University of São Paulo
Institute of Mathematics and Statistics
Bachelor of Computer Science

Folding FrodoPIR

Private Information Retrieval with optimization

Luiza Barros Reis Soezima

Final Essay

mac 499 — Capstone Project

Supervisor: Prof Dr. Hilder Vitor Lima Pereira
Co-supervisor: Prof. Dr. Alfredo Goldman

São Paulo
2024

To everyone that believed in me
and supported me in this journey.

Acknowldgement

Scientists love mysteries; They love not knowing
— Lawrence Krauss

There's real poetry in the real world; Science is the poetry of reality
— Richard Dawkins

First, I would like to thank Alfredo Goldman for being my mentor during my bachelor, for always supporting my ideas and continuously pushing me to new opportunities and new challenges and for opening the doors for the Masters at Université Grenoble Alpes. I would also like to thank Hilder Vitor for being a great supervisor and support me throughout the research and work related to not only to this project but also related to my growth in cryptologic research.

Second, I would like to thank Sofia Celi, Arantxa Zapico, Daniel Escudero, Fernando Virdia, Octavio Perez, Francisco Rodríguez-Henríquez and all the Criptolatinos community for being my role model and motivate me everyday to follow my dream of being a cryptographer. I would also like to thank Alex Davidson, Leah Namisa Rosenbloom, Pierrick Meaux and many other researchers for the insightful conversations and orientations.

Third, I would like to thank the entire Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP). All the professors, colleagues and employees for making these years in the Bachelor of Computer Science the best ones in my life. I wish to remark a few professors that made my time here greater: Routo Terada, Carlinhos, Carlos Hitoshi and Renata Wassermann.

Now, most importantly, I thank with all my heart to my family for being comprehensive and supportive of all of my choices, and for never giving up on me after so many years of study.

Without any of you, I would not be able to achieve this. Thank you.

Resumo

Luiza Barros Reis Soezima. **Folding FrodoPIR: *Recuperação de informações privadas com otimização***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2024.

Recuperar informações de bancos de dados é uma atividade constante na rotina diária. Concomitantemente, a preocupação com a privacidade, especialmente no caso de dados sensíveis, desenvolve um problema paralelo. Protocolos de Recuperação de Informações (Private Information Retrieval - PIR) permitem a um usuário fazer o download de uma mensagem desejada de um conjunto de mensagens armazenadas em um banco de dados, sem revelar o índice da mensagem desejada para os bancos de dados.

Em outras palavras, PIR é um protocolo no qual, de um lado, um servidor possivelmente não confiável mantém um banco de dados público DB com N registros. Por outro lado, um cliente deseja consultar o registro $i \in \{0 \cdots N-1\}$, sem permitir que o servidor descubra o item consultado que ele está procurando (e, consequentemente, descubra o valor v associado ao i que ele está interessado). Uma solução ingênua envolve o cliente fazer o download de todo o DB, mas isso pode ser caro: o objetivo do PIR é tanto preservar a privacidade quanto ser mais eficiente do que o custo total de baixar todo o DB. Existem muitas soluções propostas para este problema, e neste Trabalho de Conclusão de Curso, exploraremos aquelas que utilizam a Criptografia Completamente Homomórfica (Fully Homomorphic Encryption - FHE) como primitiva criptográfica.

Palavras-chave: Recuperação de Informação Privada. Protocolos Criptográficos. Criptografia Homomórfica Aplicada.

Abstract

Luiza Barros Reis Soezima. **Folding FrodoPIR: *Private Information Retrieval with optimization***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2024.

Retrieving information from databases pulls a constant activity on the daily routine, concurrently, its privacy concern when it comes to sensitive data develop a parallel problem. Private information retrieval (PIR) is a privacy protocol that allows a user to download a required message from a set of messages stored in a database without revealing the index of the required message to the databases.

In other words, PIR is protocol in which from one side, a possibly untrusted server holds a public database DB with N records. On the other side, a client wants to query for record $i \in \{0 \cdots N-1\}$, without letting the server learn the queried item they are looking up (and, hence, learning the value v associated with i they are interested in). A naive solution involves the client locally downloading the whole DB, but that can be expensive: the goal of PIR is to both preserve privacy and be more efficient than the total cost of downloading the whole DB. There are many proposed solutions for this problem, and in this Capstone Project, we will explore the ones that uses Fully Homomorphic Encryption (FHE) as cryptographic primitive.

Keywords: Private Information Retrieval. Cryptographic Protocols. Applied Homomorphic Encryption.

List of Figures

List of Tables

[1] Image from (Wikipedia contributors, 2024a)

[2] Image from (Wikipedia contributors, 2024b)

[3] Image from (Wikipedia contributors, 2024b)

List of Programs

Contents

Introduction

The world we live in can be seen as a place where data is the most valuable asset. The amount of data generated every day is increasing exponentially, and the need to store, process and analyze this data is becoming a task each day more relevant and repetitive. However, sometimes the type of data that is being manipulated must not always available to everyone, and sometimes it is necessary to access, process and analyze it without revealing anything about the queries being done to the server. A few examples of this situation could be medical data, where the patient's privacy is a must, or even a situation in which a user would like to search for a specific news without letting the server know exactly what topic the user searched for. This is where Private Information Retrieval (PIR) comes in.

Private Information Retrieval (PIR) is a protocol firstly proposed by (B. Chor *et al.*, 1995) in which a user aims to retrieve data from a public database without revealing anything about the queries being done to the server. PIR construction involves an untrusted server that holds a public database DB with n records and a client that wants to query for record $i \in \{0 \cdots n-1\}$, without the server learning the index i they are querying for.

A simple and naive solution involves the client locally downloading the whole DB, but that can be expensive, just think about a situation in which a person may want to query a 1 MB data over a database of size 100GB. The goal of PIR is to both preserve privacy and be more efficient than the total cost of downloading the whole DB.

This protocol has an large list of applications, from anonymous communications at (Mittal *et al.*, 2011) to safe browsing at (Kogan and Corrigan-Gibbs, 2021). On a broader real life set of applications, we can explore a situation in which a user would like to search for a specific news without letting the server know exactly what topic the user searched for, or even a situation in which a user would like to browse something on a public system and wishes to have protection of the qualitative information of the data being searched.

Private Information Retrieval can be divided in two types of protocols: *Information Theoretic* (Beimel and Ishai, 2001) and *Computational Theoretic* (Benny Chor and Gilboa, 1997). The first one does not cores the solution into the hardness to solve a cryptographic problem and the computation used to do it, hence most solutions rely in multi-server systems. On the other hand, the second one emphasizes on the query encoding while manipulating records and consequently considering the limitation of a computational power to solve problems. Along with the types of the PIR schemes, there are the *stateless* ones in which the client does not store any data in order to launch queries and the *stateful* ones in which the server provides a digest used as a *preprocessing* phase of amortization. These definitions will be discussed in more details in the next chapters.

The computational cost associated with the Computational Private Information Retrieval (CPIR) is one of the bottleneck that barriers most of the practical solutions to the PIR problems. This means that if, for instace, a doctor desires to fetch a one single piece of information from a patient in a hospital database, the hospital servers would have to perform some work and tasks proportional to the total length of all of the patients in the hospital databas. This setup ends up being utterly costful not only on the theoretical point of view, but also on the practical point of view. On a more precise way, what this means is that the linear server-side cost is the a problem to explore in PIR schemes in theory. This is confirmmed by the nowadays approach that still keeps sublinear, or even polylogarithmic costs in the database size, this is explained at (Corrigan-Gibbs and Kogan, 2020).

On the verge of reducing the client side costs, the idea of *preprocessing* on the server phase generated schemes where the server respond to queries in sublinear time relative to the database size, which was first presente by Amos Beimel, Yuval Ishai and Tal Malkin at (Beimel, Ishai, and Malkin, 2000). On their approach, the client sends a request to the servers in an offline phase, yet, this still kept a lot of server-side storage.

In order to avoid the linear cost on the database size, most of the efficient solutions proposed for this problem are based on Fully Homomorphic Encryption (FHE) (Aguilar-Melchor *et al.*, 2014; Angel *et al.*, 2018; Mughees *et al.*, 2021; Davidson *et al.*, 2022; Henzinger *et al.*, 2022; Menon and Wu, 2022; Celi and Davidson, 2024), specifically the Ring-Learning With Errors(RLWE) and the Learning With Errors(LWE) assumptions in order to lower amortized server time and client storage (Corrigan-Gibbs, Henzinger, *et al.*, 2022).

A remark about the cryptographic primitive used in this work is that Lattice-based cryptography, particularly schemes built on the *Learning with Errors* (LWE) and *Ring Learning with Errors* (RLWE) problems, is widely regarded as post-quantum secure due to the inherent hardness of the underlying mathematical problems even in the presence of quantum computers.

This work will approach the LWE version of the CPIR, specifically the protocol developed by Alex Davidson, Gonçalo Pestaña and Sofía Celi at (Davidson *et al.*, 2022).

Objectives and Methodology

Objectives

The primary objective of this work is to explore and optimize the computational aspects of Private Information Retrieval (PIR) protocols, specifically focusing on the *FrodoPIR* protocol (Davidson *et al.*, 2022), which is based on the Learning with Errors (LWE) assumption. This exploration involves both theoretical analysis and practical implementation to identify and address bottlenecks in computational efficiency. The specific objectives of this thesis are as follows:

1. To investigate the Cryptographic Primitives used in the Private Information Retrieval (PIR) protocols, specifically the lattice base ones, such as Learning with Errors (LWE).

2. To analyze the computational costs associated with the *FrodoPIR* protocol and identify potential areas for optimization.
3. To understand and analyze the cryptographic foundations and operational mechanisms of the *FrodoPIR* protocol.
4. To identify opportunities for optimization, both theoretically and practically, to reduce computational overhead.
5. To design optimizations aimed at reducing query response time and storage requirements while maintaining correctness and security.
6. To evaluate the optimized protocol, comparing it against the baseline and existing state-of-the-art solutions.

Methodology

The methodology follows a structured approach, ensuring both rigor and clarity in addressing the objectives. The process is outlined below:

1. Literature Review and Theoretical Analysis

The thesis begins with an extensive review of existing PIR protocols, particularly computational PIR schemes based on the LWE assumption. The focus will be on understanding the theoretical construction of *FrodoPIR*, including its reliance on cryptographic primitives such as LWE, its preprocessing phase, and its query-response mechanism. Special attention will be given to the role of preprocessing in amortizing computational costs and the use of homomorphic encryption to enable efficient query handling.

This involves a detailed analysis of bottlenecks in the current *FrodoPIR* protocol, such as the computational costs of matrix-vector multiplications, the impact of noise growth in LWE-based operations, and the storage requirements for server-side preprocessing data.

2. Understanding the FrodoPIR (Davidson *et al.*, 2022) Protocol

The *FrodoPIR* protocol will be explored, as well as existing Stateful PIR Protocols. The goal is to understand the protocol's cryptographic foundations, as well as comprehend the operational mechanisms that enable private information retrieval and the existent bottlenecks to explore in the protocol.

This will establish a reference point for evaluating the effectiveness of subsequent optimizations.

3. Design of Optimizations

Based on insights gained from the theoretical analysis and baseline implementation, this work will propose optimizations to the *FrodoPIR* protocol. These optimizations will focus on:

- **Reducing computational overhead:** Techniques such as optimized matrix representation and arithmetic operations will be explored to accelerate matrix-vector multiplications.
- **Preprocessing strategies:** Enhancing the server-side preprocessing phase to minimize query processing time without significantly increasing storage requirements.
- **Efficient use of cryptographic primitives:** Investigating alternative homomorphic encryption schemes (e.g., Paillier encryption (Paillier, 1999)) to streamline computations.
- **Amortizing costs:** Proposing mechanisms to distribute computational costs across multiple queries.

4. Result Analysis and Conclusion

Finally, the results of the experimental evaluation will be analyzed to assess the effectiveness of the proposed optimizations. The analysis will include:

- A comparison of the optimized protocol with the baseline in terms of computational efficiency and scalability.
- An evaluation of the trade-offs introduced by the optimizations, particularly in terms of storage and communication costs.
- Suggestions for future research directions to further advance the field of computational PIR.

Chapter 1

Preliminaries

This work builds upon cryptographic protocols that require a strong understanding of the fundamental principles of cryptography, cryptographic primitives, and the mathematical problems that ensure their security. In this chapter, we introduce key concepts from cryptography, cryptographic protocols, and the mathematical foundations essential for understanding the work. Each section builds upon the previous, starting from general cryptographic principles and progressing to specific mathematical problems that are considered important to the security of the systems discussed afterward.

1.1 Cryptography

Cryptography is the science of securing communication and information through mathematical techniques (Stoneburner *et al.*, 2004; Ham, 2021). Its primary goals are to ensure confidentiality, integrity, authenticity, and non-repudiation in the exchange of information. Cryptographic systems are broadly categorized into symmetric and asymmetric schemes, both of which rely on computationally hard mathematical problems to provide security guarantees. On top of these two types of cryptography, we also have the post-quantum cryptography, which is a new field that aims to develop cryptographic schemes that are secure against quantum adversaries.[1]

1.1.1 Goals of Cryptography

The primary objectives of cryptography can be summarized in four main principles as are well discussed in (Stoneburner *et al.*, 2004; Ham, 2021):

- **Confidentiality:** Ensuring that information is only accessible to authorized parties.

[1] A quantum adversary refers to a theoretical entity capable of leveraging the principles of quantum computing to perform computations and attacks. Such an adversary is modeled to exploit the unique capabilities of quantum mechanics, including superposition and entanglement, to solve problems more efficiently than classical adversaries, particularly in the context of breaking cryptographic schemes. For example, a quantum adversary could use Shor's algorithm (Shor, 1994) to factorize large integers or compute discrete logarithms in polynomial time, potentially compromising classical cryptographic systems.

- **Integrity:** Protecting information from unauthorized modification or corruption.
- **Authentication:** Verifying the identity of the entities involved in communication.
- **Non-repudiation:** Preventing entities from denying their participation in a communication or transaction.

These goals form the foundation of cryptographic protocols and serve as benchmarks for evaluating the security and robustness of cryptographic systems.

1.1.2 Modern Cryptography: Symmetric and Asymmetric Cryptography

Modern cryptography is divided into two broad categories: symmetric and asymmetric cryptography. These paradigms differ in their approach to encryption, decryption, and key management, each offering unique advantages and trade-offs.

Definition 1 (Symmetric Cryptography)**.** *A cryptographic scheme is symmetric if encryption E_k and decryption D_k use the same secret key k. Formally:*

$$\forall P \in \mathcal{M},\ k \in \mathcal{K}, D_k(E_k(P)) = P$$

Symmetric encryption is computationally efficient and well-suited for applications requiring high-speed data encryption, such as AES (Advanced Encryption Standard) (Daemen and Rijmen, 1999) and DES (Data Encryption Standard) (Standards and (NIST), 1977). However, the need for secure key exchange between parties poses a challenge in distributed systems.

Definition 2 (Asymmetric Cryptography)**.** *A cryptographic scheme is asymmetric if it uses a pair of keys (k_{pub}, k_{priv}), where:*

$$D_{k_{priv}}(E_{k_{pub}}(P)) = P \quad \textit{and} \quad E_{k_{pub}}(D_{k_{priv}}(P)) = P.$$

Asymmetric encryption, such as RSA (Rivest *et al.*, 1978) and ECC (Elliptic Curve Cryptography) (Koblitz, 1994), eliminates the need for a shared secret key, making it ideal for secure communication over untrusted networks. However, it is computationally more intensive than symmetric cryptography.

Both paradigms are essential in modern cryptographic systems and are often used together in hybrid protocols, given each of them has strengths in different applications: symmetric encryption provides high efficiency for securing large amounts of data, while asymmetric encryption ensures secure key exchange and digital signatures.

1.2 Relevant Mathematical Concepts

Many cryptographic systems rely on hard mathematical problems to ensure their security. Lattice-based cryptography, in particular, has gained prominence for its resistance to quantum attacks. To understand the cryptographic schemes discussed in this work, it

is essential to introduce some key mathematical constructs, including lattices and their associated problems (Peikert, 2015), such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP).

