Privacy-preserving protocols from lattice cryptography

Tan, Benjamin Hong Meng

2019

Tan, B. H. M. (2019). Privacy-preserving protocols from lattice cryptography. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/88691>

<https://doi.org/10.32657/10220/48357>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**PRIVACY-PRESERVING PROTOCOLS FROM LATTICE
CRYPTOGRAPHY**

TAN HONG MENG BENJAMIN

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES

2019

PRIVACY-PRESERVING PROTOCOLS FROM LATTICE CRYPTOGRAPHY

TAN HONG MENG BENJAMIN

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES

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

2019

STATEMENT OF ORIGINALITY

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

02 May 2019

.....

Date

.....

Tan Hong Meng Benjamin

SUPERVISOR DECLARATION STATEMENT

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

02 May 2019

.....

Date

.....

Prof. Wang Huaxiong

AUTHORSHIP ATTRIBUTION STATEMENT

This thesis contains material from one paper published in and one paper under submission to the following peer-reviewed journal and two papers accepted at conferences in which I am listed as an author.

Chapter 4 is published as M. Kim, H. T. Lee, S. Ling, B. H. M. Tan, and H. Wang. Private compound wildcard queries using fully homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing*, PP (99):1–1, 2017.

The contributions of the co-authors are as follows:

- Prof. Wang and Prof. Ling managed the project and all authors involved in discussions of different levels during the project and edited the manuscript.
- Prof. Kim and Prof. Lee proposed the initial protocol and drafted the manuscript.
- I improved the protocol to support the exclusion wildcard $!(c)$, implemented the protocol, conducted the experiments and revised the manuscript.

Chapter 5 is a preliminary version of the work under submission to *IEEE Transactions on Dependable and Secure Computing* as B. H. M. Tan, H. T. Lee, H. Wang, S. Ren, and K. M. M. Aung. Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields.

The contributions of the co-authors are as follows:

- Prof. Wang, Dr. Ren and Dr. Aung led the project and all authors involved in discussions of different levels during the project and edited the manuscript.
- Prof. Lee proposed the initial direction, revised and edited the manuscript.
- I designed and implemented the comparison algorithm, conducted the experiments and wrote the manuscript.

Chapter 6 is published as B. Libert, K. Nguyen, B. H. M. Tan, and H. Wang. Zero-knowledge elementary databases with more expressive queries. In *Public-Key Cryptography – PKC 2019*, pages 255–285, Cham, 2019. Springer International Publishing

The contributions of the co-authors are as follows:

- Prof. Wang managed the project and all authors involved in discussions of different levels during the project and edited the manuscript.
- Dr. Libert provided the initial direction, gave the construction of the lattice-based trapdoor mercurial commitment scheme, proposed the explanation algorithms and revised the manuscript.
- Dr. Nguyen provided the initial direction, analyzed the complexity of instantiating zero-knowledge expressive elementary database with the lattice-based TMC scheme and a ring-variant of the trapdoor mercurial commitment scheme and revised the manuscript.
- I proposed the construction of zero-knowledge expressive elementary database and wrote the manuscript.

Chapter 7 is published as K. Nguyen, B. H. M. Tan, and H. Wang. Zero-knowledge password policy check from lattices. In *Information Security*, pages 92–113, Cham, 2017. Springer International Publishing.

The contributions of the co-authors are as follows:

- Prof. Wang managed the project and all authors involved in discussions of different levels during the project and edited the manuscript.
- Khoa proposed the initial direction and revised the manuscript.
- Khoa and I designed the ZKPPC scheme.
- I wrote the draft of the manuscript.

02 May 2019

.....
Date

.....
Tan Hong Meng Benjamin

*To my parents, Eric and Irene
and loves of my life,
Xiao Tang and Rowan.*

ACKNOWLEDGEMENTS

Many have contributed to make this thesis possible, and I would like to take this opportunity to express my heartfelt gratitude to them.

First of all, my supervisor, Prof. Wang Huaxiong, and co-supervisor Dr. Khin Mi Aung for their patience, guidance and tremendous support throughout my journey towards the completion of this thesis. I am very thankful for the freedom they gave me in setting my research direction and insights on the problems I decided to tackle.

Also, I would like to thank my former supervisor, Dr. Ren Shuqin, for her counsel when I was just starting out, and without whom I would not have walked this path at all.

My sincere gratitude to Dr. Benoît Libert for hosting my visit to the AriC Crypto team at École Normale Supérieure de Lyon and for the fruitful collaboration on zero-knowledge data structures. I had a great time during my visit and am very grateful for the warm welcome and friendliness from the team despite the cold weather.

To my mentors, Prof. Lee Hyung Tae and Dr. Nguyen Ta Toan Khoa, thank you for the opportunity to work and learn with you. You two always patiently answer my questions and I greatly appreciate all of your advice.

I am also grateful to Prof. Kim Myungsun for his invitation to join the project on constructing private database queries with fully homomorphic encryption; culminating in the manuscript *Better Security for Queries on Encrypted Databases*. This project became the start of a fruitful collaboration which I hope will continue long after I graduate.

I would like to acknowledge the tremendous support from the Agency for Science, Technology and Research, awarding me the A*STAR Graduate Scholarship (Singapore) that funded the work in this thesis and giving me an opportunity to visit ENS Lyon.

Finally, I would like to show my appreciation to a few others: Fellow members of Prof Wang's research group, Drs. Frederic Ezerian Martianus, Lin Fuchun, Zhang Juanyang, Xu Yanhong, Ms. Zeng Neng and various visitors for the talks they gave and discussions we had together. My brother, Johnathan, for his encouragement and inspiration and my parents-in-law for their support and help with the care of my son during this period.

Contents

	Page
List of Figures	3
List of Tables	4
List of Work	5
Abstract	6
1 Introduction	9
1.1 Motivation	9
1.2 Our Contributions	13
1.3 Organization of the Thesis	15
1.4 Notation	15
2 Lattice Cryptography	17
2.1 Lattices	17
2.2 Computational Problems on Lattices	18
2.3 Gaussian Distributions	21
2.4 Modern Foundational Problems	22
2.5 Ideal Lattices	26
2.6 Cryptographic Primitives	31
I Private Database Queries with Fully Homomorphic Encryption	42
3 Preliminaries	43
3.1 Fully Homomorphic Encryption	43
3.2 The Brakerski-Gentry-Vaikuntanathan FHE Scheme	47
3.3 Finite Extension Fields	53
3.4 Private Database Queries	57
3.5 Experiment Platform	63

4	Private Compound Wildcard Queries	65
4.1	Introduction	65
4.2	Previous Work	67
4.3	Wildcard Pattern Matching Protocols	70
4.4	Implementation	79
5	Private Compound Comparison Queries	83
5.1	Introduction	83
5.2	Previous Work	85
5.3	Private Compound Order Comparison Protocols	86
5.4	Implementation	94
II	Zero-Knowledge and Databases	98
6	Zero-Knowledge Expressive Elementary Database	99
6.1	Introduction	99
6.2	Previous Work	101
6.3	Preliminaries	102
6.4	Zero-Knowledge Expressive Elementary Databases	113
6.5	Generic Construction from Trapdoor Mercurial Commitments	116
6.6	A Lattice-based Trapdoor Mercurial Commitment Scheme	129
7	Zero-Knowledge Password Policy Check	137
7.1	Introduction	137
7.2	Previous Work	139
7.3	Preliminaries	140
7.4	The Construction	143
8	Conclusion	155
8.1	Open Questions and Future Directions	156
	Bibliography	158

List of Figures

2.1	IND-CPA Game, $\mathcal{G}_{\mathcal{E}, \mathcal{A}}^{\text{IND-CPA}}$	33
2.2	Abstract Stern-like ZKAoK Protocol	39
3.1	Generic n -Round Protocol for $\pi_{\mathcal{Q}}$ with FHE.	60
3.2	Protocol $\pi_{\mathcal{Q}}$ for Conjunctive Query.	61
4.1	Multiplication Binary Tree for Four Ciphertexts.	80
5.1	Visual Representation of the LT Algorithm	89
6.1	Steiner Tree of $[1, 3]$	108
6.2	Merkle trees of ZK-EEDB	136
7.1	From Set Membership to Vector Well-Formedness	148
7.2	Example of an Extended Password Policy Linear Relation	150

List of Tables

3.1	Example of a Database	58
3.2	List of Multiplicative Depths	63
4.1	Performance of Wildcard Matching Algorithm <code>wcMatch</code>	81
4.2	Performance of Conjunctive Query <code>wildcardConjunction</code>	82
5.1	Performance of LT Algorithm (OEPS)	96
5.2	Performance of LT Algorithm (OBPS)	96
5.3	Performance of Range Query <code>rangeMatch</code>	97
5.4	Performance of Disjunctive Query <code>orderDisjunction</code>	97

LIST OF WORK

The following is a complete list of the papers that I worked on during my PhD studies.

1. M. Kim, H. T. Lee, S. Ling, S. Q. Ren, B. H. M. Tan, and H. Wang. Better security for queries on encrypted databases. Cryptology ePrint Archive, Report 2016/470, 2016. (Under Revision)
2. M. Kim, H. T. Lee, S. Ling, B. H. M. Tan, and H. Wang. Private compound wildcard queries using fully homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing*, PP (99):1–1, 2017.
3. K. Nguyen, B. H. M. Tan, and H. Wang. Zero-knowledge password policy check from lattices. In *Information Security*, pages 92–113, Cham, 2017. Springer International Publishing.
4. K. M. M. Aung, H. T. Lee, B. H. M. Tan, and H. Wang. Fully homomorphic encryption over the integers for non-binary plaintexts. *Theoretical Computer Science*, 2018.
5. B. Libert, K. Nguyen, B. H. M. Tan, and H. Wang. Zero-knowledge elementary databases with more expressive queries. In *Public-Key Cryptography – PKC 2019*, pages 255–285, Cham, 2019. Springer International Publishing.
6. B. H. M. Tan, H. T. Lee, H. Wang, S. Ren, and K. M. M. Aung. Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. *In Submission to IEEE Transactions on Dependable and Secure Computing*, 2019.

ABSTRACT

Data privacy is a major challenge that is not easy to solve. More and more attacks are happening and 2017 was another record year for data breaches and cyber incidents. On top of that, recent advances in quantum computers show how vulnerable the foundations of a secure Internet is today. Most public-key cryptography in use today will be completely broken once large-scale quantum computers are available.

Over the last decade, lattice cryptography received a lot of attention and is now one of the prime candidates for NIST’s post-quantum standardization efforts. Besides its conjectured resistance against quantum attacks, the main average-case problems, short integer solution and learning with errors, enjoy reductions to worst-case hardness of lattice problems. Lattice cryptography is also very versatile, having been used to construct many advanced cryptographic schemes like fully homomorphic encryption (FHE), which is seen as the “holy grail” of cryptography by some.

In this thesis, we use lattices to address these concerns and propose several privacy-preserving protocols from lattices. Our contributions can be grouped into two categories: (i) private database queries on outsourced database and (ii) zero-knowledge protocols for database queries and password policy check.

In the first scenario, we use fully homomorphic encryption for finite extension fields and design a wildcard pattern matching algorithm on encrypted strings and patterns. Our algorithm is the first to support an exclusion wildcard $!(c)$, which demands that a non-empty character that is not c be in its position. The algorithm’s usefulness is highlighted by combining it with exact matching in two protocols, evaluating conjunctive and disjunctive queries, in contrast to the similar schemes from FHE that cannot do so without expensive modifications (Yasuda *et al.* ’14 and Saha, Koshiba ’16). Experimental results show that it is highly parallelizable and especially efficient when combined with multi-threading.

Besides that, we present a private order comparison algorithm for encrypted integers by exploiting the structure of finite extension fields. Our algorithm is comparable in performance to Boolean circuit-based methods (Cheon, Kim, and Kim ’15, ’16) and improves the ciphertext expansion factor by at an order of magnitude. We obtain a private database query protocol for range conditions as well as a conjunctive query with multiple order and equality conditions, showcasing the versatility of the algorithm. The protocols do not reveal intermediate results, such as whether individual conditions are satisfied unlike some protocols based on order-preserving/revealing and partial homomorphic encryption.

In the domain of zero-knowledge protocols, we show that zero-knowledge elementary databases (ZK-EDB), which is used to prove statements of the form “ x has a non-empty value $D(x) = y$ in the database D ” or “ x does not have a corresponding value in the database D ” without revealing any additional information about the database, can support a much richer class of queries. We dub such schemes, zero-knowledge expressive elementary databases (ZK-EEDB), and prove range queries over keys, values and records of a database modulo a new security requirement that is satisfied by all previously known constructions. Moreover, we give the first quantum-resistant construction of trapdoor mercurial commitments from lattices, shown to imply ZK-EDB (Chase *et al.* ’05).

Finally, we introduce a lattice-based zero-knowledge password policy check protocol, the first of its kind that can potentially resist quantum adversaries. This is the first piece of a quantum-resistant privacy-preserving password registration, authentication and key exchange suite that allows users to register and use for authentication and key exchange passwords without ever revealing them to servers; thereby protecting this highly sensitive piece of data. We apply the KTX commitment scheme (Kawachi, Tanaka, Xagawa ’09) to construct a randomized password hashing scheme. Then, we employ an abstract Stern-like protocol (Libert *et al.* ’16) to prove that a password hashed with the randomized password hashing scheme we built satisfies the mandated password policy from the server in zero-knowledge. Notably, our techniques, adapted to the challenges of a lattice-based instantiation, do not follow the KM framework (Kiefer and Manulis ’14), using binary strings to encode passwords and relying only on standard commitments.

Chapter 1

Introduction

1.1 Motivation

In today's inter-connected world, leakage of confidential data is a major problem. 2017 saw "another worst year ever in data breaches and cyber incidents", according to the Online Trust Alliance [10], highlighting the increasing need for solutions to address the security and privacy issues around sensitive data stored in systems facing the Internet. Such attacks can potentially cause serious damage, as seen from hacks on popular websites such as LinkedIn [78] and Yahoo [175] that compromised the personal information of millions of people around the world. Therefore, it is important to design measures to mitigate the effects of such events, whether through data encryption or other techniques.

As quantum computing advances towards realization, there is a need to reassess the techniques that underlie cryptography in use today. With the ground-breaking algorithm by Shor [190], the key problems, RSA and discrete logarithm, underpinning the security of the current approach become easy once quantum computers with several thousand qubits are built. Lattice cryptography has emerged in recent years as a strong contender to replace these systems that currently protect us as there are no known efficient quantum algorithms that solve the typical problems used in lattice cryptography. The only advantage of using quantum computers to attack these problems are the generic speedups, such as the quantum search algorithm by Grover [98], which do not affect their asymptotic complexity besides reducing the constant factor. Furthermore, there is strong theoretical evidence that schemes from lattices are hard to solve. Unlike any other hard problem that we use to construct cryptographic schemes, the hard problems at the foundation of lattice cryptography have average-case to worst-case hardness reductions.

This thesis considers two cases where privacy is crucial: the first is cloud computing

and specifically, private queries on outsourced databases. The second is when one party holds the data and another party has some questions on that data, but both parties do not trust each other.

Private Queries on Outsourced Databases. In recent years, cloud computing services have become more and more attractive, promising simple system setup with high availability and scalability at low costs. However, it is a poisoned chalice. Outsourcing data to the cloud to use its vast resources is very appealing from a cost perspective but the key question is: how do we ensure that the data that is put onto the cloud cannot be compromised by third parties? Moving to the cloud does not mean that one is immune to data breaches [143], hackers can still attack systems hosted on the cloud and there is the slim (but non-zero) chance that employees or fellow tenants of the cloud service could try to steal data from within.

One of the most useful but difficult to move to the cloud is the organization's databases. These are often very large and contain sensitive information that is critical to their operations. The simplest way to secure them is to encrypt them before transferring them onto the cloud but that would render the move useless. The encryption schemes that offers the best protection for the data is also the least useful; the only way to use the encrypted data is to retrieve them onto trusted platforms and then decrypt them before use. Several practical encrypted database systems have been proposed, such as CryptDB [179] and Monomi [199], but require trading some security for functionality. For example, query access patterns of outsourced databases were shown to be vulnerable to certain attacks by Dautrich Jr. and Ravishankar [65], Kellaris *et al.* [120] and Lacharité, Minaud and Paterson [131].

Lattice cryptography offers a possible solution that does not trade security for functionality in fully homomorphic encryption (FHE). Regarded by some to be the "holy grail" of cryptography, it remained only a theoretical possibility for over thirty years until 2009, when Gentry [81] proposed the first construction. FHE allows us to evaluate arbitrary functions on encrypted data and can be used to design protocols for private database queries of arbitrary functionality with minimal round complexity. However, it is too impractical to realize this due to the large overhead that FHE would introduce. The focus for applications of FHE is on constructing efficient algorithms and protocols for a more limited set of functionality to address specific problems.

The literature on handling private database queries with FHE is quite sparse, with two main directions. First, Cheon *et al.* [53, 54] applied Boolean circuits and optimizations with large integer rings to perform efficient integer addition for sum and count function-

ality. The algorithms proposed in these work are fast and allow queries to be evaluated with low latency. However, they suffer from high ciphertext expansion because integers are encoded in their binary representation and encrypted bit-wise.

The other direction was initiated by Kim *et al.* [129] and extended by Kim *et al.* [127], who considered finite extension fields as a method to tease more performance out of FHE; almost all available space within a ciphertext is used by exploiting the natural plaintext space. Kim *et al.* [129] proposed protocols to evaluate queries with multiple equality conditions: efficient protocols for conjunctions, disjunctions and threshold conjunctions of multiple conditions with much lower ciphertext expansion factors are achieved. Kim *et al.* [127] additionally increased the security of these protocols by hiding their logical operators. An experimental study of their protocols was also performed, achieving much better amortized performance than the Boolean circuit techniques by Cheon *et al.* [53, 54]. Although the equality condition covers most basic database retrieval queries, there are many other aspects of database queries that have not been studied for this direction.

Two-Party Zero-Knowledge Interactions. In the first setting, all the data is owned by a single party, the client, and the server simply acts as a trustee who is able to perform some operations on its behalf before returning the results. Now, we consider the case where one party holds the data, called the prover, while the other asks for some information about that data, called the verifier. The verifier demands some information about the data from the prover, who will generate an answer and exchange some messages with the verifier to convince him/her on the truth of the answer. A public digest of the data in question is created before any interaction, with a cryptographic primitive called a commitment scheme, to fix it and prevent malicious tampering by the prover.

Additionally, the interaction has to fulfil several criteria. The prover, who distrusts the verifier, wishes to prevent the exchange to reveal more information than absolutely necessary. This concept is encapsulated in the notion of zero-knowledge in the cryptographic literature. The verifier too does not trust the prover to honestly compute answers based on the data. The notion of soundness captures this intent; informally, it states that the prover cannot convince the verifier of any statement that contradicts the information committed in the public digest.

We investigate two scenarios that fall under this setting, the first is where the data being proved belongs to a database. There are many different types of queries that are supported by conventional databases but there is a gap in functionality when zero-knowledge is required in interactions. Only simple questions about the existence and value of a key in the database is known to be possible when we demand maximum privacy. It

is important and interesting to realize as many of them as possible in the zero-knowledge setting.

One example is reverse lookups, which entails revealing all keys which have a particular value in the database. These find some use in practice, such as when trying to identify all persons who are at a location at a particular time. Such a functionality has not been considered previously in the literature.

Another rich class of queries are range queries, which underlie more complex questions such as k -nearest neighbours and k -minimum/maximum records. These are important tools in data analysis and deriving them from membership queries is prohibitively inefficient. Unfortunately, it is not known how to obtain range queries for zero-knowledge elementary database. Existing proposals in the literature all require weakening the notion of privacy to achieve range queries. In a three-party setting with trusted data owner, called a committer, and a third party who acts as the prover, a solution for zero-knowledge one-dimensional range queries was proposed by Ghosh *et al.* [90]. Alternatively, if we allow a polynomial-sized upper bound on the database size to be revealed, Ostrovsky *et al.* [166] showed that efficient zero-knowledge d -dimensional orthogonal range queries ($d \geq 1$) is possible.

The other scenario, orthogonal to database queries, is about authentication with passwords. Besides databases, passwords are another type of sensitive data that is ubiquitous these days. It is used as a means to authenticate a user when attempting to access private data and restricted services. However, without any restrictions, users were found by Ur *et al.* [202] to prefer choosing easy to guess passwords. Password policies were created to address this issue, forcing users to use passwords that are more random with certain constraints like including characters from several character sets, digits, upper- or lower-case letters and symbols.

Poor password management practices by servers pose a serious threat to users; the website “Have I Been Pwned?” [108] lists possible occasions where data breaches on popular services could have exposed their personal passwords. These could have been used to compromise the user’s accounts that share similar passwords on other services. Therefore, it is more secure for passwords to not be stored by servers as their practices should not be trusted. A cryptographic primitive called verifier-based password-based authenticated key exchange (VPAKE) promises better security for users by storing a related verifier on the server instead the actual password, and using said verifier to perform authenticated key exchange. However, during registration, passwords are still revealed to prove to the server that they satisfy some specified password policy.

Closing this final hole where passwords are exposed, Kiefer and Manulis [124] proposed

zero-knowledge password policy checks (ZKPPC) in 2014. This primitive enables blind registration: users register their chosen password with a server and prove that it conforms to the mandated password policy without revealing any information about the password. However, their proposal is based on the discrete logarithm problem and there is a need for alternative solutions that may be resistant to adversaries possessing quantum computers.

1.2 Our Contributions

In this thesis, the overarching theme is the use of lattice cryptography to realize privacy-preserving functionality that is potentially quantum-resistant but the contributions can be sub-divided into two parts. The first focuses on private database queries (PDQ) using fully homomorphic encryption whose plaintext spaces are finite extension fields while the second considers zero-knowledge protocols in the two-party setting.

CONTRIBUTION 1. Private Compound Wildcard Pattern Matching.

In Chapter 4, we introduce a new algorithm for encrypted pattern matching with wildcard symbols for strings whose characters are encoded as elements in a finite extension field. This algorithm is the first to support the wildcard symbol $!(c)$ which stands for a single non-empty character that is not c . Furthermore, the output of this algorithm is composable with other Boolean predicates without any additional processing. This property allows us to construct an efficient PDQ protocol for compound queries with wildcard pattern matching and exact pattern matching conditions. The protocol from this construction does not reveal any information about the underlying database, pattern and result to the server, achieving a higher level of security than previously possible. Experimental results show that the protocol is efficient and highly parallelizable.

CONTRIBUTION 2. Private Compound Order Comparison.

In Chapter 5, we propose a parameterized algorithm for evaluating comparisons for the order relations, $<$, $\leq$, $>$ and $\geq$, for integers encoded in finite extension fields. The fact that finite extension fields are vector spaces is exploited and used to reduce function evaluations on large fields to several smaller functions over a subset of the field. This yields an efficient algorithm with results that are composable with other Boolean predicates, allowing us to design efficient PDQ protocols for compound queries featuring multiple order and equality conditions. These protocols do not leak any information on the underlying database, query constants and results, ensuring a high degree of privacy for users. Experimental results show that the constructions are comparable with the state of the art and improve on its communication efficiency by at least one order of magnitude.

CONTRIBUTION 3. Zero-Knowledge Expressive Elementary Databases.

In Chapter 6, we improve on the state of the art of size-hiding zero-knowledge databases, extending previous constructions in several directions to support more expressive queries. First, we show how range queries over keys of a database can be proven in zero-knowledge by adding a new set of explain algorithms. Second, we apply this technique to handle range queries over values of the same database, which as for all records (x, y) in the database whose values y lie in the queried interval $[a_y, b_y]$. Lastly, we go one step further and handle range queries over records. These queries consists of a rectangle $[a_x, b_x] \times [a_y, b_y]$, where the height $[a_x, b_x]$ can be super-polynomial and width $[a_y, b_y]$ polynomial-sized. As a special case, with $[x, x] \times [y, y]$, we can prove efficiently that particular records (x, y) are not in the database, meaning “if x is in the database, then its value is not y ”. In addition, we provide a new construction of trapdoor mercurial commitment from standard lattices, which deviates from the generic construction of Catalano *et al.* [45] and is inspired by the first construction by Micali *et al.* [153].

CONTRIBUTION 4. Zero-Knowledge Password Policy Checks.

In Chapter 7, we design a zero-knowledge password policy check protocol from standard lattices. Particularly, we provide constructions of a randomised password hashing scheme and statistical zero-knowledge argument of knowledge for password policy compliance based on the hardness of the Short Integer Solution (SIS) problem. Our techniques deviate from the general framework proposed by Kiefer and Manulis [124] to accommodate the quirks of lattice cryptography: passwords are encoded to low-norm binary strings instead of large integers and standard commitments are used instead of homomorphic commitments.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2 introduces the basics of lattice cryptography and several fundamental cryptographic tools. Then, the thesis is split into two parts.

In Part I, Chapter 3 is devoted to fully homomorphic encryption and acts as the preliminary chapter for this part of the thesis. Chapters 4 and 5 focus on contributions 1 and 2 on private database queries outlined in earlier.

In Part II, the topic switches to lattice-based zero-knowledge protocols. Chapters 6 and 7 introduces zero-knowledge expressive elementary databases and zero-knowledge password policy checks respectively.

Finally, in Chapter 8, we conclude the thesis and discuss some future directions for our research.

1.4 Notation

Let $\mathbb{Z}_{>0}$ denote the set of positive integers and $[n]$ the set $\{1, 2, \dots, n\}$. For integers a, b , $[a, b]$, $[a, b)$ and $(a, b]$ denote the set of integers $\{a, a+1, \dots, b\}$, $\{a, a+1, \dots, b-1\}$ and $\{a+1, \dots, b\}$ respectively. In this work, $\log$ denotes base-2 logarithms.

An n -dimensional vector $\mathbf{x}$ is denoted with $(x_1, \dots, x_n)$. The ℓ_p norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is denoted by $\|\mathbf{x}\|_p = \sqrt[p]{\sum_{i=1}^n (x_i)^p}$ for $p \in \mathbb{Z}_{>0}$ and $\|\mathbf{x}\|_\infty = \max_{i \in [n]} x_i$ for $p = \infty$. $\otimes$ denotes the tensor product of two vectors.

For $x \in \{0, 1\}^\ell$, x' denotes the binary string that is identical to x except with the final position flipped, $x0$ and $x1$ respectively is the string of $\ell + 1$ bits with 0 and 1 appended to x . Besides that, we mean by $x|_i$, the length- i binary string that consists of the first i bits of x . This implies that $x|_0 = \epsilon$ and $x|_\ell = x$. For a string x , $\text{hw}(x)$ denotes the Hamming weight of x .

$\text{poly}(\lambda)$ denotes any unspecified polynomial in the parameter λ . The security parameter for cryptographic schemes is denote with λ and algorithms are efficient if they run in probabilistic polynomial time PPT, $\text{poly}(\lambda)$. A function is said to be negligible in λ , denoted with $\text{negl}(\lambda)$, if it is smaller than the reciprocal of $\text{poly}(\lambda)$.

Let $a \leftarrow \chi$ denote that the element a is sampled from the distribution χ and $\mathcal{U}(X)$ the uniform distribution over the set X . We say that an event occurs with overwhelming probability if it occurs with probability $1 - \text{negl}(\lambda)$.

For any two distributions X and Y , their statistical distance over support D is $\Delta(X, Y) = \frac{1}{2} \sum_{d \in D} |\Pr[X = d] - \Pr[Y = d]|$. X and Y are then said to be statistically

close, denoted with $X \approx_s Y$ if their statistical distance is negligible. Two distribution ensembles $\{X_n\}_{n \in \mathbb{Z}_{>0}}$ and $\{Y_n\}_{n \in \mathbb{Z}_{>0}}$ are said to be statistically indistinguishable if, for all sufficiently large n , X_n and Y_n are statistically close.

We recall the definitions of the Bachmann-Landau notations for characterizing asymptotic growth. A tilde over the symbols, such as $\tilde{\mathcal{O}}(X)$, indicates that logarithmic factors of X are suppressed.

- $f(n) = \mathcal{O}(g(n)) \Rightarrow \exists k > 0, \exists n_0 : \forall n > n_0, |f(n)| \leq k \cdot |g(n)|;$
- $f(n) = o(g(n)) \Rightarrow \forall k > 0, \exists n_0 : \forall n > n_0, |f(n)| \leq k \cdot |g(n)|;$
- $f(n) = \Omega(g(n)) \Rightarrow \exists k > 0, \exists n_0 : \forall n > n_0, |f(n)| \geq k \cdot |g(n)|;$
- $f(n) = \omega(g(n)) \Rightarrow \forall k > 0, \exists n_0 : \forall n > n_0, |f(n)| \leq k \cdot |g(n)|;$
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n)).$

In Part I, we use $\bar{p}$ to denote an encryption of a plaintext p .

Chapter 2

Lattice Cryptography

2.1 Lattices

First, we introduce the definitions and basic properties of lattices.

Definition 2.1.1. Let $\mathbf{B} = [\mathbf{b}_1 | \dots | \mathbf{b}_k] \in \mathbb{Z}^{n \times k}$, whose column vectors are linearly independent vectors in $\mathbb{R}^n$. The lattice generated by $\mathbf{B}$ is the set of all integer combinations of elements in the set,

$$\Lambda(\mathbf{B}) = \left\{ \sum_{i=1}^k x_i \cdot \mathbf{b}_i \mid \mathbf{x} \in \mathbb{Z}^k \right\}$$

$\mathbf{B}$ is called a basis of the lattice $\Lambda(\mathbf{B})$ and the integer k is known as the dimension or rank of the lattice.

A full-rank lattice has $k = n$, we restrict our attention to those lattices in this thesis. For dimensions $n > 1$, a lattice Λ has an infinite number of bases, for any unimodular matrix $\mathbf{U} \in \mathbb{Z}^{n \times n}$ (with determinant ± 1), $\mathbf{B} \cdot \mathbf{U}$ is also a basis of $\Lambda(\mathbf{B})$. For any $\mathbf{c} \in \mathbb{R}^n$, the coset $\mathbf{c} + \Lambda$ of $\mathbb{R}^n / \Lambda$ is the set $\{\mathbf{c} + \mathbf{v} \mid \mathbf{v} \in \Lambda\}$, which is the lattice Λ shifted by $\mathbf{c}$ at every point.

There are two important invariants for a lattice, the *determinant*, which is given by $\det(\Lambda) = \sqrt{\det(\mathbf{B}^T \cdot \mathbf{B})}$ for some basis $\mathbf{B}$, and successive minima.

Definition 2.1.2 (Successive Minima). For Λ , an n -dimensional lattice, the i -th successive minimum λ_i^p — for each $i \in [n]$ — with respect to the ℓ_p norm, is the smallest number r such that there exists i linearly independent vectors $\mathbf{v}_1, \dots, \mathbf{v}_i \in \Lambda$ with $\|\mathbf{v}_j\|_p \leq r$ for all $j \in [i]$.

The i -th successive minimum can be seen as the radius of the smallest ball in $\mathbb{R}^n$ containing i linearly independent lattice vectors. In particular, $\Lambda_1^p(\Lambda)$ is the length of the shortest non-zero vector in Λ for the ℓ_p norm.

Lastly, we introduce the notion of the dual of a lattice, which

Definition 2.1.3 (Dual Lattice). *The dual (or reciprocal) of a lattice $\Lambda \subset \mathbb{R}^n$ is defined as*

$$\Lambda^* := \{\mathbf{w} \in \mathbb{R}^n \mid \langle \mathbf{w}, \mathbf{v} \rangle \in \mathbb{Z}, \forall \mathbf{v} \in \Lambda\}.$$

It is the set of points in $\mathbb{R}^n$ whose inner products with every vectors in Λ are integers. A basis for Λ^ , $\mathbf{B}^\perp$, can be obtained from a basis of Λ , $\mathbf{B}$ by taking $\mathbf{B}^\perp = (\mathbf{B}^{-1})^T = (\mathbf{B}^T)^{-1}$. Note that the dual of the dual lattice is the original lattice.*

For lattice cryptography, a particular class of lattices, called q -ary lattices, is very important. Lattices belonging to this class satisfies the relation $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$ for some integer q that is possibly prime. Alternatively, Λ is a q -ary lattice if for any vector $\mathbf{v} \in \mathbb{Z}^n$, $\mathbf{v} \in \Lambda$ if $\mathbf{v} \bmod q \in \Lambda$. Although every integer lattice $\Lambda \subseteq \mathbb{Z}^n$ is a q -ary lattice for any q with $\det(\Lambda) \mid q$, only those with $q \ll \det(\Lambda)$ are really used.

Definition 2.1.4. *For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, where $q, n, m \in \mathbb{Z}$, we define two m -dimensional q -ary lattices*

$$\begin{aligned} \Lambda_q(\mathbf{A}) &= \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y} = \mathbf{A}^T \mathbf{s} \bmod q \text{ for some } \mathbf{s} \in \mathbb{Z}^n\}, \\ \Lambda_q^*(\mathbf{A}) &= \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{y} = \mathbf{0} \bmod q\}. \end{aligned}$$

The former lattice is the set of all vectors which are integer combinations of the rows of $\mathbf{A}$ modulo q while the latter contains all vectors in $\mathbb{Z}^m$ that are orthogonal modulo q to the rows of $\mathbf{A}$. The two are dual to one another up to normalization, i.e. $\Lambda_q(\mathbf{A}) = q \cdot (\Lambda_q^*(\mathbf{A}))^*$ and $\Lambda_q^*(\mathbf{A}) = q \cdot (\Lambda_q(\mathbf{A}))^*$.

2.2 Computational Problems on Lattices

Several computational problems on lattices have applications in cryptography and some important results regarding their complexities are discussed in this section. These problems depend on the norm taken, whether the Euclidean norm or ℓ_p norm, where $p \in \mathbb{Z}_{>0} \cup \{\infty\}$. Without any additional notation, the problem would refer to the one where the norm is the standard Euclidean norm. A superscript p will be used to denote problems which use the ℓ_p norm, for $p \neq 2$.

2.2.1 Definitions of the Problems

Particularly for lattice cryptography, approximation variants of the problems — parameterized by an approximation parameter $\gamma \geq 1$ — play a central role; these variants are denoted with a subscript γ . The approximation parameter is usually cast as a function of the dimension of the lattice n , i.e. $\gamma = \gamma(n)$. By setting $\gamma = 1$, the original problem can be recovered from the approximation variant. We start off with what is probably the problem that has been most studied, the shortest vector problem.

Definition 2.2.1 ((Approx.) Shortest Vector Problem ($\text{SVP}_{(\gamma)}^p$)).

- **SVP**: Given an n -dimensional lattice $\Lambda(\mathbf{B})$ defined by basis $\mathbf{B}$, find some non-zero $\mathbf{v} \in \Lambda$ for which $\|\mathbf{v}\|_p = \lambda_1(\Lambda)$.
- **SVP $_{\gamma}$** : Given an n -dimensional lattice $\Lambda(\mathbf{B})$ described by an arbitrary basis $\mathbf{B}$, find a non-zero $\mathbf{v} \in \Lambda$ for which $\|\mathbf{v}\|_p = \gamma(n) \cdot \lambda_1(\Lambda)$.

These problems are called *search* problems as they require finding a vector $\mathbf{v}$ that satisfies the conditions. However, there are no known cryptographic schemes whose security is reduced to the hardness of SVP or SVP $_{\gamma}$. Next, we introduce two problems, whose assumed hardness have been used as the basis of security for some cryptosystems: the *decision* version of approximate-SVP (**GapSVP $_{\gamma}$**) and approximate shortest independent vectors problem (**SIVP $_{\gamma}$**), which is a search problem related to the n -th successive minimum of a lattice.

Definition 2.2.2 (**GapSVP $_{\gamma}$**). Given an n -dimensional lattice $\Lambda(\mathbf{B})$ such that either $\lambda_1(\Lambda) \leq 1$ or $\lambda_1(\Lambda) > \gamma(n)$, determine which of the two is true.

Definition 2.2.3 (**SIVP $_{\gamma}$**). Given an n -dimensional lattice $\Lambda(\mathbf{B})$ of full rank, find a set $\mathcal{S} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\} \subset \Lambda$ of n linearly independent lattice vectors, such that $\|\mathbf{v}_i\| \leq \gamma(n) \cdot \lambda_n(\Lambda)$ for all $i \in [n]$.

The last problem we discuss is the bounded-distance decoding problem (**BDD $_{\gamma}$**), which demands the closest lattice vector to some given target point $\mathbf{t} \in \mathbb{R}^n$ that is guaranteed to be within a short distance away from the lattice. This problem differs from the approximate variant of the well-known closest vector problem (**CVP $_{\gamma}$**) in that the target vector in **CVP $_{\gamma}$** is an arbitrary point in $\mathbb{R}^n$ and there is no promise of closeness. **CVP $_{\gamma}$** is not formally defined in this thesis as no cryptographic scheme has security properties based on its hardness.

Definition 2.2.4 (BDD_γ). *For an n -dimensional lattice $\Lambda(\mathbf{B})$ and target point $\mathbf{t} \in \mathbb{R}^n$ such that $\text{dist}(\mathbf{t}, \Lambda) < d = \lambda_1(\Lambda)/(2\gamma(n))$, find the unique lattice vector $\mathbf{v} \in \Lambda$ with $\|\mathbf{t} - \mathbf{v}\| < d$.*

2.2.2 Worst-Case Complexity and Algorithms

Some lattice problems are known to be NP-hard with various types of randomized reductions. Some are NP-hard in their approximate variants for sub-polynomial γ , i.e. $\gamma(n) = n^{\mathcal{O}(1)}$. For example, the NP-hardness of SVP was proven by van Emde Boas [204] for the ℓ_∞ norm. The same for the ℓ_2 norm was proven by Ajtai [5] under randomized reductions. One consequence of these results is that SIVP is NP-hard for these norms. Following them, Micciancio [154] proved that for any p and for $\gamma = \sqrt[p]{2}$, SVP_γ^p is NP-hard under reverse faithful random reductions. Khot [122] and Haviv and Regev [103] reduced the approximation factor to $2^{(\log n)^{1/2-\varepsilon}}$ and $2^{(\log n)^{1-\varepsilon}}$ respectively, for any $\varepsilon > 0$.

Unfortunately, in the context of lattice cryptography, these hardness results are not directly applicable as polynomial approximation factors are required in that case, $\gamma(n) \geq n$. In fact, Goldreich and Goldwasser [92] showed that SVP_γ lies in $\text{NP} \cap \text{coAM}$ for $\gamma = \Omega(\sqrt{n/\log n})$. Later, Aharonov and Regev proved that for $\gamma = \Omega(\sqrt{n})$, the problem belongs to $\text{NP} \cap \text{coNP}$. Together, they suggest that unless $\text{P} = \text{NP}$, the problems cannot be NP-hard within such factors.

Since the problems underlying lattice cryptography are not known to be NP-hard, understanding the performance of algorithms solving them helps us gauge their difficulty better. For the state of the art to date, polynomial-time algorithms include the famous LLL algorithm, named after the inventors Lenstra, Lenstra, Lovász [134], and successors such as the BKZ algorithm proposed by Schnorr [188] combined with Ajtai, Kumar and Sivakumar's [7] sieving techniques. These algorithms only achieve slightly sub-exponential approximation factors, $\gamma = 2^{\Theta(n \log \log n / \log n)}$, for the problems discussed in Section 2.2.1. On the other hand, algorithms that approximate the solutions to polynomial factors — like the one proposed by Kannan [114], Ajtai, Kumar and Sivakumar [7] and several others [160, 1, 2] — require either super-exponential ($2^{\Theta(n \log n)}$) time or exponential ($2^{\Theta(n)}$) time and space. Time-approximate tradeoffs can also be achieved with the techniques of Schnorr [188], interpolating between the two classes of algorithms to achieve approximation factors, $\gamma = 2^k$, in time $2^{\tilde{\Theta}(n/k)}$.

On the *quantum* front, there is currently no significant improvement from the classical algorithms discussed above; Laarhoven, Mosca and van der Pol [130] showed that the hidden constant factors is slightly smaller in the quantum case.

2.3 Gaussian Distributions

The fundamental tool in modern lattice cryptography is Gaussian-like probability distributions over lattices, which are called *discrete Gaussian* distributions.

Definition 2.3.1 (Gaussian Density Function). *For any $n \in \mathbb{Z}_{>0}$ and $s \in \mathbb{R}_{>0}$, define the Gaussian density function ρ_s of parameter s*

$$\begin{aligned} \rho_s : \mathbb{R}^n &\longrightarrow \mathbb{R}_{>0} \\ \rho_s(\mathbf{x}) &\longmapsto \exp\left(-\pi \frac{\|\mathbf{x}\|^2}{s^2}\right). \end{aligned}$$

When $s = 1$, we simply use ρ to denote the density function. The probability distribution function (p.d.f.) of a Gaussian of parameter s is proportional to ρ_s .

Definition 2.3.2 ((Discrete) Gaussian Distribution). *The p.d.f. of an n -dimensional Gaussian distribution with center $\mathbf{c} \in \mathbb{R}^n$ and parameter s is*

$$D_{\mathbf{c},s}(\mathbf{x}) = \frac{1}{s^n} \cdot \exp\left(-\pi \frac{\|\mathbf{x} - \mathbf{c}\|^2}{s^2}\right) = \frac{\rho_s(\mathbf{x} - \mathbf{c})}{s^n}.$$

For a lattice Λ , the p.d.f. of the discrete Gaussian centered at $\mathbf{c}$ with parameter s and support restricted to the coset $\mathbf{c} + \Lambda$ is

$$D_{\Lambda+\mathbf{c},s}(\mathbf{x}) = \begin{cases} \frac{D_{\mathbf{c},s}(\mathbf{x})}{\sum_{\mathbf{z} \in \Lambda} D_{\mathbf{c},s}(\mathbf{z})}, & \text{if } \mathbf{x} \in \Lambda; \\ 0, & \text{otherwise.} \end{cases}$$

The Gaussian distribution centered at the origin over the entire space and discrete version restricted to the lattice Λ are denoted by D_s and $D_{\Lambda,s}$ respectively. One particular distribution, where $\Lambda = \mathbb{Z}^n$, is widely used in lattice cryptography; in this case, for $\mathbf{x} = (x_1, \dots, x_n) \leftarrow D_{\mathbb{Z}^n,s}$, each x_i is independently and identically distributed (i.i.d) from the 1-dimensional discrete Gaussian $D_{\mathbb{Z},s}$.

We recall several important lemmas on discrete Gaussians on the integer lattice from the work of Gentry, Peikert and Vaikuntanathan [86]. The first roughly bounds the maximum norm of vectors sampled from a discrete Gaussian distribution while the second characterizes the minimum entropy of samples from discrete Gaussian distributions.

Micciancio and Regev [159] proposed a quantity, called the *smoothing parameter*, for lattices. Informally, it is the amount of Gaussian “randomness” that is required to “mask” a lattice’s discrete structure.

Definition 2.3.3 (Smoothing Parameter). *Parameterized by a tolerance $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest $s > 0$ such that $\rho_{1/s}(\Lambda^\star) \leq 1 + \varepsilon$.*

This parameter is closely related to some lattice invariants; two theorems that illustrate this link are given below.

Theorem 2.3.4 ([13, 159]). *For any n -dimensional full-rank lattice Λ ,*

$$\eta_{2^{-n}}(\Lambda) \leq \frac{\sqrt{n}}{\lambda_1(\Lambda^\star)}.$$

Theorem 2.3.5 ([159, 86]). *For any n -dimensional full-rank lattice Λ and $0 < \varepsilon < 1/2$,*

$$\begin{aligned} \eta_\varepsilon(\Lambda) &\leq \min_{\text{basis } \mathbf{B} \text{ of } \Lambda} \|\tilde{\mathbf{B}}\| \cdot \sqrt{\log \mathcal{O}(n/\varepsilon)} \\ &\leq \lambda_n(\Lambda) \cdot \sqrt{\log \mathcal{O}(n/\varepsilon)}. \end{aligned}$$

$\|\tilde{\mathbf{B}}\| = \max_i \|\tilde{\mathbf{b}}_i\|$ denotes the maximum length of the Gram-Schmidt orthogonalized vectors $\{\tilde{\mathbf{b}}_i\}$ for the ordered basis $\mathbf{B} = [\mathbf{b}_1 | \dots | \mathbf{b}_n]$.

From the smoothing parameter, assuming that $0 < \epsilon = \text{negl}(n)$, we obtain several important lemmas for the discrete Gaussian distribution over $\Lambda = \mathbb{Z}^n$.

Lemma 2.3.6 (Adapted from [86]). *For any vector $\mathbf{r}$ sampled from $D_{\mathbb{Z}^n, s}$ with $s \geq \sqrt{n}$, we have $\Pr[\|\mathbf{r}\| > s\sqrt{n}] \leq 2^{-n}$.*

Lemma 2.3.7 (Adapted from [86]). *For parameter $s \geq \tilde{\mathcal{O}}(\sqrt{n})$, the minimum entropy of $D_{\mathbb{Z}^n, s}$ is at least $n - 1$.*

2.4 Modern Foundational Problems

Lattices have been used to design cryptosystems [6, 93, 106, 107] for many years. Oftentimes, the schemes lacked [93, 106, 107] direct security reductions to lattice problems and their conjectured security were only heuristic. However, following the groundbreaking work of Ajtai [4] and Regev [182] — who introduced the Short Integer Solution (SIS) and Learning with Errors (LWE) problems respectively — lattice cryptography has shifted to base security properties on the hardness of these problems.

From these new problems, many interesting primitives have been obtained such as attribute-based encryption and fully homomorphic encryption. Especially fully homomorphic encryption, which to date can only be constructed from lattice cryptography.

2.4.1 Short Integer Solution

First proposed by Ajtai [4] in 1996 and later formalized by Micciancio and Regev [159] in 2004, the SIS problem is defined as follows:

Definition 2.4.1 ($\text{SIS}_{n,m,q,\beta}$). *Given a matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{x} \in \mathbb{Z}^m$ such that*

$$\mathbf{Ax} = \mathbf{0} \bmod q \text{ and } \|\mathbf{x}\| \leq \beta.$$

The problem can be thought of as finding a small non-trivial solution for the *homogeneous* system of linear modulo q equations defined by the rows of $\mathbf{A}$. The *inhomogeneous* case, denoted by ISIS, was later formalized by Gentry, Peikert and Vaikuntanathan [86].

Definition 2.4.2 ($\text{ISIS}_{n,m,q,\beta}$). *Given a matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and uniformly random vector $\mathbf{y} \in \mathbb{Z}_q^n$, find a non-zero vector $\mathbf{x} \in \mathbb{Z}^m$ such that*

$$\mathbf{Ax} = \mathbf{y} \bmod q \text{ and } \|\mathbf{x}\| \leq \beta.$$

The constraint on the norm of the solution is vital to it being hard; if any solution is admitted, Gaussian elimination would be able to solve SIS or ISIS easily. The norm considered by the problem is usually either the ℓ_2 or ℓ_∞ norms, if the latter is used, we denote that with SIS^∞ . n is the main parameter, dependent on the level of security desired. q, m are usually polynomial in n while the bound β chosen large enough so that there is at least one solution to the $\text{SIS}_{n,m,q,\beta}$ or $\text{ISIS}_{n,m,q,\beta}$ problems. This occurs once $\beta \geq \sqrt{[n \log q]}$ and $m \geq [n \log q]$ via a pigeonhole argument. With all other parameters fixed, the problems become easier as β increases and once $\beta \geq q$, the problems become trivial. For example, $\mathbf{x} = (q, 0, \dots, 0)^T$ would be a proper though trivial solution to the SIS problem. A solution for matrix $\mathbf{A}$ can be converted to one for an extended matrix $[\mathbf{A} \mid \mathbf{A}']$ for any $\mathbf{A}'$, simply append the original solution vector with zeroes, as it does not alter the norm of the new solution. This means that we can ignore any column $\mathbf{a}_i$ we like and the problem becomes easier as m increases.

The (I)SIS problem is related to finding a short vector, with norm less than β , in the following q -ary m -dimensional lattices,

$$\begin{aligned} \Lambda^\perp(\mathbf{A}) &= \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0} \in \mathbb{Z}_q^n\}, \\ \Lambda_{\mathbf{y}}^\perp(\mathbf{A}) &= \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{y} \in \mathbb{Z}_q^n\}. \end{aligned}$$

The hardness of these problems are tied to worst-case lattice problems GapSVP_γ and

SIVP_γ since Ajtai's seminal work [4] and subsequently improved in several papers [159, 86, 158]. The theorem presented below is based off the results of Gentry, Peikert and Vaikuntanathan [86]. Micciancio and Peikert [158] improves the bound on q to as small as $\beta \cdot n^\varepsilon$ for any constant $\varepsilon > 0$. However, the approximation factor γ of the related lattice problems are less straightforward to describe.

Theorem 2.4.3 (Hardness of $\text{SIS}_{n,m,q,\beta}$, $\text{ISIS}_{n,m,q,\beta}$). *For any m that is polynomial in n , any $\beta > 0$ and sufficiently large $q \geq \beta \cdot \text{poly}(n)$, solving any random instance of the $\text{SIS}_{n,m,q,\beta}$ or $\text{ISIS}_{n,m,q,\beta}$ problem with non-negligible probability is at least as hard as solving GapSVP_γ and SIVP_γ on arbitrary n -dimensional lattices with overwhelming probability with some approximation factor $\gamma = \beta \cdot \text{poly}(n)$.*

Since the lattice problems with approximation factor $\gamma = \text{poly}(n)$ are thought to be hard, the SIS and ISIS problems too are believed to be hard. Interestingly, the values of m and q do not play a role in the worst-case lattice problems that SIS, ISIS reduces to, except that a particular lower-bound on them is required. However, when designing cryptographic schemes using the hardness of these problems, the smallest possible β would be used for efficiency.

2.4.2 Learning with Errors

The other problem which lies at the heart of many cryptographic constructions from lattices is the LWE problem. Proposed by Regev [182] in 2005 as an extension to the Learning Parity with Noise problem (corresponding to $q = 2$), LWE comes in two flavors, *search* and *decision*.

Definition 2.4.4 ((D-)LWE $_{n,m,q,\chi}$). *Let $n \geq 1$, $q \geq 2$ and χ be a probability distribution over $\mathbb{Z}$, which induces a distribution over $\mathbb{Z}_q$ as well. For $\mathbf{s} \in \mathbb{Z}_q^n$, let $A_{\mathbf{s},\chi}$ be the distribution obtained by sampling matrices uniformly at random, $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times m})$, and error vector $\mathbf{e} \leftarrow \chi^n$ then outputting $(\mathbf{A}, \mathbf{b} = \mathbf{A}^T \mathbf{s} + \mathbf{e} \bmod q)$.*

- The search variant, denoted with $\text{LWE}_{n,m,q,\chi}$, requires finding $\mathbf{s}$ with non-negligible probability.
- The decision variant, denoted with $\text{D-LWE}_{n,m,q,\chi}$, requires distinguishing (with non-negligible advantage over random choice) $A_{\mathbf{s},\chi}$ from $\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.

The parameter m corresponds to the number of LWE samples given and the basic LWE problem has $m = 1$, with most uses in cryptography requiring $m \geq n \log q$. Without an

error contribution from χ , both variants of **LWE** are easy to solve by recovering $\mathbf{s}$ using Gaussian elimination; distinguishing $A_{\mathbf{s},\chi}$ from uniform is done by noticing that if $\mathbf{b}$ is uniform in $\mathbb{Z}_q$, then with high probability there will be no solution $\mathbf{s}$. Also, similar to **SIS**, the errors sampled from χ^m should be small, bounded by a number $B \ll q$ with overwhelming probability.

Definition 2.4.5 (*B*-bounded Distribution). *A distribution χ over $\mathbb{Z}$ is *B*-bounded if for any $e \leftarrow \chi$, then $\Pr[|e| > B] = \text{negl}(n)$.*

For suitably bounded distributions, χ , **LWE** can be considered an average-case BDD_γ problem on a certain family of q -ary m -dimensional lattices. For samples drawn from $A_{\mathbf{s},\chi}$, the vector $\mathbf{b}$ is close to exactly one vector in the following **LWE** lattice,

$$\Lambda(\mathbf{A}) := \{\mathbf{A}^T \mathbf{s} \bmod q \mid \mathbf{s} \in \mathbb{Z}_q^n\}.$$

Solving **LWE** would mean finding that particular lattice vector in $\Lambda(\mathbf{A})$ if the sample is from $A_{\mathbf{s},\chi}$; if $\mathbf{b}$ is drawn from $\mathcal{U}(\mathbb{Z}_q)$, then $\mathbf{b}$ is far from all points in $\Lambda(\mathbf{A})$ with very high probability. As mentioned in Definition 2.1.4, the **SIS** lattice is dual to the **LWE** lattice above up to a scaling factor q , which is why **SIS** and **LWE** are considered dual problems.

The hardness of the **LWE** was first studied by Regev [182] — who found that with appropriate χ , it is at least as hard as solving GapSVP_γ and SIVP_γ with a *quantum* algorithm.

Theorem 2.4.6 (Quantum Hardness of $\text{LWE}_{n,m,q,\chi}$). *For any m that is polynomial in n , modulus $q \leq 2^{\text{poly}(n)}$ and χ that is a (discrete) Gaussian distribution of parameter $s = \alpha q \geq 2\sqrt{n}$, with $0 < \alpha < 1$, solving $\text{LWE}_{n,m,q,\chi}$ is at least as hard as solving GapSVP_γ and SIVP_γ on arbitrary n -dimensional lattices for some $\gamma = \tilde{O}(n/\alpha)$ with a quantum algorithm.*

The classical hardness of $\text{LWE}_{n,m,q,\chi}$ was first established in 2009 by Peikert [170] for exponential modulus $q \geq 2^{n/2}$; solving **LWE** classically is at least as hard as solving GapSVP_γ , where $\gamma = \tilde{O}(n/\alpha)$ like Theorem 2.4.6. The reduction can be altered for polynomial moduli but the underlying lattice problem is a non-standard variant of **GapSVP**. Brakerski et al. [38] obtained a *dimension-modulus* trade-off for **LWE**, finding that its hardness for a particular error rate α is almost entirely determined by $n \log q$, so long as q is lower-bounded by some small polynomial. This result, when applied to Peikert's classical reduction, mean that solving $\text{LWE}_{n,m,q,\chi}$, where q is polynomial in n , is at least as hard as solving GapSVP_γ on a $\sqrt{n}$ -dimensional lattice.

