftarrow} \chi_{HD}$. The result ciphertext is sent to $\mathscr{A}$, $\mathscr{A}$ decrypts $C_{HD'}$ and decomposes it to bits and sends back the ciphertexts $C^{(j)}_0, \dots, C^{(j)}_{l-1} (= Enc^{(1)}(b_i), i = 0, \dots, l-1)$ as well as the re-encryption $C_{HD'_0}$. $\mathscr{A}$ and $\mathscr{C}$ does **ZKPUnpack**$(C_{HD}, C_{HD'_0})$ and **ZKPBinDecomp**$(C_{HD'_0}, C^{(j)}_i)$. $\mathscr{C}$ evaluates $C''_{HD} = Enc^{(2)}(2^l + t) + \textbf{ToUnary}(C^{(j)}_i) + Enc^{(2)}(-r)$ to get $Enc^{(2)}(2^l + t - HD)$. $\mathscr{C}$ does $Enc^{(3)}(res) \leftarrow \textbf{MSBExtract}(C''_{HD})$ and sends to $\mathscr{A}$. $\mathscr{A}$ decrypts and sends the authentication result bit back. $\mathscr{C}$ and $\mathscr{A}$ does **ZKPCorrectDec**(res) to convince the authentication result. At the end, the server outputs **Accept** if all the proofs pass and $res =$ **Accept**.

Next, we discuss following games, the plan is to proceed towards the final game where we can simulate everything that $\mathscr{A}$ receives that related to X_k (without any knowledge about X_k, the only thing that known about X_k is the function $Verify(X_k,Y)$). Let $res_s = Verify(X_k,Y^{(j)})$ and S_i be the event in the game i such that $res_s = Accept$.

Game 1. In this game, we abort **ZKPValidEnc** if $Q_k^{(j)}$ is not a valid encryption of the query, but the *Prover* manages to pass the proof. Let $(*)_1$ be this event.

$$(*_1) res_s = \begin{cases} \text{Reject if } \textbf{ZKPValidEnc} \text{ fails} \\ \text{res else} \end{cases}$$

Let bad_0 be the event in game $0 : \exists j \leq q\ s.t\ (*_1)$ happens. We want to show that when we modify *game 0*, the probability of a successful forgery S_1 in this *game 1* is not much lower than what it was. Due to the modification, we observe that $Pr[S_1] \geq Pr[S_0] - Pr[bad_0]$ .For any j in the q authentication attemps, by the $\varepsilon - soundness$ property of **ZKPValidEnc** as a proof of membership in (*), we have $Pr[(*_1)\ occurs\ for\ some\ j] \leq \varepsilon_{ZK1}$. So the probability of bad_0 would be the union of these events which is bounded by $Pr[bad_0] \leq q\varepsilon_{ZK1}$. In other words, the advantage of $\mathscr{A}$ in *game 1* is

$$Pr[S_1] \geq Pr[S_0] - Pr[bad_0] \geq \varepsilon_{imp} - q\varepsilon_{ZK1}$$

Game 2. In this game, we abort **ZKPUnpack** if in one of the j^{th} runs, the $Verifier$ accepts but the ciphertext did not satisfy the relation. Let bad_1 be this event, by the same type of argument, we can derive $Pr[bad_1] \leq q\varepsilon_{ZK2}$ and therefore

$$Pr[S_2] \geq Pr[S_1] - Pr[bad_1] \geq \varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2})$$

Game 3 and Game 4. Similarly, we abort **ZKPBinDecomp** and **ZKPCorrectDec** if in any j^{th} runs, the $Verifier$ accepts even when the correctness of the ZKP is broken. We have

$$Pr[S_3] \geq \varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3})$$

and

$$Pr[S_4] \geq \varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4})$$

By the end of *Game 4*, if in any authentication attempt j, there is no abort in any of the 4 games, then by the correctness property of ZKP, the server output is equal to the output of $Verify(X_k,Y^{(j)})$. In other words, we have shown what we could get by just querying the

oracle $Verify()$. Next, we want to simulate everything related to X_k that the attacker can see using just that oracle.
Game 5. At the end of *Game 4*, $\mathscr{C}$ has the bit $b = Verify(Y^{(i)}, X_k)$. In this game, we can change $C_{res}^{(j)}$ from **MSBExtract**$(C''_{HD}) = Enc(b; e_{res})$ to $Enc(verify(Y^{(i)}, X_k); e_{res})$, given that the challenger can use the secret key to extract e_{res}. By doing this change, $\mathscr{A}$ still sees the same ciphertext, so the probability of winning of $\mathscr{A}$ in this game is the same as in *Game 4*.
Game 6. In this game, we change the way $\mathscr{C}$ computes C''_{HD}: In the orignal game 0, $Enc(-r)$ was added to remove the mask, we want to remove this r from being used anywhere in the game, so we replace this with $Enc(0)$. This change does not affect $\mathscr{A}$'s success probability: r only affects the plaintext inside C''_{HD}, since we do not use this plaintext anymore (it was replaced in *Game 5*), so this change does not affect what the attacker sees. Again, $Pr[S_6] = Pr[S_5] = Pr[S_4]$.
Game 7. We modify the way C'_{HD} is computed in this game. Instead of doing $C'_{HD} \leftarrow (-1)C_{HD} + Enc(r; e_{HD'})$, the challenger chooses a random $HD' \overset{r}{\hookleftarrow} \mathbb{Z}_p$ and encrypt it with the noise was used before: $C'_{HD} \leftarrow Enc(HD'; -e_{HD} + e_{HD'})$. In this game, the plaintext has changed from being $r + HD$ to a uniform $HD' \in \mathbb{Z}_p$. Since r is also uniform in $\mathbb{Z}_p$, the attacker sees a uniform plaintext in both cases. Therefore, $Pr[S_7] = Pr[S_6]$.
Game 8. Finally, we set $C_{HD'} = Enc(HD', e_{HD'})$ for $e_{HD'} \overset{r}{\hookleftarrow} \chi_{mask}$ instead of $-e_{HD} + e_{HD'}$. We replace the sum of the Gaussian noise with the random noise. In Section **??**, we showed that $Pr[S_8] \geq \frac{1}{c(\delta)}(Pr[S_7] - q \cdot \delta)$, where $c(\delta) = RD(-e_{HD} + e_{HD'}, e_{HD}) \leq 2 \cdot e^{1+2\delta}$ by Lemma 8 and the assumption on the parameters. After we finish *Game 8*, we can see that all the messages that the attacker sees, can be simulated with only the verified bit $b = Verify(Y^{(i)}, X_k)$. We now have an attacker A' against the biometric impersonation with advantage:

$$\varepsilon_{bio} = Adv(A') = Pr[S_8] \geq \frac{1}{c(\delta)}(\varepsilon_{imp} - q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta),$$

which gives the claimed bound $\varepsilon_{imp} \leq c(\delta) \cdot \varepsilon_{bio} + q \cdot (\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta \leq (c(\delta) + c_1) \cdot \varepsilon_{bio}$ if $q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4}) + \delta \leq c_1 \cdot \varepsilon_{bio}$.

5.5.2 Security Proof for Theorem 16: Type II Impersionation attack

The proof of Theorem 16 can be done using a sequence of games between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. We present a sequence of games as well as the relations among them to demonstrate type II security model proof. The idea is that in this type of attack, sk_k is not used to compute the view of $\mathscr{A}$. On the other hand, the soundess of the zero-knowledge proof of knowledge **ZKPValidEnc** implies the existence of an efficient witness extractor algorithm, that can be used to extract the witness (i.e. the secret key sk_k) from a cheating prover that succeeds with probability non-negligibly higher than the knowledge error of the zero-knowledge proof, thus contradicting the IND-CPA security of the BV encryption scheme.

Game 0. Game 0 is the original impersonation game for type II attack, i.e. the same as Game 0 in the proof of security against Type I attacks, except that $(T_k = (pk_k, C_k = Enc^{(1)}_{pk_k}(X_k)))$ is given by $\mathscr{C}$ to $\mathscr{A}$ at the beginning of the game, rather than sk_k. For $j \in \{1,\ldots,q\}$, let res^j denote the result of the jth authentication protocol run between $\mathscr{A}$ and $\mathscr{C}$, and S_0 be the event in the game 0 such that $res^j = Accept$ for some $j = 1,\ldots,q$. We have $\Pr[S_0] = \varepsilon_{imp,II}$, the type II success probability of $\mathscr{A}$.