1.2.1 Lattices

Lattices (view 1.1) provide a geometric framework for understanding many cryptographic problems. They are discrete structures in $\mathbb{R}^n$ formed by integer linear combinations of basis vectors. In other words, a lattice is a discrete subgroup of $\mathbb{R}^n$ formed by all integer linear combinations of a set of linearly independent vectors, called the basis vectors.

Definition 3 (Lattice). *A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^n$, defined as:*

$$\mathcal{L} = \left\{ \mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{b}_i \mid c_i \in \mathbb{Z},\ \mathbf{b}_i \in \mathbb{R}^n \right\},$$

where $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ are linearly independent vectors called the basis vectors. *Note that n vectors on $\mathbb{R}^n$ defining a lattice is a full-rank lattice. In general, we have m basis vectors in $\mathbb{R}^n$, with $m \leq n$*

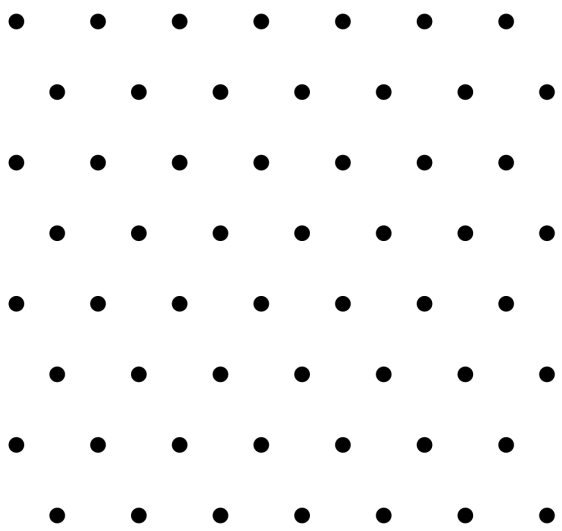


Figure 1.1: *An example of euclidean lattice*[a]

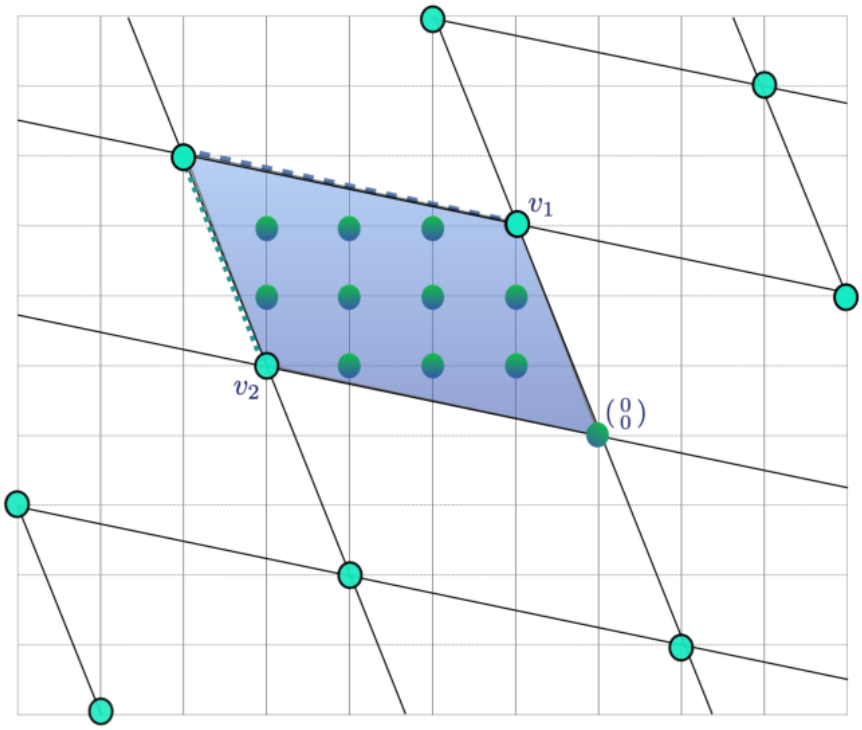
[a] Image from (Wikipedia contributors, 2024a)

Definition 4 (Dimension and Representation of a Lattice). *The* dimension *or* rank *of a lattice $\mathcal{L}$, denoted n, is the number of linearly independent basis vectors $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ spanning the lattice.*

Each point $\mathbf{x} \in \mathcal{L}$ can be uniquely represented as a linear combination of the basis vectors:

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{b}_i, \quad \text{where } c_i \in \mathbb{Z}.$$

The basis vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ define the geometric and algebraic structure of the lattice. They span a fundamental parallelepiped(view 1.2), which is the region formed by all linear combinations of the basis vectors with coefficients in $[0, 1)$. The volume of this parallelepiped, called the *determinant* of the lattice, is an invariant of the lattice, independent of the choice of basis.



a

Figure 1.2: *An example of a fundamental parallelepiped (half-open), for a lattice L, generated by the vectors v 1 and v 2*

[a] Image from (Robins, 2021)

An important property of lattices is that their structure remains invariant under any change of basis. While the basis vectors themselves are not unique, any basis for the same lattice generates the same set of points. This means that different sets of linearly independent vectors can describe the same lattice, provided they span the same discrete subset of $\mathbb{R}^n$. For example, if $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$ is a basis for a lattice $\mathcal{L}$, then $\mathbf{B}' = \mathbf{BU}$ is also a valid basis for $\mathcal{L}$, where $\mathbf{U}$ is an $n \times n$ unimodular matrix (i.e., a matrix with integer entries and determinant ± 1).

The invariance of the lattice under basis transformations is a crucial property in lattice-based cryptography. It allows algorithms to operate on different bases of the same lattice without changing the cryptographic properties. This flexibility also underpins the mathematical hardness of problems like the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP), as finding a *"good"* basis for a lattice (e.g., one with short, nearly orthogonal vectors) is computationally difficult.

1.2.2 Shortest Vector Problem (SVP)

The *Shortest Vector Problem (SVP)* is a fundamental computational problem in lattice theory. Given a lattice $\mathcal{L}$ defined by a basis $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ in $\mathbb{R}^n$, the goal of SVP is to find the shortest non-zero vector in the lattice with respect to a given norm, typically the Euclidean norm.

Definition 5 (Shortest Vector Problem (SVP))**.** *Given a lattice $\mathcal{L}$, the Shortest Vector Problem asks to find the shortest non-zero vector $\mathbf{v} \in \mathcal{L}$ under a given norm $\|\cdot\|$:*

$$\mathbf{v} = \arg \min_{\mathbf{u} \in \mathcal{L} \setminus \{0\}} \|\mathbf{u}\|.$$

The shortest vector $\mathbf{v}$ represents the smallest distance from the origin to any lattice point, excluding the origin itself (view 1.3). The Euclidean norm $\|\mathbf{v}\|$ of a vector $\mathbf{v} = (v_1, v_2, \dots, v_n)$ is given by:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}.$$

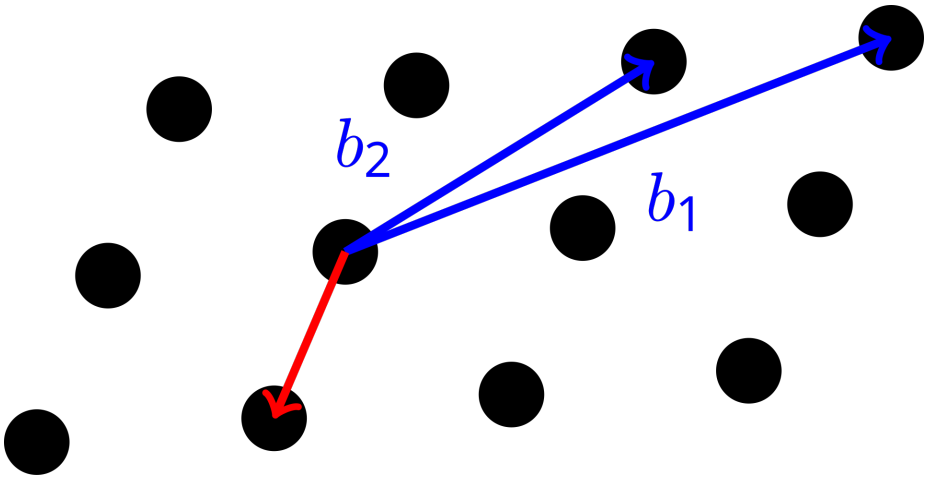


Figure 1.3: *An example of the SVP problem (basis vectors in blue, shortest vector in red)*[a]

[a] Image from (Wikipedia contributors, 2024b)

SVP is widely studied in cryptography because its computational hardness serves as the basis for many lattice-based cryptographic schemes. Finding the exact solution to SVP is known to be NP-hard in general (Micciancio and Goldwasser, 2002), and approximating it within small polynomial factors is also believed to be computationally intractable (Ajtai, 1996; Haviv and Regev, 2007). This difficulty persists even under quantum computing models, making SVP a foundation of post-quantum cryptography.

The challenge of solving SVP lies in the fact that the lattice basis vectors provided as input are typically not the shortest or the most orthogonal. A *"good"* basis with shorter, nearly orthogonal vectors would make SVP easier to solve, but obtaining such a basis is itself computationally hard.

In cryptographic applications, the hardness of SVP ensures the security of protocols by making it infeasible for adversaries to recover secret information encoded in lattice structures.

1.2.3 Closest Vector Problem (CVP)

The *Closest Vector Problem (CVP)* is another fundamental problem in lattice theory. Given a lattice $\mathcal{L}$ defined by a basis $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ in $\mathbb{R}^n$ and a target vector $\mathbf{t} \in \mathbb{R}^n$, the goal of CVP is to find the lattice vector $\mathbf{v} \in \mathcal{L}$ that is closest to the target vector $\mathbf{t}$ with respect to a given norm.

Definition 6 (Closest Vector Problem (CVP))**.** *Given a lattice $\mathcal{L}$ and a target vector $\mathbf{t} \in \mathbb{R}^n$, the Closest Vector Problem asks to find $\mathbf{v} \in \mathcal{L}$ such that:*

$$\mathbf{v} = \arg\min_{\mathbf{u} \in \mathcal{L}} \|\mathbf{t} - \mathbf{u}\|.$$

The distance $\|\mathbf{t} - \mathbf{v}\|$ represents the minimum Euclidean distance between the target vector $\mathbf{t}$ and any lattice point in $\mathcal{L}$. Unlike SVP, which focuses on the intrinsic properties of the lattice, CVP involves an external target vector and measures proximity to lattice points.

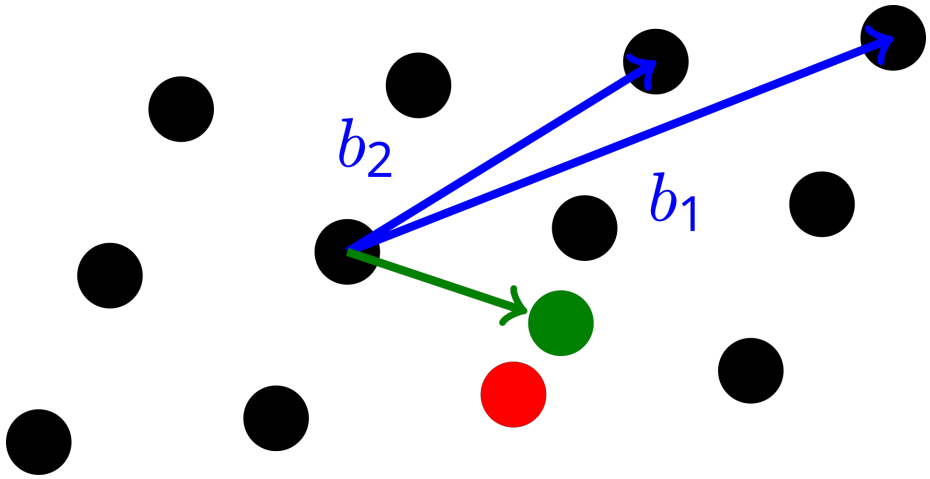


Figure 1.4: *An example of the CVP problem (basis vectors in blue, external vector in green, closest vector in red)*[a]

[a] Image from (Wikipedia contributors, 2024b)

CVP (view 1.4) is known to be at least as hard as SVP (Micciancio and Goldwasser, 2002), and in many cases, it is considered more computationally challenging. Like SVP, CVP is NP-hard and remains difficult to approximate within polynomial factors. Solving CVP is particularly challenging when the lattice basis vectors are not orthogonal, as this makes it harder to distinguish the closest lattice point to the target.

For the cryptographic contexts, CVP is significant because many lattice-based cryptographic constructions reduce to instances of CVP. And yhe computational hardness of CVP underpins the security of lattice-based cryptography, ensuring resilience against both classical and quantum attacks.

1.2.4 Decisional Composite Residuosity Assumption (DCRA)

The Decisional Composite Residuosity Assumption (DCRA) is a foundational hardness assumption in number theory and cryptography. It underpins the security of Paillier cryptosystem ((Paillier, 1999)) by asserting the computational infeasibility of deciding whether a given element is an n-residue modulo n^2.

Let $n = p \cdot q$ be a product of two large prime numbers p and q, where $p \neq q$, and consider the ring $\mathbb{Z}_{n^2}^*$, the set of integers modulo n^2 that are coprime to n^2. Define the subgroup $G_n \subseteq \mathbb{Z}_{n^2}^*$ as:

$$G_n = \{x \in \mathbb{Z}_{n^2}^* \mid x = (1+n)^r \cdot u^n \mod n^2,\ r \in \mathbb{Z}_n, u \in \mathbb{Z}_n^*\}.$$

The subgroup G_n contains the so-called n-residues modulo n^2, which are elements of $\mathbb{Z}_{n^2}^*$ that can be expressed in this form.

Definition 7 (Decisional Composite Residuosity Assumption (DCRA))**.** *Let $n = p \cdot q$ be a composite number with p and q as large primes. The DCRA asserts that, given $x \in \mathbb{Z}_{n^2}^*$, it is computationally infeasible to decide whether x is an n-residue modulo n^2 without knowledge of the factorization of n.*

The Decisional Composite Residuosity Assumption states that for a randomly chosen $x \in \mathbb{Z}_{n^2}^*$, it is computationally infeasible to decide whether $x \in G_n$ without knowledge of the factorization of n. In other words, no efficient algorithm exists that can distinguish between the following two distributions:

1. x chosen uniformly at random from G_n.
2. x chosen uniformly at random from $\mathbb{Z}_{n^2}^*$.

Note that the DCRA is closely related to the Integer Factorization Problem, which involves finding the prime factors p and q of $n = p \cdot q$. Knowledge of the factorization of n directly allows efficient membership testing for G_n. Conversely, solving the DCRA efficiently could compromise the factorization of n. Hence, the security of the Paillier cryptosystem (Paillier, 1999) , which relies on DCRA, is equivalent to the hardness of integer factorization in practice.

1.3 Hardness of Computational Problems in Cryptography

Another important topic to clarify is the hardness of computational problems in cryptography, as well as how they correlate to each other. This is what defines the core of the security of cryptographic protocols and gives directions to which approach to take when designing a new cryptographic system. The security of cryptographic protocols relies heavily on the hardness of underlying mathematical problems. These problems are designed to be computationally infeasible for adversaries to solve within a reasonable amount of time.

This section gives a brief overview of the hardness of lattice-based problems and compares them to classical cryptographic problems, such as the Discrete Logarithm Problem (DLP) and Integer Factorization.

1.3.1 Considerations about NP-Hardness

In computational complexity theory, a problem is considered *NP-hard* if solving it efficiently would allow us to solve all problems in the complexity class *NP* efficiently. Formally, NP-hardness means that every problem in NP can be reduced to the given problem in polynomial time.