Theorem 2.4.7 (Classical Hardness of $\text{LWE}_{n,m,q,\chi}$). *For $n, q \in \mathbb{Z}_{>0}$, any m that is polynomial in n , $0 < \alpha < 1$*

- *for $q \geq 2^{n/2}$ and $\chi = D_{\mathbb{Z}^n, s}$, where $s = \alpha q \geq 2\sqrt{n}$, solving $\text{LWE}_{n,m,q,\chi}$ is at least as hard as solving GapSVP_γ on arbitrary n -dimensional lattices for some $\gamma = \tilde{O}(n/\alpha)$;*
- *for q polynomial in n and $\chi = D_{\mathbb{Z}^n, s}$, where $s = \alpha q \geq 2\sqrt{n}$, solving $\text{LWE}_{n,m,q,\chi}$ is at least as hard as solving GapSVP_γ on arbitrary $\sqrt{n}$ -dimensional lattices for some $\gamma = \tilde{O}(n/\alpha)$.*

The hardness of $\text{D-LWE}_{n,m,q,\chi}$ was also studied by Regev [182]; for prime q polynomial in n , it was equivalent to $\text{LWE}_{n,m,q,\chi}$ up to a polynomial increase in m , through a simple classical reduction that generalized a result for $q = 2$ by Blum et al. [28]. The limitation on the modulus has been lifted through the years in various papers [170, 11, 156, 157, 38].

Theorem 2.4.8 (Search-Decision Equivalence of LWE). *For positive integers n, q and m , such that m is polynomial in n , and χ a discrete Gaussian over $\mathbb{Z}^n$ with parameter $s = \alpha q \geq 2\sqrt{n}$, solving $\text{D-LWE}_{n,m,q,\chi}$ is at least as hard as solving $\text{LWE}_{n,m,q,\chi}$.*

Rounding off this section, notice that in the theorems on the hardness of LWE, the value for m does not play a significant role. However, there are instances where it matters slightly, Micciancio and Peikert [158] showed that LWE is still hard for some non-Gaussian error distributions that might have small support but only if m is suitably bounded. Besides that, there is an attack by Arora-Ge [12] that can solve LWE in time and space about 2^{d^2} for errors from a domain of size d with sufficiently many LWE samples.

2.5 Ideal Lattices

Although lattices have strong security reductions and can be used to construct many advanced cryptographic primitives, it requires large matrices to represent the (q -ary) lattice. Generally, the public key of a cryptographic scheme for lattice cryptography comprises of at least one matrix. Since these matrices are random for all intents and purposes, they need $mn \log q = \tilde{O}(n^2)$ bits and may be unsuitable in practical scenarios. To reduce the size of the public keys, a class of algebraic structured lattices called *ideal* lattices are used and they only require $\tilde{O}(n)$ bits to represent. Furthermore, there are Fast Fourier Transform (FFT)-like techniques that can be used to compute operations on ideal lattices which are asymptotically faster than matrix operations, quasi-linear $\tilde{O}(n)$ compared to $\mathcal{O}(n^c)$ where $2 < c < 3$.

Both SIS and LWE have ideal lattice analogues: respectively, the Ring Short Integer Solution (RSIS) problem, first considered by Micciancio [155] in 2002, and Ring Learning with Errors (RLWE) problem, proposed by Lyubashevsky, Peikert and Regev [149] in early 2010 and concurrently in late 2009 by Stehlé et al. [195]. Stehlé et al. [195] considered a special case of RLWE where the rings have the form $R = \mathbb{Z}[X]/\langle X^n + 1 \rangle$ and n is a power-of-two, and only gave a reduction for the search variant.

2.5.1 Rings and Ideal Lattices

Ideal lattices are simply ideals in a ring R , under some fixed choice of geometric embedding. An ideal of a commutative ring R is an additive subgroup that is also closed under multiplication with any element in the ring. The multiplicative closure gives ideal lattices some symmetries that a general lattice would not have. q -ary ideal lattices would correspond to ideals in the ring $R_q = R/qR$.

For simplicity, this thesis focuses rings of the form $R = \mathbb{Z}[X]/\langle F(X) \rangle$ for $F(X) = \Phi_m(X)$, where $\Phi_m(X)$ is the m -th cyclotomic polynomial which has degree $n = \varphi(m)$ with $\varphi(\cdot)$ denoting the Euler totient function. Other rings can be used but the literature mainly considers the cyclotomic rings above. Elements in R can be represented canonically by their residues modulo $F(X)$, which correspond to integer polynomials of degree less than $\varphi(m)$. For a positive integer q , we define the ring $R_q = R/qR = \mathbb{Z}_q[X]/\langle F(X) \rangle$. We consider polynomials of degree less than n whose coefficients come from some set of canonical representatives of $\mathbb{Z}_q$, such as $\{0\} \cup [q-1]$ or $[-q/2, q/2) \cap \mathbb{Z}$, as the canonical representative for elements in R_q .

For RSIS and RLWE, there is a notion of “shortness” which depends on the choice of norm on R . That, in turn, depends on the choice of embedding of which two are most common: the *coefficient* embedding and *canonical* embedding.

Definition 2.5.1 (Coefficient Embedding of $R = \mathbb{Z}[X]/\langle F(X) \rangle$). *The coefficient embedding, ξ , maps each element $a \in R$ to the n -dimensional integer vector of coefficients of its canonical representative in $\mathbb{Z}[X]$.*

$$\begin{aligned} \xi : R &\longrightarrow \mathbb{Z}^n \\ a = a_0 + a_1x + \dots + a_{n-1}x^{n-1} &\longmapsto \xi(a) = (a_0 \ a_1 \ \dots \ a_{n-1}) \\ \text{some representative of } a \text{ in } \mathbb{Z}[X] & \end{aligned}$$

Then, the induced ℓ_p norm $\|a\|_p$ for this embedding is simply the ℓ_p norm of the resulting vector in $\mathbb{Z}^n$, $\|\xi(a)\|_p$, for any $p \in \mathbb{Z}_{>0} \cup \{\infty\}$.

Although it is very easy to understand, it is not canonical and depends on the choice of

representatives for elements in the ring R ; the representatives chosen could be polynomials that have degree greater than n . These issues necessitate constraining the form of $F(X)$ and only allows crude analysis. One important fact that affects analyzing the lattice and problems is that the norm of $a \cdot b \in R$ may be affected by a reduction modulo $F(X)$ and lose the connection to the norms of the inputs a and b .

Instead, a better choice for embedding is the canonical embedding from algebraic number theory. This maps each ring element $a \in R$ to the vector $(a(\alpha_i))_i \in \mathbb{C}^n$, where $\alpha_i \in \mathbb{C}$ are the n complex roots of $F(X)$. The reason this embedding is called canonical is that it is independent of the choice of representative for a . A good feature of the embedding is that sum and product of ring elements are translated to coordinate-wise sum and product of their embeddings; this yields simple and sharp bounds on the norms of ring elements under ring operations.

Definition 2.5.2 (Canonical Embedding of $R = \mathbb{Z}[X]/\langle F(X) \rangle$). *Let $F(X) = \prod_{i=1}^n (X - \alpha_i)$ for $\alpha_i \in \mathbb{C}$. The canonical embedding, σ , associates each element $a \in R$ to the vector $(a(\alpha_i))_i \in \mathbb{C}^n$, where the i -th coordinate is the evaluation of a at α_i .*

$$\begin{aligned} \sigma : R &\longrightarrow \mathbb{C}^n \\ a &\longmapsto (a(\alpha_1) \ a(\alpha_2) \ \cdots \ a(\alpha_n)). \end{aligned}$$

Then, the induced ℓ_p norm $\|a\|_p^c$ for this embedding is simply the ℓ_p norm of the resulting vector in $\mathbb{C}^n$, $\|\sigma(a)\|_p$, for any $p \in \mathbb{Z}_{>0} \cup \{\infty\}$.

2.5.2 Hardness of Problems on Ideal Lattices

The main downside is that the hardness of problems on ideal lattices are not as well-studied as general lattices. The complexities of the problems in Section 2.2 are slightly different in ideal lattices. One example is that GapSVP_γ , for small γ that are polynomial in the lattice dimension n and typical choices of rings, is easy on ideal lattices; algebraic symmetries force the minimum distance of an ideal to be within an easily computable range. Besides that, the same symmetries also make SVP_γ and SIVP_γ essentially equivalent, up to a small loss in the approximation factor; one short non-zero vector to be converted into n linearly independent ones with (almost) the same length from the symmetries present.

For approximation factors that are useful in cryptography, and with typical choices of rings, SVP_γ and SIVP_γ on ideal lattices seem to be hard in the worst case. Though there is additional structure to ideal lattices, no significant speed-up for these problems have been found to date compared to general lattices. Currently, the best known algorithm, quantum

or otherwise, for SVP_γ where γ is polynomial in n on ideal lattices take exponential $2^{\Omega(n)}$ time, for typical choices of rings. In spite of this evidence, it is important to temper expectations and remember that not much effort have been put into understanding ideal lattices from a computational perspective.

2.5.3 Ring Short Integer Solution

The RSIS problem has a similar set of parameters, R, m, q, β , and in general those parameters are the same as their counterparts in the SIS problem; the ring R replaces the dimension parameter n . One key difference though is that m is typically much smaller for RSIS, $m \approx \log q$ instead of $n \log q$ in SIS. This is due to the availability of an exponentially large $2^{\Omega(n)}$ number of short ring elements that can be used for each coordinate of the solution, as opposed to only several small integers in SIS. Note that the norm in the definition $\|\cdot\|$ can be any norm on the ring R .

Definition 2.5.3 ($\text{RSIS}_{R,m,q,\beta}, \text{IRIS}_{R,m,q,\beta}$). *Given a vector of ring elements $\mathbf{a} \xleftarrow{\$} R_q^m$ and a ring element $y \in R_q$ (for IRIS), find a non-zero vector $\mathbf{x} \in R^m$ such that*

$$\begin{aligned} (\text{homogeneous}) : \langle \mathbf{a}, \mathbf{x} \rangle &= 0 \bmod q \text{ and } \|\mathbf{x}\| \leq \beta; \\ (\text{inhomogeneous}) : \langle \mathbf{a}, \mathbf{x} \rangle &= y \bmod q \text{ and } \|\mathbf{x}\| \leq \beta. \end{aligned}$$

As evident from the definition, the main difference is that each row $\mathbf{a}_i$ in the matrix $\mathbf{A}$ in SIS is replaced with a single ring element a_i .

In SIS, the domain of the column vectors of the random matrix $\mathbf{A}$, $\mathbb{Z}_q^n$, is regarded as a $\mathbb{Z}$ -module; solutions are short *integer combinations* of the column vectors of $\mathbf{A}$ that sum to zero modulo q . On the other hand, in RSIS, the domain of the random elements is R_q and used as a R -module, with solutions being short *R combinations* of ring elements that sum to zero modulo q . Peikert [171, Section 4.3.2] gave a formalization of how to relate RSIS to SIS: as a special case of SIS with lattices containing some structure.

- With a fixed appropriate $\mathbb{Z}$ -basis of R , there is an additive group isomorphism between R and $\mathbb{Z}^n$, the input domains, and R_q and $\mathbb{Z}_q^n$, the output domains. This isomorphism also *approximately* preserves “shortness” of the elements.
- Left-multiplication by any fixed $a \in R_q$ is a $\mathbb{Z}$ -linear function from R to R_q and thus can be represented by a square matrix $\mathbf{A}_a$ which maps $\mathbb{Z}^n$ to $\mathbb{Z}_q^n$. This gives rise to the connection between a RSIS instance $\mathbf{a} = (a_1 \dots a_m)$ and the *structured* SIS instance $\mathbf{A} = [\mathbf{A}_{a_1} \mid \dots \mid \mathbf{A}_{a_m}]$.

The hardness results for RSIS emerged in stages: in the beginning, Micciancio [155] showed that the inhomogeneous RSIS problem (IRISIS) reduces to the worst-case hardness of certain problems on n -dimensional *cyclic* lattices, which correspond ideals in the ring $R = \mathbb{Z}[X]/\langle X^n - 1 \rangle$ with the coefficient embedding, with appropriate parameters. The hardness of RSIS for the same ring was left open but subsequently shown to be easy in two concurrent work by Peikert and Rosen [173] and Lyubashevsky and Micciancio [148]. It was due to the existence of zero divisors in R , meaning R is not an integral domain, which can be used to construct short answers.

However, that was not the end of the road. The same works [173] and [148] showed that by changing the ring R to suitable integral domains, RSIS is hard if we assume that SVP_γ is hard in the worst-case for ideal lattices in R for $\gamma = \beta \cdot \text{poly}(n)$. Among suitable rings are most cyclotomic rings $R = \mathbb{Z}[X]/\langle \Phi_m(X) \rangle$, for m a power-of-two or prime.

Finally, Peikert and Rosen, in a follow-up [174], generalized the previous results and showed that RSIS is at least as hard as SVP_γ on ideal lattices in $R = \mathcal{O}_K$, the ring of algebraic integers of any number field K . The reduction even improves the approximation factor to $\gamma = \mathcal{O}(\sqrt{\log n})$ for some families of number fields.

2.5.4 Ring Learning with Errors

As with SIS and RSIS, the parameters of LWE and RLWE are similar, with a ring R of degree n over the integers instead of $\mathbb{Z}^n$. Here, we present an RLWE variant — differing from the original definition proposed by Lyubashevsky, Peikert and Regev — that draws secrets and errors from R instead of a special fractional ideal $R^\vee$ that is dual to R . The error distribution χ can be considered a kind of discrete Gaussian in the *canonical embedding* of R , although the proper form needed to enable reductions of RLWE to the hardness of ideal lattice problems is more nuanced.

Definition 2.5.4 ((D-)RLWE $_{R,m,q,\chi}$). *For an $s \in R_q$ and error distribution χ over R , let $A_{s,\chi}$ be the distribution obtained by sampling a ring element $a \xleftarrow{\$} R_q$ and error $e \leftarrow \chi$ then outputting $(a, b = a \cdot s + e \bmod q)$.*

- The search variant, RLWE $_{R,m,q,\chi}$, requires finding s with non-negligible probability, given m independent samples from $A_{s,\chi}$.
- The decision variant, denoted with D-RLWE $_{R,m,q,\chi}$, requires distinguishing (with non-negligible advantage over random choice), whether m given independent samples $(a_i, b_i) \in R_q \times R_q$ are all drawn from $A_{s,\chi}$ or $\mathcal{U}(R_q \times R_q)$.

In a similar manner to the relationship between SIS and RSIS described by Peikert [171], by viewing a RLWE sample with a random $a \in R_q$ as replacing n LWE samples with random $\mathbf{a} \in \mathbb{Z}_q^n$, RLWE can be considered a special case of LWE with *structured* samples. The domain of the secret and random matrix in LWE is regarded as a $\mathbb{Z}$ -module and multiplication with the secret $\mathbf{s}$ is done with the inner product of vectors in $\mathbb{Z}_q^n$, which is $\mathbb{Z}$ -bilinear. In contrast, the random elements and secret in RLWE are considered as an R -module and their multiplication in R is R -bilinear.

Like $\text{LWE}_{n,m,q,\chi}$, the *quantum* hardness of $\text{RLWE}_{R,m,q,\chi}$ — where the parameter of χ is $\alpha q \geq 2 \cdot \omega(\sqrt{\log n})$ for $0 < \alpha < 1$ — is related to the *quantum* hardness of solving SVP_γ , although $\gamma = \tilde{O}(\sqrt{n}/\alpha)$. *Classical* search-to-decision reduction, from Lyubashevsky, Peikert and Regev [149] and extended through modulus-switching techniques from Langlois and Stehlé [133], also exists for RLWE and ties the hardness of $\text{D-RLWE}_{R,m,q,\chi}$ to $\text{RLWE}_{R,m,q,\chi}$ for R that are cyclotomic rings or Galois fields. This is recently improved by Peikert, Regev and Stephens-Davidowitz [172] in 2017 to work for any number field and modulus. Following that, Rosca, Stehlé and Wallet [186] unified several variants of (search and decision) RLWE by giving reductions between the original definition of RLWE (Dual-RLWE), the definition of RLWE in this thesis (Primal-RLWE) and Polynomial Learning with Errors (PLWE), which is characterized by a polynomial $F(X)$ and uses the ring $R = \mathbb{Z}[X]/\langle F(X) \rangle$. Their reductions for PLWE works for a family of polynomials that is exponentially large when considered as function of the degree.

We note that (to date) unlike LWE, there is no known *classical* reduction for RLWE to any ideal lattice problem. The reduction for LWE by Peikert [170] does not work with RLWE because GapSVP_γ on ideal lattices with polynomial γ is easy for the rings and parameters commonly used.

2.6 Cryptographic Primitives

In this section, we recall the definitions and security notions for several cryptographic primitives that will be used in later chapters. In some cases, namely for trapdoor functions, commitment schemes and zero-knowledge argument of knowledge, we also present a lattice-based construction of these primitives.

2.6.1 Public-Key Encryption

A public-key encryption scheme is a triple of algorithms, **KeyGen**, **Encrypt** and **Decrypt** with the following form:

- (pk, sk) **KeyGen** (1^λ) : Take a security parameter λ as input and output a public key pk and secret key sk .
- c **Encrypt** (pk, m) : Take a public key pk and plaintext m as inputs and output a ciphertext c that encrypts m .
- m **Decrypt** (sk, c) : Take the secret key sk and ciphertext c as inputs and output the plaintext m encrypted in c .

The encryption scheme is *correct* if for any key pair generated, encrypting a valid plaintext with the public key and then decrypting the resulting ciphertext with the associated secret key gives us the original plaintext with overwhelming probability.

Semantic security is the standard definition for encryption, which in essence captures the idea that no probabilistic polynomial time (PPT) adversary can learn anything about the plaintext encrypted in a ciphertext. First introduced by Goldwasser and Micali [94] in 1982, it was difficult to apply as they did not propose any method to prove the semantic security of an encryption scheme. Later, in 1984, Goldwasser and Micali [95] formulated another notion of security, ciphertext indistinguishability under chosen plaintext attack (IND-CPA), and showed that it was equivalent to semantic security.

Indistinguishability under chosen plaintext attack requires that an adversary cannot identify which of two plaintexts he supplied is encrypted within a given ciphertext better than randomly guessing. It is formalized as a game played between a challenger and adversary as follows.

Definition 2.6.1. *For a challenger, $\mathcal{C}$, and PPT adversary $\mathcal{A}$, the IND-CPA game, denoted $\mathcal{G}_{\mathcal{E}, \mathcal{A}}^{\text{IND-CPA}}$, with an encryption scheme $\mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ is described below. (See Figure 2.1 for an illustration)*

1. $\mathcal{C}$ generates an instance of $\mathcal{E}$ with key pair (pk, sk) and sends the public key pk to $\mathcal{A}$.
2. $\mathcal{A}$ chooses two plaintexts m_0 and m_1 from the plaintext space $\mathfrak{P}$ and sends them to $\mathcal{C}$.
3. $\mathcal{C}$ randomly chooses a bit $b \leftarrow \mathcal{U}(\{0, 1\})$, computes the ciphertext encrypting m_b , $c_b = \mathcal{E}.\text{Encrypt}(pk, m_b)$ and returns c_b to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b' and the game ends.

 Figure 2.1: IND-CPA Game, $\mathcal{G}_{\mathcal{E}, \mathcal{A}}^{\text{IND-CPA}}$.

The adversary wins if $b' = b$ and its advantage in $\mathcal{G}_{\mathcal{E}, \mathcal{A}}^{\text{IND-CPA}}$ is

$$\text{Adv}_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

The encryption scheme, $\mathcal{E}$ is then said to be λ -bit IND-CPA-secure if

$$\max_{\text{PPT } \mathcal{A}} \text{Adv}_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A}) < \text{negl}(\lambda),$$

for a security parameter λ .

2.6.2 Trapdoor Function

Roughly speaking, a trapdoor function is a function that is easy to evaluate and hard to invert, but whose hardness of inversion can be bypassed with some “trapdoor” information. Particularly, we recall a method to generate (I)SIS instances that are easy to solve given an associated trapdoor that was proposed by Micciancio and Peikert [157].

For any $k \in \mathbb{Z}_{>0}$, let $\mathbf{I}_k$ denote the identity matrix in $\mathbb{Z}^{k \times k}$. For positive integers n and q polynomial in n , let $w = n \lceil \log q \rceil$. We call $\mathbf{G} = \mathbf{I}_n \otimes (1, 2, \dots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{n \times w}$, a gadget matrix.

Definition 2.6.2 (Gadget Trapdoor). Let $m = w' + w$ for some $w' > w$. A trapdoor for the matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$ satisfying $\mathbf{A} \begin{bmatrix} \mathbf{R} \\ \mathbf{I}_w \end{bmatrix} = \mathbf{G}$. Alternatively, if

$\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$ for some $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times w'}$,

then $\mathbf{R}$ is a trapdoor for $\mathbf{A}$.

Micciancio and Peikert [157] described two algorithms, **TrapGen** and **SampleD**, such that **TrapGen** generates a matrix $\mathbf{A}$ with an associated trapdoor and **SampleD** allows one to solve (I)SIS instances defined on $\mathbf{A}$ using that trapdoor.

Lemma 2.6.3. *Let $n, q \in \mathbb{Z}_{>0}$ with q polynomial in n and $w = n\lceil \log q \rceil$. Also, let $m = w' + w$ for some $w' > w$. There exists a PPT algorithm, $(\mathbf{A}, \mathbf{R}) \leftarrow \text{TrapGen}(n, m, q)$, that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with an associated trapdoor $\mathbf{R} \in \mathbb{Z}^{w' \times w}$. The distribution of the matrix $\mathbf{A}$ output by **TrapGen** is statistically close to uniform.*

Additionally, for any $\mathbf{u} \in \mathbb{Z}_q^n$, and $s = \Omega(\sqrt{n \log q \log n})$, there is a PPT algorithm $\mathbf{r} \leftarrow \text{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$ that outputs a vector $\mathbf{r} \in \mathbb{Z}^m$ that is sampled from a distribution χ that is statistically close to $D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{A}), s}$.

2.6.3 (Trapdoor) Commitment

Commitment schemes are important ingredients in the construction of zero-knowledge proofs. It allows anyone to commit to a secret message with the **Commit** algorithm using the public parameters generated in **Setup** with the ability to reveal that message at a later stage. A prover would generate and release commitments to all the information beforehand and a verifier will specify what he wants to learn anytime after the commitments are released. The prover would then only reveal relevant information corresponding to the verifier's choices.

Formally, a commitment scheme is a four-tuple of algorithms **Setup**, **Commit**, **Open** and **Verify**.

- pk **Setup**(1^λ): Take a security parameter λ and output a public commitment key pk .
- C **Commit**($pk, M; R$): Take a commitment key pk , message M and randomness R as inputs and output a commitment C of M .
- π **Open**($pk, M; R$): Take a commitment key pk , message M and randomness R as inputs and output a decommitment π to M for the commitment, $C = \text{Commit}(pk, M; R)$.
- **Verify**(pk, M, C, π): Take a commitment key pk , message M , commitment C and decommitment π as inputs and accept if π proves that C is a valid commitment of M and reject otherwise.

Correctness of commitment schemes are straightforward, we require that the verification algorithm, $\text{Verify}(pk, M, \text{Commit}(pk, M; R), \text{Open}(pk, M; R))$, accepts with overwhelming probability. Commitment schemes have two security notions, *hiding* and *binding*.

- Hiding: No PPT adversary $\mathcal{A}$, given pk , can find two messages M_1, M_2 where he can distinguish a commitment $\text{Commit}(pk, M_1; R_1)$ of the first message from one of the other $\text{Commit}(pk, M_2; R_2)$.
- Binding: Given pk , no PPT adversary $\mathcal{A}$ can find a commitment C and two distinct message-opening pairs M, π, M', π' such that π is a valid decommitment of C to M while π' is a valid decommitment of C to M' with $M \neq M'$.

Kawachi-Tanaka-Xagawa Commitment Scheme. Kawachi, Tanaka and Xagawa [119] constructed a commitment scheme from lattices, which we denote with KTX, that comes in two flavours: the first commits to t bits and the other commits to arbitrary strings. Below, we describe the variant that commits to t -bit strings.

- $pk \leftarrow \text{KTX.Setup}(1^\lambda)$: Choose integers $n > \lambda$, prime $q = \tilde{O}(n)$ and $m = 2n \lceil \log q \rceil$ and output $pk = (\mathbf{A}, \mathbf{B})$ where $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times t})$ and $\mathbf{B} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times m})$.
- $\mathbf{c} \leftarrow \text{KTX.Commit}(pk, \mathbf{m}; \mathbf{r})$: For commitment key $pk = (\mathbf{A}, \mathbf{B})$, message $\mathbf{m} \in \{0, 1\}^t$ and randomness $\mathbf{r} \in \{0, 1\}^m$, compute and output $\mathbf{c} = \mathbf{A} \cdot \mathbf{m} + \mathbf{B} \cdot \mathbf{r} \in \mathbb{Z}_q^n$.
- $\pi \leftarrow \text{KTX.Open}(pk, \mathbf{m}; \mathbf{r})$: Output the randomness $\pi = \mathbf{r}$.
- $\text{KTX.Verify}(pk, \mathbf{m}, \mathbf{c}, \pi)$: Let $pk = (\mathbf{A}, \mathbf{B})$ and $\pi = \mathbf{r}$. Accept if $\mathbf{c} = \mathbf{A}\mathbf{m} + \mathbf{B}\mathbf{r}$ and $\|\mathbf{r}\|_\infty, \|\mathbf{m}\|_\infty \leq 1$.

The correctness of the KTX scheme is evident from the description and its security properties are summarized in the following theorems.

Theorem 2.6.4. *Valid commitments from the KTX scheme are statistically indistinguishable from random vectors in $\mathbb{Z}_q^n$, which means that the scheme is statistically hiding.*

Statistically hiding means that no adversary, even with unbounded computational power, can distinguish between commitments of two different messages. The proof of the theorem above follows from the leftover hash lemma by Regev [182, Section 5].

Theorem 2.6.5. *The KTX scheme is computationally binding assuming the hardness of the $\text{SIS}_{n,t+m,q,1}^\infty$ problem.*

The binding property of the KTX scheme is tied to the SIS^∞ problem. If two valid openings π, π' , to different messages $\mathbf{m} \neq \mathbf{m}'$, can be found for some commitment c , we have $\mathbf{A}\mathbf{m} + \mathbf{B}\pi = \mathbf{A}\mathbf{m}' + \mathbf{B}\pi'$ which means

$$\mathbf{A}(\mathbf{m} - \mathbf{m}') + \mathbf{B}(\pi - \pi') = \mathbf{0}.$$

Therefore, $(\mathbf{m} - \mathbf{m}', \pi - \pi') \in \{-1, 0, 1\}^{t+m}$ is a short solution, in the ℓ_∞ norm, to the SIS instance defined by the matrix $[\mathbf{A} \mid \mathbf{B}]$.

The KTX scheme was extended to a string commitment scheme, COM, via the Merkle-Damgård construction [152, 63]. Its security properties correspond to those of the KTX scheme with the same parameters.

Trapdoor Commitments. A trapdoor commitment scheme is a stronger variant of commitment schemes with some additional functionality. Specifically, there are two more algorithms — **Fake** and **Equiv** — and a modified setup **TSetup**. **TSetup** generates a commitment key and an associated trapdoor which is used by **Fake** and **Equiv** to generate fake commitments to nothing and decommit fake commitments to any desired message.

- (pk, sk) **TSetup**(1^λ): Take a security parameter λ as input and output a commitment key pk with associated secret key sk .
- C **Fake**($pk; R$): Take a commitment key pk and randomness R as inputs and output a fake commitment C .
- π **Equiv**($sk, M; R$): Take a secret key sk , message M and randomness R as inputs and output a decommitment π of the fake commitment $C = \text{Fake}(sk; R)$ to M .

Correctness of trapdoor commitment schemes are almost identical to regular commitment schemes, except for an additional requirement that verification of some fake commitment with a correct equivocation, $\text{Verify}(pk, M, \text{Fake}(sk; R), \text{Equiv}(sk, M; R))$, accepts with overwhelming probability. Other than hiding and binding, trapdoor commitments have one more property, *equivocation*. Catalano *et al.* [45] strengthened this definition and require that equivocation hold even when the secret key sk is known to the adversary for their construction of a primitive called trapdoor mercurial commitments.

- **Equivocation**: For any message m chosen by a PPT adversary $\mathcal{A}$, a real commitment tuple $(m, \text{Commit}(pk, M; R), \text{Open}(pk, M; R))$ is indistinguishable, over the randomness R , from the fake commitment tuple $(m, \text{Fake}(sk; R), \text{Equiv}(sk, M; R))$.

Note that the equivocation property implies that trapdoor commitment schemes satisfy the hiding property of standard commitment schemes as all commitments, regardless of the message M committed within, are indistinguishable to the distribution of fake commitments. Therefore, only the binding and equivocation properties have to be proven for the security of trapdoor commitment schemes.

2.6.4 Zero-Knowledge Argument System

A zero-knowledge argument system is an interactive protocol with three properties, completeness, soundness and zero-knowledge, where the soundness property holds against computationally bounded cheating provers. Let $R = \{(y, w)\}$, where y is a statement and w is a witness for that statement, be an NP relation. We call a two-party game $\langle \mathcal{P}, \mathcal{V} \rangle$, involving a prover $\mathcal{P}$ and verifier $\mathcal{V}$, an interactive argument system for the relation R with soundness error e if the following two conditions hold,

- Completeness: If $(y, w) \in R$, then $\mathcal{V}$ always accepts in the game $\langle \mathcal{P}(y, w), \mathcal{V}(y) \rangle$, where $\mathcal{P}$ has (y, w) and $\mathcal{V}$ has y as inputs.
- Soundness: If $(y, w) \notin R$, then for every PPT prover $\hat{\mathcal{P}}$, the probability that $\mathcal{V}$ accepts in the game $\langle \hat{\mathcal{P}}(y, w), \mathcal{V}(y) \rangle$ is at most e .

The additional zero-knowledge property requires that for any verifier $\hat{\mathcal{V}}$, there exists a PPT simulator $\mathcal{S}(y)$ that generates a simulated transcript that is indistinguishable from a real interaction between $\mathcal{P}(y, w)$ and $\hat{\mathcal{V}}(y)$. If the distribution of the transcripts are statistically close, then the zero-knowledge argument system is a statistical zero-knowledge argument system.

A related notion is the argument of knowledge, which additionally has the witness-extended emulation property. For three move protocols of the form “commit-challenge-respond”, witness-extended emulation is implied by the special soundness property, first defined by Cramer, Damgård and Schoenmakers [59]. Special soundness mandates the existence of a PPT extraction algorithm, which takes a set of valid transcripts, one for every possible value of the “challenge” to the same “commitment”, and returns a witness w' such that $(y, w') \in R$.

Lattice-based Zero-Knowledge Argument of Knowledge. There are two main families of zero-knowledge arguments of knowledge (ZKAoK) in modern lattice cryptography, the first originated from the work of Lyubashevsky [146, 147], uses “rejection sampling” and yields efficient proofs of knowledge in the ideal lattice setting. The main

relations that are supported by this family of protocols are knowledge of small secret vectors [24, 16, 60, 66] and linear and multiplicative relations [25, 17] among committed values. Protocols from this family suffer from a few drawbacks, with “rejection sampling”, there is a very slim possibility that even honest provers fail to convince verifiers, meaning the completeness property is not perfect. Moreover, proofs of knowledge of vectors with some norm bound β have a factor g called the “soundness slack” by Baum *et al.* [17] and Cramer *et al.* [60]. Extraction algorithm for these protocols are only able to produce witnesses of norm bound $g\beta$ for some $g > 1$. If extracted witnesses are to be used in security proofs for solving (R)SIS instances, then a slightly stronger assumption, where the norm bound of the (R)SIS problem is $g\beta$ rather than just β , is required.

The second, Stern-like protocols originating from the work of Ling, Nguyen, Stehlé and Wang [140], employs “decomposition-extension” with lattice-based analogues [119] of a protocol from Stern [196]. These protocols are less efficient than the first family due to a constant soundness error; to achieve negligible (in the security parameter) soundness error, protocols from this family have to be repeated about $\omega(\log \lambda)$ times. However, they have perfect completeness and support a wide range of relations from lattices, particularly when witnesses are not only small or binary, but have specific patterns in their coordinates. Besides that, these Stern-like protocols do not have soundness slack; extraction algorithms for this family output witnesses share all the properties expected of valid witnesses. These properties have been leveraged to design several advanced cryptographic primitives from lattices, such as group signatures [141, 138, 136, 142] and group encryption [137].

An Abstract Stern-like ZKAoK Protocol We revisit an abstraction of the lattice variant of Stern’s protocol by Libert *et al.* [136]. For positive integers n, q, ℓ , with $2 \leq n, q \leq \ell$, let **VALID** denote a subset of $\{0, 1\}^\ell$. Let $\mathcal{S}$ be a finite set where every element $\phi \in \mathcal{S}$ is tied to a permutation Γ_ϕ of ℓ elements such that

- $\mathbf{w} \in \text{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \text{VALID}$;
- If $\mathbf{w} \in \text{VALID}$ and ϕ is sampled from $\mathcal{U}(\mathcal{S})$, then the distribution of $\Gamma_\phi(\mathbf{w})$ over **VALID** is uniform.

The protocol is a ZKAoK of the following abstract relation, which is made up of a statement consisting of a matrix $\mathbf{M}$ and vector $\mathbf{v}$ with a witness $\mathbf{w}$, $R_{abs} = \{((\mathbf{M}, \mathbf{v}), \mathbf{w}) \in (\mathbb{Z}_q^{n \times \ell} \times \mathbb{Z}_q^n) \times \text{VALID} \mid \mathbf{M}\mathbf{w} = \mathbf{v} \bmod q\}$. The original protocol by Stern uses **VALID** = $\{\mathbf{w} \in \{0, 1\}^\ell \mid \text{hw}(\mathbf{w}) = k\}$, with $\text{hw}(\cdot)$ meaning the Hamming weight and $k < \ell$ for some

 Figure 2.2: Abstract Stern-like ZKAoK Protocol by Libert *et al.* [136]

given k . The set $\mathcal{S}$ is $\mathcal{S}_\ell$, the symmetric group on ℓ elements and $\Gamma_\phi(\mathbf{w})$ is simply the permuted vector $\phi(\mathbf{w})$ for $\phi \in \mathcal{S}_\ell$.

The conditions listed above are crucial for proving that $\mathbf{w} \in \text{VALID}$ in zero-knowledge. $\mathbf{w} \in \text{VALID}$ is proved in zero-knowledge by having the prover $\mathcal{P}$ sample $\phi \leftarrow \mathcal{U}(\mathcal{S})$ and verifier $\mathcal{V}$ check that $\Gamma_\phi(\mathbf{w}) \in \text{VALID}$, the randomness of ϕ ensures that no additional information besides $\Gamma_\phi(\mathbf{w})$ is revealed to $\mathcal{V}$. The other part of the relation, $\mathbf{M}\mathbf{w} = \mathbf{v} \bmod q$, is proven in zero-knowledge by sampling a mask for $\mathbf{w}$, $\mathbf{r}_w \leftarrow \mathbb{Z}_q^\ell$, and convincing $\mathcal{V}$ that $\mathbf{M}(\mathbf{w} + \mathbf{r}_w) = \mathbf{M}\mathbf{r}_w + \mathbf{v} \bmod q$ instead. Figure 2.2 details the interaction between $\mathcal{P}$ and $\mathcal{V}$; a statistically hiding and computationally binding string commitment scheme, like COM by Kawachi *et al.* [119], is used to obtain a statistical ZKAoK. We describe the steps of the participants in the protocol with the following three algorithms, $\Pi = (\text{Start}, \text{Finish}, \text{Check})$, assuming there is a common reference string, crs , that is a commitment key from the COM string commitment scheme.

- $\mathbf{a} \leftarrow \text{Start}(crs, \mathbf{M}, \mathbf{w})$: Sample a mask $\mathbf{r}_w \leftarrow \mathcal{U}(\mathbb{Z}_q^\ell)$, permutation $\phi \leftarrow \mathcal{U}(\mathcal{S})$ and randomness $\rho_1, \rho_2, \rho_3 \leftarrow \mathcal{U}(\{0, 1\}^m)$. Then, compute commitments $C_1 = \text{COM.Commit}(crs, (\phi, \mathbf{M}\mathbf{r}_w \bmod q); \rho_1)$, $C_2 = \text{COM.Commit}(crs, \Gamma_\phi(\mathbf{r}_w); \rho_2)$ and $C_3 = \text{COM.Commit}(crs, \Gamma_\phi(\mathbf{w} + \mathbf{r}_w \bmod q); \rho_3)$. Output $\mathbf{a} = (C_1, C_2, C_3)$.
- $\text{rsp} \leftarrow \text{Finish}(crs, \mathbf{M}, \mathbf{w}, \mathbf{a}, ch)$: Let $\mathbf{a} = (C_1, C_2, C_3)$. Compute a response, rsp , based on the challenge ch issued by $\mathcal{V}$:
 - ◊ $ch = 1$: Let $\mathbf{t}_w = \Gamma_\phi(\mathbf{w})$ and $\mathbf{t}_r = \Gamma_\phi(\mathbf{r}_w)$, compute openings of commitments C_2 and C_3 , which are given by $\pi_2 = \text{COM.Open}(crs, \mathbf{t}_w; \rho_2)$ and $\pi_3 = \text{COM.Open}(crs, (\mathbf{t}_w + \mathbf{t}_r \bmod q); \rho_3)$ respectively. Set $\text{rsp} = (\mathbf{t}_w, \mathbf{t}_r, \pi_2, \pi_3)$.
 - ◊ $ch = 2$: Let $\phi_2 = \phi$ and $\mathbf{w}_2 = \mathbf{w} + \mathbf{r}_w \bmod q$, compute openings of C_1 and C_3 , given by $\pi_1 = \text{COM.Open}(crs, (\phi_2, \mathbf{M}\mathbf{w}_2 - \mathbf{v} \bmod q); \rho_1)$ and $\pi_3 =$

- $\text{COM.Open}(crs, \Gamma_{\phi_2}(\mathbf{w}_2); \rho_3)$ respectively. Set $\mathbf{rsp} = (\phi_2, \mathbf{w}_2, \pi_1, \pi_3)$.
- ◇ $\text{ch} = 3$: Let $\phi_3 = \phi$ and $\mathbf{w}_3 = \mathbf{r}_2$, compute openings of C_1 and C_3 , $\pi_1 = \text{COM.Open}(crs, (\phi_3, \mathbf{M}\mathbf{w}_3); \rho_1)$ and $\pi_3 = \text{COM.Open}(crs, \Gamma_{\phi_3}(\mathbf{w}_3); \rho_3)$ respectively. Set $\mathbf{rsp} = (\phi_2, \mathbf{w}_2, \pi_1, \pi_2)$.
 - $\text{Check}(crs, \mathbf{M}, \mathbf{a}, \text{ch}, \mathbf{rsp})$: Let $\mathbf{a} = (C_1, C_2, C_3)$. Based on the challenge ch , accept if
 - ◇ $\text{ch} = 1$: Let $\mathbf{rsp} = (\mathbf{t}_w, \mathbf{t}_r, \pi_2, \pi_3)$ and check that $\mathbf{t}_w \in \text{VALID}$ and verifications of C_2 and C_3 , $\text{COM.Verify}(crs, \mathbf{t}_r, C_2, \pi_2)$ and $\text{COM.Verify}(crs, (\mathbf{t}_r + \mathbf{t}_w), C_3, \pi_3)$, accept.
 - ◇ $\text{ch} = 2$: Let $\mathbf{rsp} = (\phi_2, \mathbf{w}_2, \pi_1, \pi_3)$ and check that verifications of C_1 and C_3 , which are given by $\text{COM.Verify}(crs, (\phi_2, \mathbf{M}\mathbf{w}_2 - \mathbf{v} \bmod q), C_1, \pi_1)$ and $\text{COM.Verify}(crs, \Gamma_{\phi_2}(\mathbf{w}_2), C_2, \pi_2)$ respectively, accept.
 - ◇ $\text{ch} = 3$: Let $\mathbf{rsp} = (\phi_2, \mathbf{w}_2, \pi_1, \pi_2)$ and verify that C_1 and C_2 are correct with $\text{COM.Verify}(crs, (\phi_3, \mathbf{M}\mathbf{w}_3), C_1, \pi_1)$ and C_2 $\text{COM.Verify}(crs, \Gamma_{\phi_3}(\mathbf{w}_3), C_3, \pi_3)$ respectively.

The properties — completeness, soundness, zero-knowledge and witness-extended extraction — of the ZKAoK protocol are summed up in the following theorem.

Theorem 2.6.6 ([136]). *Assuming that the string commitment scheme COM is statistically hiding and computationally binding, then the protocol in Figure 2.2 with the algorithms Π is a statistical ZKAoK with perfect completeness, soundness error of $2/3$ and communication cost $\mathcal{O}(\ell \log q)$. Specifically,*

- *There exists an efficient simulator that, when given $(\mathbf{M}, \mathbf{v})$, the distribution of its output transcripts is statistically close to the distribution of the transcripts produced by a real interaction between the real prover and some verifier.*
- *There exists an efficient knowledge extractor that, when given $\mathbf{a}$ and three valid responses $\mathbf{rsp}_1, \mathbf{rsp}_2, \mathbf{rsp}_3$ for each value of $\text{ch} \in \{1, 2, 3\}$, outputs a valid witness $\mathbf{w}' \in \text{VALID}$ such that $\mathbf{M}\mathbf{w}' = \mathbf{v} \bmod q$.*

We give a brief overview of the proof of the theorem and refer to [136] for the complete proof. Completeness and communication cost are straightforward from the description of the protocol and its algorithms. For zero-knowledge, the simulator will guess a challenge $\bar{\text{ch}}$ and construct valid responses for the other two possible challenges. If the verifier's chosen challenge coincides with $\bar{\text{ch}}$, then the simulator will simply output $\perp$ and abort the protocol. This means that the protocol will not complete with probability $1/3$ and

the simulator successfully poses as an honest prover with probability $2/3$. implying that the soundness error is $2/3$. Since COM is statistically hiding, when the simulator does not abort prematurely, the distribution of the simulated transcript — which consists of commitments and random vectors — is statistically close to that from an honest prover in a real interaction.

Extracting a valid witness when given a complete set of valid responses — $\mathbf{rsp}_1 = (\mathbf{t}_w, \mathbf{t}_r, \pi_2, \pi_3)$, $\mathbf{rsp}_2 = (\phi_2, \mathbf{w}_2, \pi_1, \pi_3)$ and $\mathbf{rsp}_3 = (\phi_2, \mathbf{w}_2, \pi_1, \pi_2)$ — for some $\mathbf{a}$ from Start is fairly straightforward. As COM is computationally binding, we can assume that $\mathbf{t}_w \in \text{VALID}$ with $\phi_2 = \phi_3$, $\mathbf{t}_r = \Gamma_{\phi_3}(\mathbf{w}_3)$, $\mathbf{t}_w + \mathbf{t}_r \equiv \Gamma_{\phi_2}(\mathbf{w}_2) \bmod q$ and $\mathbf{M}\mathbf{w}_2 - \mathbf{v} \equiv \mathbf{M}\mathbf{w}_3 \bmod q$. Since $\mathbf{t}_w \in \text{VALID}$, we have $\mathbf{w}' = (\Gamma_{\phi_2})^{-1}(\mathbf{t}_w) \in \text{VALID}$. In addition, $\Gamma_{\phi_2}(\mathbf{w}') + \Gamma_{\phi_2}(\mathbf{w}_3) + \Gamma_{\phi_2}(\mathbf{w}_2) \bmod q$, implying that $\mathbf{w}' + \mathbf{w}_3 \equiv \mathbf{w}_2 \bmod q$. Finally, with $\mathbf{M}\mathbf{w}' + \mathbf{M}\mathbf{w}_3 \equiv \mathbf{M}\mathbf{w}_2 \bmod q$, we have $\mathbf{M}\mathbf{w}' \equiv \mathbf{v} \bmod q$ which means $\mathbf{w}'$ is a valid witness for the relation R_{abs} .

Part I

Private Database Queries with Fully Homomorphic Encryption

Chapter 3

Preliminaries

3.1 Fully Homomorphic Encryption

3.1.1 A Brief Introduction

First proposed by Rivest, Adleman and Dertouzos [183] in 1978 as privacy homomorphism, fully homomorphic encryption (FHE) is a public-key encryption scheme, as defined in Section 2.6.1, that additionally supports evaluation of *arbitrary* functions on the encrypted inputs. It remained an open problem until 2009 when it was finally realized by Gentry [82] based (directly) on hard problems on ideal lattices. Not only was the problem solved, a blueprint was introduced that yields an FHE scheme in three steps:

1. Design a somewhat homomorphic encryption (SHE) scheme ($\mathcal{E}$) that evaluate circuits from some set of Boolean circuits $\mathcal{C}$, comprising of AND and XOR gates, with a set of encrypted ciphertexts as inputs.
2. Squash the decryption circuit of the scheme $\mathcal{E}.\text{Dec}$, so that the augmented circuits $\mathcal{E}.\text{Dec}^{\text{AND}}, \mathcal{E}.\text{Dec}^{\text{XOR}} \in \mathcal{C}$, where $\mathcal{E}.\text{Dec}^*$ is the circuit with two encrypted inputs, $\bar{x}, \bar{y}$ and computes $\bar{x} \star \bar{y}$ and sends that through $\mathcal{E}.\text{Dec}$ for $\star \in \{\text{AND}, \text{XOR}\}$. (*Squashing*)
3. Evaluate any circuit C on encrypted inputs by replacing the gates AND and XOR with $\mathcal{E}.\text{Dec}^{\text{AND}}$ and $\mathcal{E}.\text{Dec}^{\text{XOR}}$ respectively.

This simple blueprint hinges on one key idea, called *bootstrapping*, which informally is the following.

An SHE scheme ($\mathcal{E}$) — that can evaluate $\mathcal{E}.\text{Dec}^*$, its decryption circuit *plus one additional gate* $\star \in \{\text{AND}, \text{XOR}\}$ — can evaluate arbitrary circuits with an encryption of the bits of the secret key as an additional input.

From here, there are two branches. The first yields a leveled FHE scheme, capable of evaluating depth- L circuits and the second yields pure FHE that can evaluate circuits of unbounded depth. Here, the depth of the circuit is the largest number of multiplication (AND) gates in a path from any input wire to the output wire. In the first instance, a set of $L + 1$ key pairs $\{(pk_i, sk_i)\}_{i=0}^L$ are used and we publish encryptions of each secret key (from $sk_i \in [L]$) under the public key of the previous one pk_{i-1} . Then, each time bootstrapping is used, we move from ciphertexts from key pair (pk_i, sk_i) to (pk_{i-1}, sk_{i-1}) until we reach the end at (pk_0, sk_0) . In the latter case, the bits of the secret key is encrypted under its own public key, allowing the output of bootstrapping to be used and then bootstrapped again in a loop. The trick to FHE, leveled or pure, is that the computation need not be achieved in one go, just the decryption circuit and at least one more gate is sufficient. This is because we can compose this as needed for arbitrary circuits.

Unfortunately, there are some complications to the security of Gentry’s blueprint. Besides the problem required for the semantic security of the base SHE scheme, additional hardness assumptions are required due to some of the techniques used in the construction. First of all, Step 2 of the blueprint, squashing, necessitates the assumption that a problem called the *sparse subset sum problem* (SSSP) is hard to prevent the secret key from being recovered from the additional information revealed. To achieve arbitrary computation on encrypted data, the secret key must be encrypted under its corresponding public key. The notion of circular security captures this concept for bootstrapping. Semantic security of the complete (pure) FHE scheme therefore requires assuming

1. the SHE scheme has semantic security, i.e. the associated problem which it reduces to is hard;
2. the SHE scheme is circular-secure, i.e. encryptions of the bits of the secret key under its public key do not compromise semantic security;
3. SSSP is hard so the squashed SHE scheme has semantic security.

After Gentry’s work [82], there has been an explosion in the number of FHE schemes — several types ranging from integer-based constructions [203, 52, 58, 165, 55, 22] to (ideal) lattice-based schemes, NTRU-type [145, 35] and (R)LWE-type [40, 39, 37, 36, 74, 87, 70, 105, 56, 57]. However, they follow the same basic blueprint, with minor variations in some cases, laid out by Gentry.

In this thesis, the focus is on RLWE-based FHE schemes, which are among the most widely-used FHE schemes currently. Some of the most efficient FHE schemes to date are RLWE-based and several open-source implementations of schemes from this family are available [102, 132, 177].

3.1.2 Definition and Security Notions for FHE

Now, we recall the definitions of FHE, given by the following four algorithms **KeyGen**, **Encrypt**, **Decrypt** and **Eval**. The first three algorithms are almost entirely the same as their counterparts from public-key encryption, except in key generation: we take an additional input L and possibly have an evaluation key ek besides the public and secret keys required in standard public-key encryption schemes. In some cases, **Eval** might be subdivided into two algorithms, **EvalAdd** and **EvalMult** to better present the two operations in the plaintext space. The plaintext space is denoted by $\mathcal{P}$ and the secret key space by $\mathcal{K}$.

- $(pk, ek, sk) \leftarrow \text{KeyGen}(1^\lambda, L)$: Taking as input the security parameter λ and in some schemes an upper bound on multiplicative depth L , output the public key pk , evaluation key (if required) ek and secret key sk .
- $c \leftarrow \text{Encrypt}(pk, m)$: Taking as inputs the public key pk and a plaintext m from plaintext space $\mathcal{P}$, output a ciphertext c that encrypts m .
- $m \leftarrow \text{Decrypt}(sk, c)$: Taking as inputs the secret key sk and ciphertext c , output the plaintext m encrypted in c .
- $c^* \leftarrow \text{Eval}(ek, C, c_1, \dots, c_\ell)$: Taking as inputs the evaluation key ek , circuit C and ℓ ciphertexts $\{c_i\}_{i=1}^\ell$, output a ciphertext c^* that will decrypt to $m^* = C(m_1, \dots, m_\ell)$ with sk , where m^* is the result of running the circuit C on the set of plaintexts $m_1, \dots, m_\ell$.
 - ◊ The **Eval** algorithm may be expressed as an evaluation a circuit C' on input $\{c_i\}_{i=1}^\ell$, that consists of “homomorphic gates” **EvalAdd** and **EvalMult** replacing the addition and multiplication gates of C respectively.

SHE/FHE schemes come in several different flavors, based on the set of circuits, $\mathcal{C}$, that **Eval** can evaluate:

1. Somewhat Homomorphic Encryption: $\mathcal{C}$ is a set of circuits of bounded multiplicative depth on encrypted inputs.
2. Leveled Fully Homomorphic Encryption: $\mathcal{C}$ is the set of depth- L circuits on encrypted inputs, with L a parameter of the FHE scheme.
3. (Pure) Fully Homomorphic Encryption: $\mathcal{C}$ is the set of circuits of unbounded multiplicative depth on encrypted inputs.

In this thesis, we use leveled fully homomorphic encryption to design private database query protocols in the later chapters. We drop “leveled” from here on and FHE would refer to leveled FHE unless stated otherwise.

The notion of correctness is slightly different from traditional encryption schemes, due to FHE’s homomorphic properties. An FHE scheme is said to be *correct* if fresh ciphertexts $c \leftarrow \text{Encrypt}(pk, m)$ and ciphertexts $c^* \leftarrow \text{Eval}(ek, C, c_1, \dots, c_\ell)$, for all $C \in \mathcal{C}$ and fresh ciphertexts c_i , are decrypted correctly (to m and $C(m_1, \dots, m_\ell)$ respectively) with overwhelming probability.

There are two main security notions, semantic security and circular security that are necessary to achieve pure FHE. Semantic security for FHE is the same as the definition for public-key encryption schemes since FHE is also a public-key encryption scheme.

The other security notion required is circular security, first considered by Camenish and Lysyanskaya [41]. A related notion is KDM security, first studied by Black, Rogaway and Shrimpton [27] and concurrent with [41]. Subsequently, Halevi and Krawczyk [100] extended the definitions to encompass deterministic symmetric encryption. The notion of KDM security for public key encryption was enhanced by Boneh *et al.* [32] to consider a fixed family of sets of functions $\mathcal{F}$, parameterized by the security parameter, bringing it in line with Halevi and Krawczyk [100]. For each function $f \in \mathcal{F}$, $|f(z)|$ must be the same for all inputs $z \in \mathcal{K}^n$, meaning the output length of f is independent of the input.

Here, we give the formal definition of KDM-security, proposed by Boneh *et al.* [32].

Definition 3.1.1. *For a challenger $\mathcal{C}$ and PPT adversary $\mathcal{A}$, the $\mathcal{F}$ -KDM game, denoted $\mathcal{G}_{\mathcal{E}, \mathcal{A}}^{\mathcal{F}\text{-KDM}}$, for an encryption scheme $\mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ and set of functions, $\mathcal{F} = \{f : \mathcal{K}^n \rightarrow \mathcal{P}\}$ is described below.*

1. $\mathcal{C}$ chooses a random bit $b \leftarrow \mathcal{U}(\{0, 1\})$ and generates n instances of $\mathcal{E}$ with key pairs $(pk_1, sk_1) \dots (pk_n, sk_n)$ and sends the public keys $(pk_1, \dots, pk_n)$ to $\mathcal{A}$.
2. $\mathcal{A}$ issues queries of the form (i, f) for $i \in [n]$ and $f \in \mathcal{F}$ and $\mathcal{C}$ responds by setting $y \leftarrow f(sk_1, \dots, sk_n)$ and

$$c \leftarrow \begin{cases} \mathcal{E}.\text{Encrypt}(pk_i, y), & \text{if } b = 0; \\ \mathcal{E}.\text{Encrypt}(pk_i, 0^{|y|}), & \text{if } b = 1; \end{cases}$$

and sends c to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a bit b' and the game ends.

The adversary $\mathcal{A}$ wins the game if $b' = b$ and its advantage in $\mathcal{G}_{\mathcal{E}, \mathcal{A}}^{\mathcal{F}\text{-KDM}}$ is defined as $\text{Adv}_{\mathcal{E}}^{\mathcal{F}\text{-KDM}}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|$.

The encryption scheme, $\mathcal{E}$ is then said to be λ -bit $\mathcal{F}$ -KDM-secure, or KDM-secure with respect to $\mathcal{F}$, if

$$\max_{\text{PPT } \mathcal{A}} \text{Adv}_{\mathcal{E}}^{\mathcal{F}\text{-KDM}}(\mathcal{A}) < \text{negl}(\lambda),$$

for a security parameter λ .

The standard definition of circular security can be obtained from the KDM definition by choosing $\mathcal{F}$ that contains

1. all $|\mathcal{P}|$ constant functions, i.e., mapping all inputs in $\mathcal{K}^n$ to some constant $p \in \mathcal{P}$;
2. all n selector functions $f_i(x_1, \dots, x_n) = x_i$ for $i \in [n]$.

A bootstrapped FHE scheme to be secure, we require that for secret keys of ℓ bits, the functions $g_{i,j}(x_1, \dots, x_n) = x_{i,j}$ for $i \in [n]$ and $j \in [\ell]$, where $x_{i,j}$ is the j -th bit of the i -th input, are in $\mathcal{F}$. As mentioned earlier, all FHE schemes to date could not be proven circular-secure so circular security must be assumed.