Game 1. From the definition of event S_0 in Game 0, it follows that there exists some $j^* \in \{1,\ldots,q\}$ such that $\Pr[res^{j^*} = Accept] \geq \varepsilon_{imp,II}/q$. Furthermore, by an averaging argument, there must exist a set G of (D_k, X_k, pk_k) such that $\Pr[(D_k, X_k, pk_k) \in G] \geq \varepsilon_{imp,II}/(2\cdot q)$, and for each $(D'_k, X'_k, pk'_k) \in G$, we have $\Pr[res^{j^*} = Accept | (D_k, X_k, pk_k) = (D'_k, X'_k, pk'_k)] \geq \varepsilon_{imp,II}/(2\cdot q)$. By ε_{ZK1}-soundness of the zero knowledge proof of knowledge **ZKPValidEnc** (cite Golderich's 'Foundations of Cryptography' book, volume 1, Prop. 4.7.5), there exists an witness extractor algorithm that runs in expected time $T' = O(\text{poly}(n \log Q) \cdot T/(\varepsilon_{imp,II}/(2 * q) - \varepsilon_{ZK1}))$, where T denotes the run-time of $\mathscr{A}$ and outputs a witness containing sk_k for *ZKPValidEnc*. Therefore, we obtain a (secret key recovery) attack algorithm against the IND-CPA security of the BV encryption scheme with expected run-time T' and advantage $\varepsilon' \geq \varepsilon_{imp,II}/(2\cdot q)$. Hence, if $\varepsilon_{imp,II}/(2 * q) - \varepsilon_{ZK1} > \varepsilon_{imp,II}/(4 * q)$, or equivalently, if $\varepsilon_{imp,II} > 2 \cdot q\varepsilon_{ZK1}$, then we obtain a contradiction with the assumption the BV encryption scheme with parameters Q, n, σ is IND-CPA against attacks with expected time $O(\text{poly}(n \log Q) \cdot T/\varepsilon_{ZK1})$ and advantage $\geq \varepsilon_{ZK1}$. It follows under the latter assumption that $\varepsilon_{imp,II} \leq 2q\varepsilon_{ZK1} \leq 2c_1 \cdot \varepsilon_{bio}$, under the assumption that $q(\varepsilon_{ZK1} + \varepsilon_{ZK2} + \varepsilon_{ZK3} + \varepsilon_{ZK4} + \delta) \leq c_1 \cdot \varepsilon_{subsection}$.

5.5.3 Security Proof for Theorem 15: Privacy against Server

The proof of Theorem 16 can be done using a sequence of games between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. We present a sequence of games. The idea is to proceed to remove sk_k and X_k from being used to compute the view of $\mathscr{A}$, except for $Verify(X_k, Y_k^j)$ queries, as in the ideal game, relying on the correctness of the protocol and the IND-CPA security of the BV encryption scheme.

Game 0. Game 0 is the original real privacy game, in which $(T_k = (pk_k, C_k = Enc^{(1)}_{pk_k}(X_k)))$ is given by $\mathscr{C}$ to $\mathscr{A}$ at the beginning of the game. Then, for $j = 1\ldots, q$, the attacker sends $Y_k^{(j)} \in \{0,1\}^n$ to $\mathscr{C}$, and the latter simulates a run of the authentication protocol between an honest client with input $(k, Y_k^{(j)}, sk_k)$ and an honest server with input (k, T_k), returning to $\mathscr{A}$ the protocol view $V_S^{(j)}$ of the server. Finally, $\mathscr{A}$ outputs a bit β. In the following Game i, we let S_i denote the event that $\beta = 1$.

Game 1. We change the computation of the authentication result bit $res^{(j)}$ sent by client to server from the decryption of the ciphertext **MSBExtract**$(C''_{HD}$ (its value in Game 0) to the result returned by $Verify(X_k, Y^{(j)})$. By correctness of the protocol, this does not change the value of $res^{(j)}$, so $\Pr[S_1] = \Pr[S_0]$.

Game 2. We change the computation of the zero-knowledge protocol transcripts. Instead of computing those transcripts using the secret witnesses, we simulate them using the statistical zero-knowledge simulator algorithms for the zero-knowledge proofs. By the zero-knowledge property, this is a perfect simulation, and we have $\Pr[S_2] = \Pr[S_1]$.

Game 3. We change the computation of the ciphertexts $C_i^{(j)}$ for $i = 0, \ldots, l-1$ and $Q_k^{(j)}$ for $j = 1, \ldots, q$ and C_k to encrypt zero messages, instead of encrypting the secret-related messages in the previous game. Since now sk_k is not used anywhere in generating the view of $\mathscr{A}$, it follows by a hybrid argument that $|\Pr[S_3] - \Pr[S_2]|((l+1) \cdot q + 1) \cdot \varepsilon_{BV}$, where ε_{BV} denotes the maximal advantage of an attacker against IND-CPA of BV scheme against attacks with run-time $T + \mathrm{poly}(n, \log Q)$ where T is the run-time of $\mathscr{A}$. In this game, since the only information on X_k comes via the $Verify(X_k, Y_k^j)$ queries, the challenger together with $\mathscr{A}$ constitute an efficient attacker against the ideal privacy game, which outputs 1 with probability different by at most $(l+1) \cdot q + 1) \cdot \varepsilon_{BV}$ than the probability of outputting 1 in the real privacy game, as required.

5.6 The algorithms

5.7 Evaluation and Results

Algorithm 11 HD Computation Homomorphically

```
1: procedure EvalDistance($\mathbf{T}, \mathbf{Q}$)
2:     $C_1 \leftarrow -\sum_{i=0}^{n-1} x^{n-i}$
3:     $C_2 \leftarrow 2 - C_1$
4:     $C_{HD} \leftarrow ct_1(\mathbf{T}) * Enc(C_1) + ct_2(\mathbf{Q}) * Enc(C_2) - 2 * ct_1(\mathbf{T}) * ct_2(\mathbf{Q})$
5:     return $C_{HD}$
```

Algorithm 12 ZKP of encoding transform

```
1: procedure ZKPBinDecomp($\mathbf{c}, \mathbf{c_i}$)
2:     Let $\mathbf{c}' \leftarrow \sum_{i=0}^{l-1} \mathbf{c_i} 2^i$
3:     Let $\mathbf{c}'' \leftarrow c - c'$
4:     Return ZKPExt1$((\mathbf{c}'', \mathbf{pk}), (\mathbf{Dec}(\mathbf{c}''), \mathbf{s}, \mathbf{e}, \mathbf{e}'))$
```

Algorithm 13 Most Significant bit extraction

```
1: procedure MSBExtract($\mathbf{c}$)
2:     $allOne = \{1, \dots, 1\}$
3:     $rot_{c_0} \leftarrow rot(c_0)$
4:     $lwe_0 \leftarrow allOne \times rot_{c_0}[0]$
5:     $rot_{c_1} \leftarrow rot(c_1)$
6:     $lwe_1 \leftarrow allOne \times rot_{c_1}$
7:     return $(lwe_0, lwe_1)$
```

Algorithm 14 Binary to Unary ciphertext

```
1: procedure ToUnary($\mathbf{c_i}$)
2:     for $j = 0, \dots, l-1$ do
3:         let $\mathbf{hd_j} \leftarrow (x^j - 1, 0) \times \mathbf{c_j} + (1, 0)$
4:     let $\mathbf{hd} \leftarrow \mathbf{hd_0} \times \mathbf{hd_1} \times \cdots \times \mathbf{hd_{l-1}}$
5:     return hd
```

Algorithm 15 ZKPoPK for BV

```
1: procedure ZKPBV($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e}'))$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let Z be the $n \times n$ zero matrix
6:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I), (rot_{p_1}, Z, tI, Z))$
7:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m})$
8:     $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
9:     Return SternExt$(\mathbf{A}, \mathbf{x}, \mathbf{y})$
```

Algorithm 16 ZKP for zero constant coefficient

```
1: procedure ZKPExt1($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e}')$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let I' be 1 column matrix with all 1s.
6:     let Z be the $n \times n$ zero matrix
7:     let Z' be 1 column matrix with all 0s.
8:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I'), (rot_{p_1}, Z, tI, Z'))$
9:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m})$
10:    $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
11:    Return SternExt$(\mathbf{A}, \mathbf{x}, \mathbf{y})$
```