- **NP (Nondeterministic Polynomial Time):** This class consists of decision problems for which a solution can be verified in polynomial time by a deterministic Turing machine. For example, given a solution to the Boolean satisfiability problem (SAT), it is straightforward to check its correctness in polynomial time.
- **NP-hard Problems:** A problem is NP-hard if it is at least as hard as the hardest problems in NP. It does not necessarily need to be in NP itself (i.e., it may not have a solution that can be verified in polynomial time).
- **Significance:** NP-hard problems are widely regarded as computationally infeasible to solve in the general case, as no polynomial-time algorithms are known for them.

In cryptography, computationally *hard* problems are fundamental for designing secure systems. The security of cryptographic schemes often relies on the assumption that solving these problems efficiently is infeasible for adversaries. While some problems, such as Integer Factorization or the GapSVP problem (this problem be further mentioned), are believed to be computationally intractable, they are not proven to be NP-hard. Instead, their hardness assumptions are based on the lack of efficient algorithms to solve them within classical or quantum frameworks.

1.3.2 Classical Cryptographic Problems

Before the advent of quantum computing, classical cryptographic systems relied on the hardness of problems such as:

- **Discrete Logarithm Problem (DLP):** Given a group G and elements $g, h \in G$, the problem is to find an integer x such that $g^x = h$. The hardness of DLP underpins the security of protocols like Diffie-Hellman and Digital Signature Algorithm (DSA).
- **Integer Factorization Problem:** The problem of decomposing a composite number n into its prime factors is the basis of the RSA cryptosystem.
- **Elliptic Curve Discrete Logarithm Problem (ECDLP):** A variant of DLP defined over elliptic curves, providing stronger security per bit compared to DLP.

While these problems are considered hard for classical computers, they become solvable in polynomial time on quantum computers using algorithms such as Shor's algorithm. This potential vulnerability has prompted the search for post-quantum cryptographic primitives.

1.3.3 Lattice-Based Problems

Lattice-based problems are at the core of post-quantum cryptography due to their computational hardness, even in the presence of quantum adversaries. Two fundamental problems in this domain are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) that were mentioned earlier, along with the Learning With Errors (LWE) problem.

- **Shortest Vector Problem (SVP):** The problem of finding the shortest non-zero vector in a lattice is known to be NP-hard in the exact case and remains computationally intractable for approximate solutions within small factors.
- **Closest Vector Problem (CVP):** CVP, which involves finding the lattice vector closest to a target point, is at least as hard as SVP and is also NP-hard. Its hardness persists even under approximate settings.
- **Learning With Errors (LWE):** LWE is a problem that is as hard as worst-case instances of lattice problems like GapSVP (approximating SVP). It is computationally hard for both classical and quantum adversaries and forms the foundation of many lattice-based cryptographic schemes. This problem will be further discussed in the section 3[2]
- **Ring Learning With Errors (RLWE):** RLWE is a variant of LWE defined over polynomial rings.

These problems derive their hardness from the geometric complexity of high-dimensional lattices, where efficiently solving SVP or CVP requires exponential time in the lattice dimension. This makes them ideal for constructing cryptosystems with strong security guarantees.

We will explore with more details the LWE and RLWE problems in the next sections.

1.3.4 Comparison Between Lattice-Based and Classical Problems

Lattice-based cryptography offers significant advantages in the post-quantum era:

- Problems like LWE and RLWE remain hard for quantum computers, whereas classical problems like DLP and Integer Factorization can be efficiently solved using quantum algorithms.
- Lattice problems are versatile and support advanced cryptographic functionalities such as Fully Homomorphic Encryption (FHE) and Private Information Retrieval (PIR).
- Approximate solutions to lattice problems, such as γ-SVP, provide computational hardness even for moderate γ factors, ensuring robust security.

[2] The Gap Shortest Vector Problem (GapSVP) is a decision version of the SVP, where the goal is to determine whether the shortest vector in a given lattice is smaller than a threshold or larger than another threshold. The problem remains computationally hard for approximation factors smaller than $2^{\mathcal{O}(n)}$, where n is the lattice dimension, under both classical and quantum settings.

Chapter 2

Quantum Camputation and Post-Quantum Cryptography

This chapter provides an overview of quantum computation and its implications for cryptography, speciall when it comes to our choices on using the Learning With Errors(LWE) problem as a base for the cryptographic schemes on Private INformation Retrieval.

The emergence of quantum computers poses a significant challenge to the field of cryptography. Many widely used cryptographic protocols rely on the computational intractability of problems like the Discrete Logarithm Problem (DLP) and Integer Factorization, which can be solved efficiently by quantum algorithms. This section introduces the basics of quantum computation, its implications for cryptography, and the development of post-quantum cryptographic schemes that remain secure against quantum adversaries.

2.1 Quantum Computation

Quantum computation (Nielsen and Chuang, 2010) explores the principles of quantum mechanics to process information. Unlike classical computers, which use binary states (0 and 1), quantum computers use *qubits*, which can exist in a superposition of states. This allows quantum computers to perform certain computations exponentially faster than their classical counterparts.

2.1.1 Qubit

A *qubit* (quantum bit) is the basic unit of information in quantum computation, analogous to the classical bit. However, unlike a classical bit, which can exist only in one of two states, $|0\rangle$ or $|1\rangle$, a qubit can exist in a *superposition* of both states simultaneously. Mathematically, the state of a qubit is represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers satisfying the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1.$$

Here, $|0\rangle$ and $|1\rangle$ are the computational basis states of the qubit, and $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit in the states $|0\rangle$ and $|1\rangle$, respectively.

The ability of a qubit to exist in a superposition of states is a fundamental property that enables quantum computers to perform computations on multiple possibilities simultaneously, vastly increasing their computational power for specific tasks.

2.2 Impact of Quantum Computing on Classical Cryptography

The computational power of quantum computers threatens the security of classical cryptographic systems, as several quantum algorithms can solve problems that underpin traditional cryptographic protocols efficiently:

- **Shor's Algorithm (Shor, 1994):** This algorithm efficiently solves the Integer Factorization problem and the Discrete Logarithm Problem in polynomial time. Consequently, cryptosystems like RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC) (Koblitz, 1994) are vulnerable to quantum attacks.
- **Grover's Algorithm (Grover, 1996):** This algorithm provides a quadratic speedup for unstructured search problems, reducing the security of symmetric cryptographic schemes like AES (Daemen and Rijmen, 1999) by effectively halving the key length. For example, AES-256 provides only 128 bits of security against a quantum adversary.

These algorithms created the urgent need for cryptographic systems that could be efficient against quantum adversaries, and after this the cryptographic community has shifted its focus towards developing post-quantum cryptographic schemes that are secure against quantum attacks, leading to the development of what we call post-quantum cryptography.

2.3 Post-Quantum Cryptography

Post-quantum cryptography refers to cryptographic algorithms designed to remain secure against both classical and quantum adversaries. These algorithms are based on mathematical problems believed to be resistant to quantum attacks. The main classes of post-quantum cryptographic primitives include:

- **Lattice-Based Cryptography:** Based on the hardness of problems like Learning With Errors (LWE) and Ring Learning With Errors (RLWE) (Peikert, 2015), lattice-based schemes are among the most promising candidates for post-quantum cryptography. They enable advanced functionalities such as Fully Homomorphic Encryption (FHE) and serve as the foundation for protocols like FrodoPIR.

- **Code-Based Cryptography:** Relies on the hardness of decoding random linear codes. The McEliece (McEliece, 1978) cryptosystem is a prominent example, offering security against quantum attacks.
- **Isogeny-Based Cryptography:** Uses the hardness of finding isogenies between elliptic curves. A notable example is the SIKE(Supersingular Isogeny Key Encapsulation) (J. D. Feo *et al.*, 2019; Jao and L. D. Feo, 2011) protocol.

Chapter 3

Cryptographic Primitives

Cryptographic primitives are the foundational components upon which secure cryptographic protocols and systems are built. They provide essential functionalities like encryption, decryption, key exchange, and secure computation. This section describes the cryptographic primitives relevant to this work, focusing on their mathematical definitions and the specific needs of Folding Frodo PIR.

3.1 Learning With Errors (LWE)

The Learning With Errors (LWE) (Peikert, 2015) problem is a central cryptographic primitive in lattice-based cryptography. It provides the mathematical foundation for constructing secure schemes resistant to both classical and quantum adversaries. LWE exists in two main variants: the *Search LWE* and the *Decisional LWE*:

Definition 8 (Search LWE:)**.** *The Search LWE problem involves recovering a secret vector* $\mathbf{s} \in \mathbb{Z}_q^n$ *from a system of noisy linear equations. Formally:*

$$\textit{Given } (\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \mod q), \textit{ recover } \mathbf{s},$$

where:

- $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ *is a public random matrix,*
- $\mathbf{b} \in \mathbb{Z}_q^m$ *is a vector of noisy linear combinations,*
- $\mathbf{e} \in \mathbb{Z}_q^m$ *is a noise vector, typically sampled from a discrete Gaussian distribution with small variance.*

Definition 9 (Decisional LWE:)**.** *The Decisional LWE problem asks whether a given vector* $\mathbf{b} \in \mathbb{Z}_q^m$ *is a noisy linear combination of the rows of* $\mathbf{A}$ *or is uniformly random. Formally:*

$$\textit{Given } (\mathbf{A}, \mathbf{b}), \textit{ distinguish whether } \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \mod q \textit{ or } \mathbf{b} \sim \mathcal{U}(\mathbb{Z}_q^m),$$

where $\mathcal{U}(\mathbb{Z}_q^m)$ *denotes the uniform distribution over* $\mathbb{Z}_q^m$*.*

The hardness of both Search and Decisional LWE is rooted in worst-case reductions from lattice problems such as the Shortest Vector Problem (SVP) (Regev, 2009) These properties make LWE a versatile primitive for constructing cryptographic schemes like Fully Homomorphic Encryption (FHE), Key Exchange, and, therefore, Private Information Retrieval.

3.2 Ring Learning With Errors (RLWE)

The Ring Learning With Errors (RLWE) problem is an optimization of LWE, operating over polynomial rings to achieve greater efficiency. Let $R = \mathbb{Z}[x]/\langle f(x)\rangle$ be the ring of polynomials modulo an irreducible polynomial $f(x)$ of degree n, and let q be a modulus.

Definition 10 (Ring Learning With Error (RLWE))**.** *The RLWE problem is defined as:*

$$\textit{Given } (\mathbf{a}(x), \mathbf{b}(x) = \mathbf{a}(x) \cdot \mathbf{s}(x) + \mathbf{e}(x) \mod q), \textit{ recover } \mathbf{s}(x),$$

where:

- $\mathbf{a}(x) \in R_q = R/qR$ *is a public random polynomial,*
- $\mathbf{s}(x), \mathbf{e}(x) \in R_q$ *are the secret and noise polynomials, respectively.*

The decision variant of RLWE involves distinguishing whether $\mathbf{b}(x)$ is of the form above or is uniformly random in R_q.

3.3 Homomorphic Encryption

Homomorphic Encryption (HE) is a cryptographic primitive that allows computations to be performed on encrypted data without the need for decryption. This property is essential for privacy-preserving computations, as it enables secure operations on sensitive data while maintaining confidentiality.

Homomorphic Property: For two ciphertexts c_1 and c_2 encrypting m_1 and m_2, respectively, their product modulo n^2 decrypts to the sum of the plaintexts:

$$c_1 \cdot c_2 \mod n^2 \text{ decrypts to } m_1 + m_2 \mod n.$$

3.3.1 Paillier Cryptosystem

The Paillier cryptosystem (Paillier, 1999) is an additive homomorphic encryption scheme based on the Decisional Composite Residuosity Assumption (DCRA). Its key features include the ability to add plaintexts by multiplying their ciphertexts and strong semantic security under standard assumptions.

The setting of the Pailler can be seen as the following:

Key Operations:

- **Key Generation:** Choose two large primes p and q, compute $n = p \cdot q$, and set the public key (n, g) and private key $\lambda = \text{lcm}(p-1, q-1)$.
- **Encryption:** Encrypt a message $m \in \mathbb{Z}_n$ using the public key as:

$$c = g^m \cdot r^n \mod n^2,$$

  where $r \in \mathbb{Z}_n^*$ is a random value.
- **Decryption:** Decrypt a ciphertext c using the private key as:

$$m = \frac{(c^\lambda \mod n^2) - 1}{n} \cdot (g^\lambda \mod n^2)^{-1} \mod n.$$

3.3.2 Fully Homomorphic Encryption (FHE)

Whenever we talk about privacy-preserving computations, one might think that, to protect our data, we just need to encrypt it and send it to the server or the cloud. In the cloud context, the performance of search, modification and operations among datas is constant, and naïvely, for that, one can just download the data, decrypt it, perform the operation and encrypt it again. However, this is not secure, as the data is actually totally exposed during the operation given that the server needs to perform operations on the encrypted data.

In order to bypass this problem, we can perform operations called homomorphic operations, which are operations that can be performed on encrypted data without the need of decrypting it. This is the main idea behind Fully Homomorphic Encryption (FHE), which allows arbitrary computations on encrypted data without decryption. Therefore, homomorphic operations are a fundamental property of cryptographic schemes for the current context of cloud computing.

Before exploring what does Homomorphic Encryption mean, let's first understand what is *Homomorphism* of a cipher.

Definition 11 (Homomorphic Cipher)**.** *A cipher is said to be* homomorphic *if it satisfies the following property:*

Let $\oplus$ and $\otimes$ represent operations (e.g., addition and multiplication) defined over the plaintext space. A cipher is homomorphic if, for any two plaintexts m_1 and m_2, and their respective ciphertexts $c_1 = Enc(m_1)$ and $c_2 = Enc(m_2)$, there exists an efficient operation $\star$ in the ciphertext space such that:

$$Enc(m_1 \oplus m_2) = c_1 \star c_2,$$

where $\star$ is performed directly on c_1 and c_2 without knowledge of the secret key used for encryption.

The consequence of this property is that, given the ciphertexts c_1 and c_2, we can perform operations on them without the need of decrypting them. We can extent this property to $l \in \mathcal{N}$ plaintexts, such that:

$$\text{Enc}(m_1 \oplus m_2 \oplus \ldots \oplus m_l) = c_l.$$

Fully Homomorphic Encryption (FHE) is an extension of Homomorphic Encryption that enables secure computations over encrypted data. Unlike partially homomorphic encryption, which supports only a single operation (e.g., addition or multiplication), FHE supports both addition and multiplication, allowing for arbitrary computations on encrypted data.

Definition 12 (Fully Homomorphic Encryption (FHE))**.** *Let* Enc *denote the encryption function, and* $\oplus$ *and* $\otimes$ *represent addition and multiplication in the plaintext space, respectively. The scheme is Fully Homomorphic if, for any plaintexts* $m_1, m_2, \ldots, m_l$ *and their respective ciphertexts* $c_i = \mathsf{Enc}(m_i)$ *for* $i = 1, \ldots, l$*, there exist efficient operations* $\star$ *and* $\circ$ *in the ciphertext space such that:*

$$\mathsf{Enc}(m_1 \oplus m_2 \oplus \ldots \oplus m_l) = c_1 \star c_2 \star \ldots \star c_l,$$

and

$$\mathsf{Enc}(m_1 \otimes m_2 \otimes \ldots \otimes m_l) = c_1 \circ c_2 \circ \ldots \circ c_l.$$

These operations $\star$ *and* $\circ$ *can be performed directly on the ciphertexts without requiring decryption or knowledge of the secret key, ensuring the privacy of both the queries and the processed data.*

The story the the homomorphic encryption begins with Rivest, Shamir and Adleman's RSA (Rivest *et al.*, 1978) back in 1977, which is a partially homomorphic encryption scheme, as it allows only multiplication of ciphertexts. Later in 2005, Dan Boneh, Eu-Jin Goh and Kobbi Nissim proposed at (Boneh *et al.*, 2005) the first fully homomorphic encryption scheme, which was based on the Learning With Errors (LWE) problem, yet it was not practical for real-world applications, since it could only perform a limited amount of operations.