3.2 The Brakerski-Gentry-Vaikuntanathan FHE Scheme

In this section, we describe the RLWE-based FHE scheme by Brakerski, Gentry and Vaikuntanathan (BGV) [37] and outline the batching techniques for RLWE-based schemes first analyzed by Smart and Vercauteren [192].

This scheme realizes leveled FHE without bootstrapping, unlike those preceding it, by leveraging the dimension and modulus switching techniques introduced by Brakerski and Vaikuntanathan [39]. Their crucial innovation was using a ladder of decreasing moduli, switching after every multiplication, to keep limit noise growth. The number of moduli in the ladder, L , would be set such that circuits of multiplicative depth at most L can be evaluated without bootstrapping.

The BGV FHE scheme with batching is the basis of the implementation of the protocols presented in the subsequent chapters, although we note that any other FHE scheme that supports finite extension fields as a plaintext space can technically run the algorithms. For the algorithms to be efficient, depth-free Frobenius map evaluations are necessary and further optimization is possible if multiple plaintexts can be packed into

a single ciphertext. One such example is the RLWE-based BFV scheme by Brakerski [36] and Fan and Vercauteren [74], together with the Smart-Vercauteren batching techniques.

3.2.1 The Scheme

First, we introduce two functions and an important lemma on bit representations first used by Brakerski and Vaikuntanathan [39].

- **BitRep**(x, q): Taking $x \in R_q$ and q as inputs, let $x = \sum_{i=0}^{\lfloor \log q \rfloor} 2^i \cdot x_i$ and output $(x_0, x_1, \dots, x_{\lfloor \log q \rfloor}) \in R_2^{\lfloor \log q \rfloor + 1}$, the bit representation of x .
- **PowOf2**(x, q): Taking $x \in R_q$ and q as inputs, output $(x, 2x, \dots, 2^{\lfloor \log q \rfloor} x) \in R_q^{\lfloor \log q \rfloor + 1}$.

These functions extend to vectors in R_q^n by applying them to each component separately.

Lemma 3.2.1. *For $\mathbf{c}, \mathbf{s} \in R_q^n$, we have*

$$\langle \text{BitRep}(\mathbf{c}, q), \text{PowOf2}(\mathbf{s}, q) \rangle = \langle \mathbf{c}, \mathbf{s} \rangle \bmod q.$$

Besides bit representations, the modulus switching technique uses a scaling function, **Scale** which we define below.

Definition 3.2.2. *For a vector $\mathbf{x} \in R_q^2$ and integers p, r such that $p > r$, define $\mathbf{x}' \leftarrow \text{Scale}(\mathbf{x}, q, p, r)$ as the R -vector closest to $(p/q) \cdot \mathbf{x}$ satisfying $\mathbf{x}' \equiv \mathbf{x} \bmod r$.*

Now, we give a description of the BGV scheme. The presentation differs slightly from the generic definition given in Section 3.1: **Eval** is split into two algorithms **EvalAdd** and **EvalMult** for addition and multiplication gates respectively and three new algorithms, **Refresh**, **SwKeyGen** and **SwKey** that are introduced for BGV-specific functionality.

Homomorphic addition of two encrypted plaintexts with **EvalAdd** is accomplished by simply adding two ciphertexts together component-wise. Homomorphic multiplication with **EvalMult**, on the other hand, is more complicated, involving a tensor product of two ciphertexts, and requires the dimension-modulus switching technique for the scheme to be efficient.

SwKeyGen and **SwKey** are the key switching matrix generation and key switching algorithms respectively. These are used to ensure that the output from **EvalMult** has the same form as input ciphertexts. **Refresh** is the complete dimension-modulus switching technique, scaling ciphertexts after a tensor product to control noise growth and then key switching to convert the tensored ciphertext back to the form of regular ciphertexts.

3. PRELIMINARIES

In the descriptions below, the number of samples m is omitted as it does not play a significant role in the hardness of the RLWE problem.

- $(pk, ek, sk) \leftarrow \text{BGV.KeyGen}(1^\lambda, L)$: Choose a modulus $q_L = \prod_{j=0}^L p_j$, integers $N = 3\lceil \log q_L \rceil$ and m' such that $n = \varphi(m')$ that define $R = \mathbb{Z}[X]/\langle \Phi_{m'}(X) \rangle$. Then, choose a distribution χ over R and for i from L to 0:
 1. Generate a key-pair $(\mathbf{A}_i, \mathbf{s}_i)$ with the modulus $q_i = \prod_{j=0}^i p_j$, ring R and distribution χ as follows:
 - a. Generate a secret key $s'_i \leftarrow \chi$ and set $\mathbf{s}_i = (1, s'_i)$.
 - b. Generate two vectors $\mathbf{a}_i \leftarrow \mathcal{U}(R_q^N)$ and $\mathbf{e}_i \leftarrow \chi^N$ and compute $\mathbf{b}_i = \mathbf{a}_i \cdot s'_i + 2\mathbf{e}_i$.
 - c. Set $\mathbf{A}_i = [\mathbf{b}_i | -\mathbf{a}_i]$.
 2. Compute $\mathbf{s}'_i = \mathbf{s}_i \otimes \mathbf{s}_i = (1, s', (s')^2) \in R_{q_i}^3$, the tensor product of the key $\mathbf{s}_i$ with itself viewed as a vector.
 3. Set $\mathbf{s}''_i = \text{BitRep}(\mathbf{s}'_i, q)$.
 4. If $i \neq L$, compute $\mathbf{S}_{\mathbf{s}''_{i+1} \rightarrow \mathbf{s}_i} \leftarrow \text{BGV.SwKeyGen}(\mathbf{s}''_{i+1}, \mathbf{s}_i)$.

Output $pk = \{\mathbf{A}_i\}_{i=0}^L$, $ek = \{\mathbf{S}_{\mathbf{s}''_{i+1} \rightarrow \mathbf{s}_i}\}_{i=0}^{L-1}$ and $sk = \{\mathbf{s}_i\}_{i=0}^L$.

- $\mathbf{c} \leftarrow \text{BGV.Encrypt}(pk, m)$: Taking a plaintext $m \in R_2$ and public key $pk = \{\mathbf{A}_i\}_{i=0}^L$ as inputs, set $\mathbf{m} = (m, 0)$ and sample $\mathbf{r} \leftarrow \mathcal{U}(R_2^N)$. Then, compute and output $\mathbf{c} \leftarrow \mathbf{m} + \mathbf{A}_L^T \mathbf{r}$.
- $m \leftarrow \text{BGV.Decrypt}(sk, \mathbf{c})$: Let the secret key which decrypts $\mathbf{c}$ be $\mathbf{s}_i$, compute and output $m \leftarrow (\langle \mathbf{c}, \mathbf{s}_i \rangle \bmod q) \bmod 2$.
- $\mathbf{c}^* \leftarrow \text{BGV.EvalAdd}(ek, \mathbf{c}_1, \mathbf{c}_2)$: Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be encrypted under the same secret key $\mathbf{s}_i$. If not, use BGV.Refresh to bring the one at a higher level down to the other. Compute $\mathbf{c}' = \mathbf{c}_1 + \mathbf{c}_2$ and view $\mathbf{c}'$ as a ciphertext under $\mathbf{s}'_i = \mathbf{s}_i \otimes \mathbf{s}_i$. Output the refreshed ciphertext $\mathbf{c}^* \leftarrow \text{BGV.Refresh}(\mathbf{S}_{\mathbf{s}''_{i+1} \rightarrow \mathbf{s}_i}, \mathbf{c}', q_{i+1}, q_i)$.
- $\mathbf{c}^* \leftarrow \text{BGV.EvalMult}(ek, \mathbf{c}_1, \mathbf{c}_2)$: Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be encrypted under the same secret key $\mathbf{s}_i$. If not, use BGV.Refresh to bring the one at a higher level down to the other. Compute $\mathbf{c}' = \mathbf{c}_1 \otimes \mathbf{c}_2$ and view $\mathbf{c}'$ as a ciphertext under $\mathbf{s}'_i = \mathbf{s}_i \otimes \mathbf{s}_i$. Output the refreshed ciphertext $\mathbf{c}^* \leftarrow \text{BGV.Refresh}(\mathbf{S}_{\mathbf{s}''_{i+1} \rightarrow \mathbf{s}_i}, \mathbf{c}', q_{i+1}, q_i)$.
- $\mathbf{S}_{\mathbf{s}_1 \rightarrow \mathbf{s}_2} \leftarrow \text{BGV.SwKeyGen}(\mathbf{s}_1, \mathbf{s}_2)$: Taking two secret keys $\mathbf{s}_1 = (1, s'_1) \in R_q^{n_1}$, $\mathbf{s}_2 = (1, s'_2) \in R_q^{n_2}$ as inputs, let $K = n_1 \cdot \lceil \log q \rceil$.

1. Generate a public key for $\mathbf{s}_2$, $\mathbf{A}' \leftarrow \mathcal{U}(R_q^{K \times (n_2-1)})$ and $\mathbf{e} \leftarrow \chi^K$ and compute $\mathbf{A}' = \mathbf{A}\mathbf{s}'_2 + 2\mathbf{e}$. Set $\mathbf{B} = [\mathbf{b}] - \mathbf{A}'$.
 2. Compute $\mathbf{X} = \mathbf{B} + [\text{PowOf2}(\mathbf{s}_1) | \mathbf{0}_{K \times (n_2-1)}]$, where $\mathbf{0}_{K \times (n_2-1)}$ is a $K \times (n_2-1)$ -dimensional matrix with all 0 entries.
 3. Set and output $\mathbf{S}_{\mathbf{s}_1 \rightarrow \mathbf{s}_2} = \mathbf{X}$.
- $\mathbf{c}' \leftarrow \text{BGV.SwKey}(\mathbf{S}_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c})$: Compute and output $\mathbf{c}' = \text{BitRep}(\mathbf{c})^T \cdot \mathbf{S}_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}$.
 - $\mathbf{c}^* \leftarrow \text{BGV.Refresh}(\mathbf{S}_{\mathbf{s}_{i+1}'' \rightarrow \mathbf{s}_i}, \mathbf{c}, q_{i+1}, q_i)$: Taking as input a ciphertext encrypted under $\mathbf{s}_{i+1}'$, key switching matrix $\mathbf{S}_{\mathbf{s}_{i+1}'' \rightarrow \mathbf{s}_i}$, current modulus q_{i+1} and next modulus q_i , perform the following steps:
 1. Compute $\mathbf{c}' \leftarrow \text{PowOf2}(\mathbf{c}, q_{i+1})$, expanding $\mathbf{c}$ in preparation for key switching.
 2. Switch modulus by computing $\mathbf{c}'' \leftarrow \text{Scale}(\mathbf{c}', q_{i+1}, q_i, 2)$, getting an output with reduced noise.
 3. Compute and output $\mathbf{c}^* \leftarrow \text{SwKey}(\mathbf{S}_{\mathbf{s}_{i+1}'' \rightarrow \mathbf{s}_i}, \mathbf{c}'', q_i)$, a ciphertext encrypting the same plaintext as $\mathbf{c}''$ but under key $\mathbf{s}_i$.

Correctness and Security of BGV. The correctness and security of the BGV scheme is stated in the following two theorems.

Theorem 3.2.3 ([37, Theorem 3]). *For some $\log q_L = \Theta(L(\log \lambda + \log L))$, the BGV scheme is a correct leveled FHE scheme for depth- L arithmetic circuits. More precisely, BGV correctly evaluates circuits which are made up of addition and multiplication gates over R_2 with depth at most L .*

Theorem 3.2.4 (Security of BGV). *If the $\text{D-RLWE}_{n, q_L, \chi}$ instance selected in the key generation phase, BGV.KeyGen — satisfying the condition in Theorem 3.2.3 — is hard, then the BGV scheme is IND-CPA-secure.*

Unfortunately, we do not know how to prove that the BGV scheme is circular secure and so it has to be assumed if bootstrapping is desired.

3.2.2 Smart-Vercauteren Batching for the BGV Scheme

An important optimization for FHE schemes, batching allows us to evaluate the same function simultaneously over d blocks of encrypted data. Smart and Vercauteren [192] were the first to analyze batching for FHE. They noted that the plaintext space of ideal lattice and RLWE encryption schemes actually decompose into a product of ideals.

By considering a small plaintext space $\mathcal{P}$ that can be mapped to each of the ideals, we can embed a vector in $\mathcal{P}^d$ into the native plaintext space R_2 where d is the number of distinct ideals in the factorization. Addition and multiplication on R_2 translate to component-wise addition and multiplication on $\mathcal{P}^d$. Furthermore, elements in the entries of the vectors can be rotated or shifted using appropriate key switching matrices. Here, we call $\mathcal{P}$ the plaintext space, $\mathcal{P}^d$ the batched plaintext space and R_2 the native plaintext space of the BGV scheme.

The ability to pack multiple pieces of data into a single plaintext stems from the algebraic nature of the plaintext space $R_2 = \mathbb{Z}_p[X]/\langle\Phi_m(X)\rangle$. $\Phi_m(X)$ factors into d irreducible factors of degree $d = n/d$, for some d , modulo a prime p that does not divide m :

$$\Phi_m(X) = \prod_{i=1}^d F_i(X) \bmod p$$

This induces an isomorphism, via the Chinese Remainder Theorem, between the algebra $\mathbb{A}_p = \mathbb{F}_p[X]/\langle\Phi_m(X)\rangle$ and the product of d finite fields $\mathbb{L}_i \cong \mathbb{F}_{p^d}$ for $i \in [d]$,

$$\begin{aligned} \mathbb{A}_p &\cong \mathbb{F}_p[X]/\langle F_1(X)\rangle \otimes \cdots \otimes \mathbb{F}_p[X]/\langle F_d(X)\rangle \\ &= \mathbb{L}_1 \otimes \cdots \otimes \mathbb{L}_d = A_p. \end{aligned}$$

Let $G(X) \in \mathbb{Z}_p[X]$ be a degree- d irreducible polynomial, then $\mathbb{K}_d = \mathbb{Z}_p[X]/\langle G(X)\rangle$ is a copy of the field $\mathbb{F}_{p^d}$ with a basis determined by $G(X)$. For each $\mathbb{L}_i$, there is an isomorphism, $\sigma_{d,i}$ that embeds $\mathbb{K}_d$ into $\mathbb{L}_i$.

With a batched plaintext space $\mathcal{P}^d = \mathbb{K}_d^d$ of d copies of $\mathbb{K}_d$, $\mathcal{P}^d$ is mapped to A_p as follows:

$$\begin{aligned} \Psi_d : \quad \mathcal{P}^d &\longrightarrow A_p \\ \mathbf{m} = (m_1, \dots, m_d) &\longmapsto \Psi(\mathbf{m}) = (\sigma_{d',1}(m_1), \dots, \sigma_{d',d}(m_d)). \end{aligned}$$

For a batched plaintext $\mathbf{m}$, its image under Ψ_d , $\mathbf{a} = \Psi_d(\mathbf{m}) \in A_p$ is encoded into $\alpha = \text{CRT}_p(\mathbf{a}) \in \mathbb{A}_p$ via the Chinese Remainder Theorem isomorphism, CRT_p . Since CRT_p and Ψ_d are isomorphisms, adding and multiplying elements in $\mathbb{A}_p$ is equivalent to component-wise addition and multiplication of elements in A_p and $\mathcal{P}^d$.

Rotations and Frobenius Maps. The method to compute arbitrary permutations on batched plaintexts were given by Gentry, Halevi and Smart [84] to achieve FHE with polylogarithmic overhead. We briefly describe the procedures to perform rotations and

Frobenius map evaluations and refer to Gentry *et al.* [84] for the full details.

The field $\mathbb{E} = \mathbb{Q}(\zeta_m)$, where ζ_m is a primitive m -th root of unity, has a Galois group given by $\text{Gal}(\mathbb{E}/\mathbb{Q}) \cong (\mathbb{Z}/m\mathbb{Z})^*$. The m -th root of unity ζ_m can be thought of as X in $\mathbb{E} = \mathbb{Q}[X]/\langle \Phi_m(X) \rangle$ and this translates the mapping $\kappa_i \in \text{Gal}(\mathbb{E}/\mathbb{Q})$ for $i \in (\mathbb{Z}/m\mathbb{Z})^*$ into $\kappa_i : X \mapsto X^i$. Note that $\mathbb{A}_p$ and A_p correspond to $\mathbb{E}$ modulo p .

Although Galois theory does not apply to $\mathbb{A}_p$, elements in $\text{Gal}(\mathbb{E}/\mathbb{Q})$ do commute with the functions $\Psi_{d'}$ and CRT_p . For a vector of plaintexts $\mathbf{m} = (m_1, \dots, m_d) \in \mathcal{P}^d$, we have $\alpha = \text{CRT}_p(\Psi_{d'}(\mathbf{m})) \in \mathbb{A}_p$. Then, if $\kappa_i \in \text{Gal}(\mathbb{E}/\mathbb{Q})$ is applied to α , we get $\beta = \text{CRT}_p(\Psi_{d'}(\kappa_i(m_1), \dots, \kappa_i(m_d)))$, with $\kappa_i(m_j(X)) = m_j(X^i)$.

For any field $\mathbb{F}_{p^d}$, the Galois group $\text{Gal}(\mathbb{F}_{p^d}/\mathbb{F}_p)$ is the cyclic group of order d , C_d . Since $\mathbb{A}_p$ contains $\mathbb{F}_{p^d}$ as a subfield, $\text{Gal}(\mathbb{E}/\mathbb{Q})$ contains C_d as a normal subgroup and C_d is known as the decomposition group of a prime ideal lying above p in $\mathbb{E}$. In this context, $C_d = \{\kappa_0, \kappa_p, \dots, \kappa_{p^d-1}\}$ which is generated by the element $p \in (\mathbb{Z}/m\mathbb{Z})^*$, corresponding to the Frobenius map $\kappa_p : X \mapsto X^p$. Through careful choices to represent elements in $\text{Gal}(\mathbb{E}/\mathbb{Q})$ and fixing an appropriate root of $F_i(X) \bmod p$, elements κ_i would rotate and sometimes apply Frobenius maps to certain components of encoded plaintext vectors.

The unintended application of Frobenius maps is a problem as the desired output is a rotation of the encoded plaintext vector. Therefore, instead of using a single κ_i , Gentry, Halevi and Smart [84] proposed a simple solution which uses multiple automorphisms. The desired output is obtained by piecing together portions of several ciphertexts where each had a different κ_i is applied to them.

Concretely, any automorphism κ_i is performed by applying it to the component polynomials of a ciphertext, raising the indeterminate X to the j -th power then taking the result modulo $\Phi_m(X)$ and ciphertext modulus q . This automorphism, although applied to the ciphertext, is also applied to the plaintext encrypted within. However, it has a side effect that the result is now only decryptable by the modified secret key $\mathbf{s}^{(i)} = (1, \kappa_i(\mathbf{s}))$.

Let ciphertext $\mathbf{c} = (c_0(X), c_1(X))$, where $c_0(X) = m(X) + 2e(X) - c_1(X)s(X)$, with $\|e(X)\| < q/2$ i.e. a valid encryption of plaintext $m(X)$. Then, let

$$\begin{aligned} \mathbf{c}^{(i)} &= (\kappa_i(c_0(X)), \kappa_i(c_1(X))) \\ &= (c_0(X^i), c_1(X^i)), \end{aligned}$$

and for $\mathbf{s}^{(i)} = (1, s(X^i))$, we have

$$\begin{aligned} \langle \mathbf{c}^{(i)}, \mathbf{s}^{(i)} \rangle &= m(X^i) + 2e(X^i) - c_1(X^i)s(X^i) + c_1(X^i)s(X^i) \\ &= \kappa_i(m(X)) + 2e(X^i). \end{aligned}$$

After this, a **SwKey** operation is done using the key switching matrix $\mathbf{S}_{(\mathbf{s}^{(i)})' \rightarrow \mathbf{s}}$, where $\mathbf{s}'$ is $\text{PowOf2}(\mathbf{s}^{(i)})$, to transform $\mathbf{c}^{(i)}$ into one decryptable by the original key, $\mathbf{s}$. This allows us to compose multiple automorphisms and multiplications with other ciphertexts without adding too many public elements to the evaluation key.

Depth-Free Frobenius Map Evaluation. In this thesis, we rely heavily on Frobenius automorphisms to achieve efficient algorithms for some database queries. In the BGV scheme, the key switching technique can be applied to evaluate Frobenius maps without using multiplication gates, reducing the multiplicative depth of some circuits. Monomials of the form x^i , where $i > p$, can be evaluated with some combination of Frobenius maps, κ_p and multiplications instead of purely multiplications.

Here, we introduce two new algorithms for the BGV FHE scheme that enable what we call Frobenius gates which outputs a Frobenius map evaluation on its input.

- $\mathbf{S}_{\mathbf{s}'_p \rightarrow \mathbf{s}} \leftarrow \text{BGV.FrobKeyGen}(\mathbf{s})$: Taking the key $\mathbf{s} = (1, s') \in R_q^2$ as input, let $\mathbf{s}_p = (1, \kappa_p(s'))$ and $\mathbf{s}'_p = \text{PowOf2}(\mathbf{s}_p, q)$. Compute and output $\mathbf{S}_{\mathbf{s}'_p \rightarrow \mathbf{s}} \leftarrow \text{BGV.SwKeyGen}(\mathbf{s}'_p, \mathbf{s})$.
- $\mathbf{c}' \leftarrow \text{BGV.EvalFrob}(\mathbf{S}_{\mathbf{s}'_p \rightarrow \mathbf{s}}, \mathbf{c})$: Compute and output the ciphertext $\mathbf{c}' = \text{BitRep}(\mathbf{c})^T \cdot \mathbf{S}_{\mathbf{s}'_p \rightarrow \mathbf{s}}$.

In cases where the Frobenius map has to be applied multiple times, i.e. κ_{p^j} for some $j > 1$ instead of κ_p , we can use $\mathbf{S}_{\mathbf{s}'_{p^j} \rightarrow \mathbf{s}} \leftarrow \text{BGV.SwKeyGen}(\mathbf{s}'_{p^j}, \mathbf{s})$ to reduce the amount of computation.

3.3 Finite Extension Fields

In this section, we review several important results for working in the finite extension fields $\mathbb{F}_{p^d}$. They provide a method to represent functions over $\mathbb{F}_{p^d}^n$ as polynomials, which are convenient for evaluations with arithmetic circuits. Although they are presented for general primes p , we will only use $p = 2$ in later chapters.

Definition 3.3.1. *Let p, d, n be positive integers and additionally p be prime. For any function $\nu : \mathbb{F}_{p^d}^n \rightarrow \mathbb{F}_{p^d}$, we say that g is a polynomial expression of ν , if g is a polynomial such that $g(a_1, \dots, a_n) = \nu(a_1, \dots, a_n)$ for all $(a_1, \dots, a_n) \in \mathbb{F}_{p^d}^n$.*

Polynomial expressions are handy to represent functions over finite extension fields but it raises the question: which expression is the best? This is answered by the following theorem.

Theorem 3.3.2 (Minimal Polynomial Expression, [129, Lemma 1,2]). *For any function $\nu : \mathbb{F}_{p^d}^n \rightarrow \mathbb{F}_{p^d}$ with p, d, n from Definition 3.3.1, there exists a unique polynomial expression μ_ν of ν , whose total degree is minimal out of all possible polynomial expressions of ν . The degree of μ_ν in each variable is no greater than $p^d - 1$.*

To prove Theorem 3.3.2, we introduce Fermat's Little Theorem over finite extension fields.

Lemma 3.3.3 (Fermat's Little Theorem). *For any $\alpha \in \mathbb{F}_{p^d}$, $\alpha^{p^d} = \alpha$.*

Proof. If $\alpha = 0$, then we immediately get $0 = \alpha^{p^d} = \alpha = 0$. Otherwise, $\alpha \neq 0$ and is in the multiplicative group $\mathbb{F}_{p^d}^*$. Since $\mathbb{F}_{p^d}^*$ is the cyclic group C_{p^d-1} , we have $\alpha^{p^d-1} = 1$ implying that $\alpha^{p^d} = \alpha$. $\square$

Now, we are ready to prove Theorem 3.3.2.

Proof of Theorem 3.3.2.

Existence of μ_ν : First, we define, for $\alpha \in \mathbb{F}_{p^d}$, an indicator polynomial ι_α

$$\iota_\alpha(x) = (1 - (x - \alpha)^{p^d-1}) = \begin{cases} 1, & \text{if } x = \alpha; \\ 0, & \text{otherwise.} \end{cases}$$

Then, we extend it to $\alpha = (\alpha_i)_{i=1}^n \in \mathbb{F}_{p^d}^n$ with $\iota_\alpha(x_1, \dots, x_n) = \prod_{i=1}^n \iota_{\alpha_i}(x_i)$.

Then, we obtain μ_ν from ι_α as follows,

$$\mu_\nu(x_1, \dots, x_n) = \sum_{\alpha \in \mathbb{F}_{p^d}^n} \nu(\alpha) \cdot \iota_\alpha(x_1, \dots, x_n).$$

$\mu_\nu(\alpha) = \nu(\alpha)$ for every $\alpha \in \mathbb{F}_{p^d}^n$ and thus μ_ν is a polynomial expression of ν . Also, μ_ν has degree at most $p^d - 1$ in every variable.

Uniqueness of μ_ν : It suffices to consider the case where $\nu = 0$, the constant function that maps all inputs to 0. This is because a unique minimal polynomial expression for the zero function implies that there cannot be multiple polynomial expressions for any non-zero function with the same degree in every variable. Note that the zero polynomial is a polynomial expression of a zero function, $\nu = 0$.

We prove this via induction on the number of variables, n . In the base case, where $n = 1$, because for all $\alpha \in \mathbb{F}_{p^d}$ $\mu_\nu(\alpha) = \nu(\alpha) = 0$, $x - \alpha$ divides μ_ν . Therefore, $\mu_\nu(x) = 0$ or the degree of $\mu_\nu(x)$ has to be at least p^d which implies that uniqueness holds for $n = 1$.

3. PRELIMINARIES

On to the induction step, suppose that the uniqueness of minimal polynomial expression of the zero function is true when the number of variables is at most $n - 1$. Let $g(x_1, \dots, x_n)$ be a non-zero polynomial expression of the zero function $\nu(x_1, \dots, x_n) = 0$ whose degree is at most $p^d - 1$ in each variable. Without loss of generality, for some integer k between 0 and $p^d - 1$, there is a non-zero polynomial $g_k \in \mathbb{F}_{p^d}[x_1, \dots, x_{n-1}]$ such that $g(x_1, \dots, x_n) = \sum_{j=0}^{p^d-1} g_k(x_1, \dots, x_{n-1}) \cdot x_n^j$.

When we consider g as a polynomial in a single variable x_n , g should be the zero polynomial for the same reason as the base case where $n = 1$. Therefore, $g_k(x_1, \dots, x_{n-1})$ is a polynomial expression to the zero function on $n - 1$ variables and g_k is non-zero. Since the uniqueness of minimal polynomial expression of the zero function for $(n - 1)$ -variate functions is assumed true, we have a contradiction and uniqueness has to hold for the minimal polynomial expression of the zero function in n variables.

Finally, we show that μ_ν has minimal total degree among all polynomial expressions of ν . Let $\deg(g)$ and $\deg_{x_i}(g)$ denote the total degree and degree of variable x_i , respectively, for a polynomial g . Assume that there exists a polynomial expression g_ν of ν with minimal total degree and $g_\nu \neq \mu_\nu$. Then, due to the uniqueness of μ_ν , $\deg_{x_i}(g_\nu) \geq p^d$ for some $i \in [n]$. In addition, from Lemma 3.3.3, $x_i^{p^d} = x_i$ for every variable x_i . We can obtain a polynomial h_ν such that $h_\nu = g_\nu$ are both polynomial expressions of ν except $\deg_{x_i}(h_\nu) < p^d$ by repeated substituting $x_i^{p^d}$ with x_i for all $i \in [n]$. As μ_ν is unique, $h_\nu = \mu_\nu$ and $\deg(\mu_\nu) = \deg(h_\nu) < \deg(g_\nu)$. Thus, μ_ν has the minimum total degree among all polynomial expressions of the function ν . $\square$

In some cases, we want to construct a polynomial expression for a function that is only defined on a subset $\mathcal{S}$ of the entire input space $\mathbb{F}_{p^d}^n$. The inputs to the polynomial expression are assumed to be in $\mathcal{S}$ and so its output on any point not in $\mathcal{S}$ is not of major concern. For that, we revisit Lagrange interpolation.

Theorem 3.3.4 (Lagrange Interpolation). *Given the output of a function ν on every point in $\mathcal{S} = \prod_{i=1}^n \mathcal{S}_i \subseteq \mathbb{F}_{p^d}^n$, where $\mathcal{S}_i$ is the set of values in $\mathbb{F}_{p^d}$ that can appear in the i -th coordinate of elements in $\mathcal{S}$, a polynomial expression $f(x_1, \dots, x_n)$ can be obtained with*

$$f(x_1, \dots, x_n) = \sum_{\alpha=(\alpha_1, \dots, \alpha_n) \in \mathcal{S}} \nu(\alpha) \cdot \prod_{i=1}^n \left(\prod_{\beta_i \in \mathcal{S}_i \setminus \{\alpha_i\}} \frac{x_i - \beta_i}{\alpha_i - \beta_i} \right).$$

Proof. The polynomial f is a polynomial expression of ν when restricted to the input set $\mathcal{S}$ because the term $\prod_{i=1}^n \left(\prod_{\beta_i \in \mathcal{S}_i \setminus \{\alpha_i\}} \frac{x_i - \beta_i}{\alpha_i - \beta_i} \right) = 1$ if $(x_1, \dots, x_n) = (\alpha_1, \dots, \alpha_n) \in \mathcal{S}$ and 0 if $(x_1, \dots, x_n) \in \mathcal{S}$ but not equal to $(\alpha_1, \dots, \alpha_n)$. Therefore, for any $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathcal{S}$,

we have exactly one term that is non-zero, $f(\alpha) = \nu(\alpha) \cdot \prod_{i=1}^n \left(\prod_{\beta_i \in \mathcal{S}_i \setminus \{\alpha_i\}} \frac{\alpha_i - \beta_i}{\alpha_i - \beta_i} \right) = \nu(\alpha)$. $\square$

The finite extension field $\mathbb{F}_{p^d}$ actually also has the structure of a d -dimensional vector space over $\mathbb{F}_p$. First, we define a special class of polynomials over $\mathbb{F}_{p^d}$, called linearized polynomials.

Definition 3.3.5. *A linearized polynomial over $\mathbb{F}_{p^d}[x]$ is a polynomial of the form*

$$L(x) = \sum_{i=0}^m \alpha_i x^{p^i} = \sum_{i=0}^m \alpha_i \kappa_p^i(x),$$

for some integer $m < d$ and $\alpha_i \in \mathbb{F}_{p^d}$. By adding 0 coefficients, we obtain the standard form of a linearized polynomial as a linear operator,

$$L = \sum_{i=0}^{d-1} \alpha_i \kappa_p^i.$$

$\mathbb{F}_p$ -linear maps on the vector space $\mathbb{F}_{p^d}$ are exactly characterized by these linearized polynomials. Computing a $\mathbb{F}_p$ -linear map, T , on a d -dimensional vector $\mathbf{a}$ encoded in some element $\beta_{\mathbf{a}} \in \mathbb{F}_{p^d}$ corresponds to evaluating some linearized polynomial L_T at $\beta_{\mathbf{a}}$, $\beta_{T(\mathbf{a})} = L_T(\beta_{\mathbf{a}})$.

Theorem 3.3.6 ([185, Theorem 10.4.1]). *Let $L(x) \in \mathbb{F}_{p^d}[x]$ be a linearized polynomial. For $\alpha, \beta \in \mathbb{F}_{p^d}$ and $a, b \in \mathbb{F}_p$, we have*

$$L(a\alpha + b\beta) = aL(\alpha) + bL(\beta).$$

This means that the function associated with the polynomial L , $\nu_L : \mathbb{F}_{p^d} \rightarrow \mathbb{F}_{p^d}$ is a linear operator on the $\mathbb{F}_p$ -vector space $\mathbb{F}_{p^d}$.

Proof. The proof of the theorem is straightforward. For any monomial of the form x^{p^i} for $i \in [d-1]$, $(a\alpha + b\beta)^{p^i} = a\alpha^{p^i} + b\beta^{p^i}$ since a, b lie in the base field $\mathbb{F}_p$ and we have the freshman's dream $(s+t)^{p^i} = s^{p^i} + t^{p^i}$ in characteristic p fields. Therefore, the evaluation of $L(x)$ on $a\alpha + b\beta$, $L(a\alpha + b\beta)$, can be expressed as $aL(\alpha) + bL(\beta)$ since all monomials of $L(x)$ are of the form x^{p^i} . $\square$

Theorem 3.3.7 ([185, Theorem 10.4.4]). *Every linear operator T on the $\mathbb{F}_p$ -vector space*

$\mathbb{F}_{p^d}$ can be represented by a unique linearized polynomial with the form

$$L_T(x) = \sum_{i=0}^{d-1} \alpha_i x^{p^i} = \sum_{i=0}^{d-1} \alpha_i \kappa_p^i(x),$$

for some $\alpha_i \in \mathbb{F}_{p^n}$, $0 \leq i \leq d-1$.

Proof. There are p^{d^2} linearized polynomials on $\mathbb{F}_{p^d}$ which is also the number of linear operators on $\mathbb{F}_{p^d}$ as an $\mathbb{F}_p$ -vector space. On top of that, the monomials x^{p^i} are linearly independent over $\mathbb{F}_{p^d}$ which implies that each linearized polynomial in the standard form represents a unique linear operator. Since both sets have the same cardinality, and each standard form linearized polynomial corresponds to a unique linear operator, every linear operator has a unique standard form linearized polynomial as a representation. $\square$

3.4 Private Database Queries

Private database queries emerged from the advent and proliferation of cloud computing and storage. Outsourcing databases to the cloud can provide significant cost savings and better availability and robustness. However, outsourcing comes with another set of issues, of which one of the most important is privacy of the outsourced information. To reap full advantage of the benefits offered by the cloud, almost all information should be put on it, including those most sensitive. This puts the data in a precarious situation, becoming more vulnerable to theft by hackers or employees of the cloud services. Beyond privacy of sensitive data, particularly for databases, queries sent to the cloud could reveal details of the structure of the database. Not only does the conditions of each query inform the cloud of the nature of the database, a long query history may further leak the importance of portions of the database.

To ensure that privacy of the database is preserved, the straightforward solution is to encrypt both the database and every condition in queries directed at the database. More formally, a private database query (PDQ) is a procedure that takes an encrypted database $\overline{\mathcal{D}} = (\overline{\alpha}_1, \dots, \overline{\alpha}_n)$, consisting of n records, each $\alpha_i = (k, v_1, \dots, v_\tau)$, and a query statement q for some functionality Q over the encrypted database $\overline{\mathcal{D}}$ as inputs and outputs n' ciphertexts $(\overline{\gamma}_i)$ and satisfies the following conditions. Without the correct secret key,

1. no information about the α_i 's and γ_i 's can be obtained from any intermediate or final outputs of the procedure and the encryption of the database $\overline{\mathcal{D}}$ that is available to the cloud server before any query is sent.

Index	Account No. (v)	Name (k_1)	Balance (k_2)	...	Contact (k_τ)
1	612-175	Adam Smith	\$52.15	...	9876 6421
2	201-170	Paul Baker	\$1052.32	...	8326 3514
$\vdots$		$\vdots$		$\vdots$	
n	509-254	Helen Parker	\$671.20	...	9843 6841

Table 3.1: An Example of a Database

- no one can get any information on the conditional clauses of Q . For example, the constants a_1 and a_2 in the SQL query “`select  $\alpha_i.k$  from R where  $\alpha_i.v_1 = a_1$  and  $\alpha_i.v_2 = a_2$` ”.

The first condition is achieved if the encryption scheme used has semantic security. The second is more difficult to attain; simply encrypting the constants is insufficient since the **and** logical operator and the fact that the attributes $\alpha_i.v_1$ and $\alpha_i.v_2$ are named in the conditional clauses. For simplicity, we consider a weaker security model where only the constants are encrypted and hidden from the server.

3.4.1 Database Model and Security for Database Queries

In this thesis, we consider a database $\mathcal{D} = (\alpha_1, \dots, \alpha_n)$ as a table whose rows are $\alpha_i = (v_i, k_{i,1}, \dots, k_{i,\tau})$. Using $\alpha_i.v$ to refer v_i , the value attribute of row α_i and $\alpha_i.k_j$ for its j -th key attribute $k_{i,j}$, let A_j to denote set of j -th keys for each record in $\mathcal{D}$, $(k_{1,j}, \dots, k_{n,j})$. The value attribute v_i is returned if the row α_i satisfies the search condition on its keys. Since it is easy to extend the number of value attributes and it is not our main focus, we assume that there is only one per row. An illustration of a database can be seen in Table 3.1.

Let $\mathcal{Q}$ be a query functionality that can be performed in polynomial time and $\pi_{\mathcal{Q}}$ be a PDQ protocol for $\mathcal{Q}$. We model a query q on functionality $\mathcal{Q}$ as requiring the server to run some series of functions, $f_{\mathcal{Q},i}$, on q , database $\mathcal{D}$ and auxiliary data sent by the client Δ_i , i.e. $\{f_{\mathcal{Q},i}(q, \mathcal{D}, \Delta_i)\}$. Results of these functions $f_{\mathcal{Q},i}(q, \mathcal{D})$ may be sent to the client for processing before auxiliary data for the next step Δ_{i+1} is generated. In particular, for protocols built using FHE, it is possible only use one round, i.e. the querier sends an encryption of q , the server computes $f_{\mathcal{Q}}(\bar{q}, \bar{\mathcal{D}})$ and returns the encrypted result $\bar{\mathcal{R}}$; see Figure 3.1.

The definition for the security of the PDQ protocols is adapted from those proposed by Kantarcioğlu and Clifton [115], and we introduce a security game to model the indis-

tinguishability requirement. Informally, it says any secure PDQ protocol for functionality $\mathcal{Q}$, $\pi_{\mathcal{Q}}$ must have indistinguishable transcripts, m_{q_1}, m_{q_2} , for any pair of queries q_1, q_2 that run on the same database and outputs results of the same size.

Definition 3.4.1 (Security of PDQ Protocols). *For a challenger $\mathcal{C}$ and PPT adversary $\mathcal{A}$, the indistinguishability game for a PDQ protocol $\pi_{\mathcal{Q}}$, denoted $\mathcal{G}_{\pi_{\mathcal{Q}}, \mathcal{A}}^{\text{IND}}$, for a database protocol for query $\mathcal{Q}$ is described below.*

1. $\mathcal{C}$ sets up an instance of $\pi_{\mathcal{Q}}$ and sends the public parameters (such as the public key of the FHE scheme) to $\mathcal{A}$.
2. $\mathcal{A}$ chooses a database $\mathcal{D}$, then issues two different queries q_1, q_2 , that return results of the same length on $\mathcal{D}$, and sends $\mathcal{D}, q_1$ and q_2 to $\mathcal{C}$.
3. $\mathcal{C}$ encrypts the database $\mathcal{D}$, then chooses a random bit $b \leftarrow \mathcal{U}(\{0, 1\})$ and returns the encrypted database $\mathcal{D}$ together with the transcript of $\pi_{\mathcal{Q}}$ ran with q_b on $\mathcal{D}$, m_{q_b} , to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b' and the game ends.

The adversary wins if $b' = b$ and its advantage in $\mathcal{G}_{\pi_{\mathcal{Q}}, \mathcal{A}}^{\text{IND}}$ is

$$\text{Adv}_{\pi_{\mathcal{Q}}}^{\text{IND}}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

The PDQ protocol for functionality $\mathcal{Q}$, $\pi_{\mathcal{Q}}$ is λ -bit secure if the advantage in the indistinguishability game, $\mathcal{G}_{\pi_{\mathcal{Q}}, \mathcal{A}}^{\text{IND}}$, is

$$\text{Adv}_{\pi_{\mathcal{Q}}}^{\text{IND}}(\mathcal{A}) < \text{negl}(\lambda),$$

for any PPT adversary $\mathcal{A}$ and security parameter λ .

For the protocols described in the first part of the thesis, which only consists of one round, their security follows from the semantic security of the underlying FHE scheme. Informally, this is because the transcripts of the protocol are simply a list of FHE ciphertexts; distinguishing the transcripts of the protocol for two different queries would imply that the FHE ciphertexts reveals information about the query being processed by the protocol.

3.4.2 A Private Database Query Protocol for Compound Equality Predicates

We introduce three private database query protocols for the following query functionalities proposed by Kim *et al.* [129]:

 Figure 3.1: Generic n -Round Protocol for $\pi_{\mathcal{Q}}$ with FHE.

- Disjunction of Equality Conditions: $A_1 = a_1$ or ... or $A_\tau = a_\tau$;
- Conjunction of Equality Conditions: $A_1 = a_1$ and ... and $A_\tau = a_\tau$;
- Threshold Conjunction of Equality Conditions: T out of τ conditions are satisfied out of $A_1 = a_1, \dots, A_\tau = a_\tau$.

We require an FHE scheme whose plaintext space is a finite extension field as a base and additionally rely on it having depth-free Frobenius map operations for the techniques' efficiency. Further optimization is possible if a vector of plaintexts can be packed into a single ciphertext and these vectors can be manipulated by component-wise addition, multiplication and depth-free Frobenius map evaluations and vector shift and rotations. In particular, the BGV and BFV FHE scheme from Section 3.2 supports the optimal version when the Smart-Vercauteren packing methods are used.

The message space is the set $\mathcal{M} = \{0, 1\}^\ell$ of length- ℓ binary strings, which is encoded into the plaintext space $\mathcal{P} = \mathbb{F}_2^\ell \cong \mathbb{Z}_2[X]/\langle f(X) \rangle$ for an irreducible polynomial of degree ℓ , $f(X)$, with

$$\text{Encode} : \{0, 1\}^\ell \longrightarrow \mathbb{F}_2^\ell$$

$$a = (a_1, \dots, a_\ell) \longmapsto \text{Encode}(a) = \sum_{i=0}^{\ell-1} a_{i+1} X^i.$$

3. PRELIMINARIES

Figure 3.2: Protocol π_Q for Conjunctive Query.

For simplicity, we assume that $0 \in \mathbb{F}_2^\ell$ and its pre-image represents an empty record, and values are not allowed to be 0.

Equality Predicate. Kim *et al.* [129] considered and analysed several algorithms for equality tests for various plaintext spaces available in FHE schemes. The equality test for two encrypted messages a_1, a_2 in the message space $\mathcal{M}$ is defined as

$$\text{EQ}(\overline{a}_1, \overline{a}_2) = \begin{cases} \overline{1}, & \text{if } a_1 = a_2; \\ \overline{0}, & \text{otherwise.} \end{cases}$$

They proposed an algorithm to evaluate **EQ** by applying Fermat's Little Theorem (Lemma 3.3.3) for $\mathcal{P} = \mathbb{F}_p^\ell$, which has multiplicative depth of $\lceil \log(p-1) \rceil + \lceil \log \ell \rceil$ if depth-free Frobenius map evaluations are possible.

For $\mathcal{M} = \mathbb{F}_2^\ell$, the equality test evaluates a simple polynomial

$$g(x_1, x_2) = 1 - (x_1 - x_2)^{2^\ell} = \begin{cases} 1, & \text{if } x_1 = x_2; \\ 0, & \text{otherwise.} \end{cases}$$

Then, we have

$$\text{EQ}(\overline{a}_1, \overline{a}_2) = g(\overline{a}_1, \overline{a}_2).$$

The Private Conjunctive Query Protocol. In this protocol, the client requests all $\alpha_i.v$ such that *every* $\alpha_i.k_j = \delta_j$ for $j \in J \subseteq [\tau]$ and $\delta_j \in \mathcal{M}$. The protocol is illustrated in Figure 3.2.

The server executes the following algorithm upon receiving the query:

$$- \{\overline{\gamma}_i\}_{i=1}^n \leftarrow \text{eqConjunction}(\overline{\mathcal{D}}, \{\overline{\delta}_j\}_{j \in J}):$$

1. Compute $\overline{\beta_{i,j}} = \text{EQ}(\overline{\alpha_i.k_j}, \overline{\delta_j})$ for $i \in [n], j \in J$.
2. Compute $\overline{\gamma_i} = \left(\prod_{j \in J} \overline{\beta_{i,j}} \right) \cdot \overline{v_i}$ for $i \in [n]$.

The set $\{\overline{\gamma_i}\}_{i \in [n]}$ is returned to the client, who decrypts it to obtain the result of the query in plaintext.

The Private Disjunctive Query Protocol. In this protocol, the client requests all $\alpha_i.v$ such that *at least one of* $\alpha_i.k_j = \delta_j$ for $j \in J \subseteq [\tau]$, and $\delta_j \in \mathcal{M}$.

The server executes the following algorithm upon receiving the query:

- $\{\overline{\gamma_i}\}_{i=1}^n \quad \text{eqDisjunction}(\overline{\mathcal{D}}, \{\overline{\delta_j}\}_{j \in J})$:
 1. Compute $\overline{\beta_{i,j}} = \overline{1} - \text{EQ}(\overline{\alpha_i.k_j}, \overline{\delta_j})$ for $i \in [n], j \in J$.
 2. Compute $\overline{\gamma_i} = \left(\overline{1} - \prod_{j \in J} \overline{\beta_{i,j}} \right) \cdot \overline{v_i}$ for $i \in [n]$.

The set $\{\overline{\gamma_i}\}_{i \in [n]}$ is returned to the client, who decrypts it to obtain the result of the query in plaintext.

The Private Threshold Conjunctive Query Protocol. In this protocol, the client requests all $\alpha_i.v$ such that *at least T many* $\alpha_i.k_j = \delta_j$ for $j \in J \subseteq [\rho]$, $\delta_j \in \mathcal{M}$ and some threshold $T \in [\rho]$. For the correctness of the protocol, we require that $\ell > |J|$.

The server executes the following algorithm upon receiving the query:

- $\{\overline{\gamma_i}\}_{i=1}^n \quad \text{eqThreshold}(\overline{\mathcal{D}}, \{\overline{\delta_j}\}_{j \in J}, T)$:
 1. Let t be the indeterminate in the field $\mathbb{F}_2^\ell$ and find a polynomial $h \in \mathcal{P}[X]$ of degree ρ such that $h(t^\kappa) = 1$ for $T \leq \kappa \leq |J|$ and $h(t^\kappa) = 0$ for $0 \leq \kappa < T$.
 2. Compute $\overline{\beta_{i,j}} = \text{EQ}(\overline{\alpha_i.k_j}, \overline{\delta_j}) \cdot \overline{(t+1)} + \overline{1}$ for $i \in [n], j \in J$.
 3. Compute $\overline{\zeta_i} = \left(\prod_{j \in J} \overline{\beta_{i,j}} \right)$, then $\overline{\gamma_i} = h(\overline{\zeta_i}) \cdot \overline{v_i}$ for $i \in [n]$.

The set $\{\overline{\gamma_i}\}_{i \in [n]}$ is returned to the client, who decrypts it to obtain the result of the query in plaintext.

Multiplicative Depth of the Protocols. The correctness of the protocols are evident from the description and so we focus on determining the multiplicative depth of the protocols. It is one of the most important factor in the efficiency of the protocol since larger levels mean larger parameters and longer time to compute the result.

All three protocols use the **EQ** algorithm which requires $\lceil \log \ell \rceil$ depth. The structure of the private conjunctive and disjunctive query protocols are similar and so their respective

multiplicative depths are identical. Step 2 requires $\lceil \log(1 + |J|) \rceil$ depth and so the total is $\lceil \log \ell \rceil + \lceil \log(1 + |J|) \rceil$. The private threshold conjunctive query protocol additionally evaluates the polynomial $h(\bar{\zeta}_i)$ which has $\lceil \log |J| \rceil$ depth. Therefore, we can upper bound its total multiplicative depth by $\lceil \log \ell \rceil + 2\lceil \log(1 + |J|) \rceil$.

Type	Depth
eqConjunction	$\lceil \log \ell \rceil + \lceil \log(1 + J) \rceil$
eqDisjunction	$\lceil \log \ell \rceil + \lceil \log(1 + J) \rceil$
eqThreshold	$\lceil \log \ell \rceil + 2\lceil \log(1 + J) \rceil$

$\mathcal{P} = \mathbb{F}_{2^\ell}$ and J is the set of columns in the query.

Table 3.2: List of Multiplicative Depths of the PDQ Protocols

3.5 Experiment Platform

The experiment results presented in this thesis were obtained with a Intel Xeon Platinum 8170 with 104 threads, maximum turbo frequency of 3.7 GHz and 192 GB RAM. The programs generating the results used the following libraries: OpenMP [62], GMP 6.1.2 [97], NTL 11.2.0 [191] and HELib (Commit f905e95) [102].

The parameters chosen were evaluated for their security levels with the estimator by Albrecht *et al.* [9] incorporating the improved lattice attacks on FHE instances by Albrecht [8].

Chapter 4

Private Compound Wildcard Queries

4.1 Introduction

In this chapter, we consider private database protocols for substring pattern matching queries with wildcards on a database of encrypted strings that can be compounded with other queries, such as exact string matching. The setting involves a client outsourcing his encrypted database to a server and retrieving records from that database based on conditions specified in queries at a later time.

A substring pattern matching query takes a pattern W and returns any string s in the database that contains W as a substring. The pattern W is allowed to contain wildcard characters, which for performance reasons is restricted to the following three:

1. $\$$ stands for a single non-empty character;
2. $!(\alpha)$ stands for any single, non-empty and non- α character, also called an exclusion wildcard;
3. $\%$ signifies substring endings. For a pattern W , without any $\%$ symbols,
 - $\%W$ denotes substrings that end with pattern W ;
 - $W\%$ denotes substrings that begin with pattern W ;
 - $\%W\%$ denotes substrings that contain pattern W .

There is one caveat however, the exclusion wildcard $!(c)$ cannot be the final character of a pattern. To illustrate, consider the pattern “ $\%str\$de\%$ ”, which matches words like

“strode”, “stride”, “bestride”, “astride”, “strudel”. If we wish to additionally sieve out words containing “stride”, we can use “%str!(i)de%”.

For pattern matching on encrypted strings, searchable symmetric encryption (SSE) and homomorphic encryption are the main approaches. SSE-based protocols [73, 50] are efficient but unfortunately leak some information about the query in executions, such as the length of the pattern and are possibly vulnerable to file-injection attacks as discovered by Zhang *et al.* [207]. With compound queries, SSE might be insufficient as the scheme is specialized to support equality comparisons. Systems such as CryptDB [179] and Monomi [200] bypass this issue by combining several heterogeneous encryption schemes, such as order-preserving encryption and partial homomorphic encryption. Combining multiple encryption schemes with different functionality has its own set of issues. When novel cryptographic schemes are created to enrich the set of possible private database queries, then it is unclear how these can be integrated with existing setups. Moreover, additional information may be leaked such as whether the encrypted objects in the database satisfy a subset of all queried conditions.

With homomorphic encryption, there are two related work proposed by Yasuda *et al.* [206] and an enhanced variant by Saha and Koshiha [187]. Yasuda *et al.* [206] allows pattern matching where the pattern may contain at most one \$. They used a RLWE-based SHE scheme whose plaintext space is the ring $R = \mathbb{Z}_q[X]/\langle X^N + 1 \rangle$, where N is a power of two and q a large positive integer. Pattern matching is achieved by exploiting two different encodings for strings, one for the pattern and another for the string to be compared, and evaluating a low degree polynomial on the encrypted, encoded strings. The result of the pattern matching is obtained by decrypting the output ciphertext and if one of the coefficients of the decrypted polynomial is a 0, then the input pattern occurs in the string. The protocol is very efficient for pattern matching with wildcards but does not work well when compound queries are considered. The main obstacle is that for the result of the pattern matching to be used in further computations, we have to determine if there is a coefficient that is 0 in some encrypted polynomial. This would require a huge amount of computation as there are $N - 1$ coefficients to extract. Furthermore, their techniques are not compatible with the SIMD techniques for RLWE-based FHE schemes. Modifying the protocol to support compound queries yields an inefficient protocol and poor space complexity as N has to be quite large to guarantee security but key attribute lengths are much smaller than N in general for such queries. The enhanced protocol by Saha and Koshiha [187] allows more than one \$ wildcard but otherwise follows the same structure of Yasuda *et al.* [206]. Therefore, we can expect the same problems if we try to extend it.

For this work, we treat strings as vectors of characters from some alphabet augmented with an empty character $\Sigma \cup \{\#\}$ and exploit the finite extension fields which are the native plaintext spaces of RLWE-based FHE schemes. Let $\mathcal{D} = (\alpha_1, \dots, \alpha_n)$ be a database in the model described in Section 3.4.1 and assume that key attributes are strings of maximum length η , i.e. $\alpha_i = (v_i, k_{1,i}, \dots, k_{i,\rho})$ and $k_{i,j} \in \Sigma^\eta$. We consider private compound queries, of wildcard substring pattern matching and other queries, such as equality tests with numbers. Then, for wildcards $\%$, $\$$ and $!(\cdot)$ and indices $j, \ell \in [\rho]$, an example of such a query is

select $\alpha_i.v$ from $\mathcal{D}$ where $\alpha_i.k_j$ like “ $\%W\%$ ” and $\alpha_i.k_\ell = Q$,

where W is a pattern, possibly containing some $\$$ and $!(\alpha_a)$, for substring pattern matching and Q is some constant for an equality test. We note that the pattern can also have the form “ $W\%$ ” or “ $\%W$ ”. The length of the input pattern will be kept secret from the server but we allow the maximum length of the key attribute strings to be known.

Organization of the Chapter. In Section 4.2, we review the literature on private database queries for substring pattern matching with wildcards. Following that, in Section 4.3, we describe our wildcard pattern matching algorithm and apply it to support compound queries. Finally, in Section 4.4, some experiment results for the algorithms underlying the protocols are introduced and discussed.