Algorithm 17 ZKP for only constant coefficient

```
1: procedure ZKPExt2($(\mathbf{c}, pk), (\mathbf{m}, \mathbf{s}, \mathbf{e}, \mathbf{e}')$)
2:     $rot_{c_1} \leftarrow rot(\mathbf{c_1})$
3:     $rot_{p_1} \leftarrow rot(\mathbf{pk_1})$
4:     let I be the $n \times n$ identity matrix
5:     let I' be I with the first column removed.
6:     let Z be the $n \times n$ zero matrix
7:     let Z' be Z with the first column removed.
8:     $\mathbf{A} \leftarrow ((rot_{c_1}, -tI, Z, -I'), (rot_{p_1}, Z, tI, Z'))$
9:     $\mathbf{x} \leftarrow (\mathbf{s}, \mathbf{e}', \mathbf{e}, \mathbf{m})$
10:    $\mathbf{y} \leftarrow (-\mathbf{c_0}, -\mathbf{pk_0})$
11:    Return SternExt$(\mathbf{A}, \mathbf{x}, \mathbf{y})$
```

Algorithm 18 ZKP of coefficients transform

```
1: procedure ZKPUnpack($\mathbf{c}, \mathbf{c}'$)
2:     Let $b_1 \leftarrow$ ZKPExt1$((\mathbf{c} - \mathbf{c}', \mathbf{pk}), (\mathbf{Dec}(\mathbf{c} - \mathbf{c}'), \mathbf{s}, \mathbf{e}, \mathbf{e}'))$
3:     Let $b_2 \leftarrow$ ZKPExt2$((\mathbf{c}', \mathbf{pk}), (\mathbf{Dec}(\mathbf{c}'), \mathbf{s}, \mathbf{e}, \mathbf{e}'))$
4:     Return $b_1 \wedge b_2$
```

Chapter 6

The fourth protocol - Removing The Overhead

6.1 Introduction

In the last chapter, we developed the two-parties authentication protocol that is secure against malicious client and does not leak any information to the server (the main difference with the second protocol is the server learns the value of the distance only). The final variant aims to reduce both communication size and computation time of the third variant while providing all the security features.

Reducing Communication Size We observe that the main bottle neck of the communication is the size of the Stern-based ZKPs. We propose the following approach to replace the old ZKPs

- A Schnorr-based approach to replace the Stern-based one, to prove the noise in the ciphertext is small. This variant reduces the total communication rounds from $log_{2/3}(FAR)$ to only 1.
- A challenge-response protocol to prove the binary format of the plaintext. The protocol can be summarized as follows
  1. The server samples $r_1, r_2 \overset{r}{\hookleftarrow} R_q$ and computes $ch = enc(r_1 * Q * (1 - Q) + r_2) + enc(0, r)$ and sends ch to the client.
  2. The client decrypts the result and sends back rsp.
  3. The server accepts the response if and only if $rsp = r_2$

We remark that the first operation needs to be computed bit-wise, that means we need to change the ciphertext packing method, the CRT-packing [80] can support this requirement.

Reducing Computation Time. With the application of new ciphertext packing method, we can effectively change the way of computing Hamming Distance by summing the bits of the XOR result of 2 bitstring instead of computing the inner product. Therefore, with this approach we save the computation by doing addition only without any homomorphic multiplication. The cost of this change is a set of switching keys for homomorphic rotation operations, which are generated during enrollment process. Additionally, we introduce the appplication of state-of-the-art *Secure-Multiparty-Protocol* techniques such as *Garbled Circuit* and *Oblivious Transfer* in the lattice-based context to further improve both of the computation time and the communication size overhead from previous ZKP protocols.

This approach is practical because it moves a major part of the overhead to the one-time set up phase of the protocol (the enrollment stage). Our result shows that the protocol can work efficiently during authentication stage. The technique maybe generalized to similar multi-stages protocols.

6.2 The Challenge-Response Technique

Consider this situation. Alice sends a ciphertext C of a bitstring to Bob. Bob is supposed to use homomorphic encryption technique to process C. Before doing so, Bob might want to make sure that C is an encryption of a bitstring and not something else. How can he do that? Suppose that the plaintext P is encoded as coefficients of a polynomial, or specifically, P can be an element of the ring R_2, where $R_2 = \frac{\mathbb{Z}_2[x]}{x^n+1}$, this ring is used by many RLWE based cryptosystems lately. One solution can be applying Zero Knowlege Proof technique. This is however a costly solution. There are mainly two approaches for ZKP, the Schnorr's based protocol, e.g. [8] or Stern's based protocol such as [82] or [53]. The first approach has limitation on the norm of the secret it can prove and cannot be used to prove the knowledge of binary message with norm 2. The second approach can prove binary but it has very high communication cost due to roundness error of $2/3$.

6.3 The BGV Cryptosystem and CRT packing method

Ciphertext packing We have seen that a plaintext can be packed in a specific way to support the homomorphic operations efficiently. The approach of [87] can help the inner product operations perform really fast, however, it does not support Single Instruction Multiple Data (SIMD) operation, which is required by our challenge-response protocol. This section discuss another approach for plaintext packing and the operations that it supports: the CRT packing method [80]. In this method, a message $\mathbf{m} \in R_t$ is packed by represent it in the NTT domain. In NTT domain, a single operation (addition or multiplication) of a pair of ciphertexts implicitly computed component-wise the entire plaintext vectors. Beside SIMD support, we can also *rotate* or *permute* the plaintext slots ([27]), we will need only *rotate* operation in this project.

Number Theoretic Transform In our application context, the most costly low level computation operation is the ring multiplication (polynomial multiplication). This section discusses the Number Theoretic Transform (NTT), the technique providing efficient algorithms for cyclic and nega-cyclic convolutions that can be applied to compute polynomial multiplication efficiently. Moreover, in our research context, when the message is represented in the NTT domain, the operations are computed component wise simultaneously, this important property help our challenge-response protcol to save the most communication size compared with ZKPs overhead of the previous protocols.

Notations Given $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $b(x) = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$, we denote $\cdot$ to be the convolution multiplication and $\triangle$ to be the component wise multiplication of the two polynomials.

Introduction One of the most important applications of Fast Fourier Transform (FFT) is multiplication of large integers and polynomial multiplication (which is the cyclic convolution of two integer sequences). The later operation can be computed by applying the FFT to the sequences, then multiplying the result component wise and applying the inverse FFT to the product:

$$a(x) \cdot b(x) = FFT^{-1}(FFT(a(x)) \triangle FFT(b(x)))$$

When the coefficients are elements of a finite field, the FFT is called the Number Theoretic Transform (NTT). We recall the definition of NTT: Given the parameters of our context, with n is a power of 2 and t is a prime with $t = 1 \mod 2n$, let $\mathbf{a} = [a_0, a_1, \ldots, a_{n-1}] \in R_t$ and let ω be a primitive n^{th} root of unity in $\mathbb{Z}_t$, that is, $\omega^n = 1$

$\bmod t$. The forward transform $NTT(\mathbf{a}) = \tilde{\mathbf{a}}$ is defined as $\tilde{\mathbf{a}}_i = \sum_{j=0}^{n-1} \mathbf{a}_j \omega^{ij} \mod t$ for $i = 0, \ldots, n-1$. Informally, the coefficients of $\tilde{\mathbf{a}}$ are the evaluations of the polynomial $a(x)$ at the n roots of unity. The inverse transformation is given by $\mathbf{b} = NTT^{-1}(\tilde{\mathbf{a}})$, where $\mathbf{b}_i = n^{-1} \sum_{j=0}^{n-1} \tilde{\mathbf{a}}_j \omega^{-ij} \mod t$ for $i = 0, \ldots, n-1$. We have $NTT^{-1}(NTT(\mathbf{a})) = \mathbf{a}$. The above convolution operation computes a polynomial $c(x) = a(x) \cdot b(x)$ with order at most $2n-1$, therefore in normal polynomial multiplication scenarios, $a(x)$ and $b(x)$ are padded with coefficients 0 before applying NTT. If the original $a(x)$ and $b(x)$ are not padded with 0s, the result product corresponds to the actual product modulo $x^n - 1$.

$$a(x) \cdot b(x) \mod (x^n - 1) = NTT^{-1}(NTT(a(x)) \triangle NTT(b(x)))$$

However, in our contexts, we want the operations are computed modulo $x^n + 1$, one can still pad the original ring elements with 0s and compute the product modulo $x^n + 1$, this will double the length of NTT inputs and also require an extra step of explicite reducing $x^n + 1$. The other way to avoid this issue is by exploiting the *negative wrapped convolution* [56]. Let ψ be a primitive $2n^{th}$ root of unity in $\mathbb{Z}_t$ such that $\psi^2 = \omega$ and denote $NTT_+(\mathbf{a})$ to be the special NTT transform such that $NTT_+(\mathbf{a}) = NTT(\mathbf{a} \triangle (1, \psi, \psi^2, \ldots, \psi^{n-1}))$ and $NTT_+^{-1}(\mathbf{a}) = NTT^{-1}(\mathbf{a}) \triangle (1, \psi^{-1}, \psi^{-2}, \ldots, \psi^{-(n-1)})$. It can be proved that $NTT_+(\mathbf{a})$ is the evaluations of $a(x)$ at the odd roots of the $2n^{th}$ roots of unity: $NTT_+(\mathbf{a}) = (a(\omega_0), a(\omega_1), \ldots, a(\omega_{n-1}))$, where $\omega_i = \psi^{2i+1}$. It turns out that

$$a(x) \cdot b(x) \mod (x^n + 1) = NTT_+^{-1}(NTT_+(a(x)) \triangle NTT_+(b(x)))$$