Only in 2009, 30 years after RSA, Craig Gentry proposed at (Gentry, 2009) a fully homomorphic encryption scheme based on ideal lattices. This scheme was the first to be practical for real-world applications, later being optimized by several researchers, and it is the basis for the current Homomorphic Encryption (HE), Somewhat Homomorphic Encryption (SHE) and Fully Homomorphic Encryption (FHE) schemes. The development of the FHE until an applicable state-of-the-art is largelly explored at (Chillotti, 2018).

3.3.3 Fully Homomorphic Encryption Schemes

There are several Fully Homomorphic Encryption (FHE) schemes, each with its own properties and trade-offs. The most well-known FHE schemes are the BGV, FV, and DGHV cryptosystems, each with its own set of operations and optimizations. In this section, an overview of these schemes and their key operations will be explored:

BGV Cryptosystem

The BGV(Brakerski-Gentry-Vaikuntanathan) cryptosystem (Brakerski *et al.*, 2014) is a fully homomorphic encryption (FHE) scheme based on the Learning With Errors (LWE) or Ring-LWE hardness assumption. It operates over a polynomial ring $R = \mathbb{Z}[X]/(X^d + 1)$ where d is a power of 2.

Definition 13 (BGV Cryptosystem)**.** *The BGV cryptosystem consists of the following key operations:*

- ***Key Generation:*** *Select a secret key $s \in R_q$, where q is a modulus, and generate public keys as noisy polynomials.*
- ***Encryption:*** *To encrypt a plaintext $m \in R_t$ (with t as the plaintext modulus):*

$$c = (a, b) = (a, a \cdot s + e + m \mod q),$$

  where $a \in R_q$ is uniformly random, and e is a small error polynomial.
- ***Decryption:*** *Given a ciphertext $c = (a, b)$, decrypt as:*

$$m = (b - a \cdot s) \mod t.$$

For the *Homomorphic Operations* in the BGV scheme, we have:

- **Addition:** Ciphertexts $c_1 = (a_1, b_1)$ and $c_2 = (a_2, b_2)$ can be added as:

$$c_{\text{sum}} = (a_1 + a_2, b_1 + b_2) \mod q.$$

- **Multiplication:** Ciphertexts c_1 and c_2 can be multiplied as:

$$c_{\text{prod}} = (a_1 a_2, a_1 b_2 + a_2 b_1, b_1 b_2),$$

  where the result is a "noisier" ciphertext requiring modulus switching to manage the noise growth.

What is important to notice about the BGV, is that it uses *modulus switching*, which scales down the modulus q to reduce noise while maintaining correctness.

FV Cryptosystem

The FV(Fan-Vercauteren) cryptosystem (Fan and Vercauteren, 2012) is another FHE scheme. It operates on batched plaintexts encoded as polynomials, enabling efficient computation on multiple data elements.[1]

Definition 14 (FV Cryptosystem)**.** *The FV cryptosystem is defined as follows:*

- ***Key Generation:*** *Generate a secret key $s \in R_q$. The public key is a pair (b, a) where:*

$$b = -(a \cdot s + e) \mod q,$$

  and e is a small error polynomial.

[1] Batching is a technique that allows multiple plaintexts to be packed into a single ciphertext, enabling parallel computation on these plaintext slots. For example, in the FV scheme, plaintexts are represented as polynomials modulo a carefully chosen cyclotomic polynomial, which provides N plaintext slots for batching operations.

- ***Encryption:*** *Encrypt a plaintext $m \in R_t$ as:*

$$c = (c_0, c_1) = (b \cdot u + m + e_1, a \cdot u + e_2) \mod q,$$

where u is uniformly random in R_q, and e_1, e_2 are error terms.

- ***Decryption:*** *Given a ciphertext $c = (c_0, c_1)$, compute:*

$$m = [c_0 + c_1 \cdot s] \mod t.$$

For the *Homomorphic Operations* in the FV scheme, we have:

- **Addition:** Add ciphertexts component-wise:

$$c_{\text{sum}} = (c_0^{(1)} + c_0^{(2)}, c_1^{(1)} + c_1^{(2)}) \mod q.$$

- **Multiplication:** Multiply ciphertexts and use relinearization to reduce the size of the resulting ciphertext:

$$c_{\text{prod}} = (c_0 \cdot c_0', c_0 \cdot c_1' + c_1 \cdot c_0', c_1 \cdot c_1').$$

The FV cryptosystem uses *rescaling* to reduce the modulus and control noise growth during homomorphic multiplications.

DGHV Cryptosystem

The DGHV (Van Dijk-Gentry-Halevi-Vaikuntanathan) cryptosystem (Dijk *et al.*, 2010) is one of the earliest FHE schemes designed to work over integers. Unlike lattice-based schemes, DGHV operates on plaintexts and ciphertexts that are integers modulo a secret key. Its simplicity makes it conceptually straightforward but less efficient than modern schemes.

Definition 15 (DGHV Cryptosystem)**.** *The DGHV cryptosystem is defined as follows:*

- ***Key Generation:*** *Choose a large integer p (the secret key) and a public modulus q. Publish integers $x_i = q_i \cdot p + r_i$, where $r_i \ll p$ are small random errors.*

- ***Encryption:*** *Encrypt a bit $m \in \{0, 1\}$ as:*

$$c = m + 2 \cdot r + q \cdot p,$$

where r is a small random value.

- ***Decryption:*** *Decrypt a ciphertext c as:*

$$m = (c \mod p) \mod 2.$$

For the *Homomorphic Operations* in the DGHV scheme, we have:

- **Addition:** Add ciphertexts:

$$c_{\text{sum}} = c_1 + c_2.$$

- **Multiplication:** Multiply ciphertexts:

$$c_{\text{prod}} = c_1 \cdot c_2.$$

One this that is extremelly important to underline on the DGHV scheme is that requires bootstrapping to manage noise growth during computations, which will be explained in the next section.

Bootstrapping in Fully Homomorphic Encryption

Bootstrapping (Gentry, 2009) is a noise management technique that refreshes ciphertexts, enabling arbitrarily deep computations in FHE. In FHE schemes, each homomorphic operation increases the noise in the ciphertext. Once the noise grows beyond a certain threshold, decryption fails, or in other words, if the noise is too high, the decryption of the ciphertext will not return the correct plaintext. Bootstrapping reduces this noise by homomorphically decrypting a ciphertext, effectively refreshing it.

The bootstrapping process involves three main steps:

1. **Encrypt the Secret Key:** The encryption of the secret key sk.
2. **Homomorphic Decryption:** Homomorphically evaluate the decryption function $D(c, sk)$ over an encrypted ciphertext c and encrypted secret key sk.
3. **Output Refreshed Ciphertext:** The output is a refreshed ciphertext c' with reduced noise, which can be used for further operations.

Example: Bootstrapping in the DGHV Scheme: The DGHV cryptosystem provides a simple example of bootstrapping on the cyphering scheme:

- Given a ciphertext c encrypting m, the noise e in c is:

$$c = m + t \cdot q + e,$$

  where q is a quotient and e is the noise.

- To bootstrap, compute the decryption homomorphically:

$$m = (c \mod t),$$

  using the encrypted secret key sk stored in the system.

- Re-encrypt the result m to produce a refreshed ciphertext c' with reduced noise.

A better understanding of bootstrapping and FHE schemes can be found in (Pereira and Morais, 2021).

Chapter 4

Cryptographic Protocols

4.1 Private Information Retrieval

Private Information Retrieval (PIR) protocols enable clients to retrieve data from a database while ensuring their query remains private. This chapter explores the foundational concepts of PIR, its variants, and the cryptographic techniques they use. PIR schemes are broadly categorized into Information-Theoretic Private Information Retrieval (IITPIR) and Computational Private Information Retrieval (CPIR), based on their security assumptions.

Private Information Retrieval (PIR) is a cryptographic protocol that allows a client to retrieve an element v_i from a database $DB = \{v_0, v_1, \dots, v_{n-1}\}$, where n is the total number of records in this database, without revealing the index i of the desired element to the database server. Formally:

Definition 16 (Private Information Retrieval (PIR))**.** *Let the database be represented as a function:*

$$DB : \{0, 1, \dots, n-1\} \to \mathcal{V},$$

where $\mathcal{V}$ is the set of database records. The goal is to design an interactive protocol between a client $\mathcal{C}$ and server(s) $\mathcal{S}$ such that:

$$\mathcal{C}(\mathcal{S}(DB, Q)) = v_i, \quad \forall i \in \{0, 1, \dots, n-1\},$$

where Q is the client's query, and $\mathcal{S}(DB, Q)$ is the server's response.

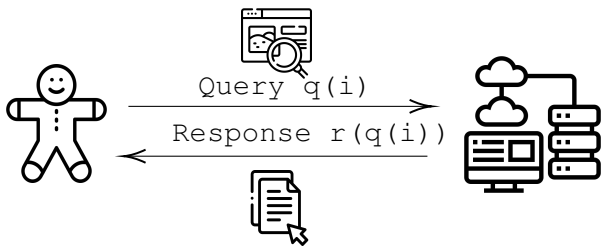


Figure 4.1: *Private Information Retrieval Protocol.*

From (Benny Chor, Goldreich, *et al.*, 1998; Benny Chor and Gilboa, 1997; Corrigan-Gibbs and Kogan, 2020; Beimel and Ishai, 2001), ideally, PIR protocols ensures two main characteristics:

- **Query Privacy:** The server (or servers) learns no information about the client's query index i.
- **Correctness:** The client retrieves v_i accurately without additional errors.

The costs of PIR protocols is measured in terms of:

- **Communication Complexity:** The total number of bits exchanged between the client and server(s).
- **Computation Complexity:** The computational effort required by the client and server(s) to generate and process queries.

4.2 Informational Theoretical Private Information Retrieval

Informational Theoretical Private Information Retrieval (ITPIR) provides unconditional query privacy by distributing the database among $k \geq 2$ non-colluding servers. Non-collusion means that the servers do not share information about their interactions with the client.

Definition 17 (Informational Theoretical Private Information Retrieval (ITPIR))**.** *Assume the database DB is replicated across k non-colluding servers $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$. The client generates k queries $Q_1, Q_2, \ldots, Q_k$ such that:*

$$\sum_{j=1}^{k} Q_j(i) = i, \quad \text{and } Q_j(i) \text{ is uniformly random for each } j.$$

Each server $\mathcal{S}_j$ independently processes its query Q_j and computes:

$$R_j = \mathcal{S}_j(DB, Q_j),$$

where R_j is the partial response from server $\mathcal{S}_j$. The client combines all responses $\{R_1, R_2, \ldots, R_k\}$ to recover v_i.

The ITPIR protocol achieves the following properties and guarantees:

- **Unconditional Privacy:** Since each query Q_j is independently random and uniformly distributed, no single server $\mathcal{S}_j$ can deduce the index i.
- **Correctness:** The summation of responses $\sum_{j=1}^{k} R_j$ guarantees the correct reconstruction of v_i.

Yet, it is important to note that ITPIR is not practical due to its high communication and computation complexity, meaning that while ITPIR provides strong privacy guarantees, it suffers from:

- **High Communication Overhead:** The client interacts with k servers, requiring significant communication.
- **Replication Requirement:** All k servers must hold identical copies of the database.

(Benny Chor and Gilboa, 1997) introduced the first formal ITPIR protocols, demonstrating the potential for unconditional privacy in multi-server settings. Their work laid the first assumptions for the use of random splitting techniques, where the query is distributed among servers such that no individual server gains information about the queried index. (Benny Chor, Goldreich, *et al.*, 1998) further refined these protocols, emphasizing the need for replication of the database across servers. This approach ensures that each server operates independently, preserving privacy through non-collusion.

On the verge of simplifying the ITPIR protocols, (Beimel and Ishai, 2001) presented a unified construction of ITPIR protocols, simplifying the design by formalizing the mathematical foundations of query splitting and response aggregation. Their work highlights the trade-off between communication cost and the number of non-colluding servers.

4.3 Computational Private Information Retrieval

Computational Private Information Retrieval (CPIR) is an approach that achieves query privacy with a single server, relying on computational hardness assumptions such as the Learning With Errors (LWE) in order to create computationally efficient PIR protocols.

Definition 18 (Computational Private Information Retrieval)**.** *In CPIR, the database* $DB = \{v_0, v_1, \ldots, v_{n-1}\}$ *is stored on a single server* $\mathcal{S}$*. The client uses a homomorphic encryption scheme to generate a query* Q *that hides the index* i*.*

Client Query: *The client encrypts its query index* i *using a public-key encryption scheme Enc:*

$$Q = Enc_{pk}(i).$$

Server Response: *The server processes the encrypted query over the database using the homomorphic properties of the encryption scheme:*

$$R = \sum_{j=0}^{n-1} v_j \cdot Q[j] \mod n^2,$$

where $Q[j] = Enc_{pk}(1)$ *if* $j = i$*, and* $Q[j] = Enc_{pk}(0)$ *otherwise.*

Client Decoding: *The client decrypts* R *using its private key to retrieve* v_i*:*

$$v_i = Dec_{sk}(R).$$

The security and efficiency of CPIR protocols can be defined as bellow:

- **Query Privacy:** The server cannot distinguish i from the encrypted query Q under the hardness of the encryption scheme (e.g., LWE or DCRA).

- **Correctness:** The client retrieves the correct record v_i without errors.

- **Efficiency:** CPIR reduces communication overhead compared to ITPIR but increases computational costs due to homomorphic operations.

(Kushilevitz and Ostrovsky, 1997) introduced the first single-server CPIR scheme, showing that replication is unnecessary if computational hardness assumptions (e.g., the difficulty of factoring) are the basis of the system security. Later, (Gentry and Ramzan, 2005) proposed CPIR protocols with a constant communication rate, emphasizing the use of advanced cryptographic primitives to reduce communication complexity while maintaining privacy.

With the doors opened to efficient CPIR protocols, researchers began exploring the trade-offs between communication and computation complexity. This led to the development of several other works such as: (Angel *et al.*, 2018) that introduced efficient query compression techniques, demonstrating how amortized query processing can make CPIR practical for large-scale databases; (Ali *et al.*, 2019) that explored trade-offs between communication and computation in CPIR, providing insights into optimizing query generation and response evaluation and (Corrigan-Gibbs and Kogan, 2020) which contributed to CPIR by introducing methods for sublinear online computation.

CPIR schemes reduce communication complexity compared to ITPIR but introduce higher computational overhead due to reliance on cryptographic operations. These trade-offs make CPIR suitable for applications where server-side computational resources are abundant, and database replication is not an option.

4.4 Stateless Private Information Retrieval

Stateless Private Information Retrieval (Stateless PIR) refers to PIR protocols where the server maintains no persistent state between queries. This is saying that each query is processed independently, making these schemes simpler to implement and scalable for large-scale deployment (very large databases for example).

In Stateless PIR, the client issues a query Q to the server for an index i, and the server processes Q without relying on any prior knowledge or preprocessing.

Definition 19 (Stateless Private Information Retrieval)**.** *Let the database be* $DB = \{v_0, v_1, \ldots, v_{n-1}\}$*, where* n *is the number of records. The goal is for the client to retrieve* v_i *without revealing* i *to the server.*

Client Query: *The client generates a query* Q *that encodes the desired index* i *in a privacy-preserving manner. For instance, using homomorphic encryption, the query* Q *may take the form:*

$$Q = Enc_{pk}(i),$$

where Enc_{pk} *is a public-key encryption scheme with homomorphic properties.*

Server Response: *The server evaluates a function f over the encrypted query Q and the database DB:*

$$R = f(Q, DB),$$

where f is designed to compute the encrypted result corresponding to the desired record v_i. For example, using additive homomorphic encryption, the server computes:

$$R = \prod_{j=0}^{n-1} Enc_{pk}(v_j)^{Q[j]} \mod n^2.$$

Client Decoding: *The client decrypts the server's response R to recover v_i:*

$$v_i = Dec_{sk}(R),$$

where Dec_{sk} is the decryption function using the client's private key sk.

Stateless PIR schemes have the advantages when it comes to scalability as the server does not need to store or manage additional metadata, making the scheme suitable for dynamic databases or large-scale systems. And is also simpler in the way that each query is processed independently, reducing implementation complexity.