4.2 Previous Work

The work that is most closely related to our substring pattern matching algorithm on encrypted strings is by Yasuda *et al.* [206] and an enhanced variant by Saha and Koshiha [187]. Their work handles retrieval queries and did not consider compound queries with other query types. They use the plaintext space $R_q = \mathbb{Z}_q[X]/\langle X^N + 1 \rangle$, where N is a power of two and q is a suitable large integer. Two types of encodings for strings, represented as a vector of characters $s = (a_1, \dots, a_\eta) \in \Sigma_\eta$, from Yasuda *et al.* [205] are exploited to achieve the results. Characters are first encoded into the set $\mathbb{Z}_q$

with 0 reserved for the wildcard \$ and transformed into one of two polynomials,

$$\begin{aligned} p_1(s) &= \sum_{i=0}^{\eta-1} a_{i+1} x^i \in R_q; \\ p_2(s) &= \sum_{i=0}^{\eta-1} a_{i+1} x^{N-i} = a_1 - \sum_{i=0}^{\eta-1} a_{i+1} x^{N-i} \in R_q. \end{aligned}$$

Denoting $s^j = (a_i^j)_{i=1}^\eta$, component-wise exponentiation by some $j \in \mathbb{Z}$, the client sends an encryption of $p_2(W)$, $p_2(W^2)$ and $p_2(W^3)$ for some pattern of length τ , $W = (w_1, \dots, w_\tau) \in \Sigma^\tau$ and the server computes an encryption of R_A

$$R_A = p_1(A)p_2(W^3) + p_1(A^3)p_2(W) - 2p_1(A^2)p_2(W^2). \quad (4.1)$$

for some plaintext key attribute string $A = (a_1, \dots, a_\eta)$. This result, Equation (4.1), is returned to the client for decryption and further processing.

The client decrypts the result and searches if there is some monomial x^i in the decrypted polynomial whose coefficient is 0. The coefficients of x^i in R_A for $i \in [\eta - \tau]$ is

$$\sum_{j=0}^{\tau-1} w_{j+1} \cdot a_{i+j+1} \cdot (w_{j+1} - a_{i+j+1})^2.$$

Since \$ is encoded to 0, the sum will be 0 if $w_{j+1} = \$$, $a_{i+j+1} = \$$ or $w_{i+1} = a_{i+j+1}$ for every $j \in \{0, 1, \dots, \tau-1\}$ which implies that the pattern W is in $A^{(i)} = (a_{i+1}, \dots, a_\eta, \#, \dots, \#)$, where $\#$ denotes an empty character, contains the pattern W and non-zero otherwise.

Yasuda *et al.* [206] gave a protocol for substring pattern matching with wildcards that is very efficient, using only a few ciphertext additions and scalar multiplications. Note that this can be easily extended to work with encrypted polynomials throughout by storing encryptions of $p_1(A)$, $p_1(A^2)$ and $p_1(A^3)$. Unfortunately, to evaluate compound queries with this algorithm, the processing that the client performs on the decrypted result has to be done homomorphically so that it can be combined with the results of other components of a compound query. In this case, the coefficients of the encrypted polynomial ($N-1$ many) has to be extracted which is quite expensive. Therefore, a direct extension of the protocol to our setting is not efficient. Furthermore, each ciphertext can only store one key attribute string due to the strict requirements on the plaintext space, whose polynomial modulus has to be $X^N + 1$ for N which is some power of two. In particular, the SIMD techniques outlined in Section 3.2.2 cannot be applied since the

polynomial modulus is some factor of $X^N + 1$ which will not be $X^k + 1$ for some $k \in \mathbb{Z}_{>0}$. For general PDQ, the large size of N , which is needed to guarantee security, mean most of the space unused as key attribute lengths are much smaller. Due to similar reasons, the enhanced protocol by Saha and Koshiha [187] cannot be efficiently extended to support compound queries as well.

Symmetric searchable encryption (SSE) is another way to achieve substring pattern matching with wildcards. SSE allows encrypted data to be matched with one another, without leaking anything other than whether two ciphertexts encrypt the same plaintext. Since the first construction by Song, Wagner and Perrig [194], many schemes [91, 61, 34, 189, 113, 112, 43, 42] have been proposed which enhanced functionality and efficiency. Faber *et al.* [73] proposed a substring query protocol using the SSE scheme proposed by Cash *et al.* [43]. However, the protocol may not hide the length of query patterns and could be vulnerable to attacks on the underlying SSE scheme. The file-injection attack, studied by Zhang, Katz and Papamanthou [207], if used on the Cash *et al.* [43] SSE scheme, would leak a substantial portion of queried strings to the server.

An alternate design for substring pattern matching was presented by Chase and Shen [50]. A user firsts expresses a string with a suffix tree [150, 201], whose nodes store suffix substrings together with their length and position. Then, the tree is encrypted and stored at the server. To send a query to the server, encryptions of all prefixes of the desired pattern are generated and sent to the server. With the additional length and position information stored in each node, the encrypted prefixes are checked for matches over the suffix tree. However, the scheme leaks the length of keyword strings and its use of deterministic encryption and extra information required for matching leaves it potentially vulnerable to the file-injection attacks by Zhang *et al.* [207].

Here, we mention another approach to substring search, using variants of predicate encryption such as an inner product encryption (IPE) scheme by Katz, Sahai and Waters [117] or hidden vector encryption scheme by Boneh and Waters [34]. Length- n strings are encrypted with an IPE scheme and stored at a server. The desired pattern in a substring matching query is be encrypted with the same IPE scheme and sent to the server together with a set of tokens. These tokens are derived from the secret key used so that the server can evaluate a predicate at the encrypted string and pattern. As strings have length n in this setting, $\mathcal{O}(2^n)$ tokens would be needed which lead to impractical query times.

Finally, we briefly mention some work handling string processing operations with secure two-party computation [104, 76, 116, 15]. The setting differs from ours, ours has one party outsourcing an encrypted database to a second party while the two-party

computation setting has the second party working with an unencrypted string. It does not seem straightforward to adapt these protocols to our setting where the second party works with encrypted strings that it cannot decrypt. Besides that, these protocols may not be extendable to support compound queries that do not leak partial results.

4.3 Wildcard Pattern Matching Protocols

We use a RLWE-based FHE scheme as the basis of the protocol, of which one example is the BGV scheme described in Section 3.2. The public key and evaluation key, if there is one, are assumed to be included in the algorithms and will not be specified. A pattern matching algorithm for encrypted strings and patterns is designed, with the option of using some auxiliary inputs to hide details of the pattern. Then, this algorithm is composed with the protocols in Section 3.4.2 to obtain a protocol for compound queries with support for wildcard pattern matching conditions.

String Encoding. We assume that strings are vectors of ASCII characters, which are 8-bit integers, so we use the plaintext space $\mathcal{P} = \mathbb{F}_2^8$ to encode the characters. In total, there are 95 characters that are used to represent Latin alphabet, numbers and symbols out of the 256 possible 8-bit strings. Among the wildcards, \$, % and !(·), we use one of the unused characters to represent both \$ and #. The input strings of the wildcard pattern matching algorithms are assumed of length η , with shorter strings are padded to length η by appending #. Note that we also assume all strings do not contain the \$ wildcard character and patterns do not include the empty space character, #. Besides that, we use one more unused ASCII characters to represent *, which denotes empty positions in patterns, and effectively matches any possible character, # included. We will add some auxiliary information to mark that a position in the pattern contains a wildcard, either \$ or !(c). The different patterns W , $\%W$, $W\%$ and $\%W\%$ will be handled by the pattern matching algorithm and there is no need to encode the % character.

Let $\text{ASCII}(c)$ denote the 8-bit string that represents the character c . Then, we encode c into the element $\text{Encode}(\text{ASCII}(c)) \in \mathbb{F}_2^8$ and, with a slight abuse of notation, denote by $\bar{c}$, an encryption of the encoded character $\text{Encode}(\text{ASCII}(c))$. For length- η encrypted strings $\bar{A} = (\bar{a}_1, \dots, \bar{a}_\eta)$ and $\bar{B} = (\bar{b}_1, \dots, \bar{b}_\eta)$, we define addition and multiplication operations on them component-wise

$$\begin{aligned}\bar{A} + \bar{B} &= (\bar{a}_1 + \bar{b}_1, \dots, \bar{a}_\eta + \bar{b}_\eta); \\ \bar{A} \cdot \bar{B} &= (\bar{a}_1 \cdot \bar{b}_1, \dots, \bar{a}_\eta \cdot \bar{b}_\eta).\end{aligned}$$

4.3.1 A Wildcard Pattern Matching Algorithm on Encrypted Data

The first step towards a private database query protocol for substring pattern matching is to design an algorithm to identify if the pattern occurs in a string. Our main idea is to express the pattern matching test for a pattern $\%W\%$ as the following Boolean formula with certain equality and inequality clauses,

$$\bigvee_{i=0}^{\eta-\xi} \left(\bigwedge_{j \in S} \underbrace{((a_{i+j} = w_j \neq \$) \wedge (w_j = \$))}_{(1)} \bigwedge_{j \notin S} \underbrace{(a_{i+j} \neq c_j)}_{(2)} \right). \quad (4.2)$$

In Equation (4.2), $S \subseteq [\xi]$ is the set of positions in the pattern W that does not contain an exclusion wildcard. As our aim is to hide the length of the pattern and locations of the wildcard characters, the equation is insufficient.

We focus on part (a) of the equation first, which considers whether the j -th character in the pattern is a $\$$ wildcard or matches the (shifted) string. It is made independent of the pattern length by appending the integers $\{\xi + 1, \dots, \eta\}$ to S , yielding $S' = S \cup \{\xi + 1, \dots, \eta\}$. The pattern W is extended by appending $\eta - \xi$ characters into $W = (w_1, \dots, w_\xi, \underbrace{*, \dots, *}_{\eta-\xi})$. Then, for every $i \in \{0, 1, \dots, \eta - \xi\}$, we extend part (1) if

Equation (4.2) into the following:

$$\bigwedge_{j \in S'} ((a_j^{(i)} = w_j \neq \$) \vee (w_j = \$)) \quad (4.3)$$

with $A^{(i)} = (a_1^{(i)}, \dots, a_\eta^{(i)}) = (a_{i+1}, \dots, a_\eta, \overbrace{\#, \dots, \#}^i)$ for $0 \leq i \leq \eta - 1$.

With the assumption that $\$$ only occurs in the input patterns and not strings, the first clause in the equation above, $(a_j^{(i)} = w_j \neq \$)$ can be determined a single equality test, $a_j^{(i)} = w_j$. To save on computation, we introduce a set of constants $\{e_j\}_{j=1}^\eta$, where $e_j = 1$ if $w_j = \$$ and 0 otherwise. The e_j 's are therefore exactly the truth value of the second clause in Equation (4.3). Combining it all together, we can evaluate Equation (4.3) over encrypted string $\bar{A} = (\bar{a}_1, \dots, \bar{a}_\eta)$ and pattern $\%\bar{W}\%$ where $\bar{W} = (\bar{w}_1, \dots, \bar{w}_\xi, \underbrace{\bar{\$}, \dots, \bar{\$}}_{\eta-\xi})$

with the following

$$\prod_{j \in S'} (\text{EQ}(\bar{a}_j^{(i)}, \bar{w}_j) + \bar{e}_j).$$

On to part (2) of Equation (4.2), which considers $!(c,)$ the exclusion wildcard with input character c . An exclusion simply requires that the string character $a_j^{(i)} \neq c_j$ for

$w_j = !(c_j)$ and can be computed with $\bar{1} + \text{EQ}(\bar{a}_j^{(i)}, \bar{c}_j)$. This formula can be integrated with the previous modifications to part (1) by setting $w_j = c_j$ and $e_j = 1$ if $w_j = !(c_j)$ for some $j \in [\xi]$. Besides this, we also set $e_j = 1$, if $j > \xi$ which is where $w_j = *$, corresponding to the padding added to patterns.

Overall, combining the modifications for both parts (1) and (2) to hide the pattern W and its length, Equation (4.2) is evaluated on encrypted strings and patterns with $\bigvee_{i=0}^{\eta-\xi} \bar{z}_i$ where

$$\bar{z}_i = \prod_{j=1}^{\eta} \left(\text{EQ}(\bar{a}_j^{(i)}, \bar{w}_j) + \bar{e}_j \right). \quad (4.4)$$

Each $\bar{z}_i$ can be evaluated without knowledge of ξ and W due to the use of some encrypted auxiliary information $\bar{e}_j$; its correctness is easily verified.

This also highlights why the $!(c)$ wildcard cannot be used at the end of a pattern. Since $\#$ only occurs at the end of strings, our current techniques cannot distinguish between $\#$ and any non-empty character that is not c . Therefore, should the exclusion wildcard occur at the end of the pattern, there is the possibility that a string will be wrongly matched to the pattern due to the first $\#$ character denoting the end of the string.

Besides the clauses in part (1) and (2), $\bigvee_{i=0}^{\eta-\xi} \bar{z}_i$ still requires knowledge of ξ to compute. To address this, we introduce a final set of auxiliary information, $\{d_i\}_{i=0}^{\eta-1}$ where $d_i = 1$ if $0 \leq i \leq \eta - \xi$ and $d_i = 0$ otherwise. Then, we evaluate Equation (4.2) without leaking any information about the pattern W by computing

$$\bar{t} = \bar{1} + \prod_{i=0}^{\eta-1} (\bar{1} + \bar{z}_i \cdot \bar{d}_i) = \begin{cases} \bar{1}, & \text{if } A \text{ contains the pattern } W; \\ \bar{0}, & \text{otherwise.} \end{cases} \quad (4.5)$$

A Formal Description. Let $A = (a_1, \dots, a_\eta) \in (\Sigma \cup \{\#\})^\eta$ be an string of length η and $W = (w_1, \dots, w_\xi) \in (\Sigma \cup \{\$, !(\cdot)\})^\xi$ be a pattern of length ξ . Denoting the entire procedure with $(W', D, E) \leftarrow \text{wcPrep}(W)$, we perform the following pre-processing steps:

1. Expand W into $W^\dagger = (w_1, \dots, w_\xi, *, \dots, *)$ by appending $\eta - \xi$ many $*$ wildcard characters.
2. Determine two lists $D = (d_0, \dots, d_{\eta-1})$ and $E = (e_1, \dots, e_\eta)$ as follows,

$$d_i = \begin{cases} 1, & \text{if } 0 \leq i \leq \eta - \xi; \\ 0, & \text{if } \eta - \xi + 1 \leq i \leq \eta - 1. \end{cases}$$

$$e_j = \begin{cases} 1, & \text{if } w_j \text{ corresponds to a wildcard } \$ \text{ or } !(c_j) \text{ or padding } *; \\ 0, & \text{otherwise.} \end{cases}$$

3. Set $W' = (w'_1, \dots, w'_\xi, *, \dots, *)$, where

$$w'_j = \begin{cases} w_j, & \text{if } w_j \neq !(c_j) \text{ for some } c_j \in \Sigma; \\ c_j, & \text{if } w_j = !(c_j) \text{ for some } c_j \in \Sigma. \end{cases}$$

From here on, we use w'_j to denote the result of pre-processing the character at w_j with **wcPrep**.

Then, the wildcard pattern matching algorithm is as follows:

– $\bar{t} \leftarrow \mathbf{wcMatch}(\bar{A}, \bar{W}', \bar{D}, \bar{E})$: Takes as input an encrypted string $\bar{A}$, encrypted pattern $\bar{W}'$ and encrypted auxiliary information $\bar{D} = (\bar{d}_0, \dots, \bar{d}_{\eta-1})$ and $\bar{E} = (\bar{e}_1, \dots, \bar{e}_\eta)$ and computes the encrypted result $\bar{t}$ as follows:

1. For $0 \leq i \leq \eta - 1$:

a. Generate shifted strings $\bar{A}^{(i)} = (\bar{a}_1^{(i)}, \dots, \bar{a}_\eta^{(i)})$, where

$$\bar{a}_j^{(i)} = \begin{cases} \bar{a}_{i+j}, & \text{if } i + j \leq \eta; \\ \#, & \text{otherwise.} \end{cases}$$

b. Set $\bar{X}_i = \bar{A}^{(i)} - \bar{W}' = (\bar{x}_{i,1}, \dots, \bar{x}_{i,\eta})$.

c. Evaluate $\bar{y}_{i,j} = \mathbf{EQ}(\bar{x}_{i,j}, \bar{0})$ for $1 \leq j \leq \eta$.

d. Let $\bar{Y}_i = (\bar{y}_{i,1}, \dots, \bar{y}_{i,\eta})$ and $\bar{S}_i = \bar{Y}_i + \bar{E} = (\bar{s}_{i,1}, \dots, \bar{s}_{i,\eta})$.

e. Compute $\bar{z}_i = \prod_{j=1}^{\eta} \bar{s}_{i,j}$.

2. Output $\bar{t} = \bar{1} + \prod_{j=1}^{\eta-1} (\bar{1} + \bar{z}_j \cdot \bar{d}_j)$.

Other Patterns. Assuming the correctness of the **wcMatch** algorithm for the input pattern $\%W\%$, we derive pattern matching algorithms for three other patterns W , $\%W$ and $W\%$. These are handled by modifying the intermediate pattern $W^\dagger$ and auxiliary information D and E in **wcPrep**. The new encrypted outputs from the modified **wcPrep**, $\bar{W}'$, $\bar{D}$ and $\bar{E}$ are then fed to **wcMatch** to compute the result of the pattern matching.

For the pattern $W\%$, we run **wcPrep** with W but modify Step 2. We instead set $d_0 = 1$ and $d_i = 0$ for $i \in [\eta - 1]$.

For input pattern W without any $\%$ characters, we break the rule that patterns cannot contain the $\#$ character and use an altered Step 1 in **wcPrep**, extending W into $W^\dagger = (w_1, \dots, w_\xi, \#, \dots, \#)$ by appending $\eta - \xi$ $\#$ characters instead. Then, we use the modified Step 2 from the pattern $W\%$. Note that for positions i where $w_i = \#$, the auxiliary information $e_i = 0$ although $\$$ and $\#$ are encoded to the same element in $\mathbb{F}_2^8$.

For the pattern $\%W$, we use the modified Step 1 that is intended for the pattern W and additionally set $w_\xi = !(c_j)$ if $w_\xi = \$$.

To see that the modifications for $\%W$ are correct, we let the attribute string $A = (a_1, \dots, a_{\eta'}, \#, \dots, \#) \in (\Sigma \cup \{\#\})^\eta$ for some $\eta' \leq \eta$. If A ends with W , then $A^{(\eta' - \xi)} = (a_{\eta' - \xi + 1}, \dots, a_{\eta'}, \#, \dots, \#)$ and we will have either $(a_j^{(\eta' - \xi)} = w_j \neq \$)$, $(w_j = \$)$ or $(a_j^{(\eta' - \xi)} \neq c_j \text{ and } w_j = !(c_j))$ for all $j \in [\xi]$. This means we have $z_{\eta' - \xi} = 1$ in Equation (4.4) and $t = 1$ in Equation (4.5) which is as we expect.

On the other hand, if A does not end with W , then there are two cases, $\eta' < \xi$ and $\eta' \geq \xi$. The first case is straightforward, for every $i \in \{0, \dots, \eta - 1\}$, there will be some index $j \in [\xi]$ such that $a_j = \#$ which means that $z_i = 0$. Therefore, $t = 0$ as well.

Otherwise, $\eta' \geq \xi$. Then, we consider the shifted strings $A^{(i)} = (a_1^{(i)}, \dots, a_{\eta'}^{(i)}) = (a_{i+1}, \dots, a_{\eta'}, \#, \dots, \#)$:

– For $i = \eta' - \xi$, there is at least an index j such that either

- ◊ $w_j \neq \$, !(c_j)$ and $a_j^{(i)} \neq w'_j$;
- ◊ $w_j = !(c_j)$ and $a_j^{(i)} = c_j$.

– For $0 \leq i < \eta' - \xi$, if $\eta' \neq \xi$, $a_{\xi+1}^{(i)} = a_{\xi+1+i} \neq \#$ but $w'_{\xi+1} = \#$.

– For $i > \eta' - \xi > 0$, $a_\xi^{(i)} = \#$ but $w'_\xi \neq \#$.

Therefore, for every $i \in \{0, \dots, \eta - 1\}$, there is at least one index j such that $(a_j^{(i)} \neq w'_j) \wedge (w_j \neq \$, !(c_j))$ or $(a_j^{(i)} = c_j) \wedge (w_j = !(c_j))$. This means that $z_i = 0$ for all $i \in \{0, \dots, \eta - 1\}$ in Equation (4.2) and $t = 0$ in Equation (4.5).

Remark 4.3.1. We can adapt **wcMatch** $(\overline{A}, \overline{W'}, \overline{D}, \overline{E})$ to compute if the string $A = W$ for W without any wildcard characters, also called an exact match, by setting $\overline{W'} = \overline{W}$, $\overline{D} = (\overline{1}, \overline{0}, \dots, \overline{0})$ and $\overline{E} = (\overline{0}, \dots, \overline{0})$. This way, the server cannot distinguish between a wildcard pattern matching query from an exact match query by the semantic security of the FHE scheme.

We can further simplify the algorithm to take only $\bar{A}$ and $\bar{W}$ for processing, computing $\bar{t} = \prod_{i=1}^{\eta} \text{EQ}(\bar{a}_1, \bar{w}_i)$ instead. We denote this algorithm with $\text{exMatch}(\bar{A}, \bar{W})$.

Analysis of the Algorithm. Now, we analyse the wcMatch algorithm, determining its correctness, security and efficiency.

Correctness. We summarize the correctness of wcMatch in the following theorem.

Theorem 4.3.2. *Let A and W be a string and pattern as defined earlier. Assuming that the auxiliary information D and E are correctly determined for a given pattern W , the algorithm wcMatch outputs an encryption of the correct matching result between an encrypted string $\bar{A}$ and encrypted pattern $\bar{W}'$.*

Proof. In Step 1b.,

$$x_{i,j} = a_j^{(i)} - w'_j = \begin{cases} a_{i+j} - w'_j, & \text{if } 1 \leq j \leq \xi; \\ a_{i+j} - \$, & \text{if } \xi + 1 \leq j \leq \eta - i; \\ \# - \$, & \text{if } \eta - i + 1 \leq j \leq \eta. \end{cases}$$

We will only consider the cases where $0 \leq i \leq \eta - \xi$ since the remaining i are eliminated using the d_i 's at a later step. Denoting the equality test algorithm on the FHE plaintext space $\mathcal{P}$ with $\text{EQ}_{\mathcal{P}}$, at Step 1c.,

$$y_{i,j} = \text{EQ}_{\mathcal{P}}(x_{i,j}, 0) = \begin{cases} 1, & \text{if } 1 \leq j \leq \xi \text{ and } a_{i+j} = w'_j; \\ 0, & \text{if } 1 \leq j \leq \xi \text{ and } a_{i+j} \neq w'_j; \\ 0, & \text{if } \xi + 1 \leq j \leq \eta. \end{cases}$$

After incorporating the auxiliary information e_j , for $0 \leq i \leq \eta - \xi$ and $1 \leq j \leq \eta$, we have

$$s_{i,j} = y_{i,j} + e_j = \begin{cases} 0, & \text{if } 1 \leq j \leq \xi, a_{i+j} \neq w'_j \text{ and } w_j \neq \$, !(c_j); \\ 0, & \text{if } 1 \leq j \leq \xi, a_{i+j} = c_j \text{ and } w_j = !(c_j); \\ 1, & \text{if } \xi + 1 \leq j \leq \eta. \end{cases}$$

where $\mathcal{P}$ is a field of characteristic 2 and $e_j = 1$ if and only if w_j is a wildcard character, whether $\$$ or $!(c_j)$.

If the pattern W is contained within A , then there would be some index $i \in \{0, \dots, \eta - \xi\}$ such that either $(a_{i+j} = w_j \neq \$), (w_j = \$)$ or $(a_{i+j} \neq c_j \text{ and } w_j = !(c_j))$. Then, for this particular i , $z_i = \prod_{j=1}^{\eta} s_{i,j} = 1$, $1 + z_i \cdot d_i = 0$ because $d_i = 1$. The result would be therefore $t = 1 + \prod_{i=0}^{\eta-1} (1 + z_i \cdot d_i) = 1$.

If not, then for all i , there will be at least one index $j \in [\eta]$ such that either $(a_{i+j} \neq w_j \text{ and } w_j \neq \$,!(c_j))$ or $(a_j = c_j \text{ and } w_j \neq !(c_j))$. Therefore, $s_{i,j} = 0$ for that j , $z_i = \prod_{j=1}^{\eta} s_{i,j} = 0$ for all i and $t = 1 + \prod_{i=0}^{\eta-1} (1 + z_i \cdot d_i) = 0$.

Hence, the algorithm is correct as the ciphertext $\bar{t}$ would contain an encryption of the correct result. $\square$

Security. It is simple to show that an honest-but-curious server cannot learn anything about the pattern, including its length, when running the algorithm. Since the inputs to **wcMatch** are encrypted with an FHE scheme, semantic security of the scheme guarantees that the encrypted information is not revealed. The algorithm only requires knowledge of the maximum length of strings η to determine the number of times loop statements should be run. Therefore, running **wcMatch** does not reveal any information about the string A and pattern W except for their maximum length η which is given to the server.

Efficiency. The main metric of efficiency we consider is the multiplicative depth of the algorithm. Recall that the equality test algorithm described in Section 3.4.2 has depth $\lceil \log \ell \rceil$, where the plaintext space is $\mathcal{P} = \mathbb{F}_2^\ell$. Therefore, with $\mathcal{P} = \mathbb{F}_2^8$ in this case, **EQ** has multiplicative depth $\log 8 = 3$. After that, we compute multiplications in Steps 1e. and 2., specifically η terms are multiplied in each case so each contributes $\lceil \log \eta \rceil$ multiplicative depth to the total. In Step 2, we additionally compute $\bar{z}_i \cdot \bar{d}_i$ which requires depth 1. In total, **wcMatch** consumes $4 + 2\lceil \log \eta \rceil$ depth.

4.3.2 Private Compound Wildcard Query Protocols

Now, we apply the algorithm, **wcMatch**, from the previous section to construct private database queries for keys attributes that are strings. For a database $\mathcal{D} = (\alpha_1, \dots, \alpha_n)$ of n records, which each have ρ keys, we consider each attribute A_i to have values that are strings of at most η length for simplicity. Then, we use $\bar{k}_{i,j}$ to denote an encryption of the key attribute string $\alpha_i.k_j$. To illustrate the protocols, we use the pattern $\%W\%$ for wildcard matching, but note that the other patterns W , $\%W$ and $W\%$ can be used by adapting **wcPrep** as mentioned previously.

The protocols' structure is identical to those in Section 3.4.2, except the equality test **EQ** is replaced with **wcMatch** and **exMatch**. In fact, protocols that use any combination of **EQ**, **wcMatch** and **exMatch** can be done since the pattern matching algorithms are designed to be composable with other Boolean predicates.

Conjunctive Case. First, we consider conjunctive queries with a mix of wildcard and

exact matching conditions. In particular, we handle the private query that seeks all $\alpha_i.v$ that satisfy

$$\left(\bigwedge_{j \in J_1} \bar{k}_{i,j} \text{ like } \overline{\%W_j\%} \right) \wedge \left(\bigwedge_{j \in J_2} \bar{k}_{i,j} = \overline{W_j^*} \right) \quad (4.Q1)$$

where $j \in J_1 \subset [\rho]$ are the indices of the columns undergoing wildcard matching and $j \in J_2 \subset [\rho]$ are the indices of those columns with exact matching; note that $J_1 \cap J_2 = \emptyset$.

- $\{\bar{\gamma}_i\}_{i \in [n]} \leftarrow \text{wildcardConjunction}(\overline{\mathcal{D}}, \{\overline{W'_j}, \overline{D_j}, \overline{E_j}\}_{j \in J_1}, \{\overline{W_j^*}\}_{j \in J_2})$:
For $i \in [n]$:

1. Compute

$$\bar{\beta}_{i,j} = \begin{cases} \text{wcMatch}(\bar{k}_{i,j}, \overline{W'_j}, \overline{D_j}, \overline{E_j}), & \text{if } j \in J_1; \\ \text{exMatch}(\bar{k}_{i,j}, \overline{W_j^*}), & \text{if } j \in J_2. \end{cases}$$

2. Compute $\bar{\gamma}_i = \overline{\alpha_i.v} \cdot \left(\prod_{j \in J_1 \cup J_2} \bar{\beta}_{i,j} \right)$.

Disjunctive Case. For the disjunctive case, we simply adjust the computation of the $\bar{\beta}_{i,j}$'s and $\bar{\gamma}_i$ to find the $\alpha_i.v$'s that satisfy

$$\left(\bigvee_{j \in J_1} \bar{k}_{i,j} \text{ like } \overline{\%W_j\%} \right) \vee \left(\bigvee_{j \in J_2} \bar{k}_{i,j} = \overline{W_j^*} \right). \quad (4.Q2)$$

The sets J_1 and J_2 are defined in the same exact manner as the conjunctive query in Equation (4.Q1).

- $\{\bar{\gamma}_i\}_{i \in [n]} \leftarrow \text{wildcardDisjunction}(\{\overline{W'_j}, \overline{D_j}, \overline{E_j}\}_{j \in J_1}, \{\overline{W_j^*}\}_{j \in J_2})$:
For $i \in [n]$:

1. Compute

$$\bar{\beta}_{i,j} = \begin{cases} \bar{1} - \text{wcMatch}(\bar{k}_{i,j}, \overline{W'_j}, \overline{D_j}, \overline{E_j}), & \text{if } j \in J_1; \\ \bar{1} - \text{exMatch}(\bar{k}_{i,j}, \overline{W_j^*}), & \text{if } j \in J_2. \end{cases}$$

2. Compute $\bar{\gamma}_i = \overline{\alpha_i.v} \cdot \left(\bar{1} - \prod_{j \in J_1 \cup J_2} \bar{\beta}_{i,j} \right)$.

Analysis of the Protocols. Both protocols share the same structure and security properties. So, we provide a proof for the conjunctive case and note that the other follows

directly.

Correctness and Security. The correctness of the protocols is evident from the description and the security of the **wcMatch** and **exMatch** algorithms. The security of the protocols is stated in the following theorem.

Theorem 4.3.3. *Assuming the the FHE scheme used in the protocol satisfies semantic security, then the protocols described in this section are secure in the manner of Definition 3.4.1.*

Proof. Since the FHE scheme used is semantically secure, no PPT party without knowledge of the secret key can distinguish between an encryption of m_0 , $\bar{m}_0$ and one of m_1 , $\bar{m}_1$, where $|m_0| = |m_1|$. Recall that the maximum length η of the key attribute strings is given to the server during the initial stage when the encrypted database $\bar{D}$ is sent to the server.

The private portion of the input to the query consists of the query pattern and auxiliary information. Pattern lengths are hidden by padding patterns up to length η . The client encrypts the private inputs with the semantically secure FHE scheme before sending them to the server or trusted third party, and so protocol inputs cannot be distinguished.

As all key attributes involved in the protocol are encrypted, a server cannot learn anything about matched strings $\bar{k}_{i,j}$'s. From the earlier discussion, the **wcMatch** and **exMatch** algorithms do not leak information about its inputs to the server. Hence, during computation on the server, none of the encrypted intermediate $(\bar{\beta}_{i,j})$ or final results $(\bar{\gamma}_i)$ reveal information about the input pattern.

In summary, the only information revealed to the server by the protocol is the encrypted results $\{\bar{\gamma}_i\}_{i=1}^n$. No information on the pattern is leaked by the encrypted input, $\{\bar{W}', \bar{D}$ and $\bar{E}\}$, $\{\bar{W}^*\}$ and all key attributes involved in the matching $\bar{k}_{i,j}$ remain hidden. These are all indistinguishable from the same computation of another input wildcard pattern string by the semantic security of the FHE scheme used. $\square$

Efficiency. In the first step, the protocols execute the **wcMatch** and **exMatch** algorithms, which have a maximum depth of $4 + 2\lceil \log \eta \rceil$. Next, the $\bar{\beta}_{i,j}$'s are combined and the value attribute multiplied with that result. This requires $1 + |J_1| + |J_2|$ multiplications in total which translates to $\lceil \log(1 + |J_1| + |J_2|) \rceil$ depth. Hence, the total depth of the protocols are $4 + 2\lceil \log \eta \rceil + \lceil \log(1 + |J_1| + |J_2|) \rceil$.

Assuming each character is encrypted in its own ciphertext, the client has to send $(3|J_1| + |J_2|)\eta$ ciphertexts to the server who replies with $n\eta$ ciphertexts. With the SIMD

techniques in Section 3.2.2, this is reduced by a factor of d if the cyclotomic polynomial has d factors modulo 2.

4.4 Implementation

In this section, we present some performance results from a basic implementation of the algorithms and protocols described in Section 4.3.

We first provide the details on the parameters chosen for the experiments. Then, we describe how the SIMD techniques in Section 3.2.2 are applied together with multi-threading and a simple optimization on computing $\bar{\gamma}_i$ in the implementation. Then, the results of performance evaluation of `wcMatch` and `wildcardConjunction` are presented. We do not consider `wildcardDisjunction` as it is very similar to the conjunctive case.

4.4.1 Implementation Details

We use the implementation provided by Kim *et al.* [128] to perform the experiments. It can be found at <https://www.github.com/benjaminhmtan/wildcardHElib>.

BGV Instance in HElib. In the experiments, we use the BGV scheme with the m -th cyclotomic polynomial modulus $\langle \Phi_m(X) \rangle$, where $m = 32767$. The plaintext space decomposes into $\prod_{i=1}^{1800} \mathbb{F}_{2^{15}}$ with 1800 slots and each slot is the finite field $\mathbb{F}_{2^{15}}$. Using 29 or 31 levels in the experiment, the security level of the instances are at least 80 bits.

Although the theoretical upper bound on the multiplicative depth is $20 = 2\lceil \log \eta \rceil + 4$ for the `wcmatch` algorithm and $23 = 2\lceil \log \eta \rceil + 4 + \lceil \log(1 + |J_1| + |J_2|) \rceil$ for `wildcardConjunction`, the actual levels we require are 29 and 31 for experiments one and two respectively. There are two main reasons to that: The first is that we use a slightly larger than required field due to a lack of parameters, more levels are needed to evaluate the `EQ` algorithm in this case. Besides this, SIMD operations such as shifts and rotations also contribute to the increase. They do add some noise to the ciphertexts which affects the multiplicative depth when they are heavily exploited. Although this means that a longer time is required to complete the processing, we can tune the parameters to provide large number of slots and match multiple attribute strings at a time.

Packing Strings with SIMD. We apply the SIMD techniques in Section 3.2.2 by Smart and Vercauteren [193], which allows us to pack multiple characters into a single ciphertext and compute on the packed characters simultaneously with every operation. In general, the number of slots available is not a multiple of the attribute string length. If there are

Figure 4.1: Multiplication Binary Tree for Four Ciphertexts.

sufficient slots, multiple strings can be packed within one ciphertext; once there are fewer empty slots than η , we pad them with the $\#$ empty character.

Multi-Threading. The `wcMatch` algorithm contains several loops whose contents can be executed independently. So, we use OpenMP [62] to distribute the computational load across many threads. This lets us better utilize the available resources and significantly reduces the time taken to run the protocol. In particular, due to the large number of threads available on the test platform, every shifted attribute string $\bar{A}^{(i)}$ is matched concurrently. Not only that, we apply multi-threading to accelerate the computation of $\bar{t}$ too.

Depth-Optimized Products. To ensure that the depth of the circuit is minimized, we compute the product $\prod_{i=0}^{\eta-1} (1 - \bar{z}_i \bar{d}_i)$ with a specific method. Individual $(\bar{1} - \bar{z}_i \bar{d}_i)$ terms pair-wise and then those results pair-wise iteratively until the result is obtained. It uses $\lceil \log \eta \rceil$ depth but the same number of multiplications compared to the sequential method. The resulting algorithm forms a binary tree structure, illustrated in Figure 4.1, where each node is the product of its children. At each level of the binary tree, each multiplication is independent of the others and so can be done in its own thread.

Conjunction Optimization. Wildcard matching requires more depth, $\lceil \log \eta \rceil$, than exact matching as there is no need to compute $\bar{z}_i$'s in the exact matching case. Thus, we combine the results of the exact matching comparisons first, exploiting the depth difference to ensure we only require one additional depth to compute the $\bar{\gamma}_i$'s. The exact matching algorithm is less complex and finishes first, so this combining operation is done

String Length (Bytes)	# of Strings in a Ciphertext	Evaluate $\overline{A}^{(i)}, \dots, \overline{S}_i$	Evaluate $\overline{z}_i$	Evaluate $\overline{t}$	Total Time	Amortised Time
35	51	25.67s	2.02s	1.17s	28.86s	0.57s
45	40	33.22s	2.29s	1.39s	36.90s	0.92s
55	32	37.33s	2.33s	1.48s	41.14s	1.29s

BGV Parameters: $m = 32767$, $L = 29$, $\mathbb{Z}_2[X]/\langle\Phi_m(X)\rangle = \prod_{i=1}^{1800} \mathbb{F}_{2^{15}}$.

Table 4.1: Performance of Wildcard Matching Algorithm `wcMatch`

while waiting for the wildcard matching to complete.

4.4.2 Experiment Results

We present the results of two experiments, using a single ciphertext worth of attributes to measure the performance of the algorithm and protocols. In the tables with the results, we combine several steps together as some of these steps do not consume much time. The first step constitutes an exact pattern match on the shifted attribute string $\overline{A}^{(i)}$, the second step computes $\overline{z}_i$ and the final step $\overline{\gamma}_i$. Within the first step, we note that the computation of $\overline{Y}_i$ takes the longest time, around 65 – 75% of the reported time, as the EQ algorithm is the most complex portion; the others are simple ciphertext additions.

Performance Evaluation of `wcMatch`. The first experiment considers the running time of the wildcard matching algorithm `wcMatch` with three different η 's, 35, 45 and 55 with results given in Table 4.1. The general trend of the results is within expectations, where the total and amortised time increases as η gets larger. This is because fewer ciphertexts operations are performed and more attribute strings can be packed into a single ciphertext when η is small. Both factors contribute to the significant difference in amortised time.

Performance Evaluation of `wildcardConjunction`. The second experiment considers the performance of Query (4.Q1) with $|J_1| = 1$ and $|J_2| = 5$ and its results are in Table 4.2. The trend largely follows that of experiment one, which is not surprising given their similarities. Due to the availability of threads, the exact matching conditions did not significantly impact the protocol's running time. Exact matchings were performed in parallel to wildcard matching, the EQ tests in both algorithms were done simultaneously and $\overline{t}$ of `exMatch` was computed when $\overline{z}_i$'s of the wildcard matching were evaluated. This caused the respective times in the first two columns to be longer than the first experiment.

String Length (Bytes)	# of Strings in a Ciphertext	Evaluate $\bar{A}^{(i)}, \dots, \bar{S}_i$	Evaluate $\bar{z}_i$	Evaluate $\bar{\gamma}$	Total Time	Amortised Time
35	51	29.98s	3.13s	1.11s	34.22s	0.67s
45	40	33.71s	3.26s	1.22s	38.19s	0.96s
55	32	37.49s	3.36s	1.29s	42.14s	1.32s

BGV Parameters: $m = 32767$, $L = 31$, $\mathbb{Z}_2[X]/\langle\Phi_m(X)\rangle = \prod_{i=1}^{1800} \mathbb{F}_{2^{15}}$.

Table 4.2: Performance of Conjunctive Query `wildcardConjunction` with 1 `wcMatch` and 5 `exMatch` Conditions

Overall Comments. The results for both experiments when the attribute string length is 45 are very close, within 10% of each other. This could be because all the `EQ` executions took similar amounts of time, or the slowest threads ended at around similar times. With even more concurrent threads when the string length is 55, their first columns report near identical times as well. This seems to indicate that multi-threading is very beneficial for the protocols.

4.4.3 Comparisons with Related Work

The experiments presented in this section is the same as that by Kim *et al.* [128]. However, there is a dramatic difference between the performance of the algorithms reported by us and them, despite running the same code. We speculate that it could be due to a couple of reasons.

The first is our test platform is much more powerful, featuring a newer CPU with higher clock speed and enough threads to concurrently evaluate all the loops in the algorithm and protocol. In contrast, Kim *et al.* [128] required 2-3 cycles to evaluate the same loops. Secondly, since their work, HELib [102] has been improved significantly. Halevi and Shoup [101] added several improvements to linear map evaluations, which is a key component of the `EQ`, `wcMatch` and `exmatch` algorithms and claimed a 30-75 times speed-up compared to their previous implementations. Some of that could have impacted the results in Tables 4.1 and 4.2.

Besides that, there were two closely related work, discussed in Section 4.2, that also relied of FHE by Yasuda *et al.* [206] and Saha and Koshiba [187]. However, the purpose of their protocols are to match encrypted patterns to strings that are in the clear. Moreover, translating them to handle compound queries like our setting incurs a lot of additional costs, making it hard to get a reasonable comparison.

Chapter 5

Private Compound Comparison Queries

5.1 Introduction

In this chapter, we consider private database protocols for order comparisons on keys which are integers. Order comparison is an important database functionality, enabling queries that desire keys within a particular range, sorted values or minimums and maximums of datasets.

A lot of work has been done to address equality conditions in PDQ; there is a large variety of techniques ranging from property-preserving [180, 199], structured [111] and homomorphic encryption [31, 83, 53, 127, 54, 129]. On the other hand, order conditions — less than $<$, less than or equal $\leq$, greater than $>$ and greater than or equal $\geq$ — are harder to achieve. Most techniques in the literature used order-preserving (OPE) [3, 29, 180, 30, 178, 198, 121] or order-revealing (ORE) encryption [51, 44, 135] or interactive protocols [64, 83, 176, 72]. The former type, OPE and ORE, leak information on encrypted data, the result of comparisons between ciphertexts and in some cases, several bits of the data as well. On the other hand, the latter type is highly interactive and requires a lot of communication between client and server. Both do not compose easily and often may reveal more information than the minimum necessary.

To the best of our knowledge, only Cheon *et al.* [53, 54] looked into the potential of FHE for this problem. They applied Boolean circuits for comparisons and constructed a PDQ protocol for comparisons with encrypted bits. The main drawback is that communication efficiency is low because the data is represented in bits; even with SIMD operations, the communication efficiency is only reduced by a constant factor.

Kim *et al.* [129] used finite extension fields to significantly improve the performance for PDQs with equality conditions but did not consider order conditions. Constructing polynomials to evaluate functions over finite extension fields is not simple and tend to be complex. Jäschke and Armknecht [110] tried to use finite extension fields for addition but found them to be less efficient due to the large polynomials involved.

In this work, we improve the performance of order comparisons with FHE by exploiting the finite extension fields that are native plaintext spaces of most RLWE FHE schemes. Instead of storing 1 bit per slot, we fit an ℓ -bit integer into a single slot by exploiting the vector space structure in finite extension fields, $\mathbb{F}_{2^\ell}$. Let the binary representation of a number $x \in [0, 2^\ell)$ be $\sum_{i=0}^{\ell-1} x_i 2^i$, we encode it into the FHE plaintext space as the vector $(x_0, \dots, x_{\ell-1}) \in (\mathbb{F}_2)^\ell \cong \mathbb{F}_{2^\ell}$. Thus, by selecting appropriate parameters such that the plaintext space is a product of d copies of $\mathbb{F}_2^\ell$, we can encode d integers into a single ciphertext, compared to ℓ/d when working with bits. Comparisons are then evaluated using a novel framework for evaluating certain functions over finite extension fields. Let x, y be two integers to be compared, the framework is as follows:

1. Decompose x, y into smaller blocks, B_i^x, B_i^y , which are vectors in “lower dimensions”;
2. Evaluate a couple of polynomials over these blocks, obtaining some intermediate results;
3. Combine the intermediate results from the previous step to obtain the final answer.

Applying this algorithm, we design PDQ protocols for compound queries where the conditions on keys are a mix of equality and order comparisons. Suppose we have a database $\mathcal{D} = (\alpha_1, \dots, \alpha_n)$ with ρ keys and a single value as defined in Section 3.4.1, some examples of compound queries we can handle include: For some indices $j, r \in [\rho]$,

```

select  $\alpha_i.v$  from  $\mathcal{D}$  where  $\alpha_i.k_j \leq 10$  and  $\alpha_i.k_r = 100$ ;
select  $\alpha_i.v$  from  $\mathcal{D}$  where  $2 \leq \alpha_i.k_j \leq 40$ ;
select  $\alpha_i.v$  from  $\mathcal{D}$  where  $\alpha_i.k_j \triangleleft 50$ ,

```

where $\triangleleft$ is one of $<, \leq, >$ or $\geq$.

Organization of the Chapter. In Section 5.2, we review the literature on private database queries for order comparisons. Following that, in Section 5.3, we describe our order comparison algorithm and apply it to support compound queries. Finally, in Section 5.4, some experiment results for the algorithms underlying the protocols are introduced and discussed.

5.2 Previous Work

A variety of techniques were used to realize PDQ protocols for comparisons, including OPE, ORE and homomorphic encryption. Order-preserving encryption was first proposed by Agrawal et al. [3] in 2004 and is a type of encryption scheme whose ciphertexts preserve the order of plaintexts. Following its introduction, many schemes were proposed [29, 30, 178, 198, 121] seeking to improve security and performance. The optimal security that OPE could achieve was formalized by Boldyreva et al. [29], indistinguishability under ordered chosen plaintext attack (IND-OCPA) which is strictly weaker than semantic security. Kerschbaum [121] then introduced a stronger security notion, indistinguishability under frequency-analysing ordered chosen plaintext attack (IND-FAOCPA) that additionally considered hiding the frequency of encrypted plaintexts. Despite its performance, OPE schemes do not satisfy standard notions of semantic security and results of any comparison done in a PDQ protocol is leaked to the server. Furthermore, with the exception of the most recent scheme by Roche et al. [184], OPE schemes were shown to be susceptible to inference attacks by Naveed et al. [164] and Grubbs et al. [99].

Generalizing OPE, Boneh et al. [33] proposed order-revealing encryption and realized it with multilinear maps, which are currently impractical and whose security is still unclear. A practical ORE scheme was proposed by Chenette et al [51], but it leaked the first bit that differs between tested encrypted plaintexts in addition to their order. Cash et al. [44] improved on the scheme but still leaked something, the “equality pattern” of the most significant different bits. Durak et al [71] studied various attacks on OPE/ORE schemes, except [135, 184] which were presented concurrently. Similar to the situation for OPE, they showed that when real world data is used, attacks can obtain meaningful information that schemes were designed not to leak. Furthermore, using OPE/ORE in practice might be tricky as most data is non-uniform and differs from the scenarios considered in formal security notions.

PDQ protocols for comparisons exploiting homomorphic encryption include the work by Damgård et al. [64] who applied additively homomorphic encryption to perform comparisons. However, the PDQ protocol obtained with Damgård et al.’s comparison protocol requires a non-constant number of rounds, every set of records in each ciphertext would require about five rounds of communication to determine if the client should recover the data within. Other work includes Gentry et al. [83] who utilized oblivious RAM and an additive homomorphic encryption scheme with ring switching capabilities. However, this protocol also requires many rounds of communication between client and server to perform queries. Using FHE, Cheon et al. [53, 54] focused on Boolean circuits to compute the

less than comparison test which impose huge communication costs, although they only require one round of communication. For fast and almost depth-free addition on numeric data, they opted to use large integer rings $\mathbb{Z}_{2^{15}}$. This differs from using finite extension field which have more algebraic structure. However, this choice increased the time taken to perform comparisons although addition is improved, and does not actually changed the communication complexity of their protocols as elements that could be compared are still encrypted as bits.

5.3 Private Compound Order Comparison Protocols

Types of Order Comparisons. For a totally ordered set $(A, \leq_A)$ and elements $x, y \in A$, we formally define four functions

$$\begin{aligned} \text{LT}_A(x, y) &= \begin{cases} 1, & \text{if } x <_A y; \\ 0, & \text{otherwise,} \end{cases} & \text{LEQ}_A(x, y) &= \begin{cases} 1, & \text{if } x \leq_A y; \\ 0, & \text{otherwise,} \end{cases} \\ \text{GT}_A(x, y) &= \begin{cases} 1, & \text{if } x >_A y; \\ 0, & \text{otherwise,} \end{cases} & \text{GEQ}_A(x, y) &= \begin{cases} 1, & \text{if } x \geq_A y; \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Unless confusion arises, we omit the set A in order notations. In practice, only two of the functions LT and GT need to be considered. This is because the other two, GEQ and LEQ , respectively, are complements. Therefore, we can use the formulas $\text{GEQ}(x, y) = 1 - \text{LT}(x, y)$ and $\text{LEQ}(x, y) = 1 - \text{GT}(x, y)$ to compute them.

Encoding Integers. We consider messages in $[0, 2^\ell)$, which can be evaluated with the Boolean circuits described by Cheon *et al.* [53] by employing their binary representation. Note that p -ary representations, for $p > 2$, can be used too but we restrict ourselves to binary for simplicity. However, their methods require ℓ ciphertexts or ℓ slots in a ciphertext with SIMD techniques.

With plaintext space $\mathbb{F}_{2^\ell}$ however, these messages can be stored in a single slot. The question becomes how should the encoding to $\mathbb{F}_{2^\ell}$ be done to allow order comparisons on

messages. For our algorithms, we simply use the encoding map

$$\begin{aligned} \text{encode} : \mathcal{M} = [0, 2^\ell) &\longrightarrow \mathcal{P} = (\mathbb{F}_2)^\ell \cong \mathbb{F}_{2^\ell} \\ x = \sum_{i=0}^{\ell-1} x_i 2^i &\longmapsto (x_0, \dots, x_{\ell-1}) = \sum_{i=0}^{\ell-1} x_i X^i \end{aligned}$$

for an indeterminate X where $\mathbb{F}_{2^\ell} = \mathbb{Z}_2[X]/\langle f(X) \rangle$ for some irreducible polynomial f of degree ℓ . The induced order on $\mathbb{F}_{2^\ell}$ is simply the lexicographic order with respect to t .

5.3.1 A Parameterized Algorithm for Order Comparisons

A Naive Order Comparison Algorithm for $\text{LT}_{\mathbb{F}_{2^\ell}}$. By casting the order relations over $\mathbb{F}_{2^\ell}$ into functions $\text{LT}_{\mathbb{F}_{2^\ell}}(x, y)$ and $\text{GT}_{\mathbb{F}_{2^\ell}}(x, y)$, we can use Lagrange interpolation (Theorem 3.3.4) to obtain bi-variate polynomial expressions, $f_{\text{LT}}(x, y)$ and $f_{\text{GT}}(x, y)$, that respectively evaluate those functions. Furthermore, with depth-free Frobenius map evaluations in RLWE FHE schemes, the depth of these polynomial expressions are a manageable $\log \log(2^\ell + 1) \approx 1 + \log \ell$. However, evaluating $f_{\text{LT}}(x, y)$ and $f_{\text{GT}}(x, y)$ is very inefficient as they each have up to $2^{2\ell} - 1$ coefficients and therefore require $O(2^{2\ell} - 1)$ multiplications by Horner's rule. The FHE schemes that support depth-free Frobenius map evaluations require a different approach which has even more multiplications.

Overview of the Parameterized Algorithm. Instead, we take advantage of the vector space representation of $\mathbb{F}_{2^\ell}$ to obtain a more efficient algorithm. Roughly speaking, we first break down the degree $\ell - 1$ polynomial encoding an integer into ℓ/r many r -bit blocks which lie in a subspace $\mathcal{P}_r = \{(a_0, \dots, a_{r-1}, 0, \dots, 0) = \sum_{i=0}^{r-1} a_i t^i \mid a_i \in \{0, 1\}\}$ of $\mathbb{F}_{2^\ell}$. Then, these blocks are evaluated with $\text{LT}_{\mathcal{P}_r}$ or $\text{GT}_{\mathcal{P}_r}$ and $\text{EQ}_{\mathcal{P}_r}$ functions defined for elements in $\mathcal{P}_r$. Finally, we compute the result of the original comparison by combining the results of the evaluations appropriately.

Extraction. We present the technique assuming that every block has the same size, but it is possible to use differently sized blocks with some adjustment. For $r \geq 1$ such that $r \mid \ell$, define $\text{Extract}_r(x, k)$ for $x = \sum_{i=0}^{\ell-1} x_i t^i \in \mathbb{F}_{2^\ell}$ and $k \in \{0, \dots, \ell/r - 1\}$ as follows

$$\text{Extract}_r(x, k) = \sum_{i=0}^{r-1} x_{kr+i} t^i.$$

Intuitively, the function projects the kr -th to $(k+1)r - 1$ -th coefficients of $x \in \mathbb{F}_{2^\ell}$ to

the first r monomials, equivalently the kr -th to $(k+1)r-1$ -th elements of the vector representation of $x \in (\mathbb{F}_2)^\ell$ to the first r dimensions. $\text{Extract}_r(x, k)$ is done in practice, by finding the correct linear map $T_{r,k}$ and associated constants $\{\rho_{T_{r,k},0}, \rho_{T_{r,k},1}, \dots, \rho_{T_{r,k},\ell-1}\}$ defined by Theorem 3.3.7, so that we can use depth-free Frobenius map evaluations.

Important Functions on $\mathcal{P}_r \times \mathcal{P}_r$. Here, we describe the functions on the subspace $\mathcal{P}_r$ that is needed to compute the comparisons. For $x, y \in \mathcal{P}_r$ and considering the inherited order from $\mathbb{F}_{2^\ell}$,

$$\begin{aligned} \text{LT}_{\mathcal{P}_r}(x, y) &= \begin{cases} 1, & \text{if } x <_{\mathcal{P}_r} y; \\ 0, & \text{otherwise,} \end{cases} & \text{GT}_{\mathcal{P}_r}(x, y) &= \begin{cases} 1, & \text{if } x >_{\mathcal{P}_r} y; \\ 0, & \text{otherwise,} \end{cases} \\ \text{EQ}_{\mathcal{P}_r}(x, y) &= \begin{cases} 1, & \text{if } x = y; \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Remark 5.3.1. To get a polynomial expression for functions defined on $\mathcal{P}_r \times \mathcal{P}_r$, we simply modify the Lagrange interpolation (Theorem 3.3.4) procedure to only consider elements in $\mathcal{P}_r$ rather than the entire field $\mathbb{F}_{2^\ell}$. The polynomial expressions obtained will have degree at most 2^r and can be evaluated at $\log r$ depth with depth-free Frobenius map evaluation and $O(2^{2r})$ multiplications.

The Comparison Algorithm. The comparison algorithm for LT and GT follow the same structure and so we present it for LT only. The algorithm for GT is obtained by replacing all instances of LT with GT. For LEQ and GEQ, we subtract the result of GT and LT respectively from an encryption of 1.

Armed with all the components, we now describe the complete algorithm for less than comparisons for elements in $\mathbb{F}_{2^\ell}$, $\text{LT}_{\mathbb{F}_{2^\ell}}(x, y)$ for $x, y \in \mathbb{F}_{2^\ell}$. Figure 5.1 offers an idea of the procedure.