Since most of our works are based on $R_q = \frac{\mathbb{Z}_q}{x^n+1}$, when we mention NTT throughout the text, we imply the operation is done with NTT_+ instead of normal NTT. In the implementation, in order to find ψ, we need to find the group element such that $\psi^2 = \omega \mod t$, where ω is the n^{th} root of unity. There are heuristic approaches to find the square root of an element, in our context, we can find it easily (the problem is finding an element with order of $2n$). Given n, t such that $2n | (t-1)$ (these parameters are set up at the initialization stage of the protocol), if we can find a generator g of order $t-1$ in $\mathbb{Z}_t^{*J}$, then $\psi = g^{\frac{t-1}{2n}}$:

$$order(g^{\frac{t-1}{2n}}) = \frac{t-1}{gcd(\frac{t-1}{2n}, t-1)} = \frac{t-1}{(\frac{t-1}{2n})} = 2n$$

6.3.1 The rotate and permute function

Slot Rotation Given a polynomial $a(x) = (a_0, a_1, \dots, a_{n-1}) \in \mathbb{Z}_t^n$, let $\mathbf{a}$ be the coefficient representation of $a(x)$ and $\hat{\mathbf{a}}$ be the NTT representation of $a(x)$. The *rotate* operation is a mapping K such as $a(x) \overset{K_j}{\mapsto} a(x^j)$, where j is the number of positions being rotated and $gcd(j,n) = 1$. This operation can be done on either the coefficient or the NTT representation of $a(x)$.

Coefficients domain In order to map $a(x) \overset{K_j}{\mapsto} a(x^j)$, or

$$\mathbf{a} = (a_0, a_1, \dots, a_{n-1}) \overset{K_j}{\mapsto} a(x^j) = a_0 + a_1(x^j)^1 + a_2(x^j)^2 + \cdots + a_{n-1}(x^j)^{n-1}$$

We can look at the relation between the coefficients before and after the mapping: Each of the mapped result coefficient $a_i(x^{j(n-i)})$ will be mapped to one of the original position a_l for $l = 0, \dots, n-1$ when we do modulo $x^n + 1$, or $x^n = -1$. In other word, the rotate operation of $\mathbf{a}$ is just a permutation of the original coefficients with some sign changes on some coefficients. However, we are more interested in the rotation operation in the NTT domain as our message was packed with CRT packing method.

NTT domain Given $\hat{\mathbf{a}}$, which is the evaluation of $a(x)$ at the n primitive $2n^{th}$ roots of unity modulo t (Section 6.3): The rotate operation maps $\hat{\mathbf{a}}$ to $\hat{\mathbf{b}} = (\hat{b}_0, \hat{b}_1, \dots, \hat{b}_{n-1})$, where $\hat{b}_i = a(x^j)|_{x=\omega_i} = a(\omega_i^j) = a(\psi^{(2i+1)j})$. We can see that the rotation on the NTT domain is also a permution of the original coefficients, the difference compared to the operation on coefficient domain is that it does not have the sign change effect on the original coefficients. However, this permutation is not the one we need for the HD computation algorithm (Algorithm 19): the i^{th} CRT component of $\hat{\mathbf{a}}$ is the i'^{th} CRT component of $\hat{\mathbf{b}}$, where i' satisfies $2i' + 1 \equiv (2i+1)j \mod 2n$.

We define $\bar{\mathbf{a}}$ to be also the evaluation of $a(x)$ at the n primitive $2n^{th}$ roots of unity modulo t, but in a different order, such that when applying the rotation operation, we will get back exactly the order we need for the HD computation algorithm. The idea is, with n a power of 2, those 2n-th roots of unity $\psi_{\mathbf{i}}$ for $i = 0, \dots, n-1$ form a group that is isomorphic to the multiplicative group of integers modulo n (the set of congruence classes relatively prime to the modulo n), denoted by $\mathbb{Z}_{2n}^*$. Note that this group $\mathbb{Z}_{2n}^*$ is not a cyclic group but it can be generated by sets of "generators". For n is a power of 2, the group $\mathbb{Z}_{2n}^*$ can be generated by $(3)^x(-1)^y$ for $x \in \mathbb{Z}_{n/2}$ and $y \in \mathbb{Z}_2$. The polynomial $\bar{\mathbf{a}}$ can be represented by

$$\begin{aligned}\bar{\mathbf{a}} = (&a(\psi^{3^0(-1)^0}), a(\psi^{3^1(-1)^0}), \ldots, a(\psi^{3^{n/2-1}(-1)^0}),\\ &a(\psi^{3^0(-1)^1}), a(\psi^{3^1(-1)^1}), \ldots, a(\psi^{3^{n/2-1}(-1)^1}))\end{aligned}$$

Each coefficient of $\bar{\mathbf{a}}$ can be indexed and denoted as follows

$$\bar{a}_{i,i'} = a(\psi^{3^i(-1)^{i'}}) \text{ for } i \in \mathbb{Z}_{n/2} \text{ and } i' \in \mathbb{Z}_2$$

Let's look at the mapping $a(x) \overset{K_j}{\mapsto} a(x^j)$ again under this new representation. Denote $b(x) = a(x^j)$, where $j \in \mathbb{Z}_{2n}^*$. We remark that every value of j can also be represented by $j = 3^{x_j}(-1)^{y_j}, (x_j \in \mathbb{Z}_{n/2}$ and $y_j \in \mathbb{Z}_2)$. We have

$$\begin{aligned}\bar{\mathbf{b}} &= (b(\psi^{3^0(-1)^0}), b(\psi^{3^1(-1)^0}), \ldots, b(\psi^{3^{n/2-1}(-1)^0}),\\ &\quad b(\psi^{3^0(-1)^1}), b(\psi^{3^1(-1)^1}), \ldots, b(\psi^{3^{n/2-1}(-1)^1}))\\ &= (a((\psi^{3^0(-1)^0})^{3^{x_j}(-1)^{y_j}}), a((\psi^{3^1(-1)^0})^{3^{x_j}(-1)^{y_j}}), \ldots, a((\psi^{3^{n/2-1}(-1)^0})^{3^{x_j}(-1)^{y_j}}),\\ &\quad a((\psi^{3^0(-1)^1})^{3^{x_j}(-1)^{y_j}}), a((\psi^{3^1(-1)^1})^{3^{x_j}(-1)^{y_j}}), \ldots, a((\psi^{3^{n/2-1}(-1)^1})^{3^{x_j}(-1)^{y_j}}))\\ &= (a(\psi^{3^{0+x_j}(-1)^{0+y_j}}), a(\psi^{3^{1+x_j}(-1)^{0+y_j}}), \ldots, a(\psi^{3^{n/2-1+x_j}(-1)^{0+y_j}}),\\ &\quad a(\psi^{3^{0+x_j}(-1)^{1+y_j}}), a(\psi^{3^{1+x_j}(-1)^{1+y_j}}), \ldots, a(\psi^{3^{n/2-1+x_j}(-1)^{1+y_j}}))\end{aligned}$$

By this representation, we see that the $\bar{\mathbf{b}}$'s coefficients $\bar{b}_{i,i'}$ are the $\bar{\mathbf{a}}$'s coefficients $\bar{a}_{i+x_j \mod n/2, i'+y_j \mod 2}$. In effect we will have a rotation here! It is not exactly a normal rotation, but it is 2 rotations on the 2 halves, each of the half rotates by x_j position and they will swap if $y_j = 1$. In our context, we can take $x_j = 2^0, 2^1, \ldots, 2^{\log n/2}$ and $y_j = 0$ to add up the bits of each half, then we swap the halves by setting $x_j = 0$ and $y_j = 1$ to add up all the bits to get the final value of Hamming Distance.

For ciphertexts that are the multiplication or addition of polynomials, when we map their rotations, we can perform each individual rotation and then do the original operation:

$$a(x).b(x) \mapsto a(x^j).b(x^j)$$
$$a(x) + b(x) \mapsto a(x^j) + b(x^j)$$

Therefore, a ciphertext of the form $c = (\mathbf{as} + t\mathbf{e} + \mathbf{m}, -\mathbf{a})$ can be rotated by doing the mapping on each of the components $\mathbf{a}, \mathbf{s}, \mathbf{e}$ and $\mathbf{m}$ before adding up the results.