This is widely explored in works such as (Mughees *et al.*, 2021; Corrigan-Gibbs and Kogan, 2020) that introduced efficient Stateless PIR protocols, demonstrating the practicality of PIR for real-world applications.

Despite their simplicity, stateless PIR schemes face certain drawbacks, such as a high computational overhead since the server cannot perform preprocessing, each query requires a full evaluation over the entire database. Also, it can be inefficient for repeated queries, as frequent queries for similar data incur redundant computations, and Stateless PIR, usually, has no mechanism to amortize costs.

4.5 Stateful Private Information Retrieval

Stateful Private Information Retrieval (Stateful PIR) enhances efficiency by introducing a preprocessing phase where the server maintains auxiliary data structures or metadata to accelerate query responses. This stateful design enables sublinear server-side computation relative to the database size, making it particularly efficient for large-scale applications.

Definition 20 (Stateful Private Information Retrieval)**.** *In Stateful PIR, the server preprocesses the database DB into auxiliary data $A(DB)$ during an offline phase. This preprocessing allows the server to answer client queries Q more efficiently in the online phase.*

Preprocessing Phase: *The server computes auxiliary data $A(DB)$ from the database $DB = \{v_0, v_1, \ldots, v_{n-1}\}$. For example:*

$$A(DB) = Preprocess(DB),$$

where Preprocess generates structured data (e.g., compressed encodings or precomputed homomorphic evaluations) that facilitates efficient query handling.

Client Query: *The client generates a query Q that specifies the index i in a privacy-preserving manner, as in Stateless PIR. For instance:*

$$Q = Enc_{pk}(i).$$

Server Response: *Using $A(DB)$, the server computes the response R more efficiently:*

$$R = Eval(f, A(DB), Q),$$

where f is the evaluation function. For example, in schemes based on lattice cryptography, $A(DB)$ might store precomputed matrix-vector products to accelerate evaluations.

Client Decoding: *As in Stateless PIR, the client decrypts R to obtain the desired record v_i:*

$$v_i = Dec_{sk}(R).$$

Stateful PIR schemes provide several advantages over Stateless PIR. The computational efficiency is reached through a preprocessing phase that reduces the server's computational cost during the online phase, enabling sublinear query evaluation times. Aditionally, it is possible to amortize the preprocessing cost across multiple queries, making these schemes particularly efficient for repeated queries. These benefits are explored in schemes such as: (Aguilar-Melchor *et al.*, 2014; Menon and Wu, 2022; Davidson *et al.*, 2022; Patel *et al.*, 2018; Henzinger *et al.*, 2023; Zhou *et al.*, 2023)

It must be noted that the benefits of Stateful PIR come at the cost of certain trade-offs, either from the increased Server Storage, in which the server must store auxiliary data $A(DB)$, which can be significant for large databases. Or also a higher complexity of updates, as modifications to the database DB require corresponding updates to $A(DB)$, increasing maintenance overhead. Also, the client may also have the additional communication and computation costs to perform additional operations such as: downloading the *hint* $A(DB)$ and also perform more computational operations during the setup and query phases.

Chapter 5

Previous works

Private Information Retrieval (PIR), as previously discussed, is a widely explored area of research in cryptography and is becoming increasingly relevant given the growing volume of information processed daily.

Among the various objectives of new PIR implementations, two primary goals drive much of the research: optimizing client-side performance and enhancing the system's ability to handle multiple queries in a single request. This includes improving the size of individual queries and the number of queries processed per request, both of which aim to enable scalable implementations for larger servers.

In this chapter we aim to present and examine five contributions in the field, exploring their core ideas, theoretical basis, and relevance to my work.

5.1 XPIR: Private Information Retrieval for Everyone (Aguilar-Melchor *et al.*, 2014)

XPIR (Aguilar-Melchor *et al.*, 2014) is a work by Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse and Marc-Olivier Killijian, and it also represents a paradigm shift in the design and practicality of single-database Computationally-Private Information Retrieval (CPIR) schemes.

The idea bases it's technical approach in lattice-based cryptography, and it challenges the long-standing belief that CPIR is impractical, as proposed by Sion and Carbunar in 2007 (Sion and Carbunar, 2007).

XPIR demonstrates that with modern cryptographic tools and efficient implementations, CPIR can achieve competitive performance while maintaining rigorous privacy guarantees.

5.1.1 Core Idea

The foundational idea behind XPIR is to utilize homomorphic encryption based on the Ring-Learning With Errors (Ring-LWE) problem to design a CPIR scheme that is both

efficient and secure. Homomorphic encryption enables computations to be performed directly on encrypted data, as mentioned before in this work, and it ensures thatthe privacy of the client's query. It is important to remark that XPIR replaces traditional number-theoretic constructions, such as RSA, with lattice-based approaches, significantly improving the computational efficiency and throughput of the protocol, and what is more noticable is that (Aguilar-Melchor *et al.*, 2014) also opened doors for many other LWE-based PIR schemes.

XPIR addresses the classic challenge of CPIR: ensuring that the server processes the entire database to avoid revealing which entry the client is interested in. This ensures that the protocol achieves the same level of privacy as downloading the entire database, but with sublinear communication costs.

5.1.2 Theoretical Contributions

XPIR's contributions can be summarized as follows:

- **Lattice-based cPIR:** By using Ring-LWE cryptography, XPIR achieves post-quantum security and improves the practicality of CPIR schemes.
- **Optimization Techniques:** XPIR introduces algorithmic optimizations, including FFT-like representations and Newton quotient pre-computation, to enhance the efficiency of homomorphic operations.
- **Efficient Implementation:** The protocol demonstrates multi-gigabit processing throughput on commodity CPUs, showcasing its practical viability in real-world scenarios.
- **Contradiction of Sion and Carbunar's Results:** XPIR disproves the widely accepted conclusion that CPIR is inherently impractical, showing that modern lattice-based cryptography enables feasible implementations. And this is extremely relevant, since athte tame this approach in CPIR was made, applied LWE was not as explored as it is today.

5.1.3 Challenges Addressed

XPIR tackles several long-standing issues in CPIR:

- **High Computational Overhead:** Traditional CPIR schemes require the server to perform expensive cryptographic operations over the entire database. XPIR mitigates this overhead by leveraging the efficiency of Ring-LWE cryptography.
- **Communication Costs:** Unlike trivial PIR (where the entire database is sent to the client), XPIR achieves sublinear communication costs while preserving privacy.
- **Scalability:** XPIR demonstrates practical performance even for large databases, making it suitable for a wide range of applications, including private keyword search.

5.1.4 Relevance to This Work

XPIR serves as a relevant reference for this work in several ways:

- The use of lattice-based cryptography in XPIR provides a robust framework for designing efficient and secure PIR protocols.
- XPIR's emphasis on practical implementations and real-world performance directly informs the design choices in this work, particularly in developing protocols suitable for constrained environments.
- The contradiction of Sion and Carbunar's conclusions about CPIR's impracticality highlights the evolving nature of cryptographic research and the potential for modern cryptographic tools to redefine the boundaries of feasibility.

It is worth mentioning that XPIR marks a significant milestone in the development of CPIR schemes. By demonstrating that CPIR can be both practical and secure, it sets a precedent for future research in the field. Its reliance on lattice-based cryptography, along with its innovative optimizations and efficient implementation, serves as a blueprint for this work, which aims to build on these principles to address remaining challenges in the domain.

5.2 SealPIR: PIR with compressed queries and amortized query processing (Angel *et al.*, 2018)

SealPIR (Angel *et al.*, 2018), a contribution by Sebastian Angel, Hao Chen, Kim Laine and Srinath Setty, also builds upon the Computationally-Private Information Retrieval (CPIR) approach by addressing one of its most significant challenges: the high computational and communication costs associated with query generation and processing.

By introducing techniques for query compression and amortized query processing, SealPIR achieves substantial improvements over prior CPIR protocols, making it more practical for deployment in bandwidth-limited and computationally constrained environments. Note that this is what we have been mentioning in this work as one of The main motivation over research over optimizations in PIR.

5.2.1 Core Idea

SealPIR refines and extends the XPIR protocol Aguilar-Melchor *et al.*, 2014. While XPIR uses a lattice-based cryptosystem to enable efficient homomorphic operations, it suffers from high network costs due to the size of client queries. In SealPIR, on the other hand, the authors introduce two complementary techniques to address this bottleneck, defining a new techical sight of the issues in PIR:

1. **Compressed Queries:** SealPIR employs a query expansion procedure that reduces the size of the query vector sent by the client to the server. With this approach, instead of sending one ciphertext per database entry, the client sends a single ciphertext encoding its desired index. The server then obliviously expands this ciphertext into

a query vector using a homomorphic expansion procedure, eliminating the need for the client to generate a large query.

2. **Probabilistic Batch Codes (PBCs):** SealPIR introduces a technique that they call PBC as a data encoding mechanism to amortize server-side computational costs when processing multiple queries. Unlike traditional batch codes, PBC are specifically designed for PIR applications, enabling efficient multi-query processing with minimal network overhead.

5.2.2 Theoretical Contributions

SealPIR's contributions, from a theoretical perspective, can be summarized as follows:

- **Oblivious Query Expansion:** The query compression mechanism significantly reduces the communication overhead between the client and the server. This is achieved by encoding the client's query as a single ciphertext, which is later expanded homomorphically at the server without revealing the client's desired index.
- **Amortized Query Processing:** Through the PBCs, SealPIR enables the server to process multiple queries from the same client in a computationally efficient manner, achieving a huge speedup compared to single-query processing.
- **Reducing the cost of expansion:** SealPIR replaces the cryptosystem used in XPIR with the Fan-Vercauteren Cryptosystem cryptosystem, which will not be explored in this work. This choice simplifies the implementation of key operations, such as homomorphic addition, multiplication, and substitution, while ensuring robust security guarantees.
- **Handling Larger Databases:** SealPIR overcomes the limitations imposed by the fixed size of query vectors by either concatenating multiple expanded vectors or using a d-dimensional hypercube representation. For example, two ciphertexts can index up to N^2 entries, enabling efficient scaling for databases with millions of entries.
- **Probabilistic Batch Codes (PBCs):** SealPIR uses PBCs to amortize server-side computational costs across multiple queries. Unlike traditional batch codes, PBC achieve this with minimal network overhead and are tailored to the specific requirements of PIR protocols.

5.2.3 Challenges Addressed

SealPIR tackles critical challenges inherent in traditional CPIR schemes:

- **High Query Size:** Traditional CPIR schemes, including XPIR, require the client to send a query vector proportional to the database size. SealPIR's compressed query mechanism significantly reduces this overhead,
- **Expensive Query Processing:** SealPIR's use of PBC allows the server to amortize computational costs across multiple queries, significantly improving throughput.

- **Bandwidth Constraints:** By reducing the query size and optimizing network usage, SealPIR is better suited for bandwidth-limited environments, such as mobile devices or constrained wired connections.

5.2.4 Relevance to This Work

SealPIR's contributions align closely with the objectives of this work, in a few lines we can mention that the compressed query mechanism opens the doors for reducing communication costs in bandwidth-constrained environments, a key focus of this work. Actually, the compression idea presented in this paper is one of inspirations for the optimizations presented in this work.

Finally, SealPIR represents a significant contribution in the field of computational PIR by addressing the dual challenges of high communication and computational costs. Its attempt to use compressed queries and amortized processing techniques makes it a practical solution for real-world applications.

5.3 Single-Server Private Information Retrieval with Sublinear Amortized Time (Corrigan-Gibbs, Henzinger, *et al.*, 2022)

This article introduces a new class of single-server Private Information Retrieval (PIR) protocols that achieve sublinear amortized server time, marking a significant theoretical advancement in the field. Unlike traditional single-server PIR schemes that require server-side computation linear in the size of the database for each query, these protocols reduce the amortized computational costs while maintaining robust security guarantees under standard cryptographic assumptions.

5.3.1 Core Idea

The key idea of (Corrigan-Gibbs, Henzinger, *et al.*, 2022) lies in dividing the server's computation into two phases, namaely the offline and online phases:

1. **Offline Phase:** During this phase, the server preprocesses the database to generate a small "hint" that the client can store.[1] The server-side computation during this phase remains linear in the size of the database, but this cost is amortized over multiple queries.
2. **Online Phase:** Using the hint from the offline phase, the client can issue adaptive queries to the server. The server processes each query in sublinear time relative to the database size, resulting in sublinear amortized cost across multiple queries.

[1] The *hint* is a compact, precomputed piece of information about the database, generated during an offline phase, that the client stores and uses to enable sublinear-time query responses during subsequent online interactions.

Note that the offline phase introduces computational overhead upfront but ensures that subsequent queries can be processed more efficiently, making the scheme well-suited for applications with multiple or sequential queries. This may no be clear, so a reading over the work is recommended.

5.3.2 Theoretical Contributions

(Corrigan-Gibbs, Henzinger, *et al.*, 2022) introduces several important theoretical advancements to the design of PIR protocols:

- **Sublinear Amortized Server Time:** The protocols achieve sublinear amortized server-side computation time, making them the first to combine this property with adaptivity in query handling and minimal additional storage requirements.
- **Client-Side Hints:** The use of a client-stored hint reduces the online computational burden on the server. The hint is computed during the offline phase with a time complexity linear in the database size but is reused across multiple queries to achieve amortized efficiency. Note that this ideia is
- **Adaptivity:** Unlike batch-PIR schemes,[2] which require the client to submit all queries at once, this protocol allows the client to issue queries adaptively over time, enabling broader applicability in real-world scenarios.
- **Two-Step Compilation:** (Angel *et al.*, 2018) demonstrates a methodology for compiling two-server PIR schemes into single-server schemes using fully homomorphic encryption (FHE) or linearly homomorphic encryption (LHE). This compilation process bridges the gap between theoretical constructs and single-server practicality.
- **Lower Bounds:** The authors establish tight lower bounds on the trade-off between client storage, server online time, and database size. These bounds prove that the proposed schemes are asymptotically optimal in terms of the trade-offs achieved.

5.3.3 Challenges Addressed

The protocols address several longstanding challenges in the field of PIR:

- **High Computational Costs:** By decoupling the offline and online phases, the protocols reduce the computational heavy work of individual queries to sublinear complexity.
- **Adaptivity vs. Efficiency:** Traditional schemes often sacrifice efficiency for adaptivity. This paper bridges that gap, supporting adaptive queries with efficient server-side computation.

[2] Batch-PIR schemes are private information retrieval protocols designed to allow a client to retrieve multiple records from a database in a single query while maintaining privacy. These schemes require the client to submit all queries at once in a non-adaptive batch, making them efficient for scenarios with predefined query sets. However, they lack flexibility in supporting dynamic or adaptive queries, limiting their applicability in real-time systems. We will not go deep in this approach in this work

- **Practical Applicability:** By relying on not too hard to implement cryptographic assumptions and requiring only sublinear additional storage, the protocols are well-positioned for practical deployment in settings with multiple clients or repeated queries.

5.3.4 Real World Applications

Two important things to mention on (Corrigan-Gibbs, Henzinger, *et al.*, 2022) is that the reduced server-side costs make the schemes suitable for large-scale systems, such as digital libraries, DNS resolution, and malware detection. This is a clear visualization of a real world cryptographic application. In addition the amortized computational efficiency is particularly beneficial for scenarios where clients make repeated queries to the same database

5.3.5 Relevance to This Work

(Corrigan-Gibbs, Henzinger, *et al.*, 2022) aligns with the objectives of my work by demonstrating the feasibility of sublinear amortized server-side computation in single-server PIR. Specifically on the efficiency improvements, where the techniques for reducing server-side costs while maintaining adaptivity provide valuable insights for designing scalable PIR protocols.

It also represents a significant theoretical advancement in PIR by achieving sublinear amortized server-side computation time while maintaining practical usability. Its innovations, specially in hint-based query handling and adaptivity establish a new benchmark for PIR protocol design. These contributions directly inform this work's efforts to balance efficiency, scalability, and security in single-server PIR systems.