For $r \leq \ell$ such that $r \mid \ell$ and $x = (x_0, x_1, \dots, x_{\ell-1}), y = (y_0, y_1, \dots, y_{\ell-1})$,

1. (*Extractions*) Compute $(B_1^x, \dots, B_{\ell/r-1}^x)$ and $(B_1^y, \dots, B_{\ell/r-1}^y)$ where the i -th block, $B_i^\alpha = \sum_{j=0}^{r-1} \alpha_{ir+j} t^j = \text{Extract}_{\mathcal{P}_r}(\alpha, i) \in \mathcal{P}_r$ with $\alpha \in \{x, y\}$.
2. (*Comparisons*) Compute $(B_1^{\text{EQ}}, \dots, B_{\ell/r-1}^{\text{EQ}})$ and $(B_1^{\text{LT}}, \dots, B_{\ell/r-1}^{\text{LT}})$, where $B_i^\beta = \beta_{\mathcal{P}_r}(B_i^x, B_i^y) \in \mathbb{F}_2$ for $\beta \in \{\text{EQ}, \text{LT}\}$.
3. (*Assembly*) $\text{LT}_{\mathbb{F}_{2^\ell}}(x, y) = B_{\ell/r-1}^{\text{LT}} + \sum_{j=1}^{\ell/r-2} B_j^{\text{LT}} \prod_{i=j+1}^{\ell/r-1} B_i^{\text{EQ}}$.

Analysis of the Algorithm. Intuitively, the algorithm is an extension of the Boolean

Figure 5.1: Visual Representation of the LT Algorithm

circuit for $\text{LT}_{\{0,1\}^\ell}$ to $\text{LT}_{\{0,\dots,2^r-1\}^{\ell/r}}$ but with arithmetic circuits in $\mathbb{F}_{2^\ell}$.

Correctness. The key observation for $a < b$, with $a, b \in [0, 2^\ell)$, is for two plaintexts $x = \text{encode}(a), y = \text{encode}(b) \in \mathbb{F}_{2^\ell}$ and block size r ,

1. if the first block of x , $B_{\ell/r-1}^x = (x_{\ell-r}, \dots, x_{\ell-1}, 0, \dots, 0)$ is smaller than the first block of y , $B_{\ell/r-1}^y = (y_{\ell-r}, \dots, y_{\ell-1}, 0, \dots, 0)$, i.e., $B_{\ell/r-1}^x <_{\mathcal{P}_r} B_{\ell/r-1}^y$, then $x <_{\mathbb{F}_{2^\ell}} y$ and correspondingly $a < b$;
2. for $j \in \{0, \dots, \ell/r-2\}$, if the j -th block of x , $B_j^x <_{\mathcal{P}_r} B_j^y$, and for $j+1 \leq i \leq \ell/r-1$, $B_i^x = B_i^y$, then for similar reasons, $a < b$;
3. Otherwise, $a \geq b$.

Since each message is encoded into a single element, we use the extraction step to obtain the necessary blocks. After that, we analyse the second step, where functions on the blocks are evaluated. The functions $\text{EQ}_{\mathcal{P}_r}(B^x, B^y)$ and $\text{LT}_{\mathcal{P}_r}(B^x, B^y)$ output 1 if two blocks $B^x, B^y \in \mathcal{P}_r$ are equal or $B^x < B^y$ in the inherited order from $\mathbb{F}_{2^\ell}$ and 0 otherwise. With correct extraction, the results of the function will be correct as well, i.e.

$$B_j^{\text{EQ}} = \begin{cases} 1, & \text{if } B_j^x = B_j^y; \\ 0, & \text{otherwise;} \end{cases} \quad B_j^{\text{LT}} = \begin{cases} 1, & \text{if } B_j^x <_{\mathcal{P}_r} B_j^y; \\ 0, & \text{otherwise;} \end{cases}$$

Then, results of these function on the blocks are combined with the following formula,

$$\text{LT}_{\mathbb{F}_2^\ell}(x, y) = B_{\ell/r-1}^{\text{LT}} + \sum_{j=1}^{\ell/r-2} B_j^{\text{LT}} \prod_{i=j+1}^{\ell/r-1} B_i^{\text{EQ}}. \quad (5.1)$$

Equation (5.1) expresses the following logic,

1. $B_{\ell/r-1}^{\text{LT}}$ captures the condition where the first block of x , $B_{\ell/r-1}^x$, is less than the same of y , $B_{\ell/r-1}^y$;
2. for $j \in [\ell/r - 2]$, the term $B_j^{\text{LT}} \prod_{i=j+1}^{\ell/r-1} B_i^{\text{EQ}}$ captures the condition where the j -th block of x is less than y and all more significant blocks are equal.

Since the conditions of each term are disjoint, a simple sum suffices to obtain the final result.

Security. It is straightforward to show that the security of the algorithm follows from the semantic security of the FHE scheme used for an honest-but-curious server. In every procedure, the algorithm only works with ciphertexts and so cannot distinguish between encryptions of different inputs of the same length, ℓ bits. Since the maximum size of the plaintext space is given to the server to run the algorithms, no new information is revealed by running the order comparison algorithms.

Efficiency. The efficiency of the algorithm is best analysed per step. In the first step, we perform ℓ/r extractions per input, each of which is a sum of $\ell - 1$ terms, where each term is a multiplication of a Frobenius mapped input with a constant. The Frobenius mapped inputs can be computed first and used for every extraction so we save on some moderately expensive operations. In total, 2ℓ depth-free Frobenius map evaluations and $2\ell/r(\ell - 1)$ which is around $2\ell^2/r$ constant multiplications are used.

In Step 2, $f_{\text{EQ}_{\mathcal{P}_r}}(x, y)$ and $f_{\text{LT}_{\mathcal{P}_r}}$ are evaluated on ℓ/r many pairs of blocks. These polynomial expressions would have degree 2^r in each variable, and we exploit depth-free Frobenius maps to compute them with the lowest multiplicative depth possible. For small r , it might be worthwhile to consider a quicker polynomial expression evaluation method at the expense of slightly more depth. First, we use the following procedure to obtain all the monomials of x and y required to evaluate $f_{\text{EQ}_{\mathcal{P}_r}}(x, y)$ and $f_{\text{LT}_{\mathcal{P}_r}}$.

1. Use $\log r$ depth-free Frobenius maps to get the some powers of 2 of each input, $(x, x^2, x^4, \dots, x^{2^{\lceil \log r \rceil}}), (y, y^2, y^4, \dots, y^{2^{\lceil \log r \rceil}})$ without multiplications.

2. For the other monomials x^i (respectively y^i), we obtain them by multiplying at most $\log i$ elements from the list we generated in the previous step. (According to the binary representation of i).

This portion requires at most $\sum_{i=1}^{2^r} \log i = \log(2^r!) \sim r2^r$ multiplications per variable. To simplify things, we consider these two polynomial expressions in $\mathbb{F}_{2^\ell}[x][y]$ and first compute the coefficients of the powers of y in $\mathbb{F}_{2^\ell}[x]$. With the pre-evaluated monomials, this does not require additional ciphertext multiplications. Finally, we use $2^r - 2$ ciphertext multiplications to obtain the evaluation of the polynomial expressions $f_{\text{EQ}_{\mathcal{P}_r}}(x, y)$ and $f_{\text{LT}_{\mathcal{P}_r}}$. In total, we use $2^r - 2 + 2(r2^r) < (2r + 1)2^r$ ciphertext multiplications and $\log r$ depth-free Frobenius map evaluations to obtain the results of one polynomial expression. Therefore, we use $2(\ell/r)(2r + 1)2^r$ ciphertext multiplications and $(\ell/r) \log r$ Frobenius map evaluations to compute Step 2 of the comparison algorithm. The last step uses $(\ell/r - 2) + (\ell/r - 1) \ll 2^\ell$ ciphertext multiplications to obtain all the terms of Equation (5.1). One of the main properties of the parameterized algorithm is that we are trading off between computing more in Steps 1 and 3 and Step 2. As r increases, the amount of computation for Steps 1 and 3 decrease proportionally to ℓ/r , but increases proportionally to 2^r for Step 2.

As for the multiplicative depth of the parameterized algorithm, Step 2 uses $1 + \lceil \log r \rceil$ depth and Step 3 uses $1 + \lceil \log(\ell/r) \rceil$ for a total of at most $4 + \lceil \log \ell \rceil$ which is comparable to the naive comparison algorithm and traditional Boolean circuit method.

Comparison for Encrypted Inputs. We call the algorithm for less than comparisons on encrypted data $\text{LT}(\bar{x}, \bar{y})$. Before applying the algorithm on encrypted data, we need to decide on its parameters and generate the required data to run the algorithm.

- For some $r \mid \ell$, the necessary constants, from Theorem 3.3.7, for the linear maps $\{T_{r,i}\}_{i=1}^{\ell/r-1}$, $\{\{\rho_{T_{r,i},0}, \dots, \rho_{T_{r,i},\ell-1}\}\}_{i=1}^{\ell/r-1}$.
- Polynomial expressions of $\text{LT}_{\mathcal{P}_r}(x, y)$ and $\text{EQ}_{\mathcal{P}_r}(x, y)$, $g_{\text{LT}_{\mathcal{P}_r}}(x, y)$ and $g_{\text{EQ}_{\mathcal{P}_r}}(x, y)$ respectively.

Then, the algorithm for LT and its complement GEQ, GT and its complement LEQ are as follows:

- $\bar{t} \leftarrow \text{LT}(\bar{x}, \bar{y})$:
 1. For $i \in \{0, \dots, \ell/r - 1\}$, decompose x and y , obtaining $\bar{B}_i^x \leftarrow \sum_{j=0}^{\ell-1} \rho_{T_{r,i},j} \bar{x}^{2^j}$ and $\bar{B}_i^y \leftarrow \sum_{j=0}^{\ell-1} \rho_{T_{r,i},j} \bar{y}^{2^j}$.

2. For $i \in \{0, \dots, \ell/r - 1\}$, compute the block-wise evaluations of LT and EQ, $\overline{B}_i^{\text{LT}} = \text{Eval}(f_{\text{LT}_{\mathcal{P}_r}}, \overline{B}_i^x, \overline{B}_i^y)$ and $\overline{B}_i^{\text{EQ}} = \text{Eval}(f_{\text{EQ}_{\mathcal{P}_r}}, \overline{B}_i^x, \overline{B}_i^y)$.
 3. Compute $\bar{t} = \overline{B}_{\ell/r-1}^{\text{LT}} + \sum_{j=1}^{\ell/r-2} \overline{B}_j^{\text{LT}} \prod_{i=j+1}^{\ell/r-1} \overline{B}_i^{\text{EQ}}$.
- $\bar{t} \quad \text{GEQ}(\bar{x}, \bar{y})$:
1. Compute $\bar{t} = \bar{1} - \text{LT}(\bar{x}, \bar{y})$.
- $\bar{t} \quad \text{GT}(\bar{x}, \bar{y})$:
1. Compute $\bar{t} = \text{LT}(\bar{y}, \bar{x})$.
- $\bar{t} \quad \text{LEQ}(\bar{x}, \bar{y})$:
1. Compute $\bar{t} = \bar{1} - \text{GT}(\bar{x}, \bar{y})$.

Remark 5.3.2. Besides simply computing order comparisons on encrypted data, FHE schemes can provide slightly more. Notice that we swapped the order of the inputs to LT to evaluate GT. We can achieve this swap without the server knowing by replacing $\text{LT}(\bar{x}, \bar{y})$ with $\text{LT}(\bar{x} + \bar{d}(\bar{y} - \bar{x}), \bar{y} + \bar{d}(\bar{x} - \bar{y}))$, where

$$d = \begin{cases} 0, & \text{if the client wishes to evaluate } \text{LT}(\bar{x}, \bar{y}); \\ 1, & \text{if the client wishes to evaluate } \text{GT}(\bar{x}, \bar{y}); \end{cases}$$

Note that this only increases the number of ciphertext multiplications and depth of the algorithm by 1.

Besides an encryption of d as above, we send another encrypted bit $\bar{e}$, $e \in \{0, 1\}$, we obtain the following hidden order comparison algorithm:

- $\bar{t} \quad \text{hiddenOrder}(\bar{x}, \bar{y}, \bar{d}, \bar{e})$:
1. Compute $\bar{t}' = \text{LT}(\bar{x}, \bar{y})$ with $\text{LT}(\bar{x} + \bar{d}(\bar{y} - \bar{x}), \bar{y} + \bar{d}(\bar{x} - \bar{y}))$.
 2. Compute $\bar{t} = \bar{e} - \bar{t}'$.

5.3.2 Private Compound Comparison Protocols

Let $\mathcal{D} = (\alpha_1, \dots, \alpha_n)$ be a database, with $\alpha_i = (\alpha_i.v, \alpha_i.k_1, \dots, \alpha_i.k_\rho)$ as defined in Section 3.4.1; we use $k_{i,j}$ to denote the key $\alpha_i.k_j$ for $i \in [n]$ and $j \in [\rho]$. As the order comparison algorithms return an encrypted bit denoting the result of the comparison, we can build protocols to support conjunctive, disjunctive and threshold conjunctive queries

using the protocol structure in Section 3.4.2. Furthermore, they are interchangeable with the EQ, wcMatch and exMatch algorithms from previous chapters.

A Private Range Query Protocol. First, we present a query, a special case of a general conjunctive protocol, which desires all $\alpha_i.v$ that satisfy for some index $j \in [\rho]$:

$$(\bar{\delta}_0 \leq \bar{k}_{i,j} \leq \bar{\delta}_1), \quad (5.Q1)$$

where δ_0, δ_1 are integers in $[0, 2^\ell)$. Formally,

– $\{\bar{\gamma}_i\}_{i=1}^n \leftarrow \text{rangeMatch}(\bar{k}_{i,j}, \bar{\delta}_0, \bar{\delta}_1)$: For $i \in [n]$:

1. Compute

$$\bar{\beta}_{i,z} = \begin{cases} \text{GEQ}(\bar{k}_{i,j}, \bar{\delta}_0), & \text{for } z = 0; \\ \text{LEQ}(\bar{k}_{i,j}, \bar{\delta}_1), & \text{for } z = 1. \end{cases}$$

2. Compute $\bar{\gamma}_i = \bar{\alpha}_{i.v} \cdot \bar{\beta}_{i,0} \cdot \bar{\beta}_{i,1}$.

A Private Disjunctive Query Protocol with Multiple Conditions. This query seeks all $\alpha_i.v$ that satisfy, for some index $j \in [\rho]$ and set of indices $J \subset [\rho] \setminus \{j\}$, the following conditions

$$(\bar{k}_{i,j} \triangleleft \bar{\delta}) \vee \bigvee_{j' \in J} (\bar{k}_{i,j'} = \bar{\delta}_{j'}), \quad (5.Q2)$$

where $\delta, \delta_{j'}$, for $j' \in J$, are integers in $[0, 2^\ell)$ and $\triangleleft$ is one of $<, \leq, >$ or $\geq$.

– $\{\bar{\gamma}_i\}_{i=1}^n \leftarrow \text{orderDisjunction}(\bar{k}_{i,j}, \bar{\delta}, \bar{d}, \bar{e}, \{\bar{\delta}_{j'}\}_{j' \in J})$: Let $\triangleleft$ be determined by $\bar{\delta}, \bar{d}$ and $\bar{e}$. For $i \in [n]$:

1. Compute

$$\bar{\beta}_{i,z} = \begin{cases} \bar{1} - \text{hiddenOrder}(\bar{k}_{i,j}, \bar{\delta}, \bar{d}, \bar{e}), & \text{for } z = j; \\ \bar{1} - \text{EQ}(\bar{k}_{i,j}, \bar{\delta}_z), & \text{for } z \in J. \end{cases}$$

2. Compute $\bar{\gamma}_i = \bar{\alpha}_{i.v} \cdot \left(\bar{1} - \bar{\beta}_{i,j} \cdot \prod_{j' \in J} \bar{\beta}_{i,j'} \right)$.

Analysis of the Protocols. Both protocols are similar to the ones from Section 3.4.2 and 4.3.2, replacing the EQ or wcMatch, exMatch algorithms with order comparisons while retaining their structure.

Correctness and Security. The correctness of the protocols follow directly from the correctness of the comparison algorithms and the description of the protocol. Although the algorithms used differ, the shared protocol structure means that we only need to check that the order comparison algorithms do not reveal any information about the data processed and result of the comparisons. The security of the order comparison algorithms are based on the semantic security of the FHE scheme used. We summarize the security of the protocols in the following theorem.

Theorem 5.3.3. *Assuming that the underlying FHE scheme is semantically secure, a PPT honest-but-curious server learns no additional information than in the ideal model where a trusted third party replaces the server and executes the protocol with the same inputs.*

The proof for the theorem omitted as it is very similar to that of Theorem 4.3.3. The main difference in the proofs lies in the different algorithms used, **wcMatch** and **exMatch** in Theorem 4.3.3 and **LT**, **LEQ**, **GT**, **LEQ** and **hiddenOrder** here.

Efficiency. Now, we determine the multiplicative depth of the protocols. In the protocol for Query (5.Q1), we use two instances of **LEQ**, which was analysed to have $5 + \lceil \log \ell \rceil$ depth earlier. Then, Step 2 multiplies 3 ciphertexts together. That yields an upper bound on the multiplicative depth of $7 + \lceil \log \ell \rceil$.

For the protocol for Query (5.Q2), **hiddenOrder**, due to the encrypted $\bar{f}_{\text{ord}}$, requires at most $5 + \lceil \log \ell \rceil$ depth, one more compared to the other four algorithms. This is greater than the $\lceil \log \ell \rceil$ depth of the **EQ** algorithm so the depth of Step 1 is that of **hiddenOrder**. The total number of conditions in the query is $|J| + 1$, then including the encrypted value, Step 2 requires $\lceil \log(|J| + 2) \rceil$ depth. Therefore, the overall multiplicative depth is at most $5 + \lceil \log(|J| + 2) \rceil + \lceil \log \ell \rceil$.

5.4 Implementation

In this section, we investigate the performance of the parameterized comparison algorithm in the PDQ protocols proposed in Section 5.3. First, we present the details of the parameters used in the experiments. Then, we describe two possible methods to use SIMD techniques with the order comparison algorithm. Finally, we present and analyse the performance results of the implementation of the order comparison algorithm and two protocols described in Section 5.3.2.

Note that unlike Section 4.4, we are only concerned with the single-threaded performance of the algorithms and protocols.

5.4.1 Implementation Details

BGV Instance in HELib. In the experiments, we use the BGV scheme with the m -th cyclotomic polynomial modulus $\Phi_m(X)$, where $m = 10261$ and $m = 13367$. The plaintext space decomposes into $\prod_{i=1}^{330} \mathbb{F}_{2^{30}}$ with 330 slots in the first case and $\prod_{i=1}^{326} \mathbb{F}_{2^{41}}$ with 326 slots in the second. In the first two experiments, where $m = 10261$, we use $L = 11$ which yields a security level of at least 80 bits. For the last experiment, as `orderDisjunction` requires larger multiplicative depth $L = 13$, we use $m = 13367$ to limit overhead on 30-bit integers without reducing the security level and this yields a security level of at least 90 bits.

Also, we note that there is a slight discrepancy between the theoretical depth of $9 = 4 + \lceil \log 30 \rceil$ needed to evaluate the comparison algorithm and the actual levels used. This is because in practice, Frobenius map evaluations, shifts and rotations are not truly depth-free. Although they do not add as much noise as ciphertext multiplications, there is still some noise growth after each evaluation. Therefore, we use slightly more levels to accommodate that.

Packing Strategies with SIMD. There are two ways to use the SIMD techniques in Section 3.2.2 proposed by Smart and Vercauteren [193]. For the first method, we encode integers into element of $\mathcal{P}$ and encrypt one element per slot (OEPS). This allows us to process a lot of data simultaneously each time we run the algorithm.

In the second method, we encode integers into a vector of blocks in $(\mathcal{P}_r)^{\ell/r}$ and encrypt one block per slot (OBPS); this case was considered by Cheon et al. [53] when the block size is 1. With this block-wise technique, we skip Step 1, evaluate only one instance of $g_{\text{LT}_{\mathcal{P}_r}}$ and $g_{\text{EQ}_{\mathcal{P}_r}}$ in Step 2 then use some depth-free shifts to compute Step 3 of the comparison algorithm. However, we then have to perform several shifts to properly compute $\bar{\gamma}_i$ instead of the straightforward multiplication of the original protocol. These two methods are compared in the first experiment of Section 5.4.2.

Optimizations for rangeMatch. Query (5.Q1) is answered by evaluating two order comparisons, which gives us some room for optimization. Each order comparison algorithm first uses the `Extract` algorithms to decompose the inputs into smaller blocks. Since both comparisons in `rangeMatch` share a common input, namely $\bar{k}_{i,j}$, we avoid redundant work by extracting it once and then using the blocks $\{\bar{B}_z^{k_{i,j}}\}_{z=0}^{\ell/r-1}$ in both order comparisons.

Block Size	# Packed	LT Evaluation			Select Value	Total Time	Amortised Time
		Step 1	Step 2	Step 3			
1	330	15.19 sec	2.30 sec	9.09 sec	0.05 sec	26.63 sec	0.08 sec
2	330	8.49 sec	12.27 sec	3.26 sec	0.05 sec	24.07 sec	0.07 sec
3	330	6.38 sec	21.69 sec	1.77 sec	0.05 sec	29.90 sec	0.09 sec

BGV Parameters: $m = 10261$, $L = 11$, $\mathbb{Z}_2[X]/\langle\Phi_m(X)\rangle = \prod_{i=1}^{330} \mathbb{F}_{2^{30}}$.

Table 5.1: Performance of LT Algorithm for 30-bit Integers (OEPS)

Block Size	# Packed	LT Evaluation		Select Value	Total Time	Amortised Time
		Step 2	Step 3			
1	11	0.062 sec	1.094 sec	0.267 sec	1.423 sec	0.129 sec
2	22	0.731 sec	0.773 sec	0.186 sec	1.690 sec	0.077 sec
3	33	2.044 sec	0.736 sec	0.208 sec	2.987 sec	0.091 sec

BGV Parameters: $m = 10261$, $L = 11$, $\mathbb{Z}_2[X]/\langle\Phi_m(X)\rangle = \prod_{i=1}^{330} \mathbb{F}_{2^{30}}$.

Table 5.2: Performance of LT Algorithm for 30-bit Integers (OBPS)

5.4.2 Experiment Results

Comparison of SIMD Packing Methods. For ease of comparison, we have used the same parameters for both packing methods. A small note on the parameter selection is that, when packed with one element per slot, the LT algorithm can be evaluated on a smaller BGV instance of $m = 7957$ and $L = 9$ with block size 1 and 2 but not 3.

In Tables 5.1 and 5.2, the results show that block size (r) of 2 is the best option regardless of the SIMD packing used. The difference between the two packing methods is also very small, ranging from 1% to 50%, with one element per slot beating the other technique by 10% when the block size is 2. Although the algorithms' running time are very close, the amount of data packed differs tremendously from 10 to 30 times. For situations where latency is important, it would be best to use both.

As block size increases, the algorithm performs better and then becomes slower. This is expected from the efficiency analysis, because larger block sizes mean evaluating exponentially larger bi-variate polynomials in Step 2 while Steps 1 and 3 vary linearly with block size. In the case, where one block is encrypted per slot, the optimized Boolean circuit evaluation turns out to be the least efficient out of the three cases. Although requiring longer running time, more integers can be packed in a single ciphertext which results in much better amortized performance.

Block Size	# Packed	rangeMatch Evaluation			Select Value	Total Time	Amortised Time
		“Step 1”	“Step 2”	“Step 3”			
1	330	22.47 sec	4.73 sec	17.86 sec	0.05 sec	45.11 sec	0.14 sec
2	330	12.52 sec	24.11 sec	6.37 sec	0.05 sec	43.06 sec	0.13 sec
3	330	9.39 sec	42.50 sec	3.44 sec	0.05 sec	55.38 sec	0.17 sec

BGV Parameters: $m = 10261$, $L = 11$, $\mathbb{Z}_2[X]/\langle\Phi_m(X)\rangle = \prod_{i=1}^{330} \mathbb{F}_{2^{30}}$.

Table 5.3: Performance of rangeMatch with 30-bit Integers

Block Size	# Packed	orderDisjunction Evaluation		Select Value	Total Time	Amortised Time
		hiddenOrder	5 EQ			
1	326	44.87 sec	4.20 sec	0.27 sec	49.34 sec	0.15 sec
2	326	38.65 sec	4.24 sec	0.27 sec	43.16 sec	0.13 sec
3	326	46.70 sec	4.30 sec	0.29 sec	51.29 sec	0.16 sec

BGV Parameters: $m = 13367$, $L = 13$, $\mathbb{Z}_2[X]/\langle\Phi_m(X)\rangle = \prod_{i=1}^{326} \mathbb{F}_{2^{41}}$.

Table 5.4: Performance of orderDisjunction with 30-bit Integers, 1 Hidden Order and 5 Equality Conditions.

Performance Evaluation of rangeMatch. The performance of rangeMatch in our experiment is presented Table 5.3. GEQ and LEQ were evaluated in parallel, with Step i of one running first followed by the other before moving to the next step. The general trend is identical to the previous experiments, the best performance comes from using the order comparison algorithms with block size 2. As explained in Section 5.4.1, the optimized protocol does not require double the time taken in Table 5.1, about $1.8\times$ is observed. This is because we are able to reduce the first step of the order comparisons to use only 3 extractions; the entries under “Step 1” reflect this optimization.

Performance Evaluation of orderDisjunction. In Table 5.4, we report the results of running orderDisjunction with $|J| = 5$. For this experiment, a larger BGV instance, $m = 13367$ and $L = 13$ is used to correctly combine the results of the order and equality comparisons with the values in the database. The general trend of the results follow those from the other experiments, with block size 2 being the most efficient overall. From the table, the order comparison takes up over 90% of the time taken to evaluate the query, despite having 5 equality conditions. Multi-threading can definitely be used to improve the situation but single-threaded performance is still important. The large gap between evaluating EQ and the order comparison algorithms highlights the room for improvement.

Part II

Zero-Knowledge and Databases

Chapter 6

Zero-Knowledge Expressive Elementary Database

6.1 Introduction

In this chapter, we study elementary databases (EDB), which are partial functions D mapping a set of keys to values, and consider privacy-preserving queries over D that require answers to come with proofs of correctness satisfying two properties: zero-knowledge and soundness. Zero-knowledge means that proofs cannot reveal any information beyond the truth value of the statement, even $|D|$ has to be hidden from verifiers. Soundness mandates that provers cannot prove contradictory statements. Particularly, we aim to achieve constructions that do not require interaction between the parties.

An EDB $D = \{(x, D(x))\}$ consists of records which are key-value pairs $(x, D(x)) \in [0, 2^\ell] \times [0, 2^\ell]$. A key x are said to be in D if there is some $D(x) \in [0, 2^\ell]$ such that $(x, D(x)) \in D$. If x is not in D , then we assign $D(x) = \perp$ to keep a common syntax. The set of keys in D is denoted with $[D]$ which extend to two-dimensional ranges where $[\mathfrak{R}] = [a_x, b_x]$ for $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$. For values y , we use $D^{-1}(y)$ to denote the set of keys, $x \in D$ such that $D(x) = y$.

Micali, Rabin and Kilian [153] initiated investigations in this area and proposed zero-knowledge sets (ZKS), which allow a prover to commit to some finite set $\mathcal{S}$ and then prove if elements x are in or not in the committed set. They also extended the notion of ZKS to EDB, where the prover proves that the value of some key x is $D(x) \in [0, 2^\ell] \cup \{\perp\}$, obtaining zero-knowledge elementary databases (ZK-EDB). The statements that can be proved by ZKS and ZK-EDB are called membership queries. In this context, soundness for ZKS proofs means that only one of two statements $x \in \mathcal{S}$ and $x \notin \mathcal{S}$ can be proven

for the same $\mathcal{S}$. For EDB, soundness require that no key x can be proven to have two different values in the EDB $\mathcal{D}$.

The construction of Micali *et al.* [153], based on the discrete logarithm assumption, was generalized by Chase *et al.* [48, 49] into a generic design for zero-knowledge elementary databases on top of a new primitive called *mercurial commitments*. Informally, mercurial commitments are commitment schemes where commitments are generated in one of two modes, hard or soft. Hard commitments satisfy the standard binding property while soft commitments are dummy commitments that do not commit the sender to any particular message.

The ZK-EDB constructions from [153, 48, 49] combine mercurial commitments with a Merkle tree [151], with each internal node of the tree containing a mercurial commitment to its two children instead of a hash. The size of the committed database is hidden in an exponentially large tree with dummy commitments which also allows us to compute the commitment of the Merkle tree in polynomial time by deferring computation of sub-trees that do not contain any values.

The literature only considered membership queries over keys for EDB, which is arguably the most basic query in an elementary database. Ostrovsky, Rackoff and Smith [166] adopted a different approach with consistent query protocols and zero-knowledge proofs to obtain zero-knowledge database protocols. In particular, they described a protocol that handled multi-dimensional range queries for committed database with d -dimensional key space. However, their techniques cannot be used to hide the size of the database as it leaks a polynomial upper-bound on the database size. We show that the ZK-EDB constructions of [153, 48, 49] can be extended to support more expressive queries, specifically, membership and range queries over values and records in addition to keys. We call ZK-EDB protocols that can handle this richer set of queries, zero-knowledge expressive elementary databases (ZK-EEDB).

In ZK-EEDB, each query has a different answer depending on the type of input given and also requires a proof of the answer's correctness. To maintain a uniform presentation for all queries, membership queries over keys x returns a record $(x, \mathcal{D}(x))$ if $x \in [\mathcal{D}]$ and an empty set otherwise. Range queries over keys are simply simultaneous membership queries for keys in some range $[a_x, b_x]$. For values, a membership with input value y , requires an answer which is a set of records $\mathcal{L} = \mathcal{D}^{-1}(y)$ that contains all records whose values are y ; a range query over values, given a range $[a_y, b_y]$, work exactly like range queries over keys. Finally, for queries over records, a membership query takes a record (x, y) and returns the record is in $\mathcal{D}$ and $\emptyset$ otherwise, while a range query with range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$ expects the set of records $\mathcal{L} = \{(x_i, y_i)\} = \mathcal{D} \cap \mathfrak{R}$.

The ZK-EEDB construction can be instantiated with any known trapdoor mercurial commitments, such as the number theoretic constructions from [153, 48]. Besides these, we give a new construction of trapdoor mercurial commitments based on the Short Integer Solution problem. This construction is inspired by the original trapdoor mercurial commitment by Micali *et al.* [153] and not implied by the generic construction by Catalano *et al.* [45]. With this, we obtain the first quantum-resistant trapdoor mercurial commitment scheme.

Organization of the Chapter. In Section 6.2, we review the literature on privacy-preserving databases. Next, we revisit important definitions of (trapdoor) mercurial commitments and zero-knowledge elementary databases in Section 6.3. Following that, in Section 6.4, we introduce the formal definitions of ZK-EEDB and next, in Section 6.5, present the main result of the chapter: a generic construction of ZK-EEDB from mercurial commitments. Finally, in Section 6.6, we describe a lattice-based trapdoor mercurial commitment scheme and when combined with the generic constructions, yield a quantum-resistant instantiation of the ZK-EEDB, ZK-EDB and ZKS schemes.

6.2 Previous Work

Zero-knowledge range queries for databases, in the two-party setting, were studied by Ostrovsky *et al.* [166], who constructed protocols for orthogonal d -dimensional range queries for committed databases with $d \geq 1$. With universal arguments from Barak and Goldreich [14] or explicit-hash Merkle trees and random oracles, these can be augmented to provide privacy. However, the protocols cannot hide the size of the database.

Size-hiding protocols stem from the pioneering work by Micali *et al.* [153]; Their techniques were abstracted by Chase *et al.* [48, 49], who gave a size-hiding construction of ZK-EDBs under general assumptions via a new primitive called mercurial commitments. Following that, Catalano, Dodis and Visconti [45] simplified security definitions for (trapdoor) mercurial commitments and showed how they can be derived from one-way functions in the shared random string model.

Besides simplifying assumptions, some have tried to add capabilities to ZK-EDB protocols. Liskov [144] augmented ZK-EDB and considered how to update zero-knowledge databases in the random oracle model. A slightly different notion to ZKS, statistically hiding sets, were proposed by Prabhakaran and Xue [181] to allow for more efficient constructions. Some work was also done to reduce the size of proofs in ZK-EDB, Catalano,

Fiore and Messina [47] introduced a technique to compress proofs of membership in ZK-EDB. Libert and Yung [139] extended their idea and compressed proofs of both membership and non-membership. Both works relied on non-standard assumptions, which Catalano and Fiore [46] managed to remove, yielding a more efficient ZK-EDB protocol under more standard assumptions.

Orthogonal to the previous works, there were studies into stronger security notions in size-hiding database commitments. Gennaro and Micali [80] introduced the notion of independent ZK-EDB, which prevents adversaries from correlating their own committed database with those belonging to honest committers. Chase and Visconti proposed zero-knowledge protocols with a stronger notion of simulation-based security, but sacrifices non-interactivity in the database commitment phase.

While the constructions above all pertain to EDBs, we mention two work that consider slightly different objects. Ghosh, Ohrimenko and Tamasia [89] formalized a new notion of zero-knowledge lists (ZKL). It allows the prover to commit to a list and prove the order in which elements appear within. They gave a construction in the random oracle model in the same work. Goyal *et al.* [96] described black-box constructions of size-hiding database commitments that supports more general queries. However, their objective is orthogonal to this work as they use the “MPC-in-the-head” technique from Ishai *et al.* [109] which requires interaction.

Last but not least, we briefly mention the literature on authenticated data structures [162, 197, 168, 167, 163, 88]. These explore a different problem as they are in a three-party setting, with an honest database owner, faithfully committing to a database but proofs that are generated by an untrusted server. Our setting combines these two parties into a single untrusted database committer. While the three-party setting suits some applications, such as certificate revocation with a trusted revocation authority, it is not universally applicable. One area where the two-party setting is more suitable is in pricing databases, where it is desirable to rule out price discrimination from the database owner. Thus, we focus on the two-party setting which gives greater security while being harder and generally resulting in less efficient schemes.

6.3 Preliminaries

6.3.1 Trapdoor Mercurial Commitments

First introduced by Chase *et al.* [48], (trapdoor) mercurial commitments were used to abstract the construction of zero-knowledge sets (ZKS) and zero-knowledge elementary databases (ZK-EDB) proposed by Micali, Rabin and Kilian [153]. It captures the proper-

ties of the ad-hoc discrete logarithm-based commitment scheme used by Micali *et al.* [153]. Catalano, Dodis and Visconti [45] refined and strengthened the definitions and security notions of the primitive. A generic construction from one-way functions of trapdoor mercurial commitments (TMC) was shown in the shared random string model.

As mentioned earlier, mercurial commitments come in one of two modes: hard commitments work like standard commitments while soft commitments are dummy commitments that do not commit to any message in particular. Not only that, openings for mercurial commitments come in two flavours as well, also called hard and soft. Hard openings function exactly like regular openings for hard commitments. Soft openings, on the other hand, tease that a commitment will hard open only to M if it can be hard opened at all.

Trapdoor mercurial commitments additionally come with a trapdoor for the commitment key that allows fake commitments to be generated and passed off as hard commitments to any desired message. As defined by Catalano *et al.* [45], trapdoor mercurial commitments consists of the following ten algorithms, TMSetup , $\mathbb{H}\text{Commit}$, $\mathbb{H}\text{Open}$, $\mathbb{H}\text{Verify}$, $\mathbb{S}\text{Commit}$, $\mathbb{S}\text{Open}$, $\mathbb{S}\text{Verify}$, MFake , $\mathbb{H}\text{Equivocate}$ and $\mathbb{S}\text{Equivocate}$.

- $(mpk, msk) \leftarrow \text{TMSetup}(1^\lambda)$: Take a security parameter λ as input and output a public mercurial commitment key mpk and secret mercurial trapdoor msk .
- $C \leftarrow \mathbb{H}\text{Commit}(mpk, M; R)$: Take a commitment key mpk , message M and randomness R as inputs and output a hard commitment C for M .
- $\pi \leftarrow \mathbb{H}\text{Open}(mpk, M; R)$: Take a commitment key mpk , message M and randomness R as inputs and output a hard-opening π for $C = \mathbb{H}\text{Commit}(mpk, M; R)$ of M .
- $\mathbb{H}\text{Verify}(mpk, M, C, \pi)$: Take a commitment key mpk , message M , commitment C and hard-opening π as inputs and accept if π proves that C is a valid hard commitment to M and reject otherwise.
- $C \leftarrow \mathbb{S}\text{Commit}(mpk; R)$: Take a commitment key mpk and randomness R as inputs and output a soft commitment C to no particular message.
- $\tau \leftarrow \mathbb{S}\text{Open}(mpk, M, \text{flag}; R)$: Take a commitment key mpk , message M , flag flag and randomness R as input and output τ according to the following:
 - ◊ if $\text{flag} = \mathbb{H}$, then $C = \mathbb{H}\text{Commit}(mpk, M; R)$ and output the soft-opening τ of C to M .

- $\diamond$ Otherwise, $\text{flag} = \mathbb{S}$ and τ is a soft-opening of the soft commitment $C = \mathbb{S}\text{Commit}(mpk; R)$ to message M .
- $\text{SVerify}(mpk, M, C, \tau)$: Take a commitment key mpk , message M , commitment C and soft-opening τ and accept if it is possible for C to be hard opened to M and reject otherwise.
- $C \leftarrow \text{MFake}(mpk; R)$: Take a commitment key mpk and randomness R as inputs and output a “fake” commitment C not tied to any particular message.
- $\pi \leftarrow \mathbb{HEquivocate}(msk, M; R)$: Take a secret trapdoor msk , message M and randomness R and output a valid hard-opening π (*hard-fake*) of the fake commitment $C = \text{MFake}(mpk; R)$ to M .
- $\tau \leftarrow \mathbb{SEquivocate}(msk, M; R)$: Take a secret trapdoor msk , message M and randomness R and output a valid soft-opening τ (*soft-fake*) of the fake commitment $C = \text{MFake}(mpk; R)$ to M .

When a trapdoor mercurial commitment scheme has soft-openings τ of some hard-commitment C consisting of a proper part of its hard-opening π , we call it a *proper* trapdoor mercurial commitment scheme.

To achieve the results in the later sections, we introduce three new algorithms, Explain , EVerify , FakeExplain . These algorithms are used to reveal and verify that some commitment was committed in the soft mode and allow fake commitments from MFake to be explained as soft commitments.

- $\text{Explain}(mpk; R)$: Take a commitment key mpk and randomness R and output R that is used to compute $C = \mathbb{S}\text{Commit}(mpk; R)$.
- $\text{EVerify}(mpk, C, R)$: Take a commitment key mpk , commitment C and randomness R as inputs, accept if $C = \mathbb{S}\text{Commit}(mpk; R)$ and reject otherwise.
- $\text{FakeExplain}(msk; R)$: Take a secret trapdoor msk and randomness R such that $C = \text{MFake}(mpk; R)$ and output randomness R' such that C passes off as a soft commitment, i.e. $C = \mathbb{S}\text{Commit}(mpk; R')$.

The general construction of trapdoor mercurial commitments from trapdoor commitments for bit messages and Σ -protocols of Catalano *et al.* [45, Section 5] can be easily modified to support the algorithms mentioned above. Therefore, these algorithms are realizable in all known constructions of trapdoor mercurial commitments from standard assumptions, such as the discrete logarithm, factoring and RSA assumptions.

TMC schemes are correct if for all $mpk \leftarrow \text{TMSetup}(1^\lambda)$, the following conditions hold

- Hard commitments: For all possible messages M and randomness R , let $C \leftarrow \mathbb{H}\text{Commit}(mpk, M; R)$. Then,
 1. for all soft-openings $\tau \leftarrow \mathbb{S}\text{Open}(mpk, M, \mathbb{H}; R)$, the soft verification algorithm $\mathbb{S}\text{Verify}(mpk, M, C, \tau)$ accepts with overwhelming probability.
 2. for all hard-openings $\pi \leftarrow \mathbb{H}\text{Open}(mpk, M; R)$, the hard verification algorithm $\mathbb{H}\text{Verify}(mpk, M, C, \pi)$ accepts with overwhelming probability.
- Soft commitments: For all randomness R , let $C \leftarrow \mathbb{S}\text{Commit}(mpk; R)$. Then, for all possible messages M and $\tau \leftarrow \mathbb{S}\text{Open}(mpk, M, \mathbb{S}; R)$, $\mathbb{S}\text{Verify}(mpk, M, C, \tau)$ accepts with overwhelming probability.
- Equivocations: For all randomness R , let $C \leftarrow \text{MFake}(mpk; R)$. Then, for all possible message M , the three conditions below are satisfied with high probability:
 1. If $\pi \leftarrow \mathbb{H}\text{Equivocate}(msk, M; R)$, then $\mathbb{H}\text{Verify}(mpk, M, C, \pi)$ accepts.
 2. If $\tau \leftarrow \mathbb{S}\text{Equivocate}(msk, M; R)$, then $\mathbb{S}\text{Verify}(mpk, M, C, \tau)$ accepts.
 3. If $R' \leftarrow \text{FakeExplain}(msk; R)$, then $C = \mathbb{S}\text{Commit}(mpk; R')$.

Trapdoor mercurial commitment schemes have three security properties, *mercurial-hiding*, *mercurial-binding* and *equivocation*. The first two are slightly modified from the hiding and binding notions of regular commitment schemes to accommodate the two modes of commitments and two types of openings.

- Mercurial-hiding: No PPT adversary $\mathcal{A}$, given mpk , can find a message M in the message space where he can distinguish between a tuple containing the message M , a hard commitment and its soft opening to M , $(M, \mathbb{H}\text{Commit}(mpk, M; R_H), \mathbb{S}\text{Open}(mpk, M, \mathbb{H}; R_H))$, from another tuple with the same message, a soft commitment and an opening of it to M , $(M, \mathbb{S}\text{Commit}(mpk; R_S), \mathbb{S}\text{Open}(mpk, M, \mathbb{S}; R_S))$, for some randomness R_H, R_S .
- Mercurial-binding: No PPT adversary $\mathcal{A}$, given mpk , can find a commitment C such that
 1. he can find two messages and corresponding hard-openings M, π, M', π' with π a valid hard-opening of C to M and π' a valid hard-opening of C to M' , where $M \neq M'$.

2. he can find a message and a soft-opening and another message with a hard-opening M, τ, M', π' with τ a valid soft-opening of C to M and π' a valid hard-opening of C to M' , where $M \neq M'$.

Besides the hiding and binding properties, trapdoor mercurial commitments have to satisfy an equivocation property. Equivocation of trapdoor mercurial commitments was formalized by Catalano *et al.* [45] and consists of three conditions, HH, HS and SS equivocations. Fake commitments from **MFake** can be equivocated in both hard and soft ways and so it should be infeasible for any adversary to identify if they are given a fake commitment and some equivocation or a real hard or soft commitment and an opening, even if the trapdoor is revealed to the adversary.

With the explanation algorithms that we introduce, we require a fourth condition for equivocation, called Soft-Explain (SE for short) equivocation. This requires that an adversary be unable to identify if he is given a real soft commitment and its explanation or a fake commitment with a fake explanation that it is a soft commitment, even if the trapdoor is known to the adversary.

The four conditions for the equivocation property of trapdoor mercurial commitments are each defined by a pair of real and ideal games. No PPT adversary $\mathcal{A}$ can identify which game he is playing, even when the trapdoor msk is given at the start of each real or ideal game. We use R to denote a set of randomness sampled from the appropriate distribution in each game.

- HH Equivocation: In the real game, $\mathcal{A}$ chooses a valid message M and receives $(M, \mathbb{H}\text{Commit}(mpk, M; R), \mathbb{H}\text{Open}(mpk, M; R))$. In the ideal game, $\mathcal{A}$ chooses a message M in the message space and receives a fake commitment with a hard equivocation $(M, \text{MFake}(mpk; R), \mathbb{H}\text{Equivocate}(mpk, M; R))$ instead.
- HS Equivocation: In the real game, $\mathcal{A}$ chooses a valid message M and receives $(M, \mathbb{H}\text{Commit}(mpk, M; R), \mathbb{S}\text{Open}(mpk, M, \mathbb{H}; R))$. In the ideal game, $\mathcal{A}$ chooses a message M in the message space and receives a fake commitment with a soft equivocation $(M, \text{MFake}(mpk; R), \mathbb{S}\text{Equivocate}(mpk, M; R))$ instead.
- SS Equivocation: In the real game, $\mathcal{A}$ chooses a valid message M and receives $(M, \mathbb{S}\text{Commit}(mpk; R), \mathbb{S}\text{Open}(mpk, M, \mathbb{H}; R))$. In the ideal game, $\mathcal{A}$ chooses a message M in the message space and receives a fake commitment with a soft equivocation $(M, \text{MFake}(mpk; R), \mathbb{S}\text{Equivocate}(mpk, M; R))$ instead.
- SE Equivocation: In the real game, $\mathcal{A}$ gets a soft commitment $C = \mathbb{S}\text{Commit}(mpk; R)$ and its randomness $R = \text{Explain}(mpk; R)$. In the ideal game, $\mathcal{A}$ gets a fake com-

mitment $C = \text{MFake}(mpk; R)$ and a fake explanation $R' = \text{FakeExplain}(msk; R)$ of C as a soft commitment.

Just as equivocation implies the hiding property in trapdoor commitment schemes, Catalano *et al.* [45] noted that HS and SS equivocations together imply mercurial-hiding. Furthermore, for proper trapdoor mercurial commitments, the HH equivocation condition implies that the HS equivocation condition is satisfied. Thus, it is sufficient to verify the HH and SS equivocation conditions and mercurial-binding for the security of any trapdoor mercurial commitment scheme that is proper.

6.3.2 Binary Trees and Zero-Knowledge Elementary Databases

Binary trees play a huge role in constructing zero-knowledge elementary databases through the use of Merkle trees [151]. We review some important notation and definitions for binary trees and then introduce the formal definitions and elements of the generic construction of zero-knowledge elementary databases from trapdoor mercurial commitments by Chase *et al.* [48].

Notation for Binary Trees. Denoting a full and complete binary tree of height ℓ with $\mathcal{T}_\ell$, we index its nodes with binary strings of length up to ℓ , according to their depth in $\mathcal{T}_\ell$. In particular, each leaf is indexed by an ℓ -bit string which is the binary representation of an integer in the interval $[0, 2^\ell)$ and the root corresponds to the empty string ϵ . We use x to mean either the binary string or integer representation of the index of a leaf, based on the context. Every node $x \in \mathcal{T}_\ell$ defines a subset of leaves, denoted with $\mathcal{S}_x$, that lies in the full and complete subtree rooted at x .

Definition 6.3.1. *A directed Steiner tree of a set of leaves $\mathcal{R}$, denoted with $ST(\mathcal{R})$, is the minimal subtree of $\mathcal{T}_\ell$ that connects every leaf in $\mathcal{R}$. The set of depth- j nodes, for j from ℓ to 0, in $ST(\mathcal{R})$ is denoted with $ST(\mathcal{R})_j$.*

The Steiner tree was exploited by Naor, Naor and Lotspiech [161] in the complete subtree algorithm for revoking users in broadcast encryption. They used it to partition the set of leaves of a full and complete binary tree, which correspond to users in their setting, into two sets. The first is the set of revoked users, $\mathcal{R}$, and the other contains all remaining users whose decryption rights are maintained.

We recall the notion of canonical coverings from Ghosh *et al.* [90], who used it to achieve zero-knowledge range queries in the three-party setting.

 Figure 6.1: Steiner Tree of $[1, 3]$

Definition 6.3.2. The canonical covering of some interval $[a, b]$, denoted by $\mathcal{P}_{[a,b]}$, is the unique minimal set of nodes of $\mathcal{T}_\ell$ such that

1. each node in $[a, b]$ is a descendant of some node in $\mathcal{P}_{[a,b]}$ and
2. for every node $x \in \mathcal{P}_{[a,b]}$, the leaves of $\mathcal{S}_x$ lie entirely within $[a, b]$.

For some set of leaves $\mathcal{R}$, let $\mathcal{R} = \bigcup_{i=1}^m [a_i, b_i]$, where m is the smallest number of intervals that $\mathcal{R}$ can be partitioned into. Then, the canonical covering of $\mathcal{R}$ in $\mathcal{T}_\ell$ is $\mathcal{P}_{\mathcal{R}} = \bigcup_{i=1}^m \mathcal{P}_{[a_i, b_i]}$.

Naor *et al.* [161] proved that for $\mathcal{R}$ which is a set of leaves, the size of the canonical covering of every leaf of $\mathcal{T}_\ell$ except those in $\mathcal{R}$ is related to the size of $\mathcal{R}$.

Lemma 6.3.3 ([161, Theorem 2]). For a set of leaves in $\mathcal{T}_\ell$, $\mathcal{R}$, the size of the canonical covering of the leaves not in $\mathcal{R}$, $[0, 2^\ell] \setminus \mathcal{R}$, is at most $|\mathcal{R}| \log(2^\ell / |\mathcal{R}|)$.

Figure 6.1 shows an example of a Steiner tree for a tree of $2^3 = 8$ leaves. The nodes in black form the Steiner tree for the interval $[1, 3]$ and the diamond nodes make up the canonical covering of the remaining nodes $[0, 8] \setminus [1, 3]$.

6.3.3 Zero-Knowledge Elementary Databases

First proposed by Micali *et al.* [153], zero-knowledge elementary databases lets a database owner secretly commit to an elementary database $D = \{(x, D(x)) \in [0, 2^\ell) \times [0, 2^\ell)\}$

and then non-interactively prove correctness of membership queries in zero-knowledge. Membership queries on D take a key x as input and expect an answer $D(x) \in [0, 2^\ell] \cup \{\perp\}$, which is the value of the key in D . We note that Micali *et al.* [153] also defined a primitive called zero-knowledge sets, which is a specialization of ZK-EDB where $D(x) = 1$ if $x \in [D]$. Zero-knowledge elementary databases formally comprise of four algorithms **Init**, **ComDB**, **ProveQ** and **VerifyQ**.

Let $D = \{(x, D(x))\}$ be the elementary database that will be committed by the scheme.

- $(crs, tk) \leftarrow \text{Init}(1^\lambda)$: Take a security parameter λ as input, then generate and output a common reference string (CRS) crs and trapdoor information tk .
- $(com, \Delta) \leftarrow \text{ComDB}(crs, D)$: Take the CRS crs and a database D as inputs and output a digest of D , com , and decommitment information Δ .
- $\Pi_x \leftarrow \text{ProveQ}(crs, (com, \Delta), x)$: Take the CRS crs , database digest and decommitment information (com, Δ) and key x as inputs and output a proof Π_x of either $x \in D$ or $x \notin D$.
- $y \leftarrow \text{VerifyQ}(crs, com, x, \Pi_x)$: Taking the CRS crs , database digest com , key x and proof Π_x as inputs, outputs y where

$$y = \begin{cases} D(x), & \text{if } x \in [D]; \\ \perp, & \text{if } x \notin [D]; \\ bad, & \text{if the prover is believed to have cheated.} \end{cases}$$

Security. ZK-EDB has three properties for security, namely *completeness*, *soundness* and *zero-knowledge*. For completeness to be satisfied, **VerifyQ** must accept all honestly generated proofs. Soundness require provers be unable to produce a key x and two valid proofs Π_x and Π'_x such that **VerifyQ** outputs different values on them. Finally, the zero-knowledge property mandates that any proof Π_x from **ProveQ** only reveal the value $D(x)$ and nothing else about D .

Formally, we have

- *Completeness*: For all databases D and for all keys x , it holds that

$$\Pr[crs \leftarrow \text{Init}(1^n); (com, \Delta) \leftarrow \text{ComDB}(crs, D);$$

$$\Pi_x \leftarrow \text{ProveQ}(crs, D, (com, \Delta), x);$$

$$\text{VerifyQ}(crs, com, x, \Pi_x) = D(x)] = 1 - \text{negl}(\lambda).$$

- *Soundness*: For all keys x and for any PPT algorithm P' ,

$$\Pr[crs \leftarrow \text{Init}(1^{1^n}); (com, x, \Pi_x, \Pi'_x) \leftarrow P'(crs);$$

$$(\text{VerifyQ}(crs, com, x, \Pi_x) = y \neq \text{bad})$$

$$\wedge (\text{VerifyQ}(crs, com, x, \Pi'_x) = y' \neq \text{bad}) \wedge (y \neq y')] = 1 - \text{negl}(\lambda).$$
- *Zero-Knowledge*: For any PPT adversary $\mathcal{A}$ and efficiently computable database D , there exists an efficient simulator for ZK-EDB $\text{Sim} = (\text{SInit}, \text{SCom}, \text{SProveQ}^D)$ such that the results of the following two experiments are indistinguishable:

Real experiment:

1. Let $crs \leftarrow \text{Init}(1^{1^n})$, $(com, \Delta) \leftarrow \text{ComDB}(crs, D)$ and initialize $s_0 = \Pi_0 = \varepsilon$.
2. For $1 \leq i \leq n$, $(x_i, s_i) \leftarrow \mathcal{A}(crs, com, \Pi_0, \dots, \Pi_{i-1}, s_{i-1})$ and $\mathcal{A}$ gets a real proof $\Pi_i = \text{ProveQ}(crs, D, (com, \Delta), x_i)$.

The experiment outputs $(crs, x_1, \Pi_1, \dots, x_n, \Pi_n)$.

Ideal experiment:

1. Let $(crs', st_0) \leftarrow \text{SInit}(1^{1^n})$, $(com', st_1) \leftarrow \text{SCom}(st_0)$ and set $s_0 = \Pi'_0 = \varepsilon$.
2. For $1 \leq i \leq n$, $(x_i, s_i) \leftarrow \mathcal{A}(crs', com', \Pi'_0, \dots, \Pi'_{i-1}, s_{i-1})$ and $\mathcal{A}$ gets a simulated proof $\Pi'_i \leftarrow \text{SProveQ}^D(cr s', st_1, x_i)$.

The experiment outputs $(crs', x_1, \Pi'_1, \dots, x_n, \Pi'_n)$.

SProveQ^D is an oracle that is allowed to invoke a database oracle D and obtain values $D(x)$ for any key x chosen by the adversary $\mathcal{A}$.

Chase *et al.* [48, 49] presented a generic construction of a ZK-EDB from trapdoor mercurial commitments. We describe four algorithms, adapted from that construction, for Merkle trees built from (trapdoor) mercurial commitment schemes **BuildTree**, **HOpenPath**, **SOpenPath** and **VerifyPath** assuming that a public commitment key crs from **TMSetup**, which is called a common reference string (CRS) in the literature.

The key idea of the construction is committing to the database D with a Merkle tree using (trapdoor) mercurial commitments. Each leaf in the Merkle tree is associated with a key x and if $x \in [D]$, then the value at leaf x is a commitment to the value $D(x)$. Then, we obtain the root of the Merkle tree by creating hard-commitments for every node on

the Steiner tree $ST([D])$ induced by the database's keys; soft commitments are generated for the canonical covering of $[0, 2^\ell] \setminus [D]$, completing the Merkle tree.