6.3.2 The flattening gadget

6.3.3 The keyswitch and modswitch operation

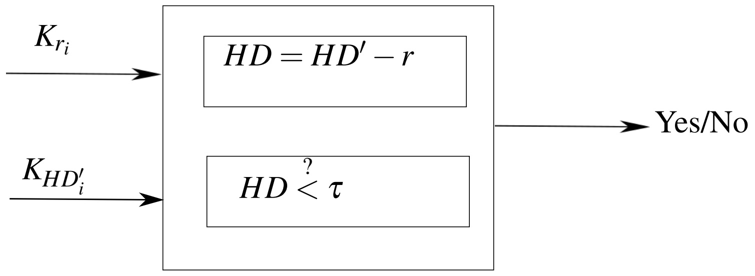
6.4 HD computation

6.5 Garbled Circuit and Oblivious Transfer

6.5.1 Oblivious Transfer

In 1-out-of-2 Oblivious Transfer protocol (denoted by $\binom{2}{1}$-OT), a *sender* has two choices m_0, m_1, and a *receiver* has one bit b as its input. At the end of the protocol, the *receiver* learns m_b and the sender learns nothing about the *receiver*'s query.

6.5.2 Yao Garbled Circuit



The concept of garbled circuit was proposed in ([REF]!)"yao generate exchange secret", it allows 2 parties to evaluate securely any function (represented as boolean circuits). We refer to the *server* as the one who prepares the circuit and the *client* is the one evaluating the circuit. The general idea is that given a function composed of logic gates that are connected by wires, the server "garbles" the circuit by assigning two random cryptographic keys K_j^0 and K_j^1 to each wire j. The keys represent 0 and 1 values, which are the possible inputs to each wire, in related literature, keys are also referred to as *labels*. The server then encrypts a truth table for each gate of the circuit such that in order to compute the *output label* of a gate, one must know the *labels*, or keys of the input, this is achieved by double encryption using possible combinations of the input keys. Figure ([REF]!) shows an example of a simple gate with its key and truth table.

The final output of the function is also represented as pair of keys, the server knows the value of the function output corresponding to each *label*. After setting up the garbled circuit, the server sends it to the client and its input's keys. The client then uses OT technique discussed to obtain the keys corresponding to its inputs. During evaluating the circuit, the client decrypt one *label* K_j^i for each output wire j, where $i \in \{0,1\}$ and use the decrypted result to continue evaluating the circuit until it get the final keys of the circuit. The client then sends the final labels to the server to verify the function's output.

Recent improvements reduce both computation time and communication overhead associated with circuit setup and evaluation. ([REF]!)"vsedvenka 30" described a technique that allow XOR gates to be evaluated *for free*: there is no communication overhead with such gates and the evaluation does not involve cryptographic operations. ([REF]!)"vsedvenka 38" proposed a mechanism to reduce the communication size of binary gates by 25%: each gate can be specified by encoding only three outcomes instead of all four. The research by ([REF]!)"vsedvenka 29" improved the complexity of basic operations (addition, multiplication, comparison, etc) by reducing the number of non-XOR gates.

6.6 The details

6.6.1 The protocol description

6.6.2 The Garble Circuit and Oblivious Transfer

We use standard substractor circuit for the operation $HD = HD' - r$. Given 2 n-bit bitstring, the circuit is built from 1 half-substractor and $n-1$ full-substractors as in Figure 6.2

Where a half-substractor is built from 1 XOR gate and 1 AND gate, a full-substractor is built from 2 half-substractors and 1 OR gate (Figure 6.3)

For our context, let l to be the bit length of the HD' and r (l is typically 10-12 bits for 1024-2048 bits biometrics data), so we need about $5l$ gates for the substractor logic gates. The comparision circuit needs not to output 3 result as standard ones (where they can output either A > B, A < B or A = B), we simply want to check whether $HD' - r < \tau$, so we can simply have another l NAND gates plus $l-1$ XOR gates for the comparsion circuit (An example is as in Figure 6.4)

Garble Circuit preparation We denote GC to be the garbled circuit prepared by the server. The inputs to the circuit include the server's GC_r and the client's GC_h, which are the cryptographic key *labels* corresponding to the server's input r and client's input HD' to the function *CompareHD*. The main function of the circuit is, it first substracts