5.4 FrodoPIR: Simple, Scalable, Single-Server Private Information Retrieval (Davidson *et al.*, 2022)

FrodoPIR (Davidson *et al.*, 2022) is a stateful, single-server Private Information Retrieval (PIR) scheme that takes in account a large numbers of clients, meaning many queries for the server in a short time (usually small queries).

The scheme developed in FrodoPIR achieves a great practical scalability and cost-efficiency by using a client-independent offline preprocessing phase, coupled with minimal online overheads, resulting in significantly reduced amortized financial and computational costs for the server. Note that the idea of client-independent is very similar to the ideas presented on the works we mentined above.

The scheme is built upon the Learning With Errors (LWE) problem, avoiding the complexity of fully homomorphic encryption (FHE) techniques used in previous state-of-the-art solutions, which focused on Rind-Learning With Erroes (RLWE) schemes.

For a better understading of this protocol, see the Appendix A .

5.4.1 Core Idea

The core innovation in FrodoPIR lies in its separation of the PIR process into two distinct phases, following the same proposition as in (Corrigan-Gibbs, Henzinger, *et al.*, 2022):

1. **Offline Phase:** The server preprocesses the database into a compressed form, independent of the number of clients or their queries. The resulting global parameters are made publicly available for download by the clients.
2. **Online Phase:** Clients use the preprocessed parameters to generate encrypted query vectors and send them to the server. The server computes the response by performing matrix-vector multiplications and sends it back to the client, who decrypts the response to retrieve the desired data.

This design in this protocol ensures that the computationally expensive preprocessing step is amortized across all client queries, resulting in significant cost savings for large-scale deployments. And actualy, the computational gains are part of the main success of this work.

5.4.2 Theoretical Contributions

FrodoPIR introduces and applies several key contributions to the design of single-server PIR schemes:

- **Client-Independent Preprocessing:** Unlike previous stateful schemes where preprocessing depends on the number of clients, FrodoPIR's offline phase is entirely client-independent. This reduces server-side costs and enhances scalability.
- **Use of LWE Instead of RLWE:** FrodoPIR is built upon the Learning With Errors (LWE) problem, avoiding ring-based lattice structures. This simplification enables modular arithmetic implementations that are both lightweight and efficient, making FrodoPIR practical for real-world applications.[3]
- **Reduced Online Costs:** FrodoPIR achieves server response times of less than 1 second for a database of 1 million elements and maintains a response size blow-up factor of less than 3.6×, demonstrating low online communication and computational overheads.
- **Configurability and Cost Optimization:** FrodoPIR is highly configurable, allowing adjustments in client download size and server-side computation to optimize performance for various deployment scenarios. This makes it adaptable to diverse applications.

[3] LWE (Learning with Errors) is often favored over RLWE (Ring Learning with Errors) in certain contexts due to its simpler underlying structure in vector spaces, which avoids the additional algebraic assumptions required by RLWE. While RLWE relies on polynomial rings and ideal lattices, enabling more compact and efficient representations, it introduces potential vulnerabilities and exploitations on ring's algebraic properties. On the other hands, LWE operates in general lattices and maintains reductions to worst-case lattice problems without introducing additional algebraic assumptions. This makes LWE more applicable and theoretically robust. This is widely explored in (Regev, 2009; Peikert, 2015)

- **Open-Source Implementation and Analysis:** FrodoPIR includes a simple Rust-based implementation and detailed experimental analyses, emphasizing transparency and reproducibility of its results.

Note that all of the above characteristics mentioned are very important for the practicality of the protocol, and this correlates with how applicable in real life the protocol is.

5.4.3 Challenges Addressed

We need to notice, though, that FrodoPIR addresses several limitations of prior PIR schemes:

- **High Preprocessing Costs:** Previous stateful schemes incurred preprocessing costs that scaled linearly with the number of clients, making them impractical for large-scale systems. FrodoPIR eliminates this dependency.
- **Complexity of FHE-Based Methods:** By avoiding FHE, FrodoPIR simplifies implementation and reduces computational costs, m aking it feasible to deploy on standard cloud-based infrastructure (The pratical experiments were done on a real-world AWS server of considerable size - c5n.2xlarge AWS EC2).
- **Financial Cost Scalability:** FrodoPIR significantly lowers server-side financial costs, with experiments showing costs of approximately $1 for processing 100,000 client queries in a database of 1 million elements.
- **Implementation Simplicity:** FrodoPIR's design relies on modular arithmetic, requiring only standard 32-bit integer operations, which are widely supported and easy to implement.

Experimental results from (Davidson *et al.*, 2022) remarks that it achieves low runtimes and communication overheads across various database configurations, making it particularly suitable for large-scale, multi-client deployments.

5.4.4 Relevance to This Work

FrodoPIR design aligns closely with the objectives of this work in a way that it implements towards a practical, scalable, cost-effective PIR protocol. Here, the client-independent preprocessing and efficient online phase serve as an utile sketch for scalable PIR systems. Also, FrodoPIR's focus on reducing financial and computational costs is very consonant to this work efforts to develop cost-effective cryptographic protocols.

5.5 SimplePIR - One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval (Henzinger *et al.*, 2023)

SimplePIR (Henzinger *et al.*, 2023) is a single-server Private Information Retrieval (PIR) scheme that achieves unprecedented server throughput, rivaling the performance of multi-server PIR protocols while relying solely on the Learning With Errors (LWE)

assumption. This scheme idea, just like the one used in (Davidson *et al.*, 2022) laverages of the preprocessing techniques, and, on top of that, SimplePIR significantly reduces server computation per query, achieving throughput rates as high as 10 GB/s per core. Note that this design redefines the efficiency benchmarks for single-server PIR protocols, though it incurs higher communication costs relative to some prior schemes.

5.5.1 Core Idea

SimplePIR uses a two-phase approach, exactly like the ones previously mentioned to achieve high throughput:

1. **Preprocessing Phase:** The server preprocesses the database, performing computationally expensive operations that depend only on the database and the public parameters of the encryption scheme. This preprocessing step enables the server to handle subsequent client queries with minimal computation.

2. **Online Phase:** Clients query the server using encrypted query vectors. The server efficiently computes matrix-vector products and returns the result, allowing the client to decrypt and retrieve the desired database entry.

5.5.2 Theoretical Contributions

SimplePIR contributes with a few innovations to the PIR literature, which are actually going to be very important for the practicality of the protocol that will be developed in this work:

- **Preprocessing for High Throughput:** By performing the bulk of matrix-vector computations during preprocessing, SimplePIR minimizes server-side computation in the online phase to approximately one 32-bit multiplication and addition per database byte.

- **Use of LWE-Based Encryption:** SimplePIR is based on standard LWE rather than the ring variant (RLWE). This choice avoids polynomial arithmetic and Fast Fourier Transforms, simplifying implementation and improving efficiency. (Note that (Davidson *et al.*, 2022) does the same)

- **Client Reusability of Preprocessed Hints:** Clients download a *hint* during the preprocessing phase, which is independent of the specific query. This hint can be reused for an unbounded number of queries, amortizing its communication cost.

- **Adaptability to Database Size:** SimplePIR supports databases of arbitrary sizes by representing the database as a matrix. The hint size scales as $\mathcal{O}(\sqrt{N})$, where N is the database size, ensuring scalability for large datasets.

- **DoublePIR Extension:** A recursive variant, DoublePIR, further reduces the hint size by applying SimplePIR recursively, achieving a trade-off between communication cost and server throughput.

5.5.3 Challenges Addressed

SimplePIR addresses critical challenges inherent in single-server PIR schemes, being the most recent work in this list. Notable, it tackles the following issues:

- **Throughput Limitations:** Previous single-server PIR schemes achieved server throughput far below memory bandwidth. SimplePIR closes this gap, achieving up to 81% of theoretical memory bandwidth.
- **Computational Overheads:** By offloading computationally expensive operations to the preprocessing phase, SimplePIR minimizes server workload during query handling.
- **Multi-Query Scalability:** The client's hint is reusable across multiple queries, ensuring that the scheme remains efficient in scenarios with repeated queries.

On (Henzinger *et al.*, 2023) implementation, there is a notable effort to show the feasibility and efficiency. So, the authors demonstrate that SimplePIR consists of the three main components: the server-side preprocessing, the client-side query generation, and the server-side query processing. With them, they explore the following:

- **Preprocessing Phase:** The server preprocesses the database using the LWE-based encryption scheme, producing a hint that can be downloaded by clients. This preprocessing work is amortized across all queries.
- **Online Phase:** During the online phase, the server performs lightweight matrix-vector multiplications using the preprocessed data, achieving high throughput for query handling.
- **Communication Costs:** For a 1 GB database, the hint size is approximately 121 MB, and each query requires 242 KB of communication. The DoublePIR variant reduces the hint size to 16 MB at the cost of slightly higher per-query communication.

In this work, we will not go deeper on SimplePIR or DoublePIR practical scheme and implementations.

5.5.4 Relevance to This work

This paper is probably the closest to the idea that will further be developed in this work. Namely, we seek for **efficiency** with a preprocessing techniques and high throughput server as a model for designing computationally efficient PIR protocols. **Scalability** by the ability to handle large databases and multi-query scenarios and the possibility of a simple implementation using standard LWE and its avoidance of complex polynomial arithmetic and FFTs.

Finally, SimplePIR represents an advance in single-server PIR design by achieving high throughput, scalability, and simplicity. It uses the advantages of the preprocessing and LWE-based encryption positions it as a benchmark for future PIR protocols.

Chapter 6

FoldingFrodo

In this chapter, we present the construction of the *FoldingFrodo* protocol with the *Paillier* (Paillier, 1999) encryption system. We start by presenting the cryptographic setup, followed by the preprocessing phase, and the online phase.

The *FoldingFrodo* protocol is a PIR protocol based on the *FrodoPIR* (Davidson *et al.*, 2022) protocol that was described in the previous chapter. The *FrodoPIR* protocol, as well as *FoldingFrodoPIR* protocol are stataful, computational PIR protocol. This means that the client maintains an internal state that is updated at each query, and the server maintains a state that is updated at the preprocessing phase.

6.1 FoldingFrodo with Paillier

The goal of this approach on the PIR protocol is to reduce the computational cost of the online query processing, allowing the client to deal with multiple indices simultaneously. By structuring the database as a $\sqrt{m} \times \sqrt{m}$ matrix $\boldsymbol{D}$, each cell representing a different element in DB, the client sends then two query vectors $\boldsymbol{v}_{\text{row}}$ and $\boldsymbol{v}_{\text{col}}$, each of size $\sqrt{m}$. These ideas were used in separate works, namelly, this idea of optimization was used in (Angel *et al.*, 2018) and (Henzinger *et al.*, 2023), and the goal is to apply this optimization to the *FrodoPIR* protocol.

In order to reduce the amount of homomorphic sums on the client query generation phase, we apply the *Paillier* (Paillier, 1999) encryption for the homomorphic sum at the indicator vector. Same is done to compute the server response.

6.1.1 Notations

On the *FoldingFrodoPIR* protocol, we have the following notations:

- All $\boldsymbol{v}$ are column vectors.
- $\lfloor x \rceil$: the nearest integer to x.
- $\boldsymbol{v} \xleftarrow{\$} \chi$ means that $\boldsymbol{v}$ is sampled from the distribution χ.

- λ is the security parameter.
- *PRG* is a pseudorandom generator function that takes a seed μ and generates a matrix $\mathbf{A}$.

6.1.2 Cryptographic Setup

From the *FrodoPIR* setup, we will keep few setups unchanged. Take the server S containing the database $\boldsymbol{DB}$ that each client access. The database $\boldsymbol{DB}$ can be read as a vector of m elements of w bits. Each $i-th$ entry of $\boldsymbol{DB}$ is associated with an index i that corresponds to a position in the vector. In the setup, let's assume that there are C clients making a maximum of c queries to the database $\boldsymbol{DB}$.

Note that the *LWE* instance is used under the setting: let n be the dimension and q the modulus, let ρ be the number of bits within each entry of the DB matrix, where $0 < \rho < q$. Let χ be the uniform distribution over $\{-1, 0, 1\}$, let λ be the security parameter, and use $\texttt{PRG}(\mu, x, y, q)$ to denote a pseudorandom generator that uses a seed $\mu \leftarrow \{0,1\}^\lambda$ in the matrix $\mathbb{Z}_q^{x \times y}$, where x and y are the dimensions of the matrix.

6.1.3 Preprocessing Phase

1. **Server Setup: `(FFPIR.ssetup)`** The server constructs the database with m elements and samples the seed $\mu \in \{0,1\}^\lambda$.

   The server also generates the parameters of the LWE encryption scheme as (q, n, m, σ) and the distribution of the secret being a ternary distribution χ over $\{-1, 0, 1\}$.

   The database is parsed as $\boldsymbol{DB} \in \mathbb{Z}_t^m$ and parses $\boldsymbol{D} \in \mathbb{Z}_t^{\sqrt{m} \times \sqrt{m}}$ as being the matrix representation of the database stored at the server. Note that t is the plaintext space and that the cipher is based in the decisional-LWE and that this cypher has the message $m \in \mathbb{Z}_t$. In other words,

   $$\boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e} + \frac{q}{t} \cdot \boldsymbol{m}$$

   Also, notice that all the $\boldsymbol{D}[i][j]$ elements on the database are all taken $\bmod t$, where a row of DB fits in $\mathbb{Z}_t$, or, in other words, the rows in $\boldsymbol{DB}$ has $log(t)$ bits.

   Then, the server generates the Paillier's parameters as (p, k, r). Note that we require Paillier's plaintext space to be $\mathbb{Z}_n$ with $n > q$ and not n^2 as the original scheme in (Paillier, 1999).

   And than the client sends a query with two $\sqrt{m} - dim$ vectors (`row, col`) instead of the one $m - dim$ vector on the *FrodoPIR* protocol.

2. **Server Preprocessing: `(FFPIR.spreproc)`** Here the server derives the matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \sqrt{m}}$ as

   $$\mathbf{A}[i, j] \leftarrow PRG(\mu, n, \sqrt{m}, q),$$

and runs

$$\boldsymbol{D} \leftarrow parse(\boldsymbol{DB}, t),$$

where the parse function encodes the database into a matrix $\boldsymbol{D} \in \mathbb{Z}_q^{\sqrt{m}\times\sqrt{m}}$

To generate the public parameters, the server then computes

$$\boldsymbol{M} \leftarrow \boldsymbol{A} \cdot \boldsymbol{D}$$

where $\boldsymbol{M} \in \mathbb{Z}_q^{n\times\sqrt{m}}$ and publishes the pair $(\mu, \boldsymbol{M}) \in \{0,1\}^{\lambda}\times\mathbb{Z}_q^{n\times\sqrt{m}}$ to a public repository accessible by clients.

3. **Client Preprocessing: `(FFPIR.cpreproc)`** Each client downloads the pair $(\mu, \boldsymbol{M})$ and runs

$$\boldsymbol{A} \leftarrow PRG(\mu, n, \sqrt{m}, q).$$

Then, for the c intended queries the index $i \in [1, c]$, the client samples c vectors $\boldsymbol{s_i} \leftarrow (\chi)^n$ and $\boldsymbol{e_i} \leftarrow (\chi)^{\sqrt{m}}$.

Later it computes:

$$\boldsymbol{b_i} \leftarrow \boldsymbol{s_i^T} \cdot \boldsymbol{A} + \boldsymbol{e_i^T} \in \mathbb{Z}_q^{\sqrt{m}}$$

and

$$\boldsymbol{c_i} \leftarrow \boldsymbol{s_i^T} \cdot \boldsymbol{M} \in \mathbb{Z}_q^{\sqrt{m}}$$

DB and stores the pair $\boldsymbol{X} = (\boldsymbol{b_j}, \boldsymbol{c_j})$

6.1.4 Online Phase

1. **Client query generation: `(FFPIR.query)`**

For each index i that the client wants to query, the client generates a vector of the following this:

Define $\boldsymbol{f}_i = \lfloor q/t \rceil \boldsymbol{e_i}$ where $\boldsymbol{e_i} \in \{0,1\}^{\sqrt{m}}$ is the i-th indicator vector.