Then, a membership proof for some key x is a hard authentication path from the leaf x to the root ϵ , which we define as a set of hard openings for the values, C_i , to their messages $D(x)$ or (C_{i0}, C_{i1}) at the nodes with indices $i \in \{x, x|_{\ell-1}, \dots, x|_1, \epsilon\}$. Non-membership proofs for key x are soft authentication paths from leaf x to the root ϵ . These are similar to hard authentication paths except the openings are soft instead of hard and the value at the leaf C_x is soft opened to $\perp$. We use Π_x to denote the (hard or soft) authentication path for a leaf x .

— $(com, \Delta) \leftarrow \text{BuildTree}(crs, D)$: Take the CRS crs and database D as input and build a Merkle tree of height ℓ , $\mathcal{MT}_\ell$, as follows:

1. View each key $x \in [D]$ as a leaf in $\mathcal{MT}_\ell$ and compute the value at leaf x , $C_x \leftarrow \mathbb{HCommit}(crs, D(x); R_x)$. For every other leaf $j \in \mathcal{MT}_\ell$, if $j' \in [D]$, compute $C_j \leftarrow \mathbb{SCommit}(crs; R_j)$ and set $C_j = nil$ otherwise.
2. At depth i of $\mathcal{MT}_\ell$, from $\ell - 1$ to 0 , define C_ρ where ρ is a length- i binary string as follows:
 - a. If $C_{\rho 0} \neq nil$ and $C_{\rho 1} \neq nil$, compute $C_\rho \leftarrow \mathbb{HCommit}(crs, (C_{\rho 0}, C_{\rho 1}); R_\rho)$, the value at node ρ .
 - b. If C_ρ was computed in the previous step but $C_{\rho'}$ was not, compute $C_{\rho'} \leftarrow \mathbb{SCommit}(crs; R_{\rho'})$.
 - c. For all remaining ρ , set $C_\rho = nil$.
3. Following Step 2, if $C_\epsilon = nil$, implying that $D = \emptyset$, then compute $C_\epsilon \leftarrow \mathbb{SCommit}(crs; R_\epsilon)$.

Output $com = C_\epsilon$ and $\Delta = \{R_j\}$, which is made up of all the randomness used in the algorithm.

— $\Pi_z \leftarrow \mathbb{HOpenPath}(crs, (com, \Delta), z)$: Take the CRS crs , database digest com with decommitment information Δ for some database D and a key $z \in [D]$, Proceed to hard open the values at the nodes $z|_\ell = z, z|_{\ell-1}, \dots, z|_1, z|_0 = \epsilon$ as follows:

1. Compute the hard opening of C_z , $\pi_z \leftarrow \mathbb{HOpen}(crs, D(z); R_z)$.
2. At depth j of $\mathcal{MT}_\ell$, from $\ell - 1$ to 0 , compute the hard opening for $C_{z|_j}$ to $(C_{z|_j 0}, C_{z|_j 1})$, $\pi_{z|_j} \leftarrow \mathbb{HOpen}(crs, (C_{z|_j 0}, C_{z|_j 1}); R_{z|_j})$.

Output $\Pi_z = (D(z), \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\pi_{z|_j}\}_{0 \leq j \leq \ell})$.

- $\Pi_z \leftarrow \text{SOpenPath}(crs, (com, \Delta), z)$: Take the CRS crs , database digest com with decommitment information Δ for a database D and a key $z \in D$ as inputs, let h be the largest value such that $C_{z|h} \neq nil$.
 1. If $h \neq \ell$, then $C_z = nil$ and we fill out the path from ϵ to z ,
 - a. Compute soft commitments for leaf z , $C_z \leftarrow \text{SCommit}(crs; R_z)$, and its sibling z' , $C_{z'} \leftarrow \text{SCommit}(crs; R_{z'})$.
 - b. At depth j of $\mathcal{MT}_\ell$, from $\ell - 1$ to $h + 1$, compute soft commitments for $z|_j$, $C_{z|_j} \leftarrow \text{SCommit}(crs; R_{z|_j})$ and $(z|_j)'$, $C_{(z|_j)'} \leftarrow \text{SCommit}(crs; R_{(z|_j)'})$.
 2. Otherwise, $C_z = \text{SCommit}(crs; R_z)$ and we move on.
 3. Compute the soft authentication path from z to ϵ of $\mathcal{MT}_\ell$.
 - a. Compute $\tau_z \leftarrow \text{SOpen}(crs, \perp, \mathbb{S}; R_z)$.
 - b. At depth j of $\mathcal{MT}_\ell$, from $\ell - 1$ to $h + 1$, compute soft openings $\tau_{z|_j} \leftarrow \text{SOpen}(crs, (C_{z|_j0}, C_{z|_j1}), \mathbb{S}; R_{z|_j})$.
 - c. At depth j of $\mathcal{MT}_\ell$, from j from h to 1 , compute soft openings $\tau_{z|_h} \leftarrow \text{SOpen}(crs, (C_{z|_h0}, C_{z|_h1}), \mathbb{H}; R_{z|_h})$.
 - d. If $D = \emptyset$, compute $\tau_\epsilon \leftarrow \text{SOpen}(crs, (C_{z|_0}, C_{z|_1}), \mathbb{S}; R_\epsilon)$. Otherwise, compute $\tau_\epsilon \leftarrow \text{SOpen}(crs, (C_{z|_0}, C_{z|_1}), \mathbb{H}; R_\epsilon)$.

Output $\Pi_z = (\perp, \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\tau_{z|_j}\}_{0 \leq j \leq \ell})$. Also, add to Δ any randomness used if the soft authentication path from z to ϵ was generated during the algorithm.

- $ans \leftarrow \text{VerifyPath}(crs, com, z, \Pi_z)$: Take the CRS crs , database digest com , a key z with proof Π_z as inputs and check the proof which has two possible forms:
 - ◊ $D(z) \neq \perp$: $\Pi_z = (D(z), \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\pi_{z|_j}\}_{0 \leq j \leq \ell})$.
 1. Run $\mathbb{H}\text{Verify}(crs, D(z), C_z, \pi_z)$, to check C_z hard opens to $D(z) \neq \perp$ and output $ans = bad$ if it rejects.
 2. For j from $\ell - 1$ to 0 , verify that $C_{z|_j}$ hard opens to $(C_{z|_j0}, C_{z|_j1})$ with $\mathbb{H}\text{Verify}(crs, (C_{z|_j0}, C_{z|_j1}), \pi_{z|_j})$ and output $ans = bad$ if any of them reject.
 If none of the verifications rejected, output $ans = D(z)$.
 - ◊ $D(z) = \perp$: $\Pi_z = (\perp, \tau_z, \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\tau_{z|_j}\}_{0 \leq j \leq \ell})$.
 1. Run $\mathbb{S}\text{Verify}(crs, \perp, C_z, \tau_z)$ verifying that C_z soft opens to $\perp$ and output $ans = bad$ if it rejects.
 2. For j from $\ell - 1$ to 0 , check if $C_{z|_j}$ soft opens to $(C_{z|_j0}, C_{z|_j1})$ with $\mathbb{S}\text{Verify}(crs, (C_{z|_j0}, C_{z|_j1}), \tau_{z|_j})$ and output $ans = bad$ if any of them reject.

If none of the verifications rejected, output $ans = \perp$.

BuildTree and VerifyPath are exactly ComDB and VerifyQ respectively. The other two algorithms, HOpenPath and SOpenPath, are used as sub-algorithms of ProveQ to compute proofs of membership and non-membership respectively.

6.4 Zero-Knowledge Expressive Elementary Databases

In this section, we introduce a new cryptographic primitive called zero-knowledge expressive elementary databases (ZK-EEDB). This protocol allows owners of elementary databases (EDBs) D to secretly commit them in public digests and then use these digests to efficiently answer certain queries on D in zero-knowledge. EDBs D are sets of records, key-value pairs $(x, D(x)) \in [0, 2^\ell) \times [0, 2^\ell)$, and queries that ZK-EEDB can answer efficiently include membership and range queries over keys and values.

ZK-EEDB expands the types of query that can be handled by size-hiding zero-knowledge databases, from membership over keys considered by Micali *et al.* [153], to range queries over records of D . This particular query generalizes range queries over both keys and values. We introduce, with ZK-EEDB, an efficient method to compute proofs of correctness for range queries over values of D in zero-knowledge.

A membership query over values of D returns the set $D^{-1}(y) = \{x \mid x \in [D], D(x) = y\}$, for some given value y . Range queries over values are simply batches of membership queries over a given range $[a_y, b_y]$. From the techniques used to answer range queries over values, we obtain the ability to answer and prove correctness of range queries over records efficiently for any super-polynomial range of keys.

Below, we introduce some new notation for values of an EDB D and query types that are considered in ZK-EEDB. Then, we formally define ZK-EEDB and describe its security properties.

The Database of Values, D^{-1} . In addition to queries over keys of an EDB D , ZK-EEDB considers queries over its values as well. To efficiently generate proofs of correctness for answers on values of D , we introduce a different representation of D , called D^{-1} , that enables efficient queries over values. It can be seen as a “reversed directory”, D^{-1} is defined as the set $\{(y, D^{-1}(y)) \mid y \in [0, 2^\ell)\}$, where $D^{-1}(y)$ is the set of keys $x \in [D]$ whose assigned value in D is y , i.e., $D(x) = y$. The key-space of D^{-1} therefore corresponds to the value-space of D . Notations defined for D are extended to D^{-1} . $[D^{-1}]$ is the set of

keys of D^{-1} and it is the set of values y where there exists at least one key x such that the record $(x, y) \in D$. For $y \in [0, 2^\ell)$, $y \in [D^{-1}]$ means that $D^{-1}(y) \neq \emptyset$.

Queries of ZK-EEDB. Here, we introduce the one query supported by ZK-EEDB that actually captures six different queries including itself. The answer to any query will be accompanied by a zero-knowledge proof of correctness.

The single query that we consider is the range query over records, which take a two-dimensional range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$ and returns all records (x, y) such that $x \in [a_x, b_x]$ and $y \in [a_y, b_y]$. From this, we get membership and range queries over the database's keys, values and records as follows:

1. Membership query over keys: With an input range of the form $\mathfrak{R} = [x, x] \times [0, 2^\ell)$, we have a query which seeks the record $(x, D(x))$ for some $D(x) \in [0, 2^\ell)$.
2. Membership query over values: With an input range of the form $\mathfrak{R} = [0, 2^\ell) \times [y, y]$, we have a query which seeks the set $\mathcal{L} = D^{-1}(y)$.
3. Membership query over records: Combining the two previous query, with an input range of the form $\mathfrak{R} = [x, x] \times [y, y]$, we have a query which seeks the record (x, y) , whether it is in D or not.
4. Range query over keys: With an input range $\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$, we have a query which seeks the set of records $\mathcal{L}$, such that $(x, D(x)) \in \mathcal{L}$ if and only if $x \in [a_x, b_x]$.
5. Range query over values: With an input range $\mathfrak{R} = [0, 2^\ell) \times [a_y, b_y]$, we have a query which seeks the set of records $\mathcal{L}$, such that $(x, D(x)) \in \mathcal{L}$ if and only if $D(x) \in [a_y, b_y]$.
6. Range query over records: With an input range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$, we have a query which seeks the set of records $\mathcal{L}$, where $(x, y) \in \mathcal{L}$ if and only if $x \in [a_x, b_x]$ and $y \in [a_y, b_y]$.

There is a subtle issue when we adapt the range query over records to queries over keys and values. Specifically, in these cases, the input range $\mathfrak{R}$ are allowed to be exponentially large, encompassing the entire key- or value-space. Also, membership query over keys in ZK-EEDB does not return $\perp$ if the queried key is not in D . Instead, an empty set is returned so that the answer is in line with the definition of range queries over records.

Formal Definition. ZK-EEDB consists of four algorithms, `Init`, `ComDB`, `ProveRQ` and `VerifyRQ`. These are the initialization, database commitment, range proof generation and verification algorithms respectively.

- $(crs, tk) \leftarrow \text{Init}(1^\lambda)$: Take a security parameter λ as input and output a common reference string (CRS) crs and trapdoor key tk .
- $(com, \Delta) \leftarrow \text{ComDB}(crs, D)$: Take a CRS crs and EDB D as input and output a commitment com and decommitment information Δ .
- $\Pi_{\mathfrak{R}} \leftarrow \text{ProveRQ}(crs, (com, \Delta), \mathfrak{R})$: Take as input a CRS crs , EDB digest and decommitment information (com, Δ) and range $\mathfrak{R}$ and output a proof of correctness $\Pi_{\mathfrak{R}}$ of the range query with input range $\mathfrak{R} \subset [0, 2^\ell) \times [0, 2^\ell)$.
- $\mathcal{L} \leftarrow \text{VerifyRQ}(crs, com, \mathfrak{R}, \Pi_{\mathfrak{R}})$: Take a CRS crs , database digest com , range $\mathfrak{R} \subset [0, 2^\ell) \times [0, 2^\ell)$ and supposed proof $\Pi_{\mathfrak{R}}$ and output

$$\mathcal{L} = \begin{cases} D \cap \mathfrak{R}, & \text{if the proof is valid;} \\ bad, & \text{if the proof is determined to be invalid.} \end{cases}$$

The security properties of ZK-EEDB are largely similar to standard ZK-EDB protocols, *completeness*, *soundness* and *zero-knowledge*. Although they share the same names, their definitions in ZK-EEDB are modified to suit the larger number of query types supported. Correctness and zero-knowledge for ZK-EEDB correspond at a high level with their counterparts in ZK-EDB.

Correctness requires that honestly and correctly computed proofs, for any input range $\mathfrak{R}$, satisfy the verification algorithm with overwhelming probability. Zero-knowledge mandates the existence of an efficient simulator that, with only oracle access to the database, outputs proofs for queries that are indistinguishable from proofs computed by a real prover with the real database.

Soundness demands that contradictory statements about the committed database cannot be proven by adversaries except with negligible probability. Informally, for ZK-EEDB, it means that no PPT adversary can find two ranges $\mathfrak{R}$ and $\mathfrak{R}'$ and corresponding proof Π and Π' such that there exists a record (x, y) in the intersection of the ranges that is only in the answer of one of the two queries and not both.

Formally, the properties are defined as follows:

- *Completeness*: For all EDBs D and keys x ,

- $\Pr[crs \leftarrow \text{Init}(1^\lambda); (com, \Delta) \leftarrow \text{ComDB}(crs, D);$
 $\Pi_{\mathfrak{R}} \leftarrow \text{ProveRQ}(crs, (com, \Delta), \mathfrak{R});$
 $\text{VerifyRQ}(crs, com, \mathfrak{R}, \Pi_{\mathfrak{R}}) \neq \text{bad}] = 1 - \text{negl}(\lambda),$
- *Soundness*: For all PPT algorithm P' ,

$$\Pr[crs \leftarrow \text{Init}(1^\lambda); (com, \mathfrak{R}, \Pi), (\mathfrak{R}', \Pi') \leftarrow P'(crs);$$

$$(\text{VerifyRQ}(crs, com, \mathfrak{R}, \Pi) = \mathcal{L} \neq \text{bad})$$

$$\wedge (\text{VerifyRQ}(crs, com, \mathfrak{R}', \Pi') = \mathcal{L}' \neq \text{bad})$$

$$\wedge (\exists (x, y) \in \mathfrak{R} \cap \mathfrak{R}' \text{ s.t. } ((x, y) \in \mathcal{L}) \wedge ((x, y) \notin \mathcal{L}'))] \leq \text{negl}(\lambda),$$
 - *Zero-Knowledge*: For any PPT adversary $\mathcal{A}$ and efficiently computable database D , there exists an efficient simulator, which is made up of three algorithms ($S\text{Init}$, $S\text{Com}$, $S\text{ProveQ}^D$) such that outputs of the following two experiments are indistinguishable:

Real experiment:

1. Let $crs \leftarrow \text{Init}(1^\lambda), (com, \Delta) \leftarrow \text{ComDB}(crs, D)$ and set $s_0 = \Pi_0 = \varepsilon$.
2. For $1 \leq i \leq n$, the adversary $\mathcal{A}$ chooses a query $\mathfrak{R}_i$ and update its state to s_i , $(\mathfrak{R}_i, s_i) \leftarrow \mathcal{A}(crs, com, \Pi_0, \dots, \Pi_{i-1}, s_{i-1})$ and gets a real proof of the query $\Pi_i = \text{ProveRQ}(crs, (com, \Delta), \mathfrak{R}_i)$.

The experiment outputs $(crs, \mathfrak{R}_1, \Pi_1, \dots, \mathfrak{R}_n, \Pi_n)$.

Ideal experiment:

1. Let $(crs', st_0) \leftarrow S\text{Init}(1^\lambda), (com', st_1) \leftarrow S\text{Com}(st_0)$ and set $s_0 = \Pi'_0 = \varepsilon$.
2. For $1 \leq i \leq n$, the adversary $\mathcal{A}$ chooses a query $\mathfrak{R}_i$ and update its state to s_i $(\mathfrak{R}_i, s_i) \leftarrow \mathcal{A}(crs', com', \Pi'_0, \dots, \Pi'_{i-1}, s_{i-1})$ and gets a simulated proof $\Pi'_i \leftarrow S\text{ProveRQ}^D(cr s', st_1, \mathfrak{R}_i)$.

The experiment outputs $(crs', \mathfrak{R}_1, \Pi'_1, \dots, \mathfrak{R}_n, \Pi'_n)$.

In the ideal experiment, $S\text{ProveQ}^D$ is allowed to call the database oracle D and receive the set of records $D \cap \mathfrak{R}$ for any range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$ queried by $\mathcal{A}$.

6.5 Generic Construction from Trapdoor Mercurial Commitments

In this section, we work towards the generic construction of ZK-EEDB from trapdoor mercurial commitments. First, we describe how to prove range queries over keys of D

with the ZK-EDB construction by Chase *et al.* [48] and the new explanation algorithms which we introduce. Then, we present our technique to enable efficient queries over values. Finally, we combine everything into a construction of ZK-EEDB.

6.5.1 Range Queries over Keys with ZK-EDB

Recall that a ZK-EDB scheme, $\mathcal{Z}$, commits to an EDB via a Merkle tree with some trapdoor mercurial commitment scheme TMC with thirteen algorithms, TMSetup for setup, hard commitment algorithms, $\mathbb{H}\text{Commit}$, $\mathbb{H}\text{Open}$ and $\mathbb{H}\text{Verify}$, soft commitment algorithms, $\mathbb{S}\text{Commit}$, $\mathbb{S}\text{Open}$, $\mathbb{S}\text{Verify}$, fake commitment algorithms, $\mathbb{M}\text{Fake}$, $\mathbb{H}\text{Equivocate}$, $\mathbb{S}\text{Equivocate}$ and the newly introduced explanation algorithms, Explain , EVerify , FakeExplain . Let $(com, \Delta) \leftarrow \mathcal{Z}.\text{ComDB}(crs, D)$ be the digest and decommitment information of an EDB D for some common reference string crs from $\mathcal{Z}.\text{Init}(1^\lambda)$. Proofs of membership of a key x is done by computing the hard authentication path from the leaf x of the Merkle tree to the root.

Optimized Proof of Membership for a Set of Leaves. Before going into range queries, we show how the Steiner tree can be adapted to generate a proof of membership for a subset of leaves, $\mathcal{S}$ with the complete subtree method by Naor *et al.* [161]. Naively, we can prove membership for a subset of leaves by returning the hard authentication path for each leaf in $\mathcal{S}$ individually. However, doing this results in some redundancy; internal nodes that are at lower depths would be opened many times since most authentication paths would pass through them.

Since the Steiner tree of $\mathcal{S}$ is the minimal subtree that connects every leaf in $\mathcal{S}$, the optimal method is to compute hard openings of every node in $ST(\mathcal{S})$ to their committed messages. We define the authentication Steiner tree of $\mathcal{S}$, $\Pi_{ST(\mathcal{S})}$ as the set of hard openings for every node in $ST(\mathcal{S})$. It is easy to see that the hard authentication path of every leaf $x \in \mathcal{S}$, Π_x is contained within $\Pi_{ST(\mathcal{S})}$. However, the size of the authentication Steiner tree is $\mathcal{O}(|\mathcal{S}| \cdot \ell)$ which is asymptotically the same as a naive proof.

Verification of the authentication Steiner tree $\Pi_{ST(\mathcal{S})}$ mirrors that of hard authentication paths: we simply check that the hard opening of each node in the Steiner tree $ST(\mathcal{S})$ is valid. These two procedures are formalized in the algorithms $\mathbb{H}\text{OpenST}$ and VerifyST below.

- $\Pi_{ST(\mathcal{S})} \leftarrow \mathbb{H}\text{OpenST}(crs, (com, \Delta), \mathcal{S})$: Take a CRS crs , ZK-EDB digest com and decommitment information Δ and set of keys $\mathcal{S} \subset [D]$ as input and output $\Pi_{ST(\mathcal{S})}$ from the following:

1. For every leaf $x \in \mathcal{S}$, compute the hard opening of C_x to its EDB value $D(x)$, $\pi_x \leftarrow \mathbb{HOpen}(crs, D(x); R_x)$.
2. At depth j , from $\ell - 1$ to 0, and for every internal node $z \in ST(\mathcal{S})_j$, compute the hard opening of C_z to the values at its children (C_{z0}, C_{z1}) , $\pi_z \leftarrow \mathbb{HOpen}(crs, (C_{z0}, C_{z1}); R_z)$.

Set $\Pi_{\mathcal{S}} = (\{(x, D(x)), C_x, \pi_x\}_{x \in \mathcal{S}}, \{C_z, C_{z'}, \pi_z\}_{z \in ST(\mathcal{S}) \setminus \mathcal{S}})$.

– $\mathcal{L} \leftarrow \text{VerifyST}(crs, com, \mathcal{S}, \Pi_{\mathcal{S}})$: Take a CRS crs , EDB digest com , set of leaves $\mathcal{S}$ and its supposed authentication Steiner tree $\Pi_{ST(\mathcal{S})}$ as input, do the following:

1. For every leaf $x \in \mathcal{S}$, check the proof π_x that C_x commits to $D(x)$ with $\mathbb{HVerify}(crs, D(x), C_x, \pi_x)$ and continue if all verifications accept. Otherwise, set and output $\mathcal{L} = bad$.
2. At depth j , from $\ell - 1$ to 0, and for every internal node $z \in ST(\mathcal{R})_j$, run $\mathbb{HVerify}(crs, (C_{z0}, C_{z1}), C_z, \pi_z)$ to check that π_z proves C_z commits to (C_{z0}, C_{z1}) . Continue if all verifications accept. Otherwise, set and output $\mathcal{L} = bad$.

Set and output $\mathcal{L} = \{(x, D(x))\}_{x \in \mathcal{S}}$ if $\mathcal{L} \neq bad$.

Proof of (Non-)Membership for an Interval. Let $\mathcal{S} = D \cap [a, b]$, the set of keys that have an associated value in D that lie in the interval $[a, b]$. We prove that $\mathcal{S} = D \cap [a, b]$ with the authentication Steiner tree, $\Pi_{ST(\mathcal{S})}$, and a key observation: If an internal node z' is a soft commitment, then no leaf in the subtree $S_{z'}$ has a valid hard authentication path and therefore cannot be in $[D]$. Building on the authentication Steiner tree for $\mathcal{S}$, which proves that the records $(x, D(x)) \in \mathcal{S}$ are in D , we use explanations, TMC.Explain , to show that no leaves in $[a, b] \setminus \mathcal{S}$ will ever be part of a membership proof. For any set of leaves $\mathcal{R}$, if $([D] \cap [a, b]) \setminus \mathcal{R}$ is the empty set, then the values at $\mathcal{P}_{[a, b] \setminus \mathcal{R}}$, the canonical covering of $[a, b] \setminus \mathcal{R}$ are either soft commitments or *nil* after the database is committed in $(com, \Delta) \leftarrow \mathcal{Z}.\text{ComDB}(crs, D)$.

Now, we elaborate on how the canonical covering, which we denote with $\mathcal{P}$, of $[a, b] \setminus \mathcal{S}$ is incorporated with the Steiner tree $ST(\mathcal{S})$. For every node $z \in \mathcal{P}$, the value C_z is explained as a soft commitment to prove that no leaf in $[a, b] \setminus \mathcal{S}$ can be in a hard authentication path. Most nodes $z \in \mathcal{P}$ have their sibling z' in $ST(\mathcal{S})$ which means that we do not need to compute additional openings. However, there are two intervals that may require more: Let $\mathcal{S} = \{x_1, \dots, x_m\}$, then the canonical covering of the intervals $[a, x_1 - 1]$ and $[x_m + 1, b]$ might contain nodes y such that $y' \notin ST(\mathcal{S})$. If so, those nodes

are shown to be properly connected to $ST(\mathcal{S})$, and by extension the root of the Merkle tree, with soft openings of their parents until some ancestor $z \in ST(\mathcal{S})$.

From Theorem 6.3.3, we have at most $|\mathcal{S}| \log(2^\ell/|\mathcal{S}|)$ explanations, which is a constant factor larger than $|ST(\mathcal{S})| = |\mathcal{S}|\ell$, and we add a constant number of soft openings to complete the proof that there are no keys in $[D] \cap ([a, b] \setminus \mathcal{S})$. Therefore, the complete range proof has size $\mathcal{O}(|\mathcal{S}|\ell)$, independent of the length of the input interval $[a, b]$ and can support exponential-sized inputs. We formalize the steps into two algorithms, **Openl** and **Verifyl** below.

- $\Pi_{[a,b]} \leftarrow \text{Openl}(crs, (com, \Delta), [a, b])$: If $a = b$, then compute the authentication path with $\mathbb{H}\text{OpenPath}(crs, (com, \Delta), a)$ if $a \in [D]$ or $\mathbb{S}\text{OpenPath}(crs, (com, \Delta), a)$ otherwise. If $a \neq b$, do the following depending on $\mathcal{S} = [D] \cap [a, b]$:
 - ◊ $\mathcal{S} = \emptyset$: We show that the value at every node of the canonical covering of $[a, b]$, $\mathcal{P}$, is a soft commitment. For each $x \in \mathcal{P}$,
 1. If $C_x = nil$, then generate soft commitments for node x and its sibling, by running $C_x \leftarrow \mathbb{S}\text{Commit}(crs; R_x)$, $C_{x'} \leftarrow \mathbb{S}\text{Commit}(crs; R_{x'})$.
 2. Compute $R_x \leftarrow \text{Explain}(crs, C_x; R_x)$.
 3. At depth i , from $|x| - 1$ to 0,
 - a. If $C_{x|_i} = nil$, generate $C_{x|_i} \leftarrow \mathbb{S}\text{Commit}(crs; R_{x|_i})$ and $C_{(x|_i)'} \leftarrow \mathbb{S}\text{Commit}(crs; R_{(x|_i)'})$.
 - b. Compute $\tau_{x|_i} \leftarrow \mathbb{S}\text{Open}(crs, (C_{x|_i0}, C_{x|_i1}); R_{x|_i})$.
 Set $\Pi_{\mathcal{P}} = \{C_x, R_x, \{C_{x|_i0}, C_{x|_i1}, \tau_{x|_i}\}_{0 \leq i < |x|}\}_{x \in \mathcal{P}}$ and $\Pi_{\mathcal{S}} = nil$.
 - ◊ $\mathcal{S} \neq \emptyset$: Compute the authentication Steiner tree $\Pi_{[a,b],\mathcal{S}}$ and explain that the value at every node of the canonical covering of $[a, b] \setminus \mathcal{S}$, $\mathcal{P}$, is a soft commitment.
 1. Compute $\Pi_{\mathcal{S}} \leftarrow \mathbb{H}\text{OpenST}(crs, (com, \Delta), \mathcal{S})$.
 2. For each $x \in \mathcal{P}$,
 - a. If $C_x = nil$, then generate $C_x \leftarrow \mathbb{S}\text{Commit}(crs; R_x)$ and $C_{x'} \leftarrow \mathbb{S}\text{Commit}(crs; R_{x'})$.
 - b. Compute $R_x \leftarrow \text{Explain}(crs, C_x; R_x)$.
 - c. Let $h_x \leq |x| - 1$ be the depth such that $x|_{h_x} = z'$ for some $z \in ST(\mathcal{S})$. Then, at depth i , from $|x| - 1$ to h_x ,
 - i. If $C_{x|_i} = nil$, first generate $C_{x|_i} \leftarrow \mathbb{S}\text{Commit}(crs; R_{x|_i})$ and $C_{(x|_i)'} \leftarrow \mathbb{S}\text{Commit}(crs; R_{(x|_i)'})$.

- ii. Compute $\tau_{x|i} \leftarrow \text{SOpen}(crs, (C_{x|i0}, C_{x|i1}); R_{x|i})$.
 Set $\Pi_{\mathcal{P}} = \{C_x, R_x, \{C_{x|i0}, C_{x|i1}, \tau_{x|i0}\}_{h_x \leq i < |x|}\}_{x \in \mathcal{P}}$.
 Set and output $\Pi_{[a,b]} = (\Pi_{\mathcal{S}}, \Pi_{\mathcal{P}})$ and add any randomness used in the algorithm to Δ .
 - $\mathcal{L}$ $\text{VerifyI}(crs, com, [a, b], \Pi_{[a,b]})$: If $a = b$, then $\Pi_{[a,b]} = \Pi_a$ and run $y \text{ VerifyPath}(crs, com, a, \Pi_a)$ then set $\mathcal{L} = (x, y)$ if $y \in [0, 2^\ell)$, $\mathcal{L} = \emptyset$ if $y = \perp$ and $\mathcal{L} = y$ if $y = bad$. Otherwise, $[a, b]$ is not a singleton set and let $\Pi_{[a,b]} = (\Pi_{\mathcal{S}}, \Pi_{\mathcal{P}})$ where $\mathcal{S}$ is the set of keys returned. Initialize $\mathcal{L}' = \emptyset$.
 - $\diamond \Pi_{\mathcal{S}} = nil$: Let $\Pi_{\mathcal{P}} = \{C_x, R_x, \{C_{x|i0}, C_{x|i1}, \tau_{x|i}\}_{0 \leq i < |x|}\}_{x \in \mathcal{P}}$. For each $x \in \mathcal{P}$, the canonical covering of $[a, b]$,
 1. Run $\text{EVerify}(crs, C_x, R_x)$ to verify that C_x is a soft commitment as claimed. Set and output $\mathcal{L} = bad$ if it fails.
 2. At depth i , from $|x| - 1$ to 0 , verify that $C_{x|i}$ commits to its children with $\text{SVerify}(crs, (C_{x|i0}, C_{x|i1}), C_{x|i}, \tau_{x|i})$ then set and output $\mathcal{L} = bad$ if it rejects.
 - $\diamond \Pi_{\mathcal{S}} \neq nil$: Let $\Pi_{\mathcal{P}} = \{C_x, R_x, \{C_{x|i0}, C_{x|i1}, \tau_{x|i0}\}_{h_x \leq i < |x|}\}_{x \in \mathcal{P}}$.
 1. Run $\mathcal{L}' \leftarrow \text{VerifyST}(crs, com, \mathcal{S}, \Pi_{\mathcal{S}})$.
 2. For each $x \in \mathcal{P}$, the canonical covering of $[a, b] \setminus \mathcal{S}$,
 - a. Run $\text{EVerify}(crs, C_x, R_x)$ then set and output $\mathcal{L} = bad$ if any verification fail.
 - b. At depth i , from $|x| - 1$ to h_x as defined previously, check that $C_{x|i}$ soft-opens to its children with $\text{SVerify}(crs, (C_{x|i0}, C_{x|i1}), C_{x|i}, \tau_{x|i})$ then set and output $\mathcal{L} = bad$ if any verifications fail.
- Return $\mathcal{L} = \mathcal{L}'$, if $\mathcal{L} \neq bad, (x, y)$.

6.5.2 The ZK-EEDB Construction

The ZK-EEDB construction exploits the standard representation of $D = \{(x, D(x))\}$ as a set of key-value pairs and the “reversed dictionary” $D^{-1} = \{(y, D^{-1}(y))\}$, a set of pairs, consisting of a value y and the set of keys $D^{-1}(y)$ whose value is y in D . While ZK-EDBs compute their digests with a single Merkle tree, ZK-EEDB uses two such trees. The first of height ℓ , which we call the key tree, commits to the value $D(x)$ at each leaf $x \in [D]$ while the other also of height ℓ and called the value tree, commits to a size-hiding digest $com_{D_y^{-1}}$ of the set $D^{-1}(y)$ at each leaf $y \in [D^{-1}]$. Recall that $D^{-1}(y) = \{x \mid (x, y) \in D\}$, $D^{-1}(y)$

is committed in $com_{D_y^{-1}}$ using a Merkle tree with a slightly different representation, $D_y^{-1} = \{(x, 1) \mid (x, y) \in D\}$. The leaves of the Merkle tree with root value $com_{D_y^{-1}}$ is a hard commitment to 1, if and only if $x \in D^{-1}(y)$.

The second Merkle tree can be regarded as a digest of the “reversed database” $D_{com}^{-1} = \{(y, com_{D_y^{-1}}) \mid D^{-1}(y) \neq \emptyset\}$. For simplicity, we will refer to D_{com}^{-1} as D^{-1} from here on. This second Merkle tree is the key to achieve efficient queries over values and records. This is because soft authentication paths in the second tree are efficient proofs of statements that some value y does not appear in D , i.e., $D^{-1}(y) = \emptyset$. For existing ZK-EDB schemes, which use a single Merkle tree, such queries require exponential time to prove in zero-knowledge as each key has to be individually proven to not be associated to y in D , i.e. $D(x) \neq y$ for all $x \in [0, 2^\ell]$. In contrast, with ZK-EEDB, such non-membership of values are proven by simply revealing that the digest of D_y^{-1} is a soft commitment and does not commit to anything in particular. This is accomplished by explaining the soft commitment stored at the leaf corresponding to y in the second Merkle tree and then producing a soft authentication path from y to the root.

The introduction of a second Merkle tree that purportedly commits to the same database adds some complication to the ZK-EEDB. In particular, we need to ensure that both Merkle trees are consistent and commit to the same database. What we do is to enforce consistency checks within every proof generated by ZK-EEDB. When some record (x, y) is proven to be in D through the first Merkle tree, the same statement is proven with the second Merkle tree, i.e. $(x, 1) \in D_y^{-1}$ and $com_{D_y^{-1}}$ is committed at leaf y of the second Merkle tree. This stops malicious provers from proving contradictory statements between the two Merkle trees since the committed database in both trees have to agree at any record proven to be in D . Now, we describe the construction of ZK-EEDB, $\mathcal{EZ}$, from a trapdoor mercurial commitment scheme, TMC , starting with the initialization and database commitment algorithms. Figure 6.2 illustrates the Merkle trees that will be computed for an example database.

- $(crs, tk) \leftarrow \mathcal{EZ}.\text{Init}(1^\lambda)$: Output $(crs, tk) \leftarrow \text{TMC}.\text{Setup}(1^\lambda)$.
- $(com, \Delta) \leftarrow \mathcal{EZ}.\text{ComDB}(crs, D)$:
 1. Create the key Merkle tree $(com_D, \Delta_D) \leftarrow \text{BuildTree}(crs, D)$.
 2. For each y with $D^{-1}(y) \neq \emptyset$, compute digests of D_y^{-1} with $(com_{D_y^{-1}}, \Delta_{D_y^{-1}}) \leftarrow \text{BuildTree}(crs, D_y^{-1})$.
 3. Create the value tree $(com_{D^{-1}}, \Delta_{D^{-1}}) \leftarrow \text{BuildTree}(crs, D^{-1})$.

Output the digest $com = (com_D, com_{D^{-1}})$ and decommitment information $\Delta = (\Delta_D, \Delta_{D^{-1}}, \{com_{D_y^{-1}}, \Delta_{D_y^{-1}}\}_{y \in [D^{-1}]})$.

To prove range queries over records, we exploit the proof generation algorithms for intervals **Openl** and **Verifyl**. There are three cases to consider, if it is a range query over keys, values or records, where the corresponding input range $\mathfrak{R}$ is $[a_x, b_x] \times [0, 2^\ell)$, $[0, 2^\ell) \times [a_y, b_y]$ and $[a_x, b_x] \times [a_y, b_y]$ respectively. We denote the answer set to the range query with $\mathcal{L}$.

If $\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$, we generate a proof for the interval $[a_x, b_x]$ in the key Merkle tree to prove that $\mathcal{L} = D \cap ([a_x, b_x] \times [0, 2^\ell))$. Then, to show that the two Merkle trees are consistent, we prove for every record $(x, y) \in \mathcal{L}$ that $(x, 1) \in D_y^{-1}$ in the digest $com_{D_y^{-1}}$, which is itself committed at leaf y of the value Merkle tree. In a similar fashion, for $\mathfrak{R} = [0, 2^\ell) \times [a_y, b_y]$, we generate a proof for the interval $[a_y, b_y]$ in the value Merkle tree. For every leaf $y \in [a_y, b_y]$ with a hard commitment to $com_{D_y^{-1}}$, we reveal the entire Merkle tree that yields the digest $com_{D_y^{-1}}$. Then, for consistency, we prove that every record $(x, y) \in \mathcal{L}$ is correctly committed in the key Merkle tree.

Lastly, for $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$, we start with the value Merkle tree. For each $y \in [a_y, b_y]$, we compute a hard (or soft) authentication path for $com_{D_y^{-1}}$ if $D^{-1}(y) \neq \emptyset$ (or $D^{-1}(y) = \emptyset$ respectively). Then, for each $com_{D_y^{-1}}$, we generate a range proof for the interval $[a_x, b_x]$. Finally, we prove that every record returned in $\mathcal{L}$ is correctly committed in the key Merkle tree as well. Unlike when $\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$, we cannot use **Openl** on the value Merkle tree as it would reveal which values $y \in [a_y, b_y]$ have $D^{-1}(y) = \emptyset$ and result in a proof that is not zero-knowledge. Therefore, we prove the contents of $D^{-1}(y) \cap [a_x, b_x]$ for every $y \in [a_y, b_y]$.

In the first two cases, the range query can support exponential-sized intervals for keys and values, $[a_x, b_x]$ and $[a_y, b_y]$ respectively. However, the range query over records can only support exponential-sized $[a_x, b_x]$ but polynomial-sized $[a_y, b_y]$. Recall that $(com = (com_D, com_{D^{-1}}), \Delta = (\Delta_D, \Delta_{D^{-1}}, \{com_{D_y^{-1}}, \Delta_{D_y^{-1}}\}_{y \in [D^{-1}]}))$ from $\text{ComDB}(crs, D)$.

— $\Pi_{\mathfrak{R}} \quad \mathcal{EZ}.\text{ProveRQ}(crs, (com, \Delta), \mathfrak{R})$:

$\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$:

1. Generate the proof for the non-empty database keys in $[a_x, b_x]$ with the key Merkle tree, $\Pi_D \leftarrow \text{Openl}(crs, (com_D, \Delta), [a_x, b_x])$.
2. Let $\mathcal{L} \leftarrow \text{Verifyl}(crs, com_D, [a_x, b_x], \Pi_D)$.
 - ◊ If $\mathcal{L} = \emptyset$, set $\Pi_{D^{-1}} = \text{nil}$.
 - ◊ Otherwise, for every $(x, y) \in \mathcal{L}$, in the value Merkle tree:

- a. Generate the hard authentication path for (x, y) in $com_{D_y^{-1}}$ by running $\Pi_{x, D_y^{-1}} \leftarrow \mathbb{H}\text{OpenPath}(crs, (com_{D_y^{-1}}, \Delta), x)$.
- b. Generate the hard authentication path for (x, y) in $com_{D^{-1}}$ by running $\Pi_{x, D^{-1}} \leftarrow \mathbb{H}\text{OpenPath}(crs, (com_{D^{-1}}, \Delta), y)$.

Set $\Pi_{D^{-1}} = \{\Pi_{x, D_y^{-1}}, \Pi_{x, D^{-1}}\}_{(x, y) \in \mathcal{L}}$.

$\mathfrak{R} = [0, 2^\ell) \times [a_y, b_y]$:

1. Generate the proof for $[a_y, b_y]$ in the value Merkle tree by running $\Pi'_{D^{-1}} \leftarrow \text{Openl}(crs, (com_{D^{-1}}, \Delta), [a_y, b_y])$.
2. Let $\mathcal{L}' \leftarrow \text{Verify}(crs, com_{D^{-1}}, [a_y, b_y], \Pi'_{D^{-1}})$
 - ◇ If $\mathcal{L}' = \emptyset$, set $\Pi_D = nil$ and $\Pi_{D^{-1}} = \Pi'_{D^{-1}}$.
 - ◇ Otherwise, for each $(y, com_{D_y^{-1}}) \in \mathcal{L}'$, compute the interval proof $\Pi_y \leftarrow \text{Openl}(crs, (com_{D_y^{-1}}, \Delta), [0, 2^\ell))$.
 - a. Let $\mathcal{L}_y \leftarrow \text{Verify}(crs, com_{D_y^{-1}}, [0, 2^\ell), \Pi_y)$ and $\mathcal{L} = \bigcup_{y \in [a_y, b_y]} \mathcal{L}_y$.
 - b. In the key Merkle tree, compute the authentication Steiner tree $\Pi_D \leftarrow \mathbb{H}\text{OpenST}(crs, (com_D, \Delta), [\mathcal{L}])$.

Set $\Pi_{D^{-1}} = (\Pi'_{D^{-1}}, \{\Pi_y\}_{y \in [\mathcal{L}]})$.

$\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$:

1. For each $y \in [a_y, b_y]$, in the value Merkle tree:
 - ◇ If $D_y^{-1} = \emptyset$:
 - a. Run $\Pi_{D_y^{-1}} \leftarrow \mathbb{S}\text{OpenPath}(crs, (com_{D^{-1}}, \Delta), y)$ to obtain the soft authentication path for leaf y of the value Merkle tree.
 - b. Let $\Pi_{D_y^{-1}} = (\perp, \{C_{y|j}, C_{(y|j)'}\}_{1 \leq j \leq \ell}, \{\tau_{y|j}\}_{0 \leq j \leq \ell})$ and denote a modified authentication path (removing the value $\perp$ and proof that C_y opens to $\perp$) with $\Pi'_{D_y^{-1}} = (\{C_{y|j}, C_{(y|j)'}\}_{1 \leq j \leq \ell}, \{\tau_{y|j}\}_{0 \leq j \leq \ell-1})$.
 - c. Compute $com_{D_y^{-1}} \leftarrow \mathbb{S}\text{Commit}(crs; R_{D_y^{-1}})$, a digest for D_y^{-1} and a soft opening of $C_y = \mathbb{S}\text{Commit}(crs; R_y)$ to $com_{D_y^{-1}}$, given by $\tau'_y \leftarrow \mathbb{S}\text{Open}(crs, com_{D_y^{-1}}; R_y)$.
 - d. Generate a proof that $[a_x, b_x] \cap D^{-1}(y) = \emptyset$ by running $\Pi_{D_y^{-1}, [\mathfrak{R}]} \leftarrow \text{Openl}(crs, (com_{D_y^{-1}}, \Delta), [a_x, b_x])$.
 - Set $\Pi_{y, D^{-1}} = (\Pi'_{D_y^{-1}}, \tau'_y, com_{D_y^{-1}}, \Pi_{D_y^{-1}, [\mathfrak{R}]})$.
 - ◇ Otherwise, $D_y^{-1} \neq \emptyset$:
 - a. Run $\Pi_{D_y^{-1}} \leftarrow \mathbb{H}\text{OpenPath}(crs, (com_{D^{-1}}, \Delta), y)$ to obtain the hard authentication path to y , $\Pi_{D_y^{-1}} = (com_{D_y^{-1}}, \{C_{y|j}, C_{(y|j)'}\}_{1 \leq j \leq \ell}, \{\pi_{y|j}\}_{0 \leq j \leq \ell})$.

- b. Let $\Pi'_{D_y^{-1}} = (\{C_{y|j}, C_{(y|j)'}\}_{1 \leq j \leq \ell}, \{\pi_{y|j}\}_{0 \leq j \leq \ell-1})$.
- c. Generate a proof for the keys in $[a_x, b_x] \cap D^{-1}(y)$, running $\Pi_{D_y^{-1}, [\mathfrak{R}]}$
 $\text{Openl}(crs, (com_{D_y^{-1}}, \Delta), [a_x, b_x])$.
 Set $\Pi_{y, D^{-1}} = (\Pi'_{D_y^{-1}}, \pi_y, com_{D_y^{-1}}, \Pi_{D_y^{-1}, [\mathfrak{R}]})$.
- 2. In the key Merkle tree:
 - a. For $y \in [a_y, b_y]$, run $\mathcal{L}_y \leftarrow \text{Verifyl}(crs, com_{D_y^{-1}}, [0, 2^\ell], \Pi_{D_y^{-1}, [\mathfrak{R}]})$ to get the keys in $[a_x, b_x] \cap D^{-1}(y)$.
 - b. Let $\mathcal{L}' = \bigcup_{y \in [a_y, b_y]} \mathcal{L}_y$ and compute the authentication Steiner tree, $\Pi_D \leftarrow \mathbb{H}\text{OpenST}(crs, (com_D, \Delta), \mathcal{L}')$.

Output $\Pi_{\mathfrak{R}} = (\Pi_D, \Pi_{D^{-1}})$ and add the randomness of any commitment computed in the algorithm to Δ .

— $\mathcal{L} \leftarrow \mathcal{EZ}.\text{VerifyRQ}(crs, com, \mathfrak{R}, \Pi_{\mathfrak{R}})$: Let $\Pi_{\mathfrak{R}} = (\Pi_D, \Pi_{D^{-1}})$.

$\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell]$:

1. In the key Merkle tree, run $\mathcal{L}' \leftarrow \text{Verifyl}(crs, com_D, [a_x, b_x], \Pi_D)$ to verify the interval proof Π_D .
 - ◇ If $\mathcal{L}' = \text{bad}$, output $\mathcal{L} = \mathcal{L}'$.
 - ◇ If $\Pi_{D^{-1}} = \text{nil}$ and $\mathcal{L}' = \emptyset$, output $\mathcal{L} = \mathcal{L}'$.
 - ◇ Otherwise, if $\Pi_{D^{-1}} = \{\Pi'_{x, D_y^{-1}}, \Pi_{x, D^{-1}}\}_{(x, y) \in \mathcal{L}^*}$, output $\mathcal{L} = \text{bad}$ should $\mathcal{L}^* \neq \mathcal{L}'$.
 - ◇ Otherwise, continue to Step 2.
2. In the value Merkle tree, for each $(x, y) \in \mathcal{L}'$:
 - a. Compute $ans \leftarrow \text{VerifyPath}(crs, com_{D^{-1}}, y, \Pi_{x, D^{-1}})$ and output $\mathcal{L} = \text{bad}$ if $ans \neq com_{D_y^{-1}}$.
 - b. Otherwise, $ans = com_{D_y^{-1}}$ and verify the hard authentication path to x , $ans' \leftarrow \text{VerifyPath}(crs, ans, x, \Pi_{x, D_y^{-1}})$ and output $\mathcal{L} = \text{bad}$ if $ans' \neq 1$.
3. Output $\mathcal{L}$ if the algorithm reaches the end without terminating.

$\mathfrak{R} = [0, 2^\ell] \times [a_y, b_y]$: Let $\Pi_{\mathfrak{R}} = (\Pi_D, \Pi_{D^{-1}})$.

- ◇ If $\Pi_D = \text{nil}$, in the value Merkle tree:
 1. Compute $\mathcal{V} \leftarrow \text{Verifyl}(crs, com_{D^{-1}}, [a_y, b_y], \Pi_{D^{-1}})$.
 - * If $\mathcal{V} \neq \emptyset$, output $\mathcal{L} = \text{bad}$.
 - * Otherwise, output $\mathcal{L} = \emptyset$.

- ◇ Otherwise, $\Pi_{D^{-1}} = (\Pi'_{D^{-1}}, \{\Pi_y\}_{y \in \mathcal{L}^*})$ for some $\mathcal{L}^*$.
 1. In the value Merkle tree:
 - a. Compute $\mathcal{V} \leftarrow \text{VerifyI}(\text{crs}, \text{com}_{D^{-1}}, [a_y, b_y], \Pi_{D^{-1}})$ and output $\mathcal{L} = \text{bad}$ if $[\mathcal{V}] \neq \mathcal{L}^*$.
 - b. For each $(y, \text{com}_{D_y^{-1}}) \in \mathcal{V}$, verify the interval proof with $\mathcal{L}_y \leftarrow \text{VerifyI}(\text{crs}, \text{com}_{D_y^{-1}}, [0, 2^\ell], \Pi_y)$ and output $\mathcal{L} = \text{bad}$ if $\mathcal{L}_y = \emptyset$ or bad .
 - c. Otherwise, let $\mathcal{L}' = \bigcup_{y \in [\mathcal{L}']} \bigcup_{x \in [\mathcal{L}_y]} \{(x, y)\}$.
 2. In the key Merkle tree, verify the authentication Steiner tree $\mathcal{L}^* \leftarrow \text{VerifyST}(\text{crs}, \text{com}_D, [\mathcal{L}], \Pi_D)$ and output $\mathcal{L} = \text{bad}$ if $\mathcal{L}^* \neq \mathcal{L}'$.
 3. Output $\mathcal{L} = \mathcal{L}^*$ if the algorithm reaches the end without terminating.
- $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$: Let $\Pi_{\mathfrak{R}} = (\Pi_D, \Pi_{D^{-1}})$ and output $\mathcal{L} = \text{bad}$ if $\Pi_{D^{-1}} \neq \{\Pi_{y, D^{-1}}\}_{y \in [a_y, b_y]}$.
 1. In the value Merkle tree, for each $y \in [a_y, b_y]$, let $\Pi_{y, D^{-1}} = (\Pi'_{D_y^{-1}}, \eta_y, \text{com}_{D_y^{-1}}, \Pi_{D_y^{-1}, [\mathfrak{R}]})$.
 - a. Run $\text{ans}' \leftarrow \text{VerifyPath}(\text{crs}, \text{com}_{D^{-1}}, y|_{\ell-1}, \Pi'_{D^{-1}})$ to verify the authentication path for $y|_{\ell-1}$ and reject if $\text{ans}' \neq (C_{y|_{\ell-1}0}, C_{y|_{\ell-1}1})$.
 - b. Compute $\mathbb{X}\text{Verify}(\text{crs}, \text{ans}, \text{com}_{D_y^{-1}}, \eta_y)$, where $\mathbb{X} = \mathbb{H}$ if η_y is a hard opening and $\mathbb{X} = \mathbb{S}$ if η_y is a soft opening, and output $\mathcal{L} = \text{bad}$ if verification fails.
 - c. Compute $\mathcal{L}_y \leftarrow \text{VerifyI}(\text{crs}, \text{com}_{D_y^{-1}}, [a_x, b_x], \Pi_{D_y^{-1}, [\mathfrak{R}]})$ and output $\mathcal{L} = \text{bad}$ if $\mathcal{L}_y = \text{bad}$.
 - d. Let $\mathcal{L}' = \bigcup_{y \in [a_y, b_y]} \bigcup_{x \in [\mathcal{L}_y]} \{(x, y)\}$.
 - ◇ If $\mathcal{L}' = \emptyset$ and $\Pi_D \neq \text{nil}$, output $\mathcal{L} = \text{bad}$.
 - ◇ If $\mathcal{L}' = \emptyset$ and $\Pi_D = \text{nil}$, output $\mathcal{L} = \emptyset$.
 - ◇ Otherwise, continue to Step 2.
 2. In the key Merkle tree, run $\mathcal{L}^* \leftarrow \text{VerifyST}(\text{crs}, \text{com}_D, [\mathcal{L}], \Pi_D)$ to verify the authentication Steiner tree and output $\mathcal{L} = \text{bad}$ if $\mathcal{L}^* \neq \mathcal{L}'$.
 3. Output $\mathcal{L} = \mathcal{L}^*$ if the algorithm reaches the end without terminating.

Proof Size. ProveRQ has three different types of proofs, depending on the input range. First, $\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$, corresponding to a range query over keys. Using $\mathcal{L}$ to denote the answer to the range query with the given $\mathfrak{R}$, the proof consists of the authentication Steiner tree of $\mathcal{L}$ in the key Merkle tree and individual hard authentication paths in the value Merkle tree and Merkle tree for D_y^{-1} . Separately, the Steiner tree has $\mathcal{O}(|\mathcal{L}|\ell)$ nodes

and each authentication path has ℓ nodes. There are two paths per record in $\mathcal{L}$ which mean that the total proof size is $\mathcal{O}(|\mathcal{L}|\ell)$.

Second, $\mathfrak{R} = [0, 2^\ell) \times [a_y, b_y]$, corresponding to a range query over values. With $\mathcal{V}$ denoting the set of distinct values in the answer set $\mathcal{L}$, we partition $\mathcal{L} = \bigsqcup_{y \in \mathcal{V}} \mathcal{L}_y$, where each $\mathcal{L}_y$ is the subset of $\mathcal{L}$ containing all records whose value is y . For this input range $\mathfrak{R}$, ProveRQ computes one authentication Steiner tree in the value Merkle tree and an authentication Steiner tree in the Merkle tree for D_y^{-1} per $y \in \mathcal{V}$. Finally, a hard authentication path per record in $\mathcal{L}$ is generated in the key Merkle tree. The authentication Steiner tree in the value Merkle tree has size $\mathcal{O}(|\mathcal{V}|\ell)$ while the same for the Merkle tree for D_y^{-1} has size $\mathcal{O}(|\mathcal{L}_y|\ell)$ per $y \in \mathcal{V}$. The authentication paths in the key Merkle tree all together have size $\mathcal{O}(|\mathcal{L}|\ell)$, which puts the total proof size at $\mathcal{O}((|\mathcal{L}| + |\mathcal{V}|K)\ell)$, where $K = \max_{y \in \mathcal{V}} |\mathcal{L}_y|$.

In the third and final case, neither the interval of keys $[a_x, b_x]$ nor values $[a_y, b_y]$ consist of the entire possible range. Let $\mathcal{L} = \bigsqcup_{y \in [a_y, b_y]} \mathcal{L}_y$ denote the answer to the range query on $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$. ProveRQ would compute $(b_y - a_y)$ authentication paths in the value Merkle tree and the same number of authentication Steiner trees; one per $y \in [a_y, b_y]$ for the Merkle tree for D_y^{-1} . Lastly, $|\mathcal{L}|$ hard authentication paths are generated in the key Merkle tree. The Steiner trees have $\mathcal{O}(|\mathcal{L}_y|\ell)$ nodes each which means that overall, the proof size has $\mathcal{O}((b_y - a_y)(1 + K) + |\mathcal{L}|\ell)$ nodes, where $K = \max_{y \in [a_y, b_y]} |\mathcal{L}_y|$. As evident from the proof sizes, ProveRQ supports super-polynomial intervals over the key-space for any query and the same for intervals over the value-space for range queries over values; only polynomial-sized intervals over the value-space are supported for range queries over records.