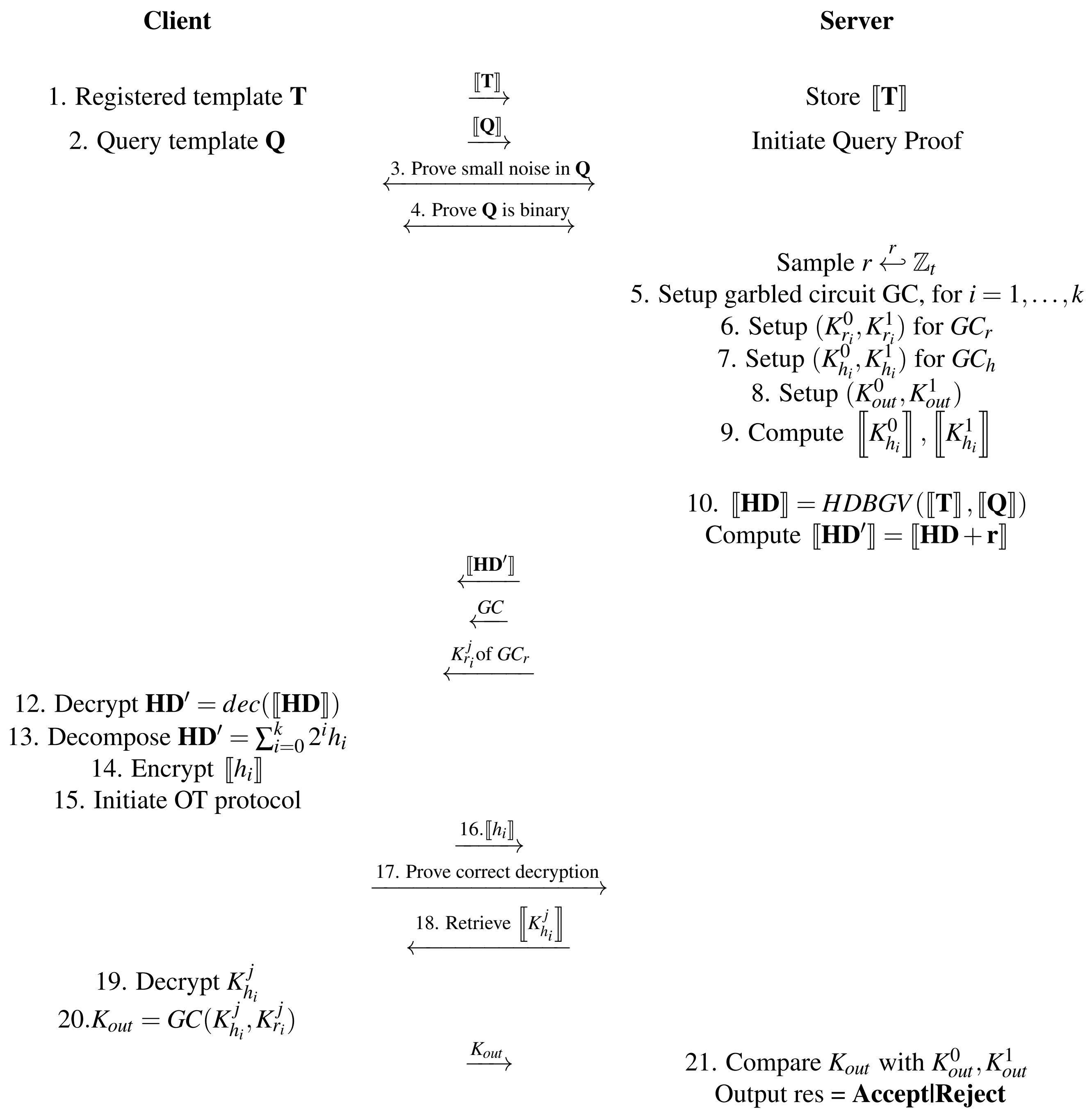

Client
Server
1. Registered template $\mathbf{T}$
$\xrightarrow{[\![\mathbf{T}]\!]}$
Store $[\![\mathbf{T}]\!]$
2. Query template $\mathbf{Q}$
$\xrightarrow{[\![\mathbf{Q}]\!]}$
Initiate Query Proof
$\xleftrightarrow{\text{3. Prove small noise in } \mathbf{Q}}$
$\xleftrightarrow{\text{4. Prove } \mathbf{Q} \text{ is binary}}$
Sample $r \overset{r}{\hookleftarrow} \mathbb{Z}_t$
5. Setup garbled circuit GC, for $i = 1, \ldots, k$
6. Setup $(K^0_{r_i}, K^1_{r_i})$ for GC_r
7. Setup $(K^0_{h_i}, K^1_{h_i})$ for GC_h
8. Setup (K^0_{out}, K^1_{out})
9. Compute $\left[\!\left[K^0_{h_i}\right]\!\right], \left[\!\left[K^1_{h_i}\right]\!\right]$
10. $[\![\mathbf{HD}]\!] = HDBGV([\![\mathbf{T}]\!], [\![\mathbf{Q}]\!])$
Compute $[\![\mathbf{HD'}]\!] = [\![\mathbf{HD} + \mathbf{r}]\!]$
$\xleftarrow{[\![\mathbf{HD'}]\!]}$
$\xleftarrow{GC}$
$\xleftarrow{K^j_{r_i} \text{of } GC_r}$
12. Decrypt $\mathbf{HD'} = dec([\![\mathbf{HD}]\!])$
13. Decompose $\mathbf{HD'} = \sum_{i=0}^{k} 2^i h_i$
14. Encrypt $[\![h_i]\!]$
15. Initiate OT protocol
$\xrightarrow{16. [\![h_i]\!]}$
$\xrightarrow{\text{17. Prove correct decryption}}$
$\xleftarrow{\text{18. Retrieve } \left[\!\left[K^j_{h_i}\right]\!\right]}$
19. Decrypt $K^j_{h_i}$
20.$K_{out} = GC(K^j_{h_i}, K^j_{r_i})$
$\xrightarrow{K_{out}}$
21. Compare K_{out} with K^0_{out}, K^1_{out}
Output res = **Accept|Reject**

Fig. 6.1 The fourth protocol

$HD' - r = HD$ then compares $HD \overset{?}{<} \tau$, where τ is the constant value of the threshold to decide the authentication result, which can be built into the circuit itself. We note that either the server or the client can do the role of setting up the garbled circuit, in our work, we let the

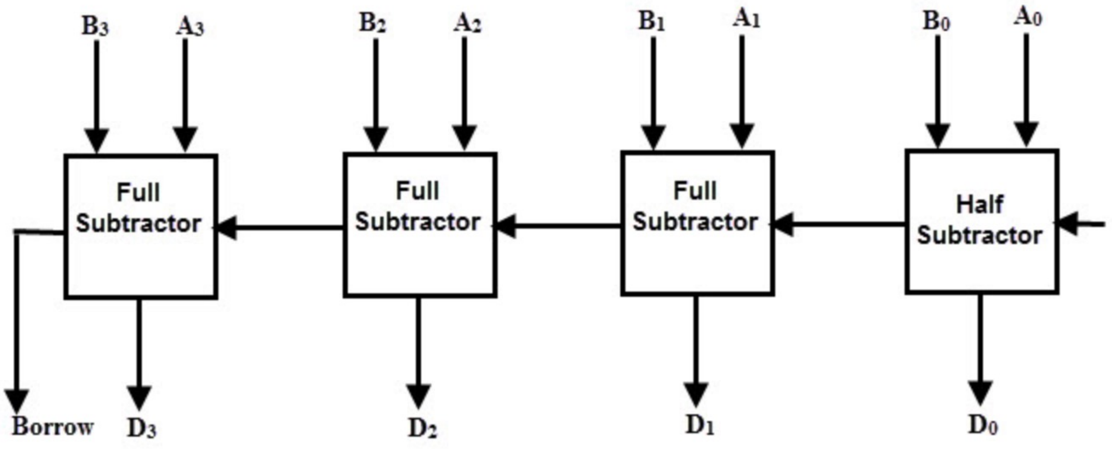


Fig. 6.2 Substractor Circuit

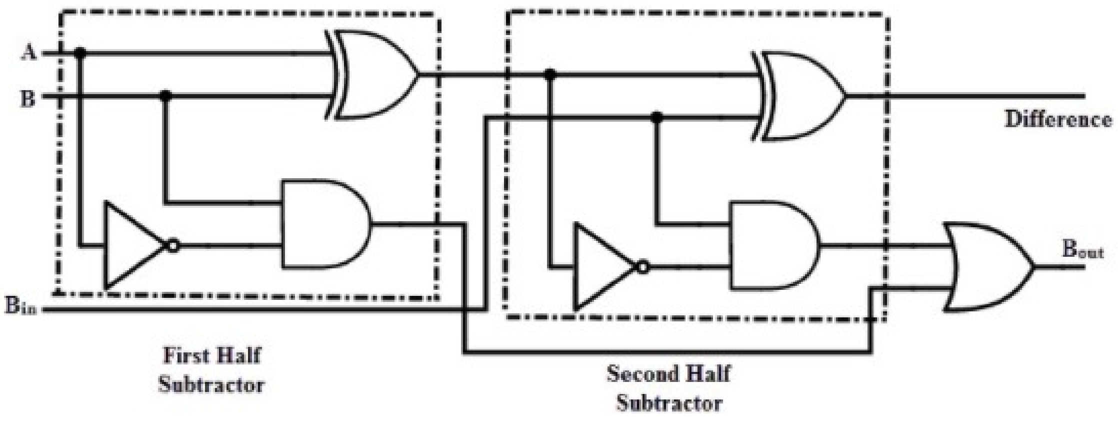


Fig. 6.3 Half-substractor and Full-substractor

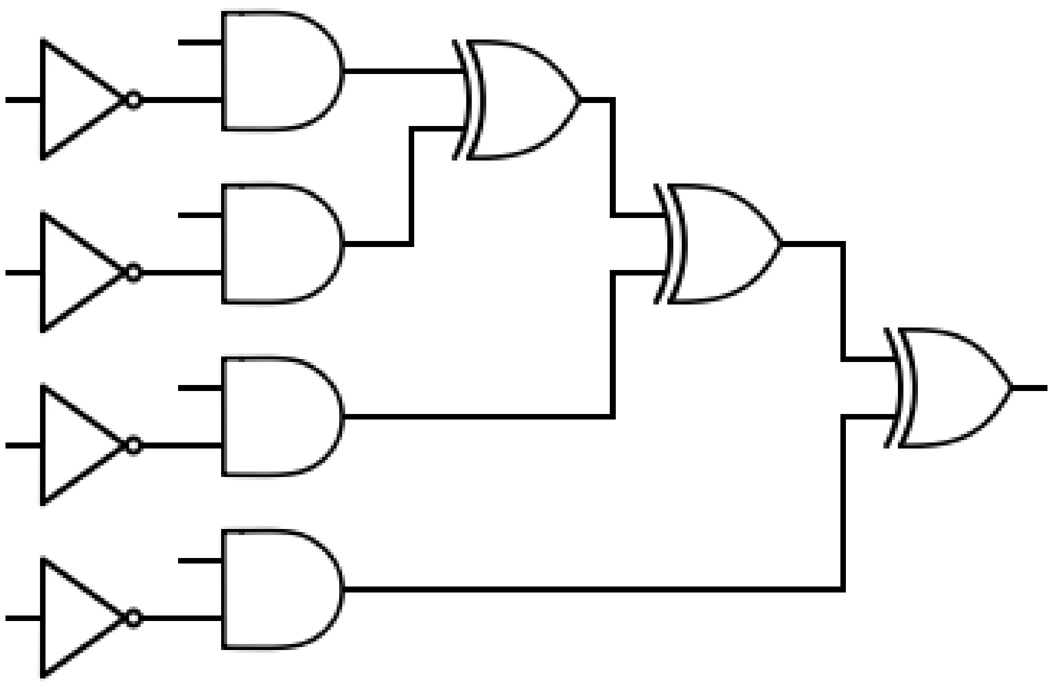


Fig. 6.4 Comparator Circuit

server take this role (Figure 6.1 - Step 5, 6, 7, 8) for the reason that we have been working on *semi-honest* server modeal and *malicious* client, this will save us one proof to prove that the circuit is generated correctly.