This means that $\boldsymbol{f}_i$ an all-zero vector in which the i- th position is taken as $\lfloor q/t \rceil$.

The client then pops two pairs $(\boldsymbol{b}, \boldsymbol{c})$ and $(\boldsymbol{b'}, \boldsymbol{c'})$ from the internal state X.

Also, the client encrypts each bit of $\mathbf{e}_j$ with *Paillier*, that is,

$$\forall 1 \leq i \leq \sqrt{m}, \hat{b}_i = \mathsf{Paillier.Enc}(\mathbf{e}_j[i])$$

and defines $\hat{\mathbf{b}} = (\hat{b}_1, ..., \hat{b}_{\sqrt{m}})$.

The client then computes:

$$\tilde{\mathbf{b}} = \boldsymbol{b} + \boldsymbol{f}_i \in \mathbb{Z}_q^{\sqrt{m}}$$

And finally send $\tilde{\boldsymbol{b}}$ and $\hat{\boldsymbol{b}}$ to the server.

2. **Server response: `(FFPIR.respond)`** The server receives $\tilde{\boldsymbol{b}}$ and $\hat{\boldsymbol{b}}$ from the client,

and computes:

$$\tilde{\mathbf{c}} = \tilde{\mathbf{b}} \cdot \mathbf{D} \in \mathbb{Z}_q^{\sqrt{m}}$$

Now, the server uses Paillier's homomorphic properties and the vector of Paillier ciphertexts $\hat{\mathbf{b}}$ received as part of the query to compute

$$\hat{\mathbf{c}} = \sum_{i=1}^{\sqrt{m}} \hat{\mathbf{b}}[i] \cdot \tilde{\mathbf{c}}[i] = \mathsf{Paillier.Enc}(\tilde{\mathbf{c}} \cdot \mathbf{e}_j) = \mathsf{Paillier.Enc}(\tilde{\mathbf{c}}[j])$$

The server then sends $\hat{\mathbf{c}}$ to the client.

3. **Client postprocessing: `(FFPIR.process)`** The client receives $\hat{\mathbf{c}}$, computes $y = \mathsf{Paillier.Dec}(\hat{\mathbf{c}})$ and outputs $\lfloor \frac{t}{q} \cdot (y - \mathbf{c}[j]) \rceil$ where $\mathbf{c}$ is the vector popped out of the state when the client sent the query.

6.1.5 Correctness

This section demonstrates the correctness of the *FoldingFrodoPIR* protocol by verifying that the final client output correctly retrieves the desired database element $\mathbf{D}[i][j]$.

6.1.6 Correctness of the Server Response (`FFPIR.respond`)

From the server's computation, the response vector $\tilde{\mathbf{c}}$ is defined as:

$$\tilde{\mathbf{c}} = \tilde{\mathbf{b}} \cdot \mathbf{D} \in \mathbb{Z}_q^{\sqrt{m}}.$$

Expanding $\tilde{\mathbf{c}}$, we have:

$$\tilde{\mathbf{c}} \equiv \mathbf{s} \cdot \mathbf{M} + \mathbf{e} \cdot \mathbf{D} + \lfloor q/t \rfloor \cdot \mathtt{row}_i(\mathbf{D}) \pmod{q}.$$

Here:

- $\mathbf{s} \cdot \mathbf{M}$: Represents the contribution from the structured matrix multiplication.
- $\mathbf{e} \cdot \mathbf{D}$: Accounts for the effect of the error vector.
- $\lfloor q/t \rfloor \cdot \mathtt{row}_i(\mathbf{D})$: Isolates the desired database row as a scaled indicator vector.

Considering the use of Paillier encryption, the server computes:

$$\hat{\mathbf{c}} = \sum_{i=1}^{\sqrt{m}} \hat{\mathbf{b}}[i] \cdot \tilde{\mathbf{c}}[i],$$

where $\hat{\mathbf{b}}$ is the vector of encrypted indicator bits received from the client. Notice that:

$$\hat{\mathbf{c}} \text{ encrypts } \mathbf{s} \cdot \mathtt{col}_j(\mathbf{M}) + \mathbf{e} \cdot \mathtt{col}_j(\mathbf{D}) + \lfloor q/t \rfloor \cdot \mathbf{D}[i][j] \pmod{q}.$$

This is valid because:

- Paillier's plaintext modulus n is larger than q, ensuring no modular wrap-around during computation.

- The norm of $\tilde{\mathbf{c}}$ satisfies $\|\tilde{\mathbf{c}}\|_\infty < q$, preventing overflow.

6.1.7 Correctness of the Client Postprocessing (`FFPIR.process`)

After receiving the server's encrypted response $\hat{\mathbf{c}}$, the client decrypts and processes it to retrieve the target database element. Specifically, the client computes:

$$y - \mathbf{c}[j] = \mathbf{e} \cdot \mathtt{col}_j(\mathbf{D}) + \lfloor q/t \rceil \cdot \mathbf{D}[i][j].$$

Breaking this down we have:

$$y - \mathbf{c}[j] = \mathbf{e} \cdot \mathtt{col}_j(\mathbf{D}) + \boldsymbol{\epsilon} + \frac{q}{t} \cdot \mathbf{D}[i][j],$$

where $\|\boldsymbol{\epsilon}\|_\infty \leq 1/2$. The term $\frac{q}{t} \cdot \mathbf{D}[i][j]$ is the target value.

The client then applies rounding to isolate $\mathbf{D}[i][j]$:

$$\left\lfloor \frac{t}{q} \cdot (y - \mathbf{c}[j]) \right\rceil = \left\lfloor \frac{t}{q} \cdot \mathbf{e} \cdot \mathtt{col}_j(\mathbf{D}) + \frac{t}{q} \cdot \boldsymbol{\epsilon} \right\rceil + \mathbf{D}[i][j].$$

The rounding succeeds because:

$$\frac{t}{q} \cdot \mathbf{e} \cdot \mathtt{col}_j(\mathbf{D}) + \frac{t}{q} \cdot \boldsymbol{\epsilon}$$

is small enough to round to zero, provided:

$$\left\| \mathbf{e} \cdot \mathtt{col}_j(\mathbf{D}) + \boldsymbol{\epsilon} \right\|_\infty < q/(2t).$$

This condition is satisfied when:

$$\|\mathbf{e}\|_\infty < q/(4t^2\sqrt{m}).$$

Thus, the client correctly retrieves:

$$\mathbf{D}[i][j].$$

Therefore, the correctness of the *FoldingFrodoPIR* protocol is ensured as the server provides a response that encodes the required row and ensures error terms are manageable. And also, the client processes the response using rounding techniques to accurately isolate the desired database element $\mathbf{D}[i][j]$.

6.2 Algorithms and Costs

In this chapter, the algorithms used in the *FoldingFrodo* protocol are presented. The algorithms are divided into five main parts: the server preprocessing, the client preprocessing, the client query generation, the server response, and the client decryption. For each part, the algorithm is presented as a pseudocode, followed by a brief analysis of the algorithm's complexity, and a discussion about the gains and losses of the protocol in terms of efficiency.

When analyzing the cost and complexities, it is needed the understanding the trade-offs between various resource requirements, including time and communication overhead. One way to verify this, is deividing the analysis into three primary categories:

- **Definition of Complexity and Big O Notation:** Complexity, in computational terms, measures the resources required for an algorithm to execute, such as time or memory, as a function of the input size. Big O notation is used to express the upper bound of an algorithm's complexity, focusing on the dominant terms that impact performance as the input grows.
- **Computational Costs:** Computational costs refer to the processing resources consumed by the server and client during each phase. In *FFPIR* case, key operations, such as matrix multiplications and cryptographic transformations, contribute significantly to this cost. It is important to analyze the impact of these operations using Big O notation, detailing how the complexity scales with parameters like database size and security levels.
- **Communication Costs:** Communication costs capture the data transmitted between the client and server during the query and response phases. These costs depend on the query size, the encrypted response size, and the overhead introduced by the protocol's structure. It is possible to detail the scalability of communication overhead with respect to the database size and the trade-offs involved in reducing this overhead.
- **Preprocessing Costs:** Preprocessing involves preparing the database and related structures before queries can be processed. This phase includes steps like encryption, data partitioning, and the generation of auxiliary data structures. The preprocessing cost is evaluated in detail, considering its one-time nature and how it affects the overall efficiency of the system.

The algorithms used as a base for the analysis are presented in the Appendix B.

6.2.1 Notations

Throughout this analysis, we use the following notations:

- m: Total number of database elements, $|DB| = m$.
- $\sqrt{m}$: Dimension of the database matrix representation, where DB is viewed as a $\sqrt{m} \times \sqrt{m}$ matrix.
- n: Dimension of the secret vector in the LWE-based scheme.

- q: Modulus used in LWE-based computations.
- λ: Security parameter.
- $\texttt{PRG}(\mu, x, y, q)$: Pseudorandom generator that outputs a matrix in $\mathbb{Z}_q^{x \times y}$.
- C: Number of concurrent clients in the system.
- c: Number of queries per client, $c = \sqrt{m}$.

6.2.2 Costs and Complexities

In this section, we analyze the costs and complexities of the Folding Frodo Private Information Retrieval (FFPIR) protocol. We consider the computational, communication, and storage costs associated with the protocol.

Operations in FFPIR

This very first analysis outlines the key operations involved in each phase of the *FFPIR* protocol. The metrics are categorized into modular multiplications (mod-q mults), additions (mod-q adds), and calls to the Pseudorandom Generator (PRG). Each formula provides the computational complexity based on the input dimensions such as n (number of entries) and $\sqrt{m}$ (square root of database blocks).

This framework assumes that PRG generation and matrix multiplications dominate the preprocessing phase, while later phases involve lightweight computations for response and processing.

The table 6.1 outlines the key operations in each phase of the *FoldingFrodo* protocol.

Phase	Metric	Formula	Explanation
`spreproc`	mod-q mults	$n \cdot m \cdot \sqrt{m}$	Precomputing $\boldsymbol{M} = \mathbf{A} \cdot \mathbf{D}$
	mod-q adds	$n \cdot m - n \cdot \sqrt{m}$	Additions in matrix multiplication
	PRG calls	$n \cdot \sqrt{m}$	Generating $\mathbf{A}$ via `PRG`
`cpreproc`	mod-q mults	$2 \cdot n \cdot \sqrt{m}$	Precomputing query-related vectors
	mod-q adds	$2 \cdot (n \cdot \sqrt{m} - \sqrt{m})$	Additions in preprocessing
	PRG calls	$n \cdot \sqrt{m}$	Regenerating $\mathbf{A}$
`query`	mod-q mults	–	No operations
	mod-q adds	2	Generating $\tilde{\mathbf{b}}$ and $\hat{\mathbf{b}}$
	PRG calls	–	No operations
`respond`	mod-q mults	$m + \sqrt{m} \cdot (2 + n)$	Computing $\tilde{\mathbf{c}}$ and homomorphic operations
	mod-q adds	$\sqrt{m} \cdot (n - 1) + (m - 1)$	Additions for response generation
	PRG calls	–	No operations
`process`	mod-q mults	$O(1)$	Decrypting one Paillier cryptogram
	mod-q adds	$O(1)$	Additions for decryption
	PRG calls	–	No operations

Table 6.1: *Detailed Operations in FFPIR*

6.2.3 Client and Server Costs

This analysis compares the computational, communication, and storage costs incurred by the client and server in both the offline and online phases of the protocol. Offline costs represent the preprocessing stage where the majority of heavy operations occur, while online costs reflect real-time interactions.

Here, there are assumptions needed, such as fixed database dimensions and a limited number of client queries, with C representing the number of concurrent queries amortized over the protocol's runtime.

Phase	Component	Client Cost	Server Cost	Storage Cost
Offline	Communication	$O(0)$	$O(n \cdot \sqrt{m})$	$O(n \cdot \sqrt{m})$
	Computation	$O(C \cdot \sqrt{m})$	$O(n \cdot m)$	–
	Storage	$O(C \cdot \sqrt{m})$	$O(n \cdot \sqrt{m})$	–
Online	Communication	$O(\sqrt{m})$	$O(\sqrt{m})$	–
	Computation	$O(1)$	$O(m + \sqrt{m})$	–
	Storage	$O(C \cdot \sqrt{m})$	$O(n \cdot \sqrt{m})$	–

Table 6.2: *Client and Server Costs for FFPIR*

6.2.4 Amortized and Big-O Costs

Amortized costs are derived by dividing the resource utilization by the number of queries, C, processed in batch mode. This provides insights into the per-query cost reduction achievable through batching. The assumptions include a uniform distribution of queries over the runtime and constant preprocessing overhead shared across the batch.

Phase	Component	Amortized Client Cost	Amortized Server Cost
Offline	Communication	$O(0)$	$O(n \cdot \sqrt{m}/C)$
	Computation	$O(\sqrt{m})$	$O(n \cdot m/C)$
Online	Communication	$O(\sqrt{m}/C)$	$O(\sqrt{m}/C)$
	Computation	$O(1/C)$	$O(m/C + \sqrt{m}/C)$

Table 6.3: *Amortized Costs for FFPIR*

The Big-O notation, on the other hand, provides a high-level summary of asymptotic costs for the client and server during offline and online phases. These costs are expressed in terms of the input size parameters n, m, and C. The assumptions include negligible overhead for constant-time operations and that communication costs scale linearly with the size of transmitted data. This table is particularly useful for understanding scalability and identifying bottlenecks in large-scale deployments. Table 6.4 summarizes the asymptotic costs of the protocol for large database sizes m, assuming fixed parameters n and C.

Phase	Component	Client Cost	Server Cost
Offline	Communication	$O(0)$	$O(n \cdot \sqrt{m})$
	Computation	$O(C \cdot \sqrt{m})$	$O(n \cdot m)$
	Storage	$O(C \cdot \sqrt{m})$	$O(n \cdot \sqrt{m})$
Online	Communication	$O(\sqrt{m})$	$O(\sqrt{m})$
	Computation	$O(1)$	$O(m + \sqrt{m})$
	Storage	$O(C \cdot \sqrt{m})$	$O(n \cdot \sqrt{m})$

Table 6.4: *Big-O Costs for FFPIR*

Chapter 7

Results and Discussion

To determine whether the *FFPIR* implementation is better in comparison to the provided alternative *FrodoPIR*, we analyze and compare their costs in terms of **communication**, **computation**, and **storage** across **offline** and **online phases** for both the **client** and **server**.

7.1 Amortization and Parameter Explanation

The comparative analysis relies on key parameters and amortization considerations to properly scale costs across multiple clients and queries. Below, we define the parameters and discuss their role in cost calculation:

7.1.1 Parameters

- m: The total number of database elements stored on the server. This parameter directly determines the size of the database and influences both communication and computation costs.
- $\sqrt{m}$: The square root of the total database size. In the FoldingFrodoPIR protocol, the database is structured as a $\sqrt{m} \times \sqrt{m}$ matrix to enable efficient querying with two smaller vectors instead of a single large one.
- C: The number of clients interacting with the server. This parameter is used to compute the total workload when multiple clients issue queries simultaneously.
- $c = \sqrt{m}$: The number of queries issued by each client. Each client generates $\sqrt{m}$ queries to retrieve data from the server, amortizing the cost over the total queries issued.

7.1.2 Amortization Considerations

In this analysis, the costs are amortized according to C clients, each issuing $c = \sqrt{m}$ queries. Amortization ensures that costs are fairly distributed across all queries. The total

number of queries is $C \cdot c = C \cdot \sqrt{m}$, which is used as the basis for comparing costs across protocols.

- **Offline Costs**: Offline costs, such as preprocessing and communication, are shared among C clients. Therefore, while the cost scales with C, it is distributed across the total queries.
- **Online Costs**: Online costs, such as query generation and response, are incurred per client query. Thus, these costs depend on $\sqrt{m}$ per client but scale to $C \cdot \sqrt{m}$ for all clients.
- **Storage Costs**: Storage costs reflect the resources needed to maintain precomputed data or state for all C clients, each issuing $\sqrt{m}$ queries.