6.5.3 Security of the Construction

The three security properties of ZK-EEDB are correctness, soundness and zero-knowledge respectively. Correctness is straight-forward from the description of the construction and we will prove the other two properties in the following theorems.

Theorem 6.5.1. *The ZK-EEDB, $\mathcal{EZ}$, from the construction in Section 6.5.2 is sound if instantiated with a trapdoor mercurial commitment scheme that satisfies the mercurial-binding property.*

Proof. Let crs be the common reference string generated with $\mathcal{EZ}.Init$ and com be the database digest from $\mathcal{EZ}.ComDB$. Assume that an adversary can produce two ranges with valid but contradicting proofs $\mathfrak{R}, \Pi, \mathfrak{R}', \Pi'$. Then, there must exist some record $(x, y) \in \mathfrak{R} \cap \mathfrak{R}'$ such that $(x, y) \in \mathcal{L} = \mathcal{EZ}.VerifyEQ(crs, com, \mathfrak{R}, \Pi)$ but somehow the

same record $(x, y) \notin \mathcal{L}' = \mathcal{EZ}.\text{VerifyEQ}(crs, com, \mathfrak{R}', \Pi')$. There are two possibilities: (i) there exists some other record $(x, y') \in \mathcal{L}'$ with $y' \neq y$ or (ii) there exists no $y' \in [0, 2^\ell)$ with $x, y' \in \mathcal{L}'$.

In the first case, both (x, y) and (x, y') have valid hard authentication paths in the key Merkle tree. Therefore, Π and Π' contain valid hard openings to two distinct values for the commitment at leaf x , C_x . This violates the mercurial-binding property of the trapdoor mercurial commitment scheme.

In the second case, there exists no record with key x in the answer of the second query $\mathcal{L}'$. This means that Π' has a proof that $(x, 1) \notin D_y^{-1}$ since Π' claims that $x \notin D^{-1}(y)$. On the other hand, Π contains a proof that $(x, 1) \in D_y^{-1}$ which is a contradiction between the range proofs Π and Π' . For this to occur, either the digest $com_{D_y^{-1}}$ of D_y^{-1} differs between Π and Π' or the digest is the same but somehow leaf x of the Merkle tree for D_y^{-1} can be (hard and soft) opened to two different values. In the former instance, Π contains a valid hard opening to one value of $com_{D_y^{-1}}$ while Π' contains a valid soft opening or explanation for the commitment at leaf y or some node along the path from y to the root. Thus, the mercurial-binding property of the trapdoor mercurial commitment scheme is contradicted since some mercurial commitment has been soft-opened to one message and hard-opened to another. In the latter instance, similar to the argument above, in the Merkle tree for D_y^{-1} , Π contains a valid hard opening for the commitment at the leaf x , C_x , to one value and Π' contains a valid soft-opening or explanation for the same commitment or some commitment at an internal node along the path from x to the root. As before, the mercurial-binding property is violated. $\square$

Theorem 6.5.2. *The ZK-EEDB, $\mathcal{EZ}$, from the construction in Section 6.5.2 satisfies the zero-knowledge property if instantiated with a trapdoor mercurial commitment scheme that satisfies the four equivocation properties, HH, HS, SS, SE equivocations.*

Proof. The ZK-EEDB simulator, SInit , SCom and SProveRQ^D , which we construct below, is similar to the ZK-EDB simulator but additionally relies on FakeExplain to simulate range proofs. Outputs of the simulator are indistinguishable from outputs of an honest prover through the equivocation properties of the trapdoor mercurial commitment scheme used.

- $(crs', st_0) \leftarrow \text{SInit}(1^\lambda)$: Run $(crs, tk) \leftarrow \text{Init}(1^\lambda)$ and output the common reference string $crs' = crs$ and add the trapdoor to the simulator state $st_0 = tk$.
- $(com', st_1) \leftarrow \text{SCom}(st_0)$: Generate fake digests with $com'_D \leftarrow \text{MFake}(crs'; R_0)$ and $com'_{D^{-1}} \leftarrow \text{MFake}(crs'; R_1)$. Output the simulated ZK-EEDB digest $com' =$

- $(com'_D, com'_{D^{-1}})$ and update the simulator state with the randomness used, $st_1 = (st_0, R_0, R_1)$.
- $\Pi' \leftarrow \text{SProveRQ}^D(crs', st_1)$: Let $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$ and obtain the set $\mathcal{S} = D \cap \mathfrak{R}$ by querying the database oracle. Denote with $\mathcal{S}' \in st_1$, the set of fake commitments and proofs generated in previous queries. Finally, let $\mathcal{S}_{D^{-1}}$ be the set of distinct values in $\mathcal{S}$ and $[\mathcal{S}_y]$ the set of keys in $[\mathcal{S}]$ whose values in D are y . Proceed to compute Π' in the following manner:
 1. The input range $\mathfrak{R}$ and set returned $\mathcal{S}$ define interval proofs, authentication Steiner trees and paths that will prove the correctness the answer.
 - a. If $[a_y, b_y] = [0, 2^\ell)$, then the answer $\mathcal{S}$ is proven with an interval proof in the key Merkle tree and $|\mathcal{S}|$ many hard authentication paths in the value Merkle tree and Merkle trees for D_y^{-1} for $y \in \mathcal{S}_{D^{-1}}$.
 - b. If $[a_x, b_x] = [0, 2^\ell)$, then $[a_y, b_y]$ requires an interval proof in the value Merkle tree. For every $y \in \mathcal{S}_{D^{-1}}$, $[a_x, b_x]$ induces an interval proof in the Merkle tree for D_y^{-1} . Lastly, $[\mathcal{S}]$ defines an authentication Steiner tree in the key Merkle tree.
 - c. For all other $\mathfrak{R}$, then $|\mathcal{S}_{D^{-1}}|$ authentication paths and $|\mathcal{S}_{D^{-1}}|$ interval proofs, one for each $y \in \mathcal{S}_{D^{-1}}$ are required. For consistency, $[\mathcal{S}]$ defines a n authentication Steiner tree in the key Merkle tree.
 2. For each of the three range types, let $\mathcal{N}$ be the set of nodes in the interval proofs, authentication Steiner trees and paths required to prove correctness of the answer $D \cap \mathfrak{R}$. For every node $x \in \mathcal{N} \setminus \mathcal{S}'$ that have not been involved in any previous proof, generate fake commitments $C_x \leftarrow \text{MFake}(crs'; R_x)$.
 3. Compute appropriate hard and soft openings and explanations for the fake commitments generated in the previous step using the equivocation algorithms HEquivocate , SEquivocate and FakeExplain to simulate an honest prover's actions.
 4. Add the fake commitments produced in Step 2 and any openings and explanations to the simulator's state st_1 .

Outputs of the simulator are indistinguishable from the same from an honest prover because the trapdoor mercurial commitment scheme used to instantiate the construction satisfies the equivocation properties. The simulator has two types of outputs, the common reference string at initialization and fake commitments and its equivocations and

explanations. The former is indistinguishable from a real common reference string as both are trapdoor mercurial commitment keys from the same distribution. From the HH, HS, SS and SE equivocation conditions, the joint distribution of fake commitments and their equivocations, hard or soft, and explanations are statistically close to that of hard and soft commitments and their respective hard or soft openings or explanations. $\square$

6.6 A Lattice-based Trapdoor Mercurial Commitment Scheme

In this section, we introduce a construction of trapdoor mercurial commitment (TMC) from standard lattices. All known TMC schemes can be derived from the generic constructions proposed by Chase *et al.* [48] and Catalano *et al.* [45]. However, that is not the case for our proposed TMC scheme, which draws inspiration from the original TMC scheme by Micali *et al.* [153] and KTX commitment scheme by Kawachi *et al.* [119].

Similar to the KTX scheme, the mercurial commitment key is made up of two random matrices $(\mathbf{A}_0, \mathbf{A}_1)$, but $\mathbf{A}_1$ has an associated trapdoor $\mathbf{T}$. Instead of directly using the public matrices, commitment algorithms apply some randomness, $\mathbf{R}$, to $\mathbf{A}_1$, transforming it into a random matrix $\mathbf{B}$; $\mathbf{R}$ additionally serves as a trapdoor in soft mode. Then, the message is committed with $\mathbf{A}_0$ and $\mathbf{B}$ with some randomness $\mathbf{r}$, drawn from a discrete Gaussian, into a vector $\mathbf{c}$ using the same procedures as `KTX.Commit`; the mercurial commitment corresponds to the pair $(\mathbf{c}, \mathbf{B})$.

Hard openings would reveal that $\mathbf{c}$ commits to the message $\mathbf{m}$ with randomness $\mathbf{r}$ and $\mathbf{B}$ is $\mathbf{A}_1$ randomized by some matrix $\mathbf{R}$ and does not have any trapdoor. Soft openings would reveal that $\mathbf{c}$ commits to some message $\mathbf{m}$ with some randomness $\mathbf{r}'$; soft commitments can generate suitable randomness to open $\mathbf{c}$ to any message using the trapdoor information $\mathbf{R}$. However, for soft commitments, $\mathbf{B}$ has a trapdoor embedded within and cannot be hard opened at all.

6.6.1 The Construction

Let the message space of the scheme be $\{0, 1\}^l$ for some $l = \text{poly}(\lambda)$. For lattice dimension $n = \mathcal{O}(\lambda)$ and prime modulus $q = \tilde{\mathcal{O}}(ln^2 + n^4)$, let $w = n\lceil \log q \rceil$, $\bar{m} = 2n\lceil \log q \rceil$ and $m = \bar{m} + w$. Let $s = \Omega(\sqrt{n \log q \log n})$ be a Gaussian parameter. We use the trapdoor generation and Gaussian sampling algorithms, `TrapGen` and `SampleD` from Lemma 2.6.3.

- $(mpk, msk) \leftarrow \text{TMSetup}(1^\lambda)$: Generate a matrix $\mathbf{A}_0 \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times l})$ and then compute $(\mathbf{A}_1, \mathbf{T}) \leftarrow \text{TrapGen}(n, m, q)$; $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$ is statistically close to uniform with

- $\mathbf{T} \in \mathbb{Z}^{\bar{m} \times w}$ as its trapdoor. Output $mpk = (\mathbf{A}_0, \mathbf{A}_1)$ and $msk = \mathbf{T}$.
- $\mathbf{C} \leftarrow \mathbb{H}\text{Commit}(mpk, \mathbf{m}; (\mathbf{R}, \mathbf{r}))$: For input message $\mathbf{m} \in \{0, 1\}^l$ and randomness $\mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}$ and $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$, let the extended matrix $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times (m+w)}$, with $\mathbf{B}_1 = \mathbf{A}_1 \mathbf{R} \in \mathbb{Z}_q^{n \times w}$. Then, compute $\mathbf{c} = \mathbf{A}_0 \mathbf{m} + \mathbf{B} \mathbf{r} \in \mathbb{Z}_q^n$ and output $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$.
 - $\pi \leftarrow \mathbb{H}\text{Open}(mpk, \mathbf{m}; (\mathbf{R}, \mathbf{r}))$: Output the randomness used to generate the hard commitment, $\pi = (\mathbf{R}, \mathbf{r}) \in \mathbb{Z}^{m \times w} \times \mathbb{Z}^{m+w}$.
 - $\mathbb{H}\text{Verify}(mpk, \mathbf{m}, \mathbf{C}, \pi)$: For input commitment $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$ and supposed hard opening $\pi = (\mathbf{R}, \mathbf{r})$,
 1. Reject if $\mathbf{R} = [\mathbf{r}_1 \mid \dots \mid \mathbf{r}_w]$ such that $\|\mathbf{r}_i\| > s\sqrt{m}$ for some $i \in [w]$ or $\|\mathbf{r}\| > s\sqrt{m+w}$.
 2. Let $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times (m+w)}$ and accept if $\mathbf{B}_1 = \mathbf{A}_1 \mathbf{R}$ and $\mathbf{c} = \mathbf{A}_0 \mathbf{m} + \mathbf{B} \mathbf{r}$.
 - $\mathbf{C} \leftarrow \mathbb{S}\text{Commit}(mpk; (\mathbf{R}, \mathbf{r}))$: For input randomness $\mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}$ and $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$, compute the extended matrix $\mathbf{B} = [\mathbf{A} \mid \mathbf{G} - \mathbf{A}_1 \mathbf{R}] \in \mathbb{Z}_q^{n \times (m+w)}$, where $\mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1 \mathbf{R}$, and vector $\mathbf{c} = \mathbf{B} \mathbf{r} \in \mathbb{Z}_q^n$. Output $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times (m+w)}$. Observe that the matrix $\mathbf{R}$ is a trapdoor for $\mathbf{B}$, in the sense of Definition 2.6.2.
 - $\tau \leftarrow \mathbb{S}\text{Open}(mpk, \mathbf{m}, \text{flag}; (\mathbf{R}, \mathbf{r}))$:

flag = $\mathbb{S}$: The soft commitment $\mathbf{C}$ has to be of the form $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \mathbf{r}, \mathbf{G} - \mathbf{A}_1 \mathbf{R})$, where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$. Compute $\mathbf{c}' = \mathbf{c} - \mathbf{A}_0 \mathbf{m}$ and sample a short vector $\mathbf{r}' \leftarrow \text{SampleD}(\mathbf{R}, \mathbf{B}, \mathbf{c}', s)$ using the trapdoor $\mathbf{R}$. From the properties of SampleD from Lemma 2.6.3, $\mathbf{r}'$ satisfies the equation $\mathbf{C} = \mathbf{A}_0 \mathbf{m} + \mathbf{B} \mathbf{r}'$ and $\|\mathbf{r}'\| \leq s\sqrt{m+w}$ with overwhelming probability. Output $\tau = \mathbf{r}' \in \mathbb{Z}^{m+w}$.

flag = $\mathbb{H}$: Output the randomness $\tau = \mathbf{r} \in \mathbb{Z}^{m+w}$.
 - $\mathbb{S}\text{Verify}(mpk, \mu, \mathbf{C}, \tau)$: Let $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$ and $\tau = \mathbf{r} \in \mathbb{Z}^{m+w}$. Accept if $\mathbf{c} = \mathbf{A}_0 \mathbf{m} + \mathbf{B} \mathbf{r}$, where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$ and $\|\mathbf{r}\| \leq s\sqrt{m+w}$ and reject otherwise.
 - $\mathbf{C} \leftarrow \mathbb{M}\text{Fake}(mpk; (\mathbf{R}, \mathbf{r}))$: For input randomness $\mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}$ and $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$, compute the extended matrix $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times m+w}$, where $\mathbf{B}_1 = \mathbf{A}_1 \mathbf{R}$ and then the vector $\mathbf{c} = \mathbf{B} \mathbf{r}$. Output $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1)$.
 - $\pi \leftarrow \mathbb{H}\text{Equivocate}(msk, \mathbf{m}; (\mathbf{R}, \mathbf{r}))$: Let $msk = \mathbf{T} \in \mathbb{Z}^{\bar{m}+w}$ and the fake commitment $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \mathbf{r}, \mathbf{A}_1 \mathbf{R})$, where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$ and compute $\mathbf{c}' =$

- $\mathbf{c} \leftarrow \mathbf{A}_0 \mathbf{m}$. Extend $\mathbf{T}$ into $\mathbf{T}_\mathbf{B}$, a trapdoor for $\mathbf{B}$, and sample a short vector $\mathbf{r}' \leftarrow \text{SampleD}(\mathbf{T}_\mathbf{B}, \mathbf{B}, \mathbf{c}', s)$. Output $\pi = (\mathbf{R}, \mathbf{r}') \in \mathbb{Z}^{m \times w} \times \mathbb{Z}^{m+w}$.
- $\tau \leftarrow \text{SEquivocate}(\text{msk}, \mathbf{m}; (\mathbf{R}, \mathbf{r}'))$: Obtain the hard equivocation with $(\mathbf{R}, \mathbf{r}') \leftarrow \text{HEquivocate}(\text{msk}, \mathbf{m}; (\mathbf{R}, \mathbf{r}'))$ and output $\tau = \mathbf{r}'$.
 - $(\mathbf{R}', \mathbf{r}') \leftarrow \text{FakeExplain}(\text{msk}; (\mathbf{R}, \mathbf{r}'))$: For input randomness $\mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}$ and $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$, let $\text{msk} = \mathbf{T} \in \mathbb{Z}^{\tilde{m} \times w}$ and $\mathbf{C} \leftarrow \text{MFake}(\text{msk}; (\mathbf{R}, \mathbf{r}))$. Sample a matrix $\mathbf{R}' = [\mathbf{r}'_1 \mid \dots \mid \mathbf{r}'_w]$ such that $\mathbf{A}_1 \mathbf{R}' = \mathbf{G} - \mathbf{A}_1 \mathbf{R}$ as follows:
 1. Let $\mathbf{G} = [\mathbf{g}_1 \mid \dots \mid \mathbf{g}_w]$ and $\mathbf{R} = [\mathbf{r}_1 \mid \dots \mid \mathbf{r}_w]$.
 2. For each $i \in [w]$, compute $\mathbf{r}'_i \leftarrow \text{SampleD}(\mathbf{T}, \mathbf{A}_1, \mathbf{g}_i - \mathbf{A}_1 \mathbf{r}_i)$ to obtain short vectors $\mathbf{r}'_i$.

Output $(\mathbf{R}', \mathbf{r})$, which satisfies $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B}\mathbf{r}, \mathbf{G} - \mathbf{A}_1 \mathbf{R}')$.

6.6.2 Security Properties of the Construction

Now, we prove that the construction in Section 6.6.1 is correct and satisfies the security properties described in Section 6.3.1.

Theorem 6.6.1. *The trapdoor mercurial commitment scheme in Section 6.6.1 is correct.*

Proof. From Lemma 2.3.6, samples from the discrete Gaussian distributions $D_{\mathbb{Z}^m, s}$ and $D_{\mathbb{Z}^{m+w}, s}$, with overwhelming probability, have ℓ_2 norms bounded by $s\sqrt{m}$ and $s\sqrt{m+w}$ respectively. Besides that, vectors sampled with the **SampleD** algorithm are statistically close to being sampled from the discrete Gaussian above. Therefore, correctly computed proofs π and τ satisfy the conditions of **HVerify** and **SVerify** except with negligible probability. In addition, the equations modulo q are correct by construction so the scheme is correct with overwhelming probability. $\square$

Recall the three security properties for trapdoor mercurial commitment schemes are mercurial-binding, mercurial-hiding and equivocation. In the following theorems, we state and prove that the scheme satisfies mercurial-binding and equivocation properties. The mercurial-hiding property is implied by the HH and SS equivocation conditions since the scheme is proper (i.e., soft openings are a proper subset of hard openings of any commitment).

Theorem 6.6.2. *The scheme is computationally mercurial-binding if the $\text{SIS}_{n, m, q, \beta}$ problem is hard for $\beta = s \cdot (l\sqrt{m} + \sqrt{s \cdot m\tilde{m}(s^2 w^3 + 2m)})$.*

Before proving the theorem, we introduce an important lemma from Gentry *et al.* [86] necessary for the proof.

Lemma 6.6.3 ([86, Lemma 5.2]). *For $n, q \in \mathbb{Z}_{>0}$ and q prime, let integer $m \geq 2n \log q$ and Gaussian parameter $s \geq \sqrt{n \log q \log n}$.*

1. *For $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, s}$, $\mathbf{x} = \mathbf{A}\mathbf{e} \bmod q$ is statistically close to $\mathcal{U}(\mathbb{Z}_q^n)$.*
2. *For fixed $\mathbf{x} \in \mathbb{Z}_q^n$, let $\mathbf{t} \in \mathbb{Z}^m$ be an arbitrary vector such that $\mathbf{A}\mathbf{t} = \mathbf{x} \bmod q$, then the distribution of $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, s}$ conditioned on $\mathbf{A}\mathbf{e} = \mathbf{x} \bmod q$ is exactly $D_{\Lambda_{\mathbf{x}}^\perp(\mathbf{A}), s}$.*

Proof. Since the scheme is a proper trapdoor mercurial commitment, we only have to consider the case where a commitment can be hard-opened to one message and soft-opened to another distinct message. Assume that an adversary, $\mathcal{A}$, can produce a commitment $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1)$ and two openings, a hard opening $\pi = (\mathbf{R}, \mathbf{r})$ to a message $\mathbf{m}$ and soft opening $\tau = \mathbf{r}'$ to a different message $\mathbf{m}' \neq \mathbf{m}$. Then, we have $\mathbf{B}_1 = \mathbf{A}_1 \mathbf{R}$ and

$$\mathbf{c} = \mathbf{A}_0 \mathbf{m} + [\mathbf{A}_1 \mid \mathbf{A}_1 \mathbf{R}] \mathbf{r} = \mathbf{A}_0 \mathbf{m}' + [\mathbf{A}_1 \mid \mathbf{A}_1 \mathbf{R}] \mathbf{r}'. \quad (6.1)$$

With such an adversary, we build an $\text{SIS}_{n, m, q, \beta}^\infty$ solver, $\mathcal{B}$, that takes an instance $\mathbf{A} \in \mathbb{Z}_q^{n \times \bar{m}}$ and finds a non-zero vector $\mathbf{v}^* \in \mathbb{Z}^{\bar{m}}$ in $\Lambda^\perp(\mathbf{A})$ such that $\|\mathbf{v}^*\|_\infty \leq \beta$. First, $\mathcal{B}$ samples $\mathbf{R}_0 \leftarrow D_{\mathbb{Z}, s}^{\bar{m} \times l}$ and $\mathbf{R}_1 \leftarrow D_{\mathbb{Z}, s}^{\bar{m} \times m}$ and computes $\mathbf{A}_0 = \mathbf{A} \mathbf{R}_0 \in \mathbb{Z}_q^{n \times l}$ and $\mathbf{A}_1 = \mathbf{A} \mathbf{R}_1 \in \mathbb{Z}_q^{n \times m}$. By Lemma 6.6.3, $\mathbf{A}_0$ and $\mathbf{A}_1$ are statistically close to $\mathcal{U}(\mathbb{Z}_q^{n \times l})$ and $\mathcal{U}(\mathbb{Z}_q^{n \times m})$ respectively. $\mathcal{A}$ is then given $\text{mpk} = (\mathbf{A}_0, \mathbf{A}_1)$ and since it can output two message-proof pairs $\mathbf{m}, \pi = (\mathbf{R}, \mathbf{r})$ and $\mathbf{m}', \tau = \mathbf{r}'$ that satisfy Equation (6.1) such that $\mathbf{m} \neq \mathbf{m}'$, we have

$$\mathbf{A}_0(\mathbf{m} - \mathbf{m}') = \mathbf{A}_1[\mathbf{I}_m \mid \mathbf{R}](\mathbf{r} - \mathbf{r}') \bmod q.$$

Therefore, we get a short vector in $\Lambda^\perp(\mathbf{A})$,

$$\mathbf{v}^* = \mathbf{R}_0(\mathbf{m} - \mathbf{m}') + \mathbf{R}_1[\mathbf{I}_m \mid \mathbf{R}](\mathbf{r} - \mathbf{r}') \in \mathbb{Z}^{\bar{m}} \quad (6.2)$$

with $\|\mathbf{v}^*\|_\infty \leq s \cdot (l\sqrt{\bar{m}} + \sqrt{s \cdot m\bar{m}(s^2 w^3 + 2m)})$. Furthermore, it is non-zero with overwhelming probability as $\mathbf{m} - \mathbf{m}' \in \{-1, 0, 1\}^l$ has at least one non-zero coordinate since the two vectors are distinct. The columns of $\mathbf{R}_0$, conditioned on $\mathbf{A}_0 = \mathbf{A} \mathbf{R}_0$, have minimum entropy of at least n bits by Lemma 6.6.3 and 2.3.7. This means that the columns of $\mathbf{R}_0$ contain a completely random term. Since the product $\mathbf{R}_0(\mathbf{m} - \mathbf{m}')$ is a linear combination of columns of $\mathbf{R}_0$, the right hand side of Equation (6.2) cancel over $\mathbb{Z}^{\bar{m}}$ with negligible probability. $\square$

Theorem 6.6.4. *The scheme satisfies the four equivocation properties, HH, HS, SS and SE equivocation, in the statistical sense.*

Proof. First, we show that the HH equivocation condition is satisfied, i.e., the distributions of fake commitments and their hard equivocations and hard commitments and their hard openings are statistically close. Let $(mpk = (\mathbf{A}_0, \mathbf{A}_1), msk = \mathbf{T}) \leftarrow \text{TMSetup}(1^\lambda)$ be a mercurial commitment key and trapdoor information.

Recall the real and ideal experiments of the HH equivocation condition, where the adversary chooses a message $\mathbf{m}$ and receives the following:

- real^{HH} : $\mathbf{m}, \mathbf{C} = (\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{A}_1\mathbf{R})$ and $\pi = (\mathbf{R}, \mathbf{r})$ such that $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$;
- ideal^{HH} : $\mathbf{m}, \mathbf{C} = (\mathbf{c} = \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{A}_1\mathbf{R})$ and $\pi = (\mathbf{R}, \mathbf{r})$ where $\mathbf{r} \leftarrow \text{SampleD}(\mathbf{T}_B, \mathbf{B}, \mathbf{c} - \mathbf{A}_0\mathbf{m}, s)$;

where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$ and $\mathbf{R} \leftarrow D_{\mathbb{Z}^{n \times (m+w)}, s}$ in both experiments and $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{B}\mathbf{r}$ in real^{HH} and $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{B}\mathbf{r}'$ in ideal^{HH} respectively.

As $\mathbf{A}_1$ is statistically close to $\mathcal{U}(\mathbb{Z}_q^{n \times m})$, Lemma 6.6.3 implies that the distribution of $(\mathbf{A}_1, \mathbf{B}_1)$, where $\mathbf{B}_1 = \mathbf{A}_1\mathbf{R}$ with $\mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}$, is statistically close to $\mathcal{U}(\mathbb{Z}_q^{n \times m}) \times \mathcal{U}(\mathbb{Z}_q^{n \times w})$. This means that $\mathbf{B}$ is distributed statistically close to $\mathcal{U}(\mathbb{Z}_q^{n \times (m+w)})$ for both hard and fake commitments. Considering the entire commitment, we see that the distribution of fake commitments $(\mathbf{c}, \mathbf{B}_1) = ([\mathbf{A}_1 \mid \mathbf{B}_1]\mathbf{r}, \mathbf{B}_1)$ — when $\mathbf{r}$ is drawn from the discrete Gaussian $D_{\mathbb{Z}^{m+w}, s}$ — is statistically close to $\mathcal{U}(\mathbb{Z}_q^n) \times \mathcal{U}(\mathbb{Z}_q^{n \times (m+w)})$. In particular, we cannot distinguish between ideal^{HH} and a modified ideal experiment, dubbed ideal_1^{HH} . In ideal_1^{HH} , instead of $\mathbf{c} = \mathbf{B}\mathbf{r}$ in ideal , we compute $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{B}\mathbf{r}$;

- ideal_1^{HH} : $\mathbf{m}, \mathbf{C} = (\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{A}_1\mathbf{R})$ and $\pi = (\mathbf{R}, \mathbf{r}')$ such that $\mathbf{r}' \leftarrow \text{SampleD}(\mathbf{T}_B, \mathbf{B}, \mathbf{c} - \mathbf{A}_0\mathbf{m}, s)$.

Furthermore, invoking Lemma 6.6.3 again, we find that, for any statistically uniform matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times (m+w)}$, the distribution of

$$(\mathbf{A}_0, \mathbf{A}\mathbf{r}, \mathbf{r}) \in \mathbb{Z}_q^{n \times (m+w)} \times \mathbb{Z}_q^n \times \mathbb{Z}^{m+w} \text{ where } \mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s} \quad (6.3)$$

is statistically close to that of

$$(\mathbf{A}_0, \mathbf{u}, \mathbf{r}) \in \mathbb{Z}_q^{n \times (m+w)} \times \mathbb{Z}_q^n \times \mathbb{Z}^{m+w} \text{ where } \mathbf{u} \leftarrow \mathcal{U}(\mathbb{Z}_q^n), \mathbf{r} \leftarrow D_{\Lambda_{\mathbf{u}}^\perp(A), s}. \quad (6.4)$$

Let ideal_2^{HH} be modified from ideal_1^{HH} so that the vector $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$ used to compute

$\mathbf{c} = \mathbf{A}\mathbf{m} + \mathbf{B}\mathbf{r}$ is revealed instead of using the extended trapdoor $\mathbf{T}_\mathbf{B}$ with `SampleD` to sample from a coset of the lattice $\Lambda^\perp(\mathbf{B})$.

- ideal_2^{HH} : $\mathbf{m}, \mathbf{C} = (\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{A}_1\mathbf{R})$ and $\pi = (\mathbf{R}, \mathbf{r})$ such that $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$.

Then, due to the statistical closeness of the distributions (6.3) and (6.4), ideal_2^{HH} is statistically indistinguishable from ideal_1^{HH} . To conclude, we observe that the experiment ideal_2^{HH} is simply the real experiment real^{HH} . Therefore, ideal^{HH} is statistically indistinguishable from real^{HH} since each is indistinguishable from the adjacent experiment ideal_1^{HH} .

Since the scheme is proper, HH equivocation implies HS equivocation and there is no need for to prove it separately. As for SS equivocation, the proof proceeds in almost the exact same manner and so we do not explicitly provide a proof. Lastly, we consider the SE equivocation condition which is introduced to construct ZK-EEDB.

First, the experiments in the SE case:

- real^{SE} : $\mathbf{C} = (\mathbf{c} = \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1\mathbf{R})$ and randomness $(\mathbf{R}, \mathbf{r})$;
- ideal^{SE} : $\mathbf{C} = (\mathbf{c} = \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{A}_1\mathbf{R})$ and randomness $(\mathbf{R}', \mathbf{r})$ where $\mathbf{R}' = [\mathbf{r}'_1 \mid \dots \mid \mathbf{r}'_w]$ and $\mathbf{r}'_i \leftarrow \text{SampleD}(\mathbf{T}, \mathbf{A}_1, \mathbf{g}_i - \mathbf{A}_1\mathbf{r}_i, s)$ for $i \in [w]$;

where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$, $\mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}$ and $\mathbf{r} \leftarrow D_{\mathbb{Z}^{m+w}, s}$ in both experiments.

The experiments are statistically indistinguishable from the observation that both distributions

$$\mathcal{D}_{\text{fake}} = \{\mathbf{A}_1\mathbf{R} \mid \mathbf{R} \leftarrow D_{\mathbb{Z}^{m \times w}, s}\} \text{ and } \mathcal{D}_{\text{soft}} = \{\mathbf{G} - \mathbf{A}_1\mathbf{R}'' \mid \mathbf{R}'' \leftarrow D_{\mathbb{Z}^{m \times w}, s}\}$$

are statistically close to $\mathcal{U}(\mathbb{Z}_q^{n \times w})$. With this, when ideal^{SE} is modified to ideal_1^{SE} , which samples $\mathbf{B}_1$ from $\mathcal{D}_{\text{soft}}$ instead of $\mathcal{D}_{\text{fake}}$, the adversary's view remains the same in a statistical sense.

- ideal_1^{SE} : $\mathbf{C} = (\mathbf{c} = \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1\mathbf{R}'')$ and randomness $(\mathbf{R}', \mathbf{r})$ where $\mathbf{R}' = [\mathbf{r}'_1 \mid \dots \mid \mathbf{r}'_w]$ and $\mathbf{r}'_i \leftarrow \text{SampleD}(\mathbf{T}, \mathbf{A}_1, \mathbf{g}_i - \mathbf{A}_1\mathbf{r}_i'', s)$ for $i \in [w]$;

Furthermore, since the distributions (6.3) and (6.4) are statistically close, a further modification, ideal_2^{SE} , where the explanation sampled with the trapdoor information $\mathbf{T}$ is replaced with the Gaussian matrix $\mathbf{R}''$ that is used to compute $\mathbf{B}_1$, remains indistinguishable from ideal_1^{SE} .

— ideal_2^{SE} : $\mathbf{C} = (\mathbf{c} = \mathbf{B}\mathbf{r}, \mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1\mathbf{R}'')$ and randomness $(\mathbf{R}'', \mathbf{r})$ where $\mathbf{R}'' \leftarrow D_{\mathbb{Z}^m \times w}$

ideal_2^{SE} is identical to real^{SE} and so SE equivocation is satisfied. $\square$

6.6.3 Instantiating ZK-EEDB

The trapdoor mercurial commitment scheme from Section 6.6.1 has commitments of the form $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$, which are $k = n(w + 1)\lceil \log q \rceil$ bits long. In the instantiation of ZK-EEDB using this scheme, the message space $\{0, 1\}^l$ of the TMC will differ depending on the position in the Merkle tree. A total of four different message lengths $\{l_1, l_2, l_3, l_4\}$ are used in the system.

1. $l_1 = \ell$ at the leaves of the key Merkle tree since the values of the database have size ℓ bits;
2. $l_2 = 2k$ at internal nodes of all the Merkle trees as two mercurial commitments are inputs;
3. $l_3 = k$ at the leaves of the value Merkle tree because we commit to the digest of D_y^{-1} , $\text{com}_{D_y^{-1}}$ which is a single mercurial commitment;
4. $l_4 = 1$ as we commit to a single bit at the leaves of the Merkle trees used to compute a digest of $D_y^{-1} = \{(x, 1) \mid x \in D^{-1}(y)\}$.

Although there are several message lengths, we can handle them by simply adjusting the number of columns in the matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times l}$, setting $l = \max\{l_1, l_2, l_3, l_4\}$. For each $i \in [4]$, we use the submatrix $\mathbf{A}_{0,i} \in \mathbb{Z}_q^{n \times l_i}$ which is made up of the first l_i columns of $\mathbf{A}_0$.

Efficiency of the Instantiation. To describe a path in the Merkle tree and its associated commitments, we require $\zeta = \mathcal{O}(l \cdot k)$ bits. With $l = \mathcal{O}(\lambda)$, $\zeta = \mathcal{O}(\lambda^3)$ which is quite large. For more efficiency, the TMC can be transformed from using SIS to RSIS and rely on ideal lattices instead. It was shown by Micciancio and Peikert [157, Section 4.3] and Ducas and Micciancio [69, Section 2 and Appendix] a couple of years later, that the standard lattice-based cryptographic tools used can be adapted to the ring setting. In this case, the parameter w is reduced to $w = \mathcal{O}(\log q)$ instead of $\mathcal{O}(n \log q)$ reducing the commitment size and ζ by a factor of $\mathcal{O}(\lambda)$ to a more reasonable $\mathcal{O}(\lambda^2)$.

Figure 6.2: Merkle trees of ZK-EEDB

Chapter 7

Zero-Knowledge Password Policy Check

7.1 Introduction

In this chapter, we consider the privacy of passwords in user authentication. One of the most ubiquitous mechanisms for authentication for online services, passwords must be chosen such that they have sufficient entropy and are hard to guess for security. Password policies were introduced to help users generate passwords that are harder to guess; Ur *et al.* [202] found that password policies were effective and users are more likely to choose weaker, easy to guess passwords in their absence. Florêncio and Herley [75] studied the types of password policies that were used in over 70 websites and found that most policies require passwords with characters from at least one of four character sets, upper and lower case letters, digits and symbols as well as a minimum password length. Therefore, we focus on password policies with a minimum password length, various sets of permitted characters and possibly some constraints on the variety of characters used.

On the other hand, strong passwords and good policies cannot prevent servers from being compromised and leaking this sensitive data if they are not properly stored. As evident from hacks on LinkedIn [78] and Yahoo [175] and the website “Have I Been Pwned?” [108], improperly managed passwords on servers can cause serious damage to users. Baumann *et al.* [18] discovered that a lot of popular websites do not follow good practices when it comes to passwords, storing them in plaintext form. This puts the trustworthiness of servers in question and should endeavour to move sensitive information away from them. Thus, it becomes desirable — arguably even necessary — to design protocols that do not need secret user information to be stored at external and untrusted

servers.

However, these protocols are at odds with password policies: without password policies, the security of passwords are questionable but enforcing password policies require knowledge of the password to be given to the untrusted server. To reconcile the differences, a protocol, called blind registration, where users register a password with a server and prove policy compliance of that password without revealing any information about them is required. Kiefer and Manulis [124] showed that this can be achieved with a new primitive called zero knowledge password policy check (ZKPPC). With ZKPPC, the server stores some randomized information for password verification which does not reveal anything about the password, thus preventing server compromises from affecting user passwords. Users can use the ZKPPC protocol to prove that their chosen password satisfies the server's password policy without the server gaining any knowledge on the password. Combined with a verifier-based password-based authenticated key exchange (VPAKE) protocol, a complete system for privacy-preserving password-based registration, authentication and key exchange can be achieved.

First proposed by Bellovin and Merritt [20] and spawning more subsequent work [19, 79, 118, 67, 23], password-based authenticated key exchange (PAKE) is a protocol that enables users to authenticate their identity and exchange keys in a single run. However, the primitive allows servers to hold users' passwords and so they have to be trusted to handle them securely. VPAKE [21, 19, 85, 26] takes into account the potential for information leakage by the server and has the server store a randomized verifier instead of the actual password. Verifiers are used to check that users gave the correct password in the protocol and are usually generated by hashing the password with a salt, making attempts to extract the password from it very difficult.

A generic construction of ZKPPC was proposed by Kiefer and Manulis [124] using homomorphic commitments and set membership proofs. They also instantiated ZKPPC with Pedersen commitments [169], with security reduced to the discrete logarithm problem. Other than [124], there is a lack of concrete ZKPPC schemes. Since Shor's algorithm [190] can solve the problem in quantum polynomial time, a quantum-resistant ZKPPC scheme is an important and interesting open problem.

We address this by constructing a ZKPPC protocol from lattices, which is one of the main candidates for quantum-resistant cryptography. This is accomplished in two parts: first, a randomized password hashing scheme based on the hardness of the SIS problem is constructed to compute verifiers. Then, a SIS-based statistical zero-knowledge argument of knowledge (ZKAoK) is designed. The client would run the protocol with the server to convince it that the secret password, committed in a given verifier, satisfies the server's

designated password policy. The constructions deviate from the framework established by Kiefer and Manulis [124] because their use of large integers to encode passwords are incompatible with lattice-based techniques which use small norm objects for security and efficiency.

Organization of the Chapter. In Section 7.2, we review the literature on zero-knowledge password policy checks. Then, in Section 7.3, we introduce important preliminaries for the chapter: the model of passwords and password policies as well as formal definitions of randomized password hashing and ZKPPC. Finally, we describe a construction of the components of ZKPPC from lattices in Section 7.4.

7.2 Previous Work

There is only one concrete instantiation of ZKPPC, which was given by Kiefer and Manulis [124], using Pedersen commitments and a randomized password hashing scheme constructed in the same work. One commitment is generated per character in the password and set membership proofs with these commitments are used to prove compliance of the password to a password policy. The verifier given to the server is a commitment to sum of the committed characters obtained by using the homomorphic property of the commitment scheme. This property also ties the set membership proofs to the verifier.

To get better efficiency for secure password registration than VPAKE [125] and two server PAKE [126], Kiefer and Manulis introduced a new family of protocols called blind password registration (BPR). Adapting the techniques in [124], they applied an efficient shuffling proof from Furukawa [77] to reduce the number of set membership proofs to a constant number instead of being dependent on the maximum allowed password length. However, BPR trades security to achieve this improvement; the model only considers honest but curious participants which is a lot weaker than the ZKPPC model where all parties might be malicious. Malicious users might attempt to register bad, non-compliant passwords while malicious servers might try to glean information on registered passwords during the execution of the ZKPPC protocol. As with ZKPPC, BPR does not yet have a quantum-resistant instantiation.

With symmetric key primitives, secure set-based policy checking (SPC) was proposed to verify password policy compliance. Policies are represented with monotone access structures by translating them to sets and then mapped to linear secret sharing schemes (LSSS). This way, policy compliance is verified by determining if some set is in the access structure; in terms of the LSSS, the set of shares representing the password

can be used to reconstruct the shared secret. To transform this into a privacy-preserving protocol, oblivious bloom intersection (OBI) from Dong, Chen and Wen [68] is used. The server initializes an LSSS which corresponds to its password policy and password registration is done by running OBI, with the user obtaining a set of shares based on his desired password. Knowledge of the secret, implying a valid password for registration, is then proved by hashing the secret with the transcript of the OBI execution and identities of the two parties — which ensures that the password from the OBI protocol is tied to the user executing it. In the proposed SPC protocol, soundness of the protocol, in the event that the client is malicious, is proved with the one-more-RSA assumption. Therefore, should the adversary have access to quantum computers, the SPC protocol becomes insecure.

7.3 Preliminaries

In this section, we recall the models of password strings and password policies from Kiefer and Manulis [124] as well as the definitions of randomised password hashing schemes and zero-knowledge password policy checks.

7.3.1 Password Strings and Password Policies

Password Strings. In our setting, password strings pw are comprised of characters from the set of 94 printable characters, Σ_{all} , in the ASCII alphabet, Σ_{ASCII} , divided into four disjoint subsets,

Σ_D : The set of 10 digits, $\{0, 1, \dots, 9\}$;

Σ_U : The set of 26 upper case letters, $\{A, B, \dots, Z\}$;

Σ_L : The set of 26 lower case letters, $\{a, b, \dots, z\}$;

Σ_S : The set of 32 symbols, $\{ !"#$\%&'()*+,-./ :;<=>?@ [\]_`{|} ~ \}$.

A general dictionary containing all strings in Σ_{all}^* is denoted with Dict . A password string $pw = (c_1, c_2, \dots, c_k) \in \Sigma_{\text{all}}^k \subset \text{Dict}$, with length k , is an ordered multi-set of k characters from Σ_{all} .

Password Policies. A password policy $f = ((k_D, k_U, k_L, k_S), n_{\min}, n_{\max})$ is a six-tuple, where $k_D, k_U, k_L, k_S \in \mathbb{Z}_{\geq 0}$ denote the minimum number of digits, upper and lower case letters and symbols a password must contain. The other two positive integers with $n_{\min} \leq n_{\max}$ indicate the minimum and maximum password lengths respectively. We say

that $f(pw) = \text{true}$ if and only if the password string pw satisfied the password policy f . For example,

1. The policy $f = ((1, 1, 1, 0), 8, 16)$ is satisfied only by password strings that have at least one digit, upper and lower case letter and between 8 and 16 characters inclusive;
2. Another policy $f = ((1, 0, 0, 2), 9, 12)$ requires valid password strings to contain at least 1 digit and 2 symbols and be between 9 and 12 characters long inclusive.

In practice, a maximum password length $n_{\max}$ may not be specified but we can just fix a reasonable upper bound, such as 32 or 64.

7.3.2 Randomised Password Hashing and Zero-Knowledge Password Policy Check

Here, we present the notions of randomised password hashing and zero-knowledge password policy check from [124, 123].

Randomised Password Hashing. The scheme is used to compute password verification information that can be applied in other protocols such as ZKPPC and VPAKE. To prevent offline dictionary attacks, the password hashing algorithm is randomised with a pre-hash and hash salt. Formally, a randomised password hashing scheme $\mathcal{H}$ is a five-tuple $\mathcal{H} = (\text{Setup}, \text{PreSalt}, \text{PreHash}, \text{Salt}, \text{Hash})$ as follows

- **Setup**(1^λ): Take a security parameter λ as input, then generate and output public parameters pp , with descriptions of the pre-hash and hash salt spaces $\mathbb{S}_P$ and $\mathbb{S}_H$.
- **PreSalt**(pp): Take the public parameters pp as input and output a random pre-hash salt $s_P \in \mathbb{S}_P$.
- **PreHash**(pp, pw, s_P): Take the public parameters pp , password string pw and pre-hash salt s_P as inputs and output a pre-hash value P .
- **Salt**(pp): Take the public parameters pp as inputs and output a random hash salt $s_H \in \mathbb{S}_H$.
- **Hash**(pp, P, s_P, s_H): Take the public parameters pp , pre-hash value P , pre-hash salt s_P and hash salt s_H as inputs and output a hash value $\mathbf{h}$.

For a randomised password hashing scheme $\mathcal{H}$ to be secure, five properties must be satisfied, namely *pre-image resistance* (or *tight one-wayness* in [26]), *second pre-image resistance*, *pre-hash entropy preservation*, *hash entropy preservation* and *password hiding*.

- Pre-Image Resistance: Let $pp \leftarrow \text{Setup}(1^\lambda)$ and Dict be a dictionary with minimum entropy β . Denote with $(H_i, s_{H_i}) \leftarrow \text{Hash}(\cdot)$ the function which outputs hash salt $s_{H_i} \leftarrow \text{Salt}(pp)$ and hash value $H_i \leftarrow \text{Hash}(pp, P_i, s_{P_i}, s_{H_i})$, where $P_i \leftarrow \text{PreHash}(pp, pw_i, s_{P_i})$ for pre-hash salt $s_{P_i} \leftarrow \text{PreSalt}(pp)$ and password $pw_i \leftarrow \mathcal{U}(\text{Dict})$. P_i is stored by $\text{Hash}(\cdot)$ and let $\text{Verify}(i, P)$ be a function such that $\text{Verify}(i, P) = 1$ if $P = P_i$.

For all PPT adversaries $\mathcal{A}$, running in time at most t , we have

$$\Pr[(i, P) \leftarrow \mathcal{A}^{\text{Hash}(\cdot)}; \text{Verify}(i, P) = 1] \leq \frac{\alpha \cdot t}{2^\beta \cdot t_{\text{PreHash}}} + \text{negl}(\lambda),$$

for some small α and t_{PreHash} which is the running time of the PreHash algorithm.

- Second Pre-Image Resistance: For all PPT adversaries $\mathcal{A}$ and $P' \leftarrow \mathcal{A}(pp, P, s_H)$, we have

$$\Pr[(P' \neq P) \wedge (\text{Hash}(pp, P, s_H) = \text{Hash}(pp, P', s_H))] \leq \text{negl}(\lambda),$$

with $pp \leftarrow \text{Setup}(1^\lambda)$, $s_P \leftarrow \text{PreSalt}(pp)$, $P \leftarrow \text{PreHash}(pp, pw, s_P)$ and $s_H \leftarrow \text{Salt}(pp)$ for any $pw \in \text{Dict}$.

- Pre-Hash Entropy Preservation: For all polynomial-time sampleable dictionaries with minimum entropy β , Dict and any PPT adversary $\mathcal{A}$ and $H, s_P, s_H \leftarrow \mathcal{A}(pp)$, we have

$$\Pr[(s_P \in \mathbb{S}_p) \wedge (P = \text{PreHash}(pp, pw, s_P))] \leq 2^{-\beta} + \text{negl}(\lambda),$$

with $pp \leftarrow \text{Setup}(1^\lambda)$ and $pw \leftarrow \mathcal{U}(\text{Dict})$.

- Entropy Preservation: For all polynomial-time sampleable dictionaries with minimum entropy β , Dict and $(H, s_P, s_H) \leftarrow \mathcal{A}(pp)$, we have

$$\Pr[(s_P \in \mathbb{S}_P) \wedge (s_H \in \mathbb{S}_H) \wedge (H = \text{Hash}(pp, pw, s_P, s_H))] \leq 2^{-\beta} + \text{negl}(\lambda),$$

with $pp \leftarrow \text{Setup}(1^\lambda)$ and $pw \leftarrow \mathcal{U}(\text{Dict})$.

- Password Hiding: For $pp \leftarrow \text{Setup}(1^\lambda)$ and all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1(pp)$ outputs two passwords pw_0 and pw_1 of equal length and $\mathcal{A}_2(H)$ outputs a bit b' for $H \leftarrow \text{Hash}(pp, P, s_P, s_H)$ with $s_H \leftarrow \text{Salt}(pp)$, $s_P \leftarrow \text{PreSalt}(pp)$ and $P \leftarrow \text{PreHash}(pp, pw_b, s_P)$ for some random bit $b \leftarrow \mathcal{U}(\{0, 1\})$, we have

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

Zero-Knowledge Password Policy Check. Let $\mathcal{H} = (\text{Setup}, \text{PreSalt}, \text{PreHash}, \text{Salt}, \text{Hash})$ be a randomised password hashing scheme. Password policy check (PPC) is an two-party interactive protocol between a server and client with common inputs $f = ((k_d, k_U, k_L, k_S), n_{\min}, n_{\max})$ and $pp \leftarrow \mathcal{H}.\text{Setup}(1^\lambda)$, being the password policy and public parameters of $\mathcal{H}$ respectively. At the end of the protocol's execution, the server accepts a hash value $\mathbf{h}$ of some password pw that the client chose if and only if $f(pw) = \text{true}$.

In other words, a PPC protocol is an argument of knowledge of the password pw and randomness (salts) $s_P \leftarrow \text{PreSalt}(pp)$ and $s_H \leftarrow \text{Salt}(pp)$ used in the randomised password hash value $\mathbf{h} \leftarrow \text{Hash}(pp, P, s_P, s_H)$, with $P \leftarrow \text{PreHash}(pp, pw, s_P)$. The goal is to argue this knowledge without revealing the password, the protocol has to be zero-knowledge as well. More formally, a zero-knowledge PPC protocol is an interactive protocol between a client (prover) and server (verifier), where given $(pp, f, \mathbf{h})$, the prover convinces the verifier in zero-knowledge that some password pw and randomness (s_P, s_H) is known to him such that

$$f(pw) = \text{true} \quad \text{and} \quad \text{Hash}(pp, P, s_P, s_H) = \mathbf{h},$$

where $P \leftarrow \text{PreHash}(pp, pw, s_P)$.

7.4 The Construction

In this section, we describe our construction of a lattice-based randomised password hashing scheme and zero-knowledge argument of knowledge for password policy compliance. First, we introduce the definitions of several important sets and permutation used in the section. Following that, we describe the lattice-based randomised password hashing scheme. Finally, we present a zero-knowledge argument of knowledge for proving that the hashed password satisfies the password policy specified by a server.

The first obstacle is to find a password encoding method that allows us to work with lattice-based techniques for constructing hashing schemes and zero-knowledge argu-

ments. Kiefer and Manulis [124], in the discrete logarithm setting, map password strings to integers and then embed them in a group of large order. In the lattice setting, such large norm objects result in much lower security and efficiency, necessitating a different approach. The natural objects, for lattice-based schemes, are binary vectors and so we introduce the character encoding which we will use in this work.

Character Encoding. Let $\text{bin}(x)$ be the function mapping non-negative integers $x \in [0, X]$ to their binary representation of length $\lceil \log X \rceil$. For any character c encoded in ASCII, let $\text{ASCII}(c) \in [0, 255]$ denote its ASCII code. Then, for a character $c \in \Sigma_{\text{ASCII}}$ and password string $p = (c_1, \dots, c_k) \in \Sigma_{\text{all}}^k$, we define the character encoding function enc and its extension to password strings encode ,

$$\begin{aligned} \text{enc}(c) &= \text{bin}(\text{ASCII}(c)) \in \{0, 1\}^8; \\ \text{encode}(pw) &= (\text{enc}(c_1), \dots, \text{enc}(c_k)) \in \{0, 1\}^{8k}. \end{aligned}$$

7.4.1 Some Sets and Permutations

For $\mathbf{m}, \mathbf{n} \in \mathbb{Z}_{>0}$, let $\mathcal{S}_a$ denote the set of permutations over a elements and define

- $\mathbf{B}_{\mathbf{m}}^2 = \{\mathbf{x} \mid \mathbf{x} \in \{0, 1\}^{2\mathbf{m}} \text{ such that } \text{hw}(\mathbf{x}) = \mathbf{m}\}$, the set of vectors in $\{0, 1\}^{2\mathbf{m}}$ with Hamming weight exactly $\mathbf{m}$. Note that vectors in $\mathbf{B}_{\mathbf{m}}^2$ have some important properties,

$$\begin{cases} \mathbf{x} \in \mathbf{B}_{\mathbf{m}}^2 & \Leftrightarrow & (\mathbf{x}) \in \mathbf{B}_{\mathbf{m}}^2; \\ \mathbf{x} \in \mathbf{B}_{\mathbf{m}}^2, \psi \leftarrow \mathcal{U}(\mathcal{S}_{2\mathbf{m}}) & \Rightarrow & \psi(\mathbf{x}) \text{ is uniform in } \mathbf{B}_{\mathbf{m}}^2. \end{cases} \quad (7.1)$$

- $T_{\psi, \mathbf{n}}$: For $\psi \in \mathcal{S}_{\mathbf{m}}$ and $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{\mathbf{m}}) \in \mathbb{Z}^{\mathbf{m}\mathbf{n}}$ where each $\mathbf{v}_i \in \mathbb{Z}^{\mathbf{n}}$, it is the permutation that sends $\mathbf{v}$ to

$$T_{\psi, \mathbf{n}}(\mathbf{v}) = (\mathbf{v}_{\psi(1)}, \mathbf{v}_{\psi(2)}, \dots, \mathbf{v}_{\psi(\mathbf{m})}).$$

For a simpler exposition, we introduce some notation and define some sets for working with the password alphabet $\Sigma_{\text{all}} = \Sigma_D \sqcup \Sigma_U \sqcup \Sigma_L \sqcup \Sigma_S$ and password policy $f = ((k_D, k_U, k_L, k_S), n_{\min}, n_{\max})$.

- For $\alpha \in \{D, U, L, S, \text{all}\}$, let

$$\diamond \eta_{\alpha} = |\Sigma_{\alpha}|;$$

- ◇ $\text{Enc}_\alpha = \{\text{enc}(w) \mid w \in \Sigma_\alpha\};$
- ◇ Set_α be the set of vectors $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_{\eta_\alpha}) \in \{0, 1\}^{8\eta_\alpha}$, such that the η_α blocks $\mathbf{v}_1, \dots, \mathbf{v}_{\eta_\alpha} \in \mathbb{Z}^8$ together comprise the binary encodings of every character in Σ_α . In other words, $\{\mathbf{v}_1, \dots, \mathbf{v}_{\eta_\alpha}\} = \{\text{enc}(w) \mid w \in \Sigma_\alpha\}$.
- $\text{Set}_{n_{\max}}$ is the set of vectors $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_{n_{\max}}) \in \{0, 1\}^{n_{\max} \lceil \log n_{\max} \rceil}$, such that the $n_{\max}$ blocks $\mathbf{v}_1, \dots, \mathbf{v}_{n_{\max}} \in \mathbb{Z}^{\lceil \log n_{\max} \rceil}$ together comprise the binary representations of the integers in $[0, n_{\max})$ of length $\lceil \log n_{\max} \rceil$. Basically, $\{\mathbf{v}_1, \dots, \mathbf{v}_{n_{\max}}\} = \{\text{bin}(0), \dots, \text{bin}(n_{\max} - 1)\}$.

Similar to $\mathcal{B}_m^2$, the sets defined above have important interactions with certain classes of permutations.