As described before in Section 6.5.2, the server garbles the circuit GC and sends to the client together with its *labels* of r, which are $K_{r_i}^j$. Next, the client uses our OT technique (([REF]!)) to select the correct *labels* of HD': the first part of the OT will ensure that the client will get back the encryptions $\left[\!\left[K_{h_i}^j\right]\!\right]$. The second part of the OT makes sure that the protocol is secure against *malicious* clients: it needs to prove that it decrypted and decompose $[\![\mathbf{HD}']\!]$ correctly (Figure 6.1 - Step 12,13). In other words, the client wants to prove that the encryptions $[\![h_i]\!]$ that he sent is actually the encryption of the bits of HD'_i. Recall that the server still has $[\![\mathbf{HD}']\!]$, so the proof can be a homomorphically check of $[\![\mathbf{HD}']\!] - \sum 2^i [\![h_i]\!] = [\![0]\!]$. This proof is part of the proof we did in the third variant of the protocol ([REF]!). However, in order to do this proof to support the homomorphic operations, we need to use the same encryption scheme for the OT protocol, in other words, we need an OT protocol based on BGV.

We note that inside the garbled circuit, AES is normally used to encrypt the *labels*, this encryption is a part of the garbled circuit and not related to the OT protocol we just described and can still be used independently to ensure the performance of circuit evaluations. Next, we discuss an approach to design an OT protocol that is compatible with BGV cryptosystem

ew PVW08, ivious trans- LWE

BGV-based OT

This section describes a technique to use BGV homomorphic operations to implement an OT protocol. The technique can be used together with the garbled circuit protocol discussed earlier. The problem can be stated as follows: Given a BGV ciphertext of a bit selection h_i: $[\![h_i]\!]$, the server computes and returns a BGV encryption of the corresponding key selected by the bit. We observe that the following homomorphic operations will obtain such requirement:

$$\left[\!\left[K_i^j\right]\!\right] = \left[\!\left[h_i^j\right]\!\right] \cdot [\![K_i^1]\!] + (1 - \left[\!\left[h_i^j\right]\!\right]) \cdot [\![K_i^0]\!]$$

Where K_i^0, K_i^1 are the *labels* corresponding to the wire inputs 0 or 1. The function include one level of homomorphic multiplication, two additions and one multiplication with a constant. This approach is simple enough and it is secure against *malicious client* model in our context because the client had to do the proof that he actually encrypted $[\![h_i]\!]$ correctly already. Other applications of this OT should take into account that a malicious client can encrypt $[\![h_i]\!]$ incorrectly to learn about K_i^0 and/or K_i^1.

The other issue that we have to be careful about is *Circuit Privacy*: for a multi-factor attacker that has access to the secret key but not the biometric template, it can learn about information of the randomness of the operation. The randomness of the above operation is of the form $K_i^0 + (K_i^1 - K_i^0)r_0$, which is correlated to both of the keys. Again, we can mask the randomness of this leakage similarly to what we did in the second variant of the protocol. By similar analysis, we can still show that computing the HD cannot be done with significant probability (the privacy security requirement is relaxed from what the client sees is indistinguishable to the probability that he can compute the HD is not much more than random guessing). This will be detailed in the security analysis ([REF]!).

6.6.3 HD Computation Homomorphically - BGV

Given two bistring **a** and **b** of length n (for n is a power of 2), the Hamming Distance $HD_{(\mathbf{a},\mathbf{b})}$ can be computed by adding up and rotating all the bits of $\mathbf{a} \oplus \mathbf{b}$ (Algorithm 19). Provided that the BGV cryptosystem used supports SIMD operations, the algorithm can run with homomorphic addition and multiplication. ($\triangle$ denotes component-wise homomorphic multiplication)

Algorithm 19 HD computation

```
1: procedure HDBGV($\mathbf{a},\mathbf{b}$)
2:     $\mathbf{c} \leftarrow \mathbf{a} + \mathbf{b} - 2\mathbf{a}\triangle\mathbf{b}$
3:     let $n \leftarrow length(\mathbf{a})$
4:     for $i = 0,\ldots,n-1$ do
5:         let $\mathbf{t} \leftarrow rotate(\mathbf{c}, 2^i)$
6:         let $\mathbf{c} \leftarrow add(\mathbf{c},\mathbf{t})$
7:     return $\mathbf{c}[0]$
```

6.7 Results

References

[Fuj] Fujitsu develops world's first slide-style vein authentication technology based on palm veins - fujitsu global. http://www.fujitsu.com/global/about/resources/news/press-releases/2017/0110-01.html. (Accessed on 01/23/2017).

[2] Ajtai, M. (1996). Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM.

[3] Ateniese, G., Camenisch, J., Joye, M., and Tsudik, G. (2000). A practical and provably secure coalition-resistant group signature scheme. In *Annual International Cryptology Conference*, pages 255–270. Springer.

[4] Bai, S., Langlois, A., Lepoint, T., Stehlé, D., and Steinfeld, R. (2015). Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–24. Springer.

[5] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., and Shacham, H. (2009). Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009*, pages 108–125. Springer.

[6] Belguechi, R., Alimi, V., Cherrier, E., Lacharme, P., Rosenberger, C., et al. (2011). An overview on privacy preserving biometrics. *Recent Application in Biometrics*, pages 65–84.

[7] Bellare, M. and Goldreich, O. (1992). On defining proofs of knowledge. In *Annual International Cryptology Conference*, pages 390–420. Springer.

[8] Benhamouda, F., Camenisch, J., Krenn, S., Lyubashevsky, V., and Neven, G. (2014). Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 551–572. Springer.

[9] Boneh, D. and Boyen, X. (2004). Short signatures without random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 56–73. Springer.

[10] Boneh, D. and Shacham, H. (2004). Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 168–177. ACM.

[11] Brakerski, Z. and Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer.

[12] Camenisch, J. and Groß, T. (2008). Efficient attributes for anonymous credentials. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 345–356. ACM.

[13] Camenisch, J. and Lysyanskaya, A. (2001). An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 93–118. Springer.

[14] Camenisch, J. and Stadler, M. (1997). Efficient group signature schemes for large groups. In *Advances in Cryptology—CRYPTO'97*, pages 410–424. Springer.

[15] Cappelli, R., Ferrara, M., and Maltoni, D. (2010). Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2128–2141.

[16] Chase, M., Kohlweiss, M., Lysyanskaya, A., and Meiklejohn, S. (2013). Malleable signatures: Complex unary transformations and delegatable anonymous credentials. *IACR Cryptology ePrint Archive*, 2013:179.

[17] Chen, Y. and Nguyen, P. (2011). Bkz 2.0: Better lattice security estimates. *Advances in Cryptology–ASIACRYPT 2011*, pages 1–20.

[18] Damgard, I., Geisler, M., and Kroigard, M. (2008). Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31.

[19] Daugman, J. (2003). The importance of being random: statistical principles of iris recognition. *Pattern recognition*, 36(2):279–291.

[20] Ducas, L. and Micciancio, D. (2015). Fhew: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer.

[21] Feige, U., Fiat, A., and Shamir, A. (1988). Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94.

[22] Feng, J. and Jain, A. K. (2011). Fingerprint reconstruction: from minutiae to phase. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):209–223.

[23] Fiat, A. and Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 186–194. Springer.

[FVC] FVC. Fvc-ongoing. https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx. (Accessed on 12/04/2016).

[25] Galil, Z., Haber, S., and Yung, M. (1985). Symmetric public-key encryption. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 128–137. Springer.

[26] Gentry, C. (2009). *A fully homomorphic encryption scheme*. PhD thesis, Stanford University. crypto.stanford.edu/craig.

[27] Gentry, C., Halevi, S., and Smart, N. P. (2012). Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–482. Springer.

[28] Gentry, C., Peikert, C., and Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM.

[29] Gentry, C., Sahai, A., and Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer.

[github] github. rimrim/rqbv. https://github.com/rimrim/rqbv. (Accessed on 03/04/2017).

[31] Goldreich, O. (2009). *Foundations of cryptography: volume 2, basic applications*. Cambridge university press.

[32] Goldwasser, S. and Kharchenko, D. (2005). Proof of plaintext knowledge for the ajtai-dwork cryptosystem. In *Theory of Cryptography Conference*, pages 529–555. Springer.

[33] Goldwasser, S., Micali, S., and Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208.

[34] Groth, J. (2007). Fully anonymous group signatures without random oracles. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 164–180. Springer.

[35] Guillou, L. C. and Quisquater, J.-J. (1990). A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *Proceedings on Advances in cryptology*, pages 216–231. Springer-Verlag New York, Inc.

[36] Hanrot, G., Pujol, X., and Stehlé, D. (2011). Terminating bkz. *IACR Cryptology ePrint Archive*, 2011:198.

[37] Higo, H., Isshiki, T., Mori, K., and Obana, S. (2015). Privacy-preserving fingerprint authentication resistant to hill-climbing attacks. In *International Conference on Selected Areas in Cryptography*, pages 44–64. Springer.