- For all $\alpha \in \{D, U, L, S, \text{all}\}$, vectors $\mathbf{x} \in \mathbb{Z}^{8\eta_\alpha}$ and permutations $\psi \in \mathcal{S}_{\eta_\alpha}$,

$$\begin{cases} \mathbf{x} \in \text{Set}_\alpha & \Leftrightarrow T_{\psi, 8}(\mathbf{x}) \in \text{Set}_\alpha; \\ \mathbf{x} \in \text{Set}_\alpha, \psi \leftarrow \mathcal{U}(\mathcal{S}_{\eta_\alpha}) & \Rightarrow T_{\psi, 8}(\mathbf{x}) \text{ is uniform in } \text{Set}_\alpha. \end{cases} \quad (7.2)$$

- For all vectors $\mathbf{x} \in \mathbb{Z}^{n_{\max} \lceil \log n_{\max} \rceil}$ and permutations $\psi \in \mathcal{S}_{n_{\max}}$,

$$\begin{cases} \mathbf{x} \in \text{Set}_{n_{\max} \lceil \log n_{\max} \rceil} & \Leftrightarrow T_{\psi, \lceil \log n_{\max} \rceil}(\mathbf{x}) \in \text{Set}_{n_{\max}}; \\ \mathbf{x} \in \text{Set}_{n_{\max}}, \psi \leftarrow \mathcal{U}(\mathcal{S}_{n_{\max}}) & \Rightarrow T_{\psi, \lceil \log n_{\max} \rceil}(\mathbf{x}) \text{ is uniform in } \text{Set}_{n_{\max}}. \end{cases} \quad (7.3)$$

7.4.2 Randomized Password Hashing Scheme from Lattices

Now we present the first component of our lattice-based ZKPPC scheme, a randomized password hashing scheme, $\mathcal{LH}$. The scheme maps password characters to binary vectors, which we divide into blocks, each derived from one character. Then, we use a random permutation χ to re-arrange the blocks and finally use the KTX commitment [119] described in Section 2.6.3 to compute the password hash from the pre-hash value and permutation χ . Formally,

- $pp \leftarrow \mathcal{LH}.\text{Setup}(1^\lambda)$: Choose parameters $n = \mathcal{O}(\lambda)$, prime $q = \tilde{\mathcal{O}}(n)$, $m = 2n \lceil \log q \rceil$ and sample matrices $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n_{\max}(\lceil \log n_{\max} \rceil + 8)})$ and $\mathbf{B} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times m})$. The pre-hash salt space is $\mathbb{S}_p = \mathcal{S}_{n_{\max}}$ and hash salt space is $\mathbb{S}_H = \{0, 1\}^m$. Output $pp = (n, q, m, \mathbb{S}_p, \mathbb{S}_H, \mathbf{A}, \mathbf{B})$.
- $s_P \leftarrow \mathcal{LH}.\text{PreSalt}(pp)$: Sample a random permutation $\chi \leftarrow \mathcal{U}(\mathcal{S}_{n_{\max}})$ and output $s_P = \chi$.

- P $\mathcal{LH}.\text{PreHash}(pp, pw, s_P)$: Let the pre-hash salt $s_P = \chi \in \mathcal{S}_{n_{\max}}$ and the length of the password string $t = |pw| \in [n_{\min}, n_{\max}]$.
 1. Encode the password string into $\mathbf{v} = \text{encode}(pw) \in \{0, 1\}^{8t}$, consisting of t blocks of 8 bits each.
 2. Insert $n_{\max} - t$ blocks of the form $\text{enc}(g)$, where $g \in \Sigma_{\text{ASCII}} \setminus \Sigma_{\text{all}}$ i.e., ASCII codes of non-printable characters, into $\mathbf{v}$ to get $\mathbf{e} \in \{0, 1\}^{8n_{\max}}$, masking the password length.
 3. Apply the block-wise permutation $T_{\chi, 8}$, obtaining $\mathbf{e}' = T_{\chi, 8}(\mathbf{e})$.
 4. Output $P = \mathbf{e}'$.
- s_H $\text{Salt}(pp)$: Sample a random binary vector $\mathbf{r} \leftarrow \mathcal{U}(\{0, 1\}^m)$ and output $s_H = \mathbf{r}$.
- H $\mathcal{LH}.\text{Hash}(pp, P, s_P, s_H)$: Let the pre-hash value $P = \mathbf{e}' \in \{0, 1\}^{8n_{\max}}$, pre-hash salt $s_P = \chi \in \mathcal{S}_{n_{\max}}$ and hash salt $s_H = \mathbf{r} \in \{0, 1\}^m$.
 1. Let $\chi = [\chi(1), \dots, \chi(n_{\max})]$, where $\chi(i)$ is the result of applying χ on $i \in [n_{\max}]$. Then, convert χ into a binary vector $\mathbf{e}_0$ where

$$\mathbf{e}_0 = (\text{bin}(\chi(1) - 1), \dots, \text{bin}(\chi(n_{\max}) - 1)) \in \{0, 1\}^{n_{\max} \lceil \log n_{\max} \rceil}.$$
 2. Let $\mathbf{x} = (\mathbf{e}_0, \mathbf{e}')$ and compute $\mathbf{h} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{r} \in \mathbb{Z}_q^n$.
 3. Output $H = \mathbf{h}$.

The security of $\mathcal{LH}$, as defined by the five properties in Section 7.3.2, is captured in the following theorem.

Theorem 7.4.1. *Assuming the hardness of the $\text{SIS}_{n, n_{\max}(\lceil \log n_{\max} \rceil + 8), q, 1}^\infty$ problem, the randomised password hashing scheme $\mathcal{LH}$ satisfies the five security requirements of pre-image resistance, second pre-image resistance, pre-hash entropy preservation, entropy preservation and password hiding.*

Proof. First, we show that the scheme has the pre-hash entropy and entropy preservation properties. By construction, given the pre-hash salt $s_P = \chi$, the password string can be recovered from the pre-hash procedure by applying the inverse permutation $T_{\chi^{-1}, 8}$ induced by χ^{-1} to the pre-hash value $P = \mathbf{e}'$ and then discarding all blocks corresponding to non-printable characters in the string and converting the binary blocks back to ASCII characters. Therefore, if s_P is hidden, the minimum entropy of the pre-hash value P is

larger than that of the pre-hashed password pw . This means that the two properties are fulfilled.

Next, the password hiding property of $\mathcal{LH}$ is satisfied since the hash value, which is $\mathbf{h} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{r} \bmod q$, is a proper KTX commitment of the message $\mathbf{x}$ with randomness $\mathbf{r}$. Hence, the statistical hiding property of the KTX scheme implies the password hiding property of $\mathcal{LH}$. Moreover, if distinct pre-hash values P, P' can be produced such that both yield the same hash value $\mathbf{h}$, then P, P' can be used to violate the computational binding property of the KTX scheme. Therefore, the second pre-image resistance property is satisfied assuming the hardness of the $\text{SIS}_{n, n_{\max}(\lceil \log n_{\max} \rceil + 8), q, 1}^\infty$ problem.

Lastly, except with negligible probability where a solution to the SIS problem with the matrix $\mathbf{A}$ is accidentally found, over the randomness of $\mathbf{A}$, pw and s_P , the vector $\mathbf{A}\mathbf{x}$ accepts at least 2^β values in $\mathbb{Z}_q^n$, where β is the minimum entropy of the dictionary Dict that the password is chosen from. Therefore, even when given $\mathbf{A}\mathbf{x} = h - \mathbf{B} \cdot s_H \bmod q$, to find $P = \mathbf{e}'$, at least 2^β calls of PreHash is necessary to find $\mathbf{x}$, implying that $\mathcal{LH}$ satisfies the pre-image resistance property. $\square$

7.4.3 Zero-Knowledge Password Policy Check Protocol

The second component of ZKPPC is a protocol to prove password policy compliance in zero-knowledge, where the password string is hashed via a randomised password hashing scheme. Here, we design such a protocol for password strings hashed with the randomised password hashing scheme $\mathcal{LH}$ from Section 7.4.2.

Our construction exploits the sets and permutations defined in Section 7.4.1 to reduce the statement that a password string pw satisfies a given password policy f to an instance of the abstract relation R_{abs} in Section 2.6.4. The Stern-like protocol for R_{abs} in Figure 2.2 is then used to obtain a ZKAoK for password policy compliance.

Proving Set Membership in Lattice Cryptography. To obtain a ZKAoK for password policy compliance, we prove that a linear relation with the form

$$\sum_i \mathbf{M}_i \cdot \mathbf{s}_i = \mathbf{h} \bmod q,$$

such that

- the matrices $\mathbf{M}_i$ are public;
- the vectors $\mathbf{s}_i$ are secret, binary and are elements of small sets, such as Enc_α for $\alpha \in \{D, U, L, S, \text{all}\}$.

Figure 7.1: From Set Membership to Vector Well-Formedness

Therefore, we have to incorporate sub-protocols for proving membership of these secret $\mathbf{s}_i$'s to some sets.

For lattice cryptography, Libert *et al.* [138] exploited Stern-like protocols and a Merkle hash tree to construct a set membership ZKAoK with complexity logarithmic in the size of the set. Although it is asymptotically efficient, it does not perform well when the size of the set involved is some small constant. Therefore, we use a different method, with linear complexity but simpler to explain and more efficient in practice for our purposes.

Let $\mathbf{s}_i$ be an n -dimensional vector that belongs to some set $\{\mathbf{v}_1, \dots, \mathbf{v}_m\}$ of size m and suppose we want to prove that $\mathbf{s}_i$ belongs in this set. To achieve that, we append $m - 1$ blocks to $\mathbf{s}_i$ to form a new vector $\mathbf{s}_i^* \in \mathbb{Z}^{nm}$ such that the m blocks of $\mathbf{s}_i^*$ make up the entire set $\{\mathbf{v}_1, \dots, \mathbf{v}_m\}$. Similarly, we append $n(m - 1)$ columns of the n -dimensional zero vector to the public matrix $\mathbf{M}_i$, obtaining $\mathbf{M}_i^*$ with $\mathbf{M}_i^* \mathbf{s}_i^* = \mathbf{M}_i \mathbf{s}_i$ and preserving the linear relations involved.

With this, proving that $\mathbf{s}_i$ belongs in some set of vectors is reduced to proving that $\mathbf{s}_i^*$ is well-formed. The well-formedness can be proven with random permutations of the blocks of the vector, fitting perfectly within the techniques of Stern's protocol. Figure 7.1 gives a pictorial representation of the extension.

From Password Policy Compliance to the R_{abs} Form. The protocol allows a client, who is the prover $\mathcal{P}$ in this setting, to convince the server (verifier $\mathcal{V}$) in zero-knowledge that $\mathcal{P}$ knows a password string pw that hashes to a given hash value $\mathbf{h}$ with randomness $\chi, \mathbf{r}$ and satisfies the password policy $f = ((k_D, k_U, k_L, k_S), n_{\min}, n_{\max})$. We consider the case where each $k_\alpha \geq 1$ for $\alpha \in \{D, U, L, S\}$ but the procedure can be easily modified to accommodate any number of $k_\alpha = 0$. Recall that the policy specifies that valid password strings are between $n_{\min}$ and $n_{\max}$ characters long with at least k_D digits, k_U upper and k_L lower case letters and k_S symbols.

Let $k_{\text{all}} = n_{\min} - (k_D + k_U + k_L + k_S)$. The common input of both parties for the protocol comprises of the matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times n_{\max}(\lceil \log n_{\max} \rceil + 8)}$, $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, hash value $\mathbf{h} \in \mathbb{Z}_q^n$ and some additional information

$$\Delta = \{\delta_{D,i}\}_{i=1}^{k_D} \cup \{\delta_{U,i}\}_{i=1}^{k_U}, \{\delta_{L,i}\}_{i=1}^{k_L} \cup \{\delta_{S,i}\}_{i=1}^{k_S} \cup \{\delta_{\text{all},i}\}_{i=1}^{k_{\text{all}}},$$

where each $\delta_{\alpha,i}$, for $\alpha \in \{D, U, L, S, \text{all}\}$ and $i \in [k_\alpha]$, indicates the position of an 8-bit block from Enc_α inside the block vector $P = \mathbf{e}'$. In all, Δ reveals to the verifier $\mathcal{V}$ that there are sufficient characters from each set Σ_α for $\alpha \in \{D, U, L, S, \text{all}\}$ as required by the password policy. This information does not affect the privacy of the password since the original position of those blocks are hidden due to the secret permutation (pre-hash salt) χ and padding in the PreHash algorithm.

The prover possesses a witness, consisting of vectors $\mathbf{x} = (\mathbf{e}_0, \mathbf{e}') \in \{0, 1\}^{n_{\max}(\lceil \log n_{\max} \rceil + 8)}$ and $\mathbf{r} \in \{0, 1\}^m$ such that

1. $\mathbf{Ax} + \mathbf{Br} = \mathbf{h} \bmod q$;
2. $\mathbf{e}_0 = (\text{bin}(\chi(1) - 1), \dots, \text{bin}(\chi(n_{\max}) - 1))$;
3. $\mathbf{e}' = (\mathbf{x}_1, \dots, \mathbf{x}_{n_{\max}})$ such that $\mathbf{x}_{\delta_{\alpha,i}} \in \text{Enc}_\alpha$ for every $\delta_{\alpha,i} \in \Delta$.

First, notice that if the matrix $\mathbf{A}$ is expressed as a block matrix, $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \dots \mid \mathbf{A}_{n_{\max}}]$ where $\mathbf{A}_0 \in \mathbb{Z}_q^{n_{\max} \times \lceil \log n_{\max} \rceil}$ and $\mathbf{A}_i \in \mathbb{Z}_q^{n \times 8}$ for $i \in [n_{\max}]$, then $\mathbf{Ax} + \mathbf{Br} = \mathbf{h} \bmod q$ is identical to

$$\mathbf{A}_0 \mathbf{e}_0 + \sum_{\delta_{\alpha,i} \in \Delta} \mathbf{A}_{\delta_{\alpha,i}} \mathbf{x}_{\delta_{\alpha,i}} + \sum_{j \in [n_{\max}] \setminus \Delta} \mathbf{A}_j \mathbf{x}_j = \mathbf{h} \bmod q. \quad (7.4)$$

This form is already compatible with the ZKAoK from Section 2.6.4 but we now have to incorporate set membership proofs.

- For $\mathbf{e}_0$, nothing else needs to be done since $\mathbf{e}_0 \in \text{Set}_{n_{\max}}$ already.
- For each $\mathbf{x}_{\delta_{\alpha,i}}$ with $\delta_{\alpha,i} \in \Delta$, we append $\eta_\alpha - 1$ appropriate blocks from Enc_α to obtain a vector $\mathbf{x}_{\delta_{\alpha,i}}^* \in \text{Set}_\alpha$.
- For each $\mathbf{x}_j$ where $j \in [n_{\max}] \setminus \Delta$ and similarly $\mathbf{r}$, these vectors only need to be proven to be binary. Let $\zeta = |[n_{\max}] \setminus \Delta| = n_{\max} - n_{\min}$.
 1. Let $\mathbf{y} = (\mathbf{x}_{j_1}, \dots, \mathbf{x}_{j_\zeta})$, where $j_i \in [n_{\max}] \setminus \Delta$ and $j_1 < \dots < j_\zeta$ and $\mathbf{z} = (\mathbf{y}, \mathbf{r}) \in \{0, 1\}^{8\zeta + m}$.

Figure 7.2: Example of an Extended Password Policy Linear Relation

2. Append a suitable binary vector to $\mathbf{z}$ to obtain $\mathbf{z}^* \in \{0, 1\}^{2(8\zeta+m)}$ such that $\text{hw}(\mathbf{z}^*) = 8\zeta + m$, meaning $\mathbf{z}^* \in \mathbb{B}_{8\zeta+m}^2$.

From this, we construct the witness vector for the relation to be proven, a binary vector $\mathbf{w}$ of length ℓ , where

$$\ell = n_{\max} \lceil \log n_{\max} \rceil + 8 \sum_{\alpha \in \{D, U, L, S, \text{all}\}} k_\alpha \eta_\alpha + 2(8\zeta + m).$$

and $\mathbf{w}$ has the following form,

$$\mathbf{w} = (\mathbf{e}_0, \mathbf{x}_{\delta_{D,1}}^*, \dots, \mathbf{x}_{\delta_{D,k_D}}^*, \mathbf{x}_{\delta_{U,1}}^*, \dots, \mathbf{x}_{\delta_{U,k_U}}^*, \mathbf{x}_{\delta_{L,1}}^*, \dots, \mathbf{x}_{\delta_{L,k_L}}^*, \mathbf{x}_{\delta_{S,1}}^*, \dots, \mathbf{x}_{\delta_{S,k_S}}^*, \mathbf{x}_{\delta_{\text{all},1}}^*, \dots, \mathbf{x}_{\delta_{\text{all},k_{\text{all}}}}^*, \mathbf{z}^*). \quad (7.5)$$

As explained earlier in the section, when extending the secret vectors, their corresponding public matrices, $\mathbf{A}_{\delta_{\alpha,i}}$ for all $\delta_{\alpha,i} \in \Delta$ and $[\mathbf{A}_{j_1} \mid \dots \mid \mathbf{A}_{j_\zeta} \mid \mathbf{B}]$, also have to be extended with a suitable matrix of all zero entries to preserve Equation (7.4). Using $\mathbf{M} \in \mathbb{Z}_q^{n \times \ell}$ to denote the concatenation of the properly extended matrices, with $\mathbf{v} = \mathbf{h} \in \mathbb{Z}_q^n$, we have the final linear relation

$$\mathbf{M}\mathbf{w} = \mathbf{v} \bmod q.$$

We present a simple example in Figure 7.2 for a password policy $f = ((1, 0, 0, 1), 2, 16)$.

With the appropriate linear relation fixed, the final step is to determine the set of valid witness vectors **VALID** containing $\mathbf{w}$, set $\mathcal{S}$ and permutations $\{\Gamma_\phi \mid \phi \in \mathcal{S}\}$ so that the conditions specified in Section 2.6.4 are satisfied.

First, we define **VALID** as the set of vectors $\mathbf{w} \in \{0, 1\}^\ell$ with the form in Equation (7.5) such that

1. $\mathbf{e}_0 \in \text{Set}_{n_{\max}}$;
2. $\mathbf{x}_{\delta_{\alpha,i}}^* \in \text{Set}_\alpha$, for all $\delta_{\alpha,i} \in \Delta$;
3. $\mathbf{z}^* \in \mathbb{B}_{8\zeta+m}^2$.

It is easily seen that the vector $\mathbf{w}$ obtained from the extensions described above belong to **VALID** specified. Next, we define the set $\mathcal{S}$ as

$$\mathcal{S} = \mathcal{S}_{n_{\max}} \times \prod_{\alpha \in \{D, U, L, S, \text{all}\}} (\mathcal{S}_{\eta_\alpha})^{k_\alpha} \times \mathcal{S}_{2(8\zeta+m)}$$

and denote any $\phi \in \mathcal{S}$ as

$$\phi = (\pi, \tau_{D,1}, \dots, \tau_{D,k_D}, \tau_{U,1}, \dots, \tau_{U,k_U}, \tau_{L,1}, \dots, \tau_{L,k_L}, \tau_{S,1}, \dots, \tau_{S,k_S}, \tau_{\text{all},1}, \dots, \tau_{\text{all},k_{\text{all}}}, \theta).$$

Each $\phi \in \mathcal{S}$ induces a permutation of blocks Γ_ϕ that transforms a block vector $\mathbf{w} \in \mathbb{Z}^\ell$ of the form in Equation (7.5),

$$\mathbf{w} = (\mathbf{e}_0, \mathbf{x}_{\delta_{D,1}}^*, \dots, \mathbf{x}_{\delta_{D,k_D}}^*, \mathbf{x}_{\delta_{U,1}}^*, \dots, \mathbf{x}_{\delta_{U,k_U}}^*, \mathbf{x}_{\delta_{L,1}}^*, \dots, \mathbf{x}_{\delta_{L,k_L}}^*, \mathbf{x}_{\delta_{S,1}}^*, \dots, \mathbf{x}_{\delta_{S,k_S}}^*, \mathbf{x}_{\delta_{\text{all},1}}^*, \dots, \mathbf{x}_{\delta_{\text{all},k_{\text{all}}}}^*, \mathbf{z}^*),$$

where $\mathbf{e}_0 \in \mathbb{Z}^{n_{\max} \lceil \log n_{\max} \rceil}$, $\mathbf{x}_{\delta_{\alpha,i}}^* \in \mathbb{Z}^{8\eta_\alpha}$ for all $\delta_{\alpha,i} \in \Delta$ and $\mathbf{z}^* \in \mathbb{Z}^{2(8\zeta+m)}$ as follows:

1. $\mathbf{e}_0 \mapsto T_{\pi, \lceil \log n_{\max} \rceil}(\mathbf{e}_0)$;
2. For all $\delta_{\alpha,i} \in \Delta$, $\mathbf{x}_{\delta_{\alpha,i}}^* \mapsto T_{\tau_{\alpha,i}, 8}(\mathbf{x}_{\delta_{\alpha,i}}^*)$;
3. $\mathbf{z}^* \mapsto \theta(\mathbf{z}^*)$.

The conditions on **VALID** in Section 2.6.4 and listed below, specified by the abstract relation Stern-like protocol by Libert *et al.* [136], can be seen to be satisfied from the properties in (7.1), (7.2) and (7.3).

$$- \mathbf{w} \in \text{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \text{VALID};$$

- If $\mathbf{w} \in \text{VALID}$ and ϕ is sampled from $\mathcal{U}(\mathcal{S})$, then the distribution of $\Gamma_\phi(\mathbf{w})$ over VALID is uniform.

The Protocol. With password policy compliance reduced to an instance of the abstract relation, R_{abs} , we now describe the procedure for executing the protocol for proving password policy compliance using the string commitment scheme **COM** by Kawachi *et al.* [119], which is statistically hiding and computationally binding. Before the protocol is run, both the prover $\mathcal{P}$ and verifier $\mathcal{V}$ construct the extended matrix $\mathbf{M}$ and vector $\mathbf{v}$ with the common inputs $(\mathbf{A}, \mathbf{B}, \mathbf{h}, \Delta)$ and $\mathcal{P}$ additionally constructs the vector $\mathbf{w} \in \text{VALID}$ from the secret vectors $\mathbf{x}$ and $\mathbf{r}$. Then, both parties interact according to Figure 2.2 with the algorithms described in Section 2.6.4. As a corollary of Theorem 2.6.6, we obtain the security of the ZKPPC protocol.

Theorem 7.4.2. *The protocol described above is a ZKPPC protocol for the randomised password hashing scheme $\mathcal{LH}$ from Section 7.4.2 and password policy of the type $f = ((k_D, k_U, k_L, k_S), n_{\min}, n_{\max})$ assuming the hardness of the SIS problem. The protocol is a statistical ZKAoK with perfect completeness, soundness error of $2/3$ and communication cost $\mathcal{O}(\ell \log q)$.*

Proof. Since the ZKPPC protocol uses the abstract Stern-like protocol in Figure 2.2 with the algorithms described in Section 2.6.4, its properties of perfect completeness, soundness error of $2/3$ and communication costs follow from Theorem 2.6.6. The simulator for the ZKPPC protocol is simply the simulator of the abstract protocol in the proof of Theorem 2.6.6.

To recover a valid witness for knowledge extraction, the knowledge extractor of the abstract Stern-like protocol is run to obtain $\mathbf{w}' \in \text{VALID}$ such that $\mathbf{M}\mathbf{w}' = \mathbf{v} \bmod q$. Then, following the structure of $\mathbf{w}'$ in the set VALID , we can "reverse" the transformations used to extend the secrets to obtain vectors $\mathbf{x}' = (\mathbf{e}'_0, \mathbf{x}'_1, \dots, \mathbf{x}'_{n_{\max}}) \in \{0, 1\}^{n_{\max}(\lceil \log n_{\max} \rceil + 8)}$ and $\mathbf{r}' \in \{0, 1\}^m$ where

1. $\mathbf{A}\mathbf{x}' + \mathbf{B}\mathbf{r}' = \mathbf{h} \bmod q$;
2. $\mathbf{e}'_0 \in \text{Set}_{n_{\max}}$;
3. For all $\delta_{\alpha,i} \in \Delta$, we have $\mathbf{x}'_{\delta_{\alpha,i}} \in \text{Enc}_\alpha$.

From any element in $\text{Set}_{n_{\max}}$, we can recover a permutation over $n_{\max}$ elements. So let χ' be the permutation recovered from $\mathbf{e}'_0 \in \text{Set}_{n_{\max}}$, then we recover $\mathbf{e}' \in \{0, 1\}^{8n_{\max}}$ by applying the inverse permutation $T_{\chi',8}^{-1}$ to the vector $\mathbf{x}' = (\mathbf{x}'_1, \dots, \mathbf{x}'_{n_{\max}})$. Lastly, by removing blocks of length 8 corresponding to encodings of ASCII codes for non-printable

characters from $\mathbf{e}'$, we get a vector that encodes some password string pw' such that $f(pw') = \text{true}$. $\square$

Now, we briefly estimate the practical efficiency of the protocol. From the description of the protocol, we can see that each round of communication costs in bits is roughly

$$\ell \log q = (n_{\max} \lceil \log n_{\max} \rceil + 8 \sum_{\alpha \in \{D, U, L, S, \text{all}\}} k_{\alpha} \eta_{\alpha} + 2(8\zeta + m)) \log q.$$

In practice, the KTX commitment scheme can use relatively small parameters, such as $n = 256, \log q = 10$ and $m = 5120$. Using a rather common password policy $f = ((1, 1, 1, 1), 8, 16)$, communication costs amount to about 17 KB. With a soundness error of $2/3$ per run of the protocol, achieving a high level of confidence in the proofs requires repeating the protocol many times. As an example, if a soundness error of 2^{-30} is desired, then 52 repetitions of the protocol would be needed, meaning a final cost of about 900 KB. This communication complexity can be reduced in implementations with some optimizations; for example, sending the seed for the pseudorandom number generator that yields the random vector instead of said vector directly.

Chapter 8

Conclusion

In this thesis, we applied lattice cryptography to develop several privacy-preserving protocols for databases and password policy checks. Particularly, the focus is on private database queries on outsourced databases in Part I and zero-knowledge protocols for databases and password policy check in Part II.

For private database queries, fully homomorphic encryption for finite extension fields were applied to improve functionality and communication complexity compared to the Boolean circuit approach. In Chapter 4, we proposed a private wildcard pattern matching algorithm that used some auxiliary information to support various wildcards, \$, % and !(c). Notably, our algorithm is the only one that supports the exclusion wildcard !(c) so far. Attribute strings and patterns are encrypted character-wise, with the \$ wildcard and # empty character sharing the same encoding. The auxiliary information consists of two lists of bits: the first D hides the pattern length and determines which set of shifted attribute strings are relevant to the matching algorithm; the second denotes the positions in the pattern that contain wildcards and is used to correct the result of equality tests. The algorithm is then composed with a simple procedure to evaluate conjunctive or disjunctive queries over multiple conditions to obtain private compound query protocols with wildcard pattern matching capabilities.

Following that, in Chapter 5, we designed a order comparison algorithm for integers encoded in finite field elements. This lets us significantly reduce communication costs for PDQ protocols involving order comparisons by decreasing the ciphertext expansion factor. Exploiting the vector space structure of finite fields, we introduced an alternate method to evaluate certain functions. Finite field elements are decomposed into smaller blocks, which are then put through smaller, simpler functions. Their outputs are finally combined to obtain the result of the original function. From this, we can use two order comparisons

to obtain a private range query protocol and even compound queries with different types of conditions. Experiment results indicate that the finite field-based technique is efficient and compares well with the Boolean circuit based comparison algorithm, achieving slightly better asymptotic performance with much lower communication costs.

In Part II, we constructed zero-knowledge protocols for database queries and password policy checks. A new primitive called zero-knowledge expressive elementary databases (ZK-EEDB) are defined in Chapter 6 that enabled range queries over keys, values and records of a committed database in zero-knowledge. The key technique we introduced is a novel algorithm, **Explain** to prove that some given commitment is a soft commitment without sacrificing security. Using **Explain**, we applied the complete subtree method by Naor *et al.* [161] to compute efficient proofs for (non-)membership of database keys in an interval. Then, we used a second tree to support efficient queries over values and records of the database, with consistency between the two trees enforced by proving membership of any record in both trees. We also presented an instantiation of trapdoor mercurial commitments, that is used to construct ZK-EEDB, based on the hardness of the SIS problem.

Lastly, in Chapter 7, we proposed the first zero-knowledge password policy check (ZKPPC) protocol from lattices. Using the Kawachi-Tanaka-Xagawa commitment scheme [119], we designed a randomised password hashing scheme that took passwords encoded in binary strings. Then, we constructed a Stern-like zero-knowledge argument of set membership for proving that a password hashed with the earlier scheme satisfies the mandated password policy. To prove set membership of some secret vector, we extended the secret vector to a longer one containing every member of the set and modified the public matrix so that the linear relation remained the same. Then, permutations with a particular structure and a random mask is used to randomize the witness and remove any correlation of the result with the original witness. The two components together make up the ZKPPC protocol and are based on the hardness of the SIS problem.

8.1 Open Questions and Future Directions

In this section, we highlight several open questions and possible future directions that arise from the work in this thesis.

Chapter 4: One important open question is a wildcard pattern matching algorithm that supports wildcards involving sets or ranges of characters. The main ones include $[S]$ that ask for a character within the set S and $!(S)$, which extends the $!(c)$ wildcard to a set

of characters. $\mathcal{S}$ can also be replaced with a range of characters such as $[a\dots z]$. Current techniques allow for exclusion of a single character and cannot handle $!(c)$ in the final position of any pattern. It stems from the need to determine if the character at the final position is the empty character $\#$ or c , in which case the position should be marked 0, or any other character, which should be marked with 1 as usual. Supporting sets of size two with exclusion wildcards would suffice to allow $!(c)$ to be used at the end of patterns.

Chapter 5: With equality and order conditions possible with FHE for finite extension fields, the main obstacle to a fully functional private database query system is integer arithmetic for count, sum and statistical queries. Cheon *et al.* [53, 54] modified the plaintext modulus from 2 to 2^{15} to get efficient integer arithmetic which translated to very fast count and sum query protocols. This technique can be applied to our work but we need to use some large enough prime p instead. In this case, the equality test algorithm EQ would have much lower performance as the degree of the polynomial to evaluate increases significantly.

The framework introduced in this chapter could possibly be applied to evaluate the integer addition circuit but its efficiency is a big question. Cheon *et al.* [53, 54] found that the integer addition Boolean circuit is a couple of magnitudes slower than the Boolean circuits for equality and order comparisons. It would still be interesting to determine the performance of a finite field integer addition algorithm. A PDQ protocol for count and sum queries would be the next logical step but new techniques may need to be found to get an efficient construction.

Chapter 6: With ZK-EEDB, we get range queries over keys, values and records of a database. The main open question would be whether other queries can be realized in this setting. For example, ZK-EEDB keys are one-dimensional unlike Ostrovsky *et al.* [166], who achieved d -dimensional orthogonal range queries in a weaker setting. An important question is the possibility and efficiency of extending ZK-EEDB to support d -dimensional range queries.

In an orthogonal work to ZK-EEDB, Ghosh *et al.* [89] proposed zero-knowledge lists, generalizing zero-knowledge sets, to support order queries. For some list $\mathcal{L}$ committed to a public digest, an order query takes a set of elements $\mathcal{S}$ and asks for the subset $\mathcal{S} \cap \mathcal{L}$ ordered according to their rank in $\mathcal{L}$, i.e., the answer is an ordered list $(s_1, s_2, \dots, s_m)$ such that $s_1 <_{\mathcal{L}} s_2 <_{\mathcal{L}} \dots <_{\mathcal{L}} s_m$, where $<_{\mathcal{L}}$ denotes the order of the list $\mathcal{L}$. We can try to adapt this to databases, and return the values, or possibly a portion of the entire value, ordered according to the rank of the records in the database. In ZK-EEDB, this

is trivial since entire records are revealed and the order of any answer set can be directly seen. This means that we have to consider multi-dimensional values in the database, which will come with another set of complexities.

Chapter 7: Although we designed a lattice-based zero-knowledge password policy check protocol, there are no known constructions of verifier-based password-based authenticated key exchange (VPAKE) from lattices. Besides that, there are other variants of password-based authenticated key exchange to which we can extend our lattice-based ZKPPC to support. A final interesting open problem, posed by Professor Adi Shamir, is to design a ZKPPC protocol that can support password policies that require passwords that are different from the currently registered one. This has not been realized by the only other ZKPPC scheme by Kiefer and Manulis [124] as well.

Bibliography

- [1] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In *47th ACM STOC*, pages 733–742. ACM Press, 2015.
- [2] D. Aggarwal and N. Stephens-Davidowitz. Just take the average! an embarrassingly simple 2^n -time algorithm for SVP (and CVP). In *1st Symposium on Simplicity in Algorithms*, page 1, 2018.
- [3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *ACM SIGMOD 2004*, pages 563–574. ACM, 2004.
- [4] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, 1996.
- [5] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, 1998.
- [6] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th ACM STOC*, pages 284–293. ACM Press, 1997.
- [7] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd ACM STOC*, pages 601–610. ACM Press, 2001.
- [8] M. R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HELib and SEAL. Cryptology ePrint Archive, Report 2017/047, 2017. <http://eprint.iacr.org/2017/047>.
- [9] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.

- [10] O. T. Alliance. Online trust alliance reports doubling of cyber incidents in 2017, 2017.
- [11] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, 2009.
- [12] S. Arora and R. Ge. New algorithms for learning in presence of errors. In *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, 2011.
- [13] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.
- [14] B. Barak and O. Goldreich. Universal arguments and their applications. In *Proceedings 17th IEEE Annual Conference on Computational Complexity, CCC '02*, pages 162–171. IEEE Computer Society, 2002.
- [15] J. Baron, K. E. Defrawy, K. Minkovich, R. Ostrovsky, and E. Tressler. 5pm: Secure pattern matching. In I. Visconti and R. D. Prisco, editors, *Security and Cryptography for Networks (SCN) 2012*, volume 7485 of *LNCS*, pages 222–240. Springer, 2012.
- [16] C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 478–498. Springer, Heidelberg, 2016.
- [17] C. Baum, I. Damgård, S. Oechsner, V. Lyubashevsky, and C. Peikert. More efficient commitments from structured lattice assumptions. Cryptology ePrint Archive, Report 2016/997, 2016. <http://eprint.iacr.org/2016/997>.
- [18] E. Bauman, Y. Lu, and Z. Lin. Half a century of practice: Who is still storing plaintext passwords? In *ISPEC 2015*, volume 9065 of *Lecture Notes in Computer Science*. Springer, 2015.
- [19] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000.
- [20] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *IEEE Symposium on Security and Privacy*. IEEE, 1992.

- [21] S. M. Bellare and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM CCS 1993*. ACM, 1993.
- [22] D. Benarroch, Z. Brakerski, and T. Lepoint. FHE over the integers: Decomposed and batched in the post-quantum regime. In *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 271–301. Springer, Heidelberg, 2017.
- [23] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for sphfs and efficient one-round pake protocols. In *CRYPTO 2013*, volume 8402 of *Lecture Notes in Computer Science*. Springer, 2013.
- [24] F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 551–572. Springer, Heidelberg, 2014.
- [25] F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 305–325. Springer, Heidelberg, 2015.
- [26] F. Benhamouda and D. Pointcheval. Verifier-based password-authenticated key exchange: New models and constructions. Cryptology ePrint Archive, Report 2013/833, 2013.
- [27] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, 2003.
- [28] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO’93*, volume 773 of *LNCS*, pages 278–291. Springer, Heidelberg, 1994.
- [29] A. Boldyreva, N. Chenette, Y. Lee, and A. O’Neill. Order-preserving symmetric encryption. In *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, Heidelberg, Apr. 2009.
- [30] A. Boldyreva, N. Chenette, and A. O’Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 578–595. Springer, Heidelberg, Aug. 2011.

- [31] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu. Private database queries using somewhat homomorphic encryption. In M. J. Jacobson Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 102–118. Springer, Heidelberg, June 2013.
- [32] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Heidelberg, 2008.
- [33] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, Apr. 2015.
- [34] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *Theory of Cryptography (TCC) 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007.
- [35] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, Heidelberg, 2013.
- [36] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, 2012.
- [37] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012*, pages 309–325. ACM, 2012.
- [38] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *45th ACM STOC*, pages 575–584. ACM Press, 2013.
- [39] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *52nd FOCS*, pages 97–106. IEEE Computer Society Press, 2011.
- [40] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, 2011.

- [41] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, 2001.
- [42] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *Network and Distributed System Security Symposium (NDSS) 2014*. The Internet Society, 2014.
- [43] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 Part I*, volume 8042 of *LNCS*, pages 353–373. Springer, 2013.
- [44] D. Cash, F.-H. Liu, A. O’Neill, and C. Zhang. Reducing the leakage in practical order-revealing encryption. Cryptology ePrint Archive, Report 2016/661, 2016. <http://eprint.iacr.org/2016/661>.
- [45] D. Catalano, Y. Dodis, and I. Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In *TCC 2006*, 2006.
- [46] D. Catalano and D. Fiore. Vector commitments and their applications. In *PKC*, 2013.
- [47] D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. In *Eurocrypt*, 2008.
- [48] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *EUROCRYPT 2005*, 2005.
- [49] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *J. of Cryptology*, 2013.
- [50] M. Chase and E. Shen. Substring-searchable symmetric encryption. *PoPETs*, 2015(2):263–281, 2015.
- [51] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *FSE 2016*, volume 9783 of *LNCS*, pages 474–493. Springer, Heidelberg, Mar. 2016.

- [52] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 315–335. Springer, Heidelberg, 2013.
- [53] J. H. Cheon, M. Kim, and M. Kim. Search-and-compute on encrypted data. In *FC 2015 Workshops*, volume 8976 of *LNCS*, pages 142–159. Springer, Heidelberg, 2015.
- [54] J. H. Cheon, M. Kim, and M. Kim. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Trans. Information Forensics and Security*, 11(1):188–199, 2016.
- [55] J. H. Cheon and D. Stehlé. Fully homomorphic encryption over the integers revisited. In *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 513–536. Springer, Heidelberg, 2015.
- [56] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, 2016.
- [57] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 377–408. Springer, Heidelberg, 2017.
- [58] J.-S. Coron, T. Lepoint, and M. Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *PKC 2014*, volume 8383 of *LNCS*, pages 311–328. Springer, Heidelberg, 2014.
- [59] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO’94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, 1994.
- [60] R. Cramer, I. Damgård, C. Xing, and C. Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 479–500. Springer, Heidelberg, 2017.
- [61] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security (ACM CCS) 2006*, pages 79–88. ACM, 2006.

- [62] L. Dagum and R. Menon. OpenMP: An industry standard API for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.
- [63] I. Damgård. A design principle for hash functions. In *CRYPTO’89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, 1990.
- [64] I. Damgård, M. Geisler, and M. Kroigard. Homomorphic encryption and secure comparison. *Int. J. Appl. Cryptol.*, 1(1), 2008.
- [65] J. L. Dautrich, Jr. and C. V. Ravishankar. Compromising privacy in precise query protocols. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT ’13, pages 155–166. ACM, 2013.
- [66] R. del Pino and V. Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 365–394. Springer, Heidelberg, 2017.
- [67] Y. Ding and L. Fan. Efficient password-based authenticated key exchange from lattices. In *CIS 2011*. IEEE, 2011.
- [68] C. Dong, L. Chen, and Z. Wen. When private set intersection meets big data: an efficient and scalable protocol. In *ACM CCS 2013*. ACM, 2013.
- [69] L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014*, 2014.
- [70] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, 2015.
- [71] F. B. Durak, T. M. DuBuisson, and D. Cash. What else is revealed by order-revealing encryption? In *ACM CCS 16*, pages 1155–1166. ACM Press, Oct. 2016.
- [72] M. Egorov and M. Wilkison. ZeroDB white paper. *CoRR*, abs/1602.07168, 2016.
- [73] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner. Rich queries on encrypted data: Beyond exact matches. In G. Pernul, P. Y. A. Ryan, and E. R. Weippl, editors, *European Symposium on Research in Computer Security (ESORICS) 2015 Part II*, volume 9327 of *LNCS*, pages 123–145. Springer, 2015.
- [74] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. <http://eprint.iacr.org/2012/144>.

- [75] D. Florêncio and C. Herley. Where do security policies come from? In *SOUPS 2010*. ACM, 2010.
- [76] K. B. Frikken. Practical private DNA string searching and matching through efficient oblivious automata evaluation. In E. Gudes and J. Vaidya, editors, *DBSec 2009*, volume 5645 of *LNCS*, pages 81–94. Springer, 2009.
- [77] J. Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88-A(1):172–188, 2005.
- [78] S. Gates. LinkedIn password hack: Check to see if yours was one of the 6.5 million leaked, 2012. http://www.huffingtonpost.com/2012/06/07/linkedin-password-hack-check_n_1577184.html.
- [79] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.
- [80] R. Gennaro and S. Micali. Independent zero-knowledge sets. In *ICALP*, 2006.
- [81] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *ACM Symposium on Theory of Computing (STOC) 2009*, pages 169–178. ACM, 2009.
- [82] C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178. ACM Press, 2009.
- [83] C. Gentry, S. Halevi, C. S. Jutla, and M. Raykova. Private database access with HE-over-ORAM architecture. In T. Malkin, V. Kolesnikov, A. B. Lewko, and M. Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 172–191. Springer, Heidelberg, June 2015.
- [84] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, Heidelberg, 2012.
- [85] C. Gentry, P. MacKenzie, and Z. Ramzan. A method for making password-based key exchange resilient to server compromise. In *CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*. Springer, 2006.

- [86] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206. ACM Press, 2008.
- [87] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, 2013.
- [88] E. Ghosh, O. Ohrimenko, and R. Tamassia. Verifiable order queries and order statistics on a list in zero-knowledge. In *ACNS*, 2015.
- [89] E. Ghosh, O. Ohrimenko, and R. Tamassia. Zero-knowledge authenticated order queries and order statistics on a list. In *ACNS*, 2015.
- [90] E. Ghosh, O. Ohrimenko, and R. Tamassia. Efficient verifiable range and closest point queries in zero-knowledge. *PoPETs*, 2016(4), 2016.
- [91] E. Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.
- [92] O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540 – 563, 2000.
- [93] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO’97*, volume 1294 of *LNCS*, pages 112–131. Springer, Heidelberg, 1997.
- [94] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, 1982.
- [95] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
- [96] V. Goyal, R. Ostrovsky, A. Scafuro, and I. Visconti. Black-box non-black-box zero knowledge. In *STOC*, 2014.
- [97] T. Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library Version 6.1.2*, 2017.
- [98] L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997.

- [99] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE Symposium on Security and Privacy*, pages 655–672. IEEE Computer Society Press, May 2017.
- [100] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In *ACM CCS 07*, pages 466–475. ACM Press, 2007.
- [101] S. Halevi and V. Shoup. Faster homomorphic linear transformations in helib. Cryptology ePrint Archive, Report 2018/244, 2018. <https://eprint.iacr.org/2018/244>.
- [102] S. Halevi and V. Shoup. HELib: Software library for homomorphic encryption. <http://github.com/shaih/HElib>, 2018.
- [103] I. Haviv and O. Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012.
- [104] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In R. Canetti, editor, *Theory of Cryptography (TCC) 2008*, volume 4948 of *LNCS*, pages 155–175. Springer, 2008.
- [105] R. Hiromasa, M. Abe, and T. Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. In *PKC 2015*, volume 9020 of *LNCS*, pages 699–715. Springer, Heidelberg, 2015.
- [106] J. Hoffstein, J. Pipher, and J. H. Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, pages 267–288. Springer Berlin Heidelberg, 1998.
- [107] J. Hoffstein, J. Pipher, and J. H. Silverman. NSS: An NTRU lattice-based signature scheme. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 211–228. Springer, Heidelberg, 2001.
- [108] T. Hunt. Have i been pwned, 2017. <https://haveibeenpwned.com/>, accessed on 7 July 2017.
- [109] Y. Ishai, E. Kushilevitz, R. Ostrovksy, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.

- [110] A. Jäschke and F. Armknecht. (finite) field work: Choosing the best encoding of numbers for the computation. Cryptology ePrint Archive, Report 2017/582, 2017. <https://eprint.iacr.org/2017/582>, Also appeared in CANS 2017.
- [111] S. Kamara and T. Moataz. SQL on structurally-encrypted databases. Cryptology ePrint Archive, Report 2016/453, 2016. <http://eprint.iacr.org/2016/453>.
- [112] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In A. Sadeghi, editor, *Financial Cryptography and Data Security (FC) 2013*, volume 7859 of *LNCS*, pages 258–274. Springer, 2013.
- [113] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In T. Yu, G. Danezis, and V. D. Gligor, editors, *ACM Conference on Computer and Communications Security (ACM CCS) 2012*, pages 965–976. ACM, 2012.
- [114] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *15th ACM STOC*, pages 193–206. ACM Press, 1983.
- [115] M. Kantarcioğlu and C. Clifton. Security issues in querying encrypted data. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, DBSec’05, pages 325–337, Berlin, Heidelberg, 2005. Springer-Verlag.
- [116] J. Katz and L. Malka. Secure text processing with applications to private DNA matching. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security (ACM CCS) 2010*, pages 485–492. ACM, 2010.
- [117] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.
- [118] J. Katz and V. Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*. Springer, 2009.
- [119] A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT 2008*, 2008.

- [120] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic attacks on secure outsourced databases. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 1329–1340. ACM Press, 2016.
- [121] F. Kerschbaum. Frequency-hiding order-preserving encryption. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 15*, pages 656–667. ACM Press, Oct. 2015.
- [122] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *45th FOCS*, pages 126–135. IEEE Computer Society Press, 2004.
- [123] F. Kiefer. *Advancements in password-based cryptography*. PhD thesis, University of Surrey, 2016.
- [124] F. Kiefer and M. Manulis. Zero-knowledge password policy checks and verifier-based PAKE. In *ESORICS 2014*, volume 8713 of *Lecture Notes in Computer Science*. Springer, 2014.
- [125] F. Kiefer and M. Manulis. Blind password registration for two-server password authenticated key exchange and secret sharing protocols. In *ISC 2016*, volume 9866 of *Lecture Notes in Computer Science*. Springer, 2016.
- [126] F. Kiefer and M. Manulis. Blind password registration for verifier-based PAKE. In *AsiaPKC@AsiaCCS 2016*. ACM, 2016.
- [127] M. Kim, H. T. Lee, S. Ling, S. Q. Ren, B. H. M. Tan, and H. Wang. Better security for queries on encrypted databases. Cryptology ePrint Archive, Report 2016/470, 2016.
- [128] M. Kim, H. T. Lee, S. Ling, B. H. M. Tan, and H. Wang. Private compound wildcard queries using fully homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2017.
- [129] M. Kim, H. T. Lee, S. Ling, and H. Wang. On the efficiency of fhe-based private queries. *IEEE Transactions on Dependable and Secure Computing*, 15(2):357–363, 2018.
- [130] T. Laarhoven, M. Mosca, and J. van de Pol. Finding shortest lattice vectors faster using quantum search. *Design, Codes and Cryptography*, 77(2-3):375–400, 2015.
- [131] M. Lacharite, B. Minaud, and K. G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 1–18.

- [132] K. Laine. SEAL: Simple encrypted arithmetic library 2.3.1. <https://www.microsoft.com/en-us/research/project/simple-encrypted-arithmetic-library/>, 2017.
- [133] A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Design, Codes and Cryptography*, 75(3):565–599, 2015.
- [134] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
- [135] K. Lewi and D. J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *ACM CCS 16*, pages 1167–1178. ACM Press, Oct. 2016.
- [136] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 373–403. Springer, Heidelberg, 2016.
- [137] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 101–131. Springer, Heidelberg, 2016.
- [138] B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Heidelberg, 2016.
- [139] B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC 2010*, 2010.
- [140] S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, Heidelberg, 2013.
- [141] S. Ling, K. Nguyen, and H. Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *PKC 2015*, volume 9020 of *LNCS*, pages 427–449. Springer, Heidelberg, 2015.
- [142] S. Ling, K. Nguyen, H. Wang, and Y. Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS 17*, volume 10355 of *LNCS*, pages 293–312. Springer, Heidelberg, 2017.

- [143] D. Linthicum. Safer but not immune: Cloud lessons from the equifax breach, 2017.
- [144] M. Liskov. Updatable zero-knowledge databases. In *Asiacrypt*, 2005.
- [145] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In H. J. Karloff and T. Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.
- [146] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC 2008*, volume 4939 of *LNCS*, pages 162–179. Springer, Heidelberg, 2008.
- [147] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, 2012.
- [148] V. Lyubashevsky and D. Micciancio. Generalized compact Knapsacks are collision resistant. In *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 144–155. Springer, Heidelberg, 2006.
- [149] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, 2010.
- [150] E. M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, 1976.
- [151] R. C. Merkle. A Certified Digital Signature. In *CRYPTO*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.
- [152] R. C. Merkle. One way hash functions and DES. In *CRYPTO’89*, volume 435 of *LNCS*, pages 428–446. Springer, Heidelberg, 1990.
- [153] S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. In *44th FOCS*, 2003.
- [154] D. Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001.
- [155] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *43rd FOCS*, pages 356–365. IEEE Computer Society Press, 2002.
- [156] D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 465–484. Springer, Heidelberg, 2011.

- [157] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, 2012.
- [158] D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, 2013.
- [159] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, 2004.
- [160] D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.
- [161] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO 2001*, 2001.
- [162] M. Naor and K. Nissim. Certificate revocation and certificate update. In *7th USENIX Security Symposium*, 1998.
- [163] M. Naor and A. Ziv. Primary-secondary-resolver membership proof systems. In *TCC*, 2015.
- [164] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *ACM CCS 15*, pages 644–655. ACM Press, Oct. 2015.
- [165] K. Nuida and K. Kurosawa. (Batch) fully homomorphic encryption over integers for non-binary message spaces. In *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 537–555. Springer, Heidelberg, 2015.
- [166] R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs for generalized queries on a committed database. In *ICALP 2004*, 2004.
- [167] D. Papadopoulos, S. Papadopoulos, and N. Triandopoulos. Taking authenticated range queries to arbitrary dimensions. In *ACM-CCS*, 2014.
- [168] C. Papamanthou, R. Tamassia, and N. Triandopoulos. Optimal verification of operations on dynamic sets. In *Crypto*, 2011.
- [169] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*. Springer, 1991.

- [170] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *41st ACM STOC*, pages 333–342. ACM Press, 2009.
- [171] C. Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.
- [172] C. Peikert, O. Regev, and N. Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *49th ACM STOC*, pages 461–473. ACM Press, 2017.
- [173] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC 2006*, volume 3876 of *LNCS*, pages 145–166. Springer, Heidelberg, 2006.
- [174] C. Peikert and A. Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *39th ACM STOC*, pages 478–487. ACM Press, 2007.
- [175] N. Perlroth. More than half a billion yahoo accounts have been hacked, yahoo confirms, 2016. <https://www.nytimes.com/2016/09/23/technology/yahoo-hackers.html>.
- [176] R. Poddar, T. Boelter, and R. A. Popa. Arx: A strongly encrypted database system. Cryptology ePrint Archive, Report 2016/591, 2016. <http://eprint.iacr.org/2016/591>.
- [177] Y. Polyakov, K. Rohloff, and G. W. Ryan. PALISADE lattice cryptography library user manual. <https://git.njit.edu/palisade/PALISADE>, 2017.
- [178] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*, pages 463–477. IEEE Computer Society Press, May 2013.
- [179] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In T. Wobber and P. Druschel, editors, *ACM Symposium on Operating Systems Principles (SOSP) 2011*, pages 85–100. ACM, 2011.
- [180] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In T. Wobber and P. Druschel, editors, *ACM Symposium on Operating Systems Principles (SOSP) 2011*, pages 85–100. ACM, 2011.
- [181] M. Prabhakaran and R. Xue. Statistically hiding sets. In *CT-RSA*, 2009.

- [182] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, 2005.
- [183] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.
- [184] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich. POPE: Partial order preserving encoding. In *ACM CCS 16*, pages 1131–1142. ACM Press, Oct. 2016.
- [185] S. Roman. *Field Theory*. Graduate Texts in Mathematics. Springer New York, 2005.
- [186] M. Rosca, D. Stehlé, and A. Wallet. On the ring-LWE and polynomial-LWE problems. In *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 146–173. Springer, Heidelberg, 2018.
- [187] T. K. Saha and T. Koshiha. An enhancement of privacy-preserving wildcards pattern matching. In F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. García-Alfaro, editors, *Foundations and Practice of Security (FPS) 2016*, volume 10128 of *LNCS*, pages 145–160. Springer, 2016.
- [188] C. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201 – 224, 1987.
- [189] S. Sedghi, P. van Liesdonk, S. Nikova, P. H. Hartel, and W. Jonker. Searching keywords with wildcards on encrypted data. In J. A. Garay and R. D. Prisco, editors, *Security and Cryptography for Networks (SCN) 2010*, volume 6280 of *LNCS*, pages 138–153. Springer, 2010.
- [190] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.
- [191] V. Shoup. NTL: A library for doing number theory version 11.2.0. Available at <http://www.shoup.net/ntl/>, 2018.
- [192] N. P. Smart and F. Vercauteren. Fully homomorphic simd operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.
- [193] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Design, Codes and Cryptography*, 71(1):57–81, 2014.

- [194] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy 2000*, pages 44–55. IEEE Computer Society, 2000.
- [195] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, 2009.
- [196] J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
- [197] R. Tamassia. Authenticated data structures. In *European Symp. on Algorithms (ESA)*, 2003.
- [198] I. Teranishi, M. Yung, and T. Malkin. Order-preserving encryption secure beyond one-wayness. In *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 42–61. Springer, Heidelberg, Dec. 2014.
- [199] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. *PVLDB*, 6(5):289–300, 2013.
- [200] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. *PVLDB*, 6(5):289–300, 2013.
- [201] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.
- [202] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor. How does your password measure up? the effect of strength meters on password creation. In *USENIX Security Symposium 2012*. USENIX Association, 2012.
- [203] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, Heidelberg, 2010.
- [204] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Mathematics Department, University of Amsterdam. Technical report, TR 81-04, 1981.
- [205] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshihara. Secure pattern matching using somewhat homomorphic encryption. In A. Juels and B. Parno,

- editors, *ACM Cloud Computing Security Workshop (CCSW) 2013*, pages 65–76. ACM, 2013.
- [206] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption. In W. Susilo and Y. Mu, editors, *ACISP 2014*, volume 8544 of *LNCS*, pages 338–353. Springer, 2014.
- [207] Y. Zhang, J. Katz, and C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In T. Holz and S. Savage, editors, *USENIX Security 2016*, pages 707–720. USENIX Association, 2016.