[38] Hirano, T., Hattori, M., Ito, T., and Matsuda, N. (2013). Cryptographically-secure and efficient remote cancelable biometrics based on public-key homomorphic encryption. In *International Workshop on Security*, pages 183–200. Springer.

[39] Hoffstein, J., Pipher, J., and Silverman, J. (1996). Ntru: A ring-based public key cryptosystem. *Algorithmic number theory*, pages 267–288.

[40] Ishai, Y., Kilian, J., Nissim, K., and Petrank, E. (2003). Extending oblivious transfers efficiently. In *Crypto*, volume 2729, pages 145–161. Springer.

[41] Jain, A., Flynn, P., and Ross, A. A. (2007). *Handbook of biometrics*. Springer Science & Business Media.

[42] Jain, A. K., Nandakumar, K., and Nagar, A. (2008). Biometric template security. *EURASIP Journal on Advances in Signal Processing*, 2008:113.

[43] Jain, A. K., Nandakumar, K., and Ross, A. (2016). 50 years of biometric research: Accomplishments, challenges, and opportunities. *Pattern Recognition Letters*.

[44] Jin, A. T. B., Ling, D. N. C., and Goh, A. (2004). Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern recognition*, 37(11):2245–2255.

[45] Katz, J. (2003). Efficient and non-malleable proofs of plaintext knowledge and applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 211–228. Springer.

[46] Kawachi, A., Tanaka, K., and Xagawa, K. (2008). Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 372–389. Springer.

[47] Kolesnikov, V., Sadeghi, A.-R., and Schneider, T. (2009). Improved garbled circuit building blocks and applications to auctions and computing minima. *Cryptology and Network Security*, pages 1–20.

[48] Kolesnikov, V. and Schneider, T. (2008). Improved garbled circuit: Free xor gates and applications. *Automata, Languages and Programming*, pages 486–498.

[49] Lacharme, P., Cherrier, E., and Rosenberger, C. (2013). Preimage attack on biohashing. In *Security and Cryptography (SECRYPT), 2013 International Conference on*, pages 1–8. IEEE.

[50] Langlois, A., Stehlé, D., and Steinfeld, R. (2014). Gghlite: More efficient multilinear maps from ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 239–256. Springer.

[51] Lee, Y., Chung, Y., and Moon, K. (2009). Inverse operation and preimage attack on biohashing. In *Computational Intelligence in Biometrics: Theory, Algorithms, and Applications, 2009. CIB 2009. IEEE Workshop on*, pages 92–97. IEEE.

[52] Lenstra, A. K., Lenstra, H. W., and Lovász, L. (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534.

[53] Ling, S., Nguyen, K., Stehlé, D., and Wang, H. (2013). Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *Public-Key Cryptography–PKC 2013*, pages 107–124. Springer.

[54] Lyubashevsky, V. (2008). Lattice-based identification schemes secure under active attacks. In *International Workshop on Public Key Cryptography*, pages 162–179. Springer.

[55] Lyubashevsky, V. (2009). Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 598–616. Springer.

[56] Lyubashevsky, V., Micciancio, D., Peikert, C., and Rosen, A. (2008). Swifft: A modest proposal for fft hashing. In *International Workshop on Fast Software Encryption*, pages 54–72. Springer.

[57] Lyubashevsky, V., Peikert, C., and Regev, O. (2010). On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer.

[58] Mandal, A., Roy, A., and Yasuda, M. (2015). Comprehensive and improved secure biometric system using homomorphic encryption. In *International Workshop on Data Privacy Management*, pages 183–198. Springer.

[59] Micciancio, D. (2002). Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411.

[60] Micciancio, D. and Regev, O. (2008). Lattice-based cryptography, book chapter in postquantum cryptography, dj bernstein and j. buchmann.

[61] Micciancio, D. and Vadhan, S. P. (2003). Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *Annual International Cryptology Conference*, pages 282–298. Springer.

[62] Nagar, A., Nandakumar, K., and Jain, A. K. (2010). A hybrid biometric cryptosystem for securing fingerprint minutiae templates. *Pattern Recognition Letters*, 31(8):733–741.

[63] Naor, M. and Pinkas, B. (2001). Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics.

[64] Nguyen, P. Q. and Stehlé, D. (2006). Lll on the average. In *International Algorithmic Number Theory Symposium*, pages 238–256. Springer.

[OPM] OPM. Opm says 5.6 million fingerprints stolen in cyberattack, five times as many as previously thought, the washington post. https://www.washingtonpost.com/news/the-switch/wp/2015/09/23/opm-now-says-more-than-five-million-fingerprints-compromised-in-breaches/. (Accessed on 10/23/2016).

[66] Pedersen, T. P. (1991). Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer.

[67] Peikert, C., Vaikuntanathan, V., and Waters, B. (2008). A framework for efficient and composable oblivious transfer. In *Annual International Cryptology Conference*, pages 554–571. Springer.

[68] Pinkas, B., Schneider, T., Smart, N. P., and Williams, S. C. (2009). Secure two-party computation is practical. In *Asiacrypt*, volume 9, pages 250–267. Springer.

[69] Rackoff, C. and Simon, D. R. (1991). Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Annual International Cryptology Conference*, pages 433–444. Springer.

[70] Ratha, N. K., Chikkerur, S., Connell, J. H., and Bolle, R. M. (2007). Generating cancelable fingerprint templates. *IEEE Transactions on pattern analysis and machine intelligence*, 29(4).

[71] Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34.

[72] Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34.

[73] Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

[74] Roberts, C. (2007). Biometric attack vectors and defences. *Computers & Security*, 26(1):14–25.

[75] Schnorr, C.-P. (1989). Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer.

[76] Schnorr, C.-P. (1991). Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174.

[77] Šeděnka, J., Govindarajan, S., Gasti, P., and Balagani, K. S. (2015). Secure outsourced biometric authentication with performance evaluation on smartphones. *IEEE Transactions on Information Forensics and Security*, 10(2):384–396.

[78] Shahandashti, S. F., Safavi-Naini, R., and Ogunbona, P. (2012). Private fingerprint matching. In *Australasian Conference on Information Security and Privacy*, pages 426–433. Springer.

[79] Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. Ieee.

[80] Smart, N. P. and Vercauteren, F. (2014). Fully homomorphic simd operations. *Designs, codes and cryptography*, pages 1–25.

[81] Stehlé, D. and Steinfeld, R. (2011). Making ntru as secure as worst-case problems over ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 27–47. Springer.

[82] Stern, J. (1993). A new identification scheme based on syndrome decoding. In *Annual International Cryptology Conference*, pages 13–21. Springer.

[83] Teoh, A. B., Kuan, Y. W., and Lee, S. (2008). Cancellable biometrics and annotations on biohash. *Pattern recognition*, 41(6):2034–2044.

[84] Uludag, U. and Jain, A. K. (2004). Attacks on biometric systems: a case study in fingerprints. In *Proceedings of SPIE*, volume 5306, pages 622–633.

[85] Uludag, U., Pankanti, S., Prabhakar, S., and Jain, A. K. (2004). Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960.

[86] Yao, A. C.-C. (1986). How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE.

[87] Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., and Koshiba, T. (2014). Practical packing method in somewhat homomorphic encryption. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 34–50. Springer.

[88] Zhang, Y., Chen, Z., Xue, H., and Wei, T. (2015). Fingerprints on mobile devices: Abusing and eaking. In *Black Hat Conference*.

Appendix A

How to install LaTeX

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from
   https://www.tug.org/texlive/
2. Download WinCDEmu (if you don't have a virtual drive) from
   http://wincdemu.sysprogs.org/download/
3. To install Windows CD Emulator follow the instructions at
   http://wincdemu.sysprogs.org/tutorials/install/
4. Right click the iso and mount it using the WinCDEmu as shown in
   http://wincdemu.sysprogs.org/tutorials/mount/
5. Open your virtual drive and run setup.pl

or

Basic MikTeX - TeX distribution

1. Download Basic-MiKTeX(32bit or 64bit) from
   http://miktex.org/download
2. Run the installer
3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin) and choose Package Manager

4. Select or search for packages to install

TexStudio - TeX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Run the installer

Mac OS X

MacTeX - TeX distribution

1. Download the file from
   https://www.tug.org/mactex/

2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - TeX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Extract and Start

Unix/Linux

TeXLive - TeX distribution

Getting the distribution:

1. TexLive can be downloaded from
   http://www.tug.org/texlive/acquire-netinstall.html.

2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

   ```
   mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
   ```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

   ```
   cd /your/download/directory
   ./install-tl
   ```

4. Enter command ‘i’ for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64bit OS

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
```

```
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CentOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.