By considering the combined workload of C clients, the use of $C \cdot \sqrt{m}$ ensures an accurate representation of the system's overall cost.

7.1.3 Comparison Methodology

The comparison is structured around the following key components:

- **Communication Costs:**
  - *Offline Phase:* Cost of precomputing and transmitting data.
  - *Online Phase:* Cost of transmitting query vectors and responses.
- **Computation Costs:**
  - *Client:* Preprocessing vectors and generating encrypted queries.
  - *Server:* Preprocessing the database and responding to queries.
- **Storage Costs:**
  - *Client:* Precomputed query data.
  - *Server:* Precomputed matrices and database.

Component	FrodoPIR	*FFPIR*	Advantage
Offline Client Communication	-	$O(C \cdot \sqrt{m})$	FrodoPIR (No offline communication)
Online Client Communication	$O(m)$	$O(\sqrt{m})$	FFPIR (Reduced query communication)
Offline Client Computation	$O(m)$	$O(C \cdot \sqrt{m})$	FrodoPIR (Lower offline cost)
Online Client Computation	$O(1)$	$O(\sqrt{m})$	FrodoPIR (Minimal online computation)
Offline Server Communication	$O(\lambda \cdot \sqrt{m})$	$O(C \cdot \sqrt{m})$	FrodoPIR (Smaller offline data sent)
Online Server Communication	$O(1)$	$O(\sqrt{m})$	FrodoPIR (Smaller server response)
Offline Server Computation	$O(\sqrt{m}/C)$	$O(C \cdot \sqrt{m})$	FrodoPIR (More efficient offline prep)
Online Server Computation	$O(m)$	$O(\sqrt{m})$	FFPIR (Better scalability)
Client Storage	$O(\lambda)$	$O(C \cdot \sqrt{m})$	FrodoPIR (Less storage required)
Server Storage	$O(\lambda \cdot \sqrt{m})$	$O(\lambda \cdot \sqrt{m})$	Tie

Table 7.1: *Comparison Between Frodo and FFPIR*

7.1.4 Comparative Analysis

The comparison of *FrodoPIR* and *FFPIR* highlights distinct advantages and trade-offs in their design:

Offline Phase

- **Client Communication:** In the offline phase, *FrodoPIR* requires no communication from the client, giving it an edge over *FFPIR*, where the client communicates $O(C \cdot \sqrt{m})$.
- **Server Computation and Communication:** *FrodoPIR* is more efficient due to its lighter preprocessing, requiring only $O(\sqrt{m}/C)$, compared to $O(n \cdot \sqrt{m})$ for *FFPIR*.

Online Phase

- **Client Communication:** *FFPIR* significantly outperforms *FrodoPIR*, as it transmits only $O(\sqrt{m})$ query vectors compared to $O(m)$ in *FrodoPIR*.
- **Client Computation:** *FrodoPIR* maintains an advantage here, requiring $O(1)$ operations, whereas *FFPIR* involves $O(\sqrt{m})$.
- **Server Computation:** Both protocols scale well, but *FFPIR* can handle larger databases more efficiently during query processing.

Storage Costs

- **Client Storage:** *FrodoPIR* has a minimal storage requirement of $O(\lambda)$, compared to $O(C \cdot \sqrt{m})$ for *FFPIR*.
- **Server Storage:** Both protocols require $O(\lambda \cdot \sqrt{m})$ for preprocessed data, making this a tie.

7.2 Comparison of Function Costs Between FrodoPIR and FFPIR

To better understand the differences between *FrodoPIR* and *FFPIR*, we can also compare the temporal complexities and communication costs of each function: server preprocessing (`spreproc`), client preprocessing (`cpreproc`), query generation (`query`), server response (`respond`), and client postprocessing (`process`).

The following table summarizes these comparisons:

Phase/Function	FrodoPIR Timing Complexity	FrodoPIR Communication	FFPIR Timing Complexity	FFPIR Communication
spreproc	$O(nm)$	$O(\lambda \cdot \sqrt{m})$	$O(n\sqrt{m^2})$	$O(C \cdot \sqrt{m})$
cpreproc	$O(c \cdot (nm + m))$	$O(\lambda)$	$O(c \cdot (n\sqrt{m} + \sqrt{m}))$	$O(\lambda)$
query	$O(1)$	$O(m)$	$O(1)$	$O(\sqrt{m})$
respond	$O(m \cdot \omega)$	$O(1)$	$O(\sqrt{m^2})$	$O(\sqrt{m})$
process	$O(\omega)$	$O(1)$	$O(1)$	$O(1)$

Table 7.2: *Timing and Communication Complexities of FrodoPIR and FFPIR*

7.2.1 Explanation of Function Comparisons

Server Preprocessing (`spreproc`)

- **FrodoPIR:** The server constructs matrix $\mathbf{A}$ and computes $\mathbf{M} = \mathbf{A} \cdot \mathbf{D}$, which involves $O(nm)$ operations.
- **FFPIR:** The server constructs a smaller $\mathbf{A}$ for $n \times \sqrt{m}$ and computes $\mathbf{M} = \mathbf{A} \cdot \mathbf{D}$, where $\mathbf{D}$ is represented as a $\sqrt{m} \times \sqrt{m}$ matrix. This leads to $O(n\sqrt{m}^2)$ complexity.
- **Communication:** FrodoPIR requires $O(\lambda \cdot \sqrt{m})$ communication, while FFPIR involves $O(C \cdot \sqrt{m})$ communication for preprocessing results.

Client Preprocessing (`cpreproc`)

- **FrodoPIR:** The client regenerates $\mathbf{A}$, computes $b_j = \mathbf{s}_j^T \mathbf{A} + \mathbf{e}_j^T$, and $c_j = \mathbf{s}_j^T \mathbf{M}$, for c queries, resulting in $O(c \cdot (nm + m))$ complexity.
- **FFPIR:** Similar steps are performed but over $n \times \sqrt{m}$ and $\sqrt{m} \times \sqrt{m}$ matrices, resulting in $O(c \cdot (n\sqrt{m} + \sqrt{m}))$.
- **Communication:** Both protocols require $O(\lambda)$ communication for downloading public parameters.

Query Generation (`query`)

- **FrodoPIR and FFPIR:** Both protocols involve simple vector operations to generate $\tilde{b}$, resulting in $O(1)$ complexity.
- **Communication:** FrodoPIR sends $O(m)$, while FFPIR reduces this to $O(\sqrt{m})$.

Server Response (`respond`)

- **FrodoPIR:** The server multiplies $\tilde{b}^T$ with $\mathbf{D}$, resulting in $O(m \cdot \omega)$, where $\omega = \lceil w / \log(\rho) \rceil$.
- **FFPIR:** The server multiplies $\tilde{b}^T$ with the $\sqrt{m} \times \sqrt{m}$ matrix $\mathbf{D}$, yielding $O(\sqrt{m}^2)$.
- **Communication:** FrodoPIR sends $O(1)$, whereas FFPIR sends $O(\sqrt{m})$.

Client Postprocessing (`process`)

- **FrodoPIR:** Decrypting $\tilde{c}$ to retrieve x involves $O(\omega)$ complexity.
- **FFPIR:** Since only one Paillier cryptogram is decrypted, the complexity is $O(1)$.
- **Communication:** Both protocols have $O(1)$ communication for this phase.

7.3 Discussion

The comparison at at the table 7.1 outputs distinct advantages for each protocol depending on the context:

- **FFPIR:** The primary advantage of *FFPIR* lies in its reduced online communication costs, which scale as $O(\sqrt{m})$, compared to $O(m)$ for *FrodoPIR*. This makes *FFPIR* particularly suitable for large databases or scenarios with limited bandwidth.
- **FrodoPIR:** *FrodoPIR*, on the other hand, is better suited for small databases or systems with constrained resources. Its simplicity and minimal offline and online costs for the client make it ideal for lightweight deployments.

The comparison at the table 7.2 is another way to analyze the costs of *FrodoPIR* and *FFPIR* functions. The results show:

- **FrodoPIR:** Performs better in offline preprocessing for the server due to simpler matrix structures and avoids complexities introduced by the folded database representation.
- **FFPIR:** Excels in reducing online communication and server computation during query response, making it highly suitable for large database systems with multiple queries.

The choice between *FrodoPIR* and *FFPIR* depends on application requirements. Using *FFPIR* seems more acceptable when handling large datasets with multiple queries, where online communication efficiency is critical. Whereas *FrodoPIR* demonstrates a significant improvement in scalability for large-scale applications offering an alternative for simpler use cases.

Chapter 8

Conclusion

This thesis presented a detailed exploration and implementation of the *FoldingFrodo* protocol, a privacy-preserving system leveraging homomorphic encryption and lattice-based cryptography to optimize Private Information Retrieval (PIR). The work focused on the theoretical foundations, practical implementation, and comprehensive analysis of the protocol's efficiency and trade-offs compared to traditional PIR schemes, such as Frodo.

8.1 Summary of Contributions

The main contributions of this thesis can be summarized as follows:

- **Protocol Design and Implementation:** The *FoldingFrodo* protocol was implemented in a modular and extensible manner, incorporating lattice-based encryption through LWE and homomorphic operations using the Paillier cryptosystem. Each component, including server preprocessing, client query generation, server response, and client postprocessing, was carefully designed and implemented.

- **Cost Analysis:** Comprehensive analyses of computational, communication, and storage costs were conducted. The trade-offs between offline and online phases were examined, providing insights into the efficiency improvements offered by *FoldingFrodo*. Amortized costs were also calculated, highlighting the protocol's scalability for multiple queries.

- **Comparison with Traditional PIR:** The protocol was benchmarked against the FrodoPIR scheme. The results demonstrated significant gains in online communication efficiency, with a reduction from $O(m)$ to $O(\sqrt{m})$, while acknowledging the increased computational overhead during the offline phase.

- **Pseudocode and Formalization:** The thesis provided a formalized presentation of the *FoldingFrodo* protocol, including pseudocode for all major components. This structured documentation facilitates further development and understanding of the protocol.

8.2 Key Findings

From the analyses and experiments conducted in this thesis, the following key findings emerged:

- The use of matrix-based decomposition and indicator vectors allows *FoldingFrodo* to achieve reduced online communication costs, making it more practical for scenarios with bandwidth constraints.
- The computational cost of server preprocessing, though high, can be amortized over multiple queries, making the protocol suitable for batch operations.
- Homomorphic encryption, while introducing computational overhead, enables secure and efficient query processing without compromising privacy.
- Storage requirements for both client and server scale with the parameters $\sqrt{m}$ and n, offering a manageable trade-off for improved performance.

8.3 Challenges and Limitations

Despite its advantages, the *FoldingFrodo* protocol has certain challenges and limitations:

- **Computational Overhead:** The reliance on lattice-based cryptography and homomorphic encryption introduces significant computational demands, particularly during the server preprocessing and response phases.
- **Scalability Concerns:** While the protocol performs well for moderate values of m, larger database sizes may require further optimization to maintain efficiency.
- **Parameter Tuning:** The security and performance trade-offs depend heavily on the choice of parameters (n, m, q, t). Balancing these parameters for different application scenarios remains a challenge.

8.4 Future Work

Building on the foundations laid in this thesis, several avenues for future research and development are proposed:

- **Full Protocol Implementation:** Develop a complete end-to-end implementation of the *FoldingFrodo* as a standalone system, integrating all components and optimizations.
- **Optimization of Homomorphic Operations:** Investigate alternative homomorphic encryption schemes or optimizations to reduce the computational cost of encryption, multiplication, and addition operations.
- **Dynamic Parameter Adaptation:** Develop techniques for dynamically adjusting protocol parameters based on real-time workload and security requirements.

- **Extended Applications:** Explore the application of *FoldingFrodo* in multi-party computation, secure database queries, and privacy-preserving machine learning.
- **Benchmarking and Real-World Deployment:** Conduct extensive benchmarking against other PIR protocols in diverse network and database settings to evaluate real-world performance and adaptability.
- **Quantum Resistance:** Further analyze the protocol's resilience against quantum attacks, especially concerning the choice of LWE parameters and cryptographic primitives.
- **Handling Larger Database Entries:** Investigate adaptations of the protocol for scenarios where database entries exceed the plaintext modulus t (i.e., do not fit in $\mathbb{Z}_t$), requiring alternative encoding schemes or multiple modular computations to accommodate larger data sizes.

8.5 Concluding Remarks

The *FoldingFrodo* protocol represents a significant step forward in the development of efficient, privacy-preserving systems. By leveraging advances in lattice-based cryptography and homomorphic encryption, the protocol achieves a balance between security and performance, making it a practical deployment in modern data-intensive applications. The reductions in online communication costs, combined with the scalability offered through precomputations, provide a strong foundation for its use in scenarios with strict privacy and efficiency requirements.

Through this thesis, the theoretical and practical foundations of *FoldingFrodo* have been established, paving the way for further innovation and research in privacy-preserving computation. The implementation idea and analysis provided in this work have highlighted the potential of matrix-based approaches and hybrid encryption methods to address traditional limitations in Private Information Retrieval (PIR) protocols.

Looking ahead, this thesis is not the end but rather a continuously ongoing research into PIR and related privacy-preserving technologies. I intend to continue refining and optimizing the *FoldingFrodo* protocol as part of an effort to further reduce computational and communication overheads. The goal is to eventually develop and publish a paper detailing these improvements and benchmarking the protocol against state-of-the-art PIR systems.

In addition, PIR remains a central focus of my research interests, as I aim to explore its intersections with other fields, such as privacy-preserving machine learning, secure multiparty computation, and quantum-resilient cryptography. The work presented in this thesis will serve as a strong foundation for future projects, contributing to the broader goal of enabling secure, efficient, and privacy-preserving access to sensitive data in diverse application domains.

It is hoped that the findings and implementations presented here will serve as a valuable resource not only for myself but also for other researchers in the field. By addressing current challenges and identifying promising directions for future work, this thesis contributes

to the collective effort of advancing privacy-preserving technologies for the benefit of individuals and organizations in an increasingly data-driven world.

Annex A

An overview of the FrodoPIR protocol

FrodoPIR (Davidson *et al.*, 2022) implementation demonstrates its practicality and scalability:

- **Server Preprocessing:** The server preprocesses the database into a compressed matrix format using a global public seed and a pseudorandom generator. This reduces the storage size to approximately m/λ, where m is the database size and λ is the security parameter.
- **Client Query Generation:** Clients generate encrypted query vectors using the preprocessed parameters. The query vectors appear uniformly random to the server, preserving client privacy.
- **Server Response and Client Decryption:** The server performs efficient matrix-vector multiplications to respond to client queries. The client decrypts the response using precomputed parameters to retrieve the desired database entry.

In this annex, the FrodoPIR protocol is described in detail as exactly in (Davidson *et al.*, 2022), including the setup, preprocessing, and online phases.

A.1 FrodoPIR Scheme

The protocol consists of 5 parts, where:

- **Offline**

    1. In the **offline** phase, the server interprets the database as a matrix and applies a **compression** function to reduce its size, creating a global parameter. This compression function reduces the size of the DB by m/λ, where λ is the security parameter and m is the number of elements in the DB. Therefore, note that the parameter **is not** linear in the size of the DB.

2. Also in the **offline** phase, the client downloads the public parameters and computes c sets of preprocessed query parameters.

- **Online**

1. In the **online** phase, the client uses a set of preprocessed query parameters to create the encrypted query vector and sends it to the server.
2. Still in the **online** phase, the server responds to this query by multiplying the query vector with the DB matrix.
3. Finally, the client returns the result by decrypting the response using the preprocessed query parameters.

A.1.1 Setup

Let $\mathcal{S}$ be the server containing the database DB that each client tries to access. The database DB is a vector of m elements of w bits. Each entry is associated with an index i that corresponds to a position in the vector. For now, let's assume that the client knows which index it would like to query during the online phase of the protocol. We assume that there are C clients making a maximum of c queries to the database DB. Note that the LWE instance is used: let n be the dimension and q the modulus, let ρ be the number of bits within each entry of the DB matrix, where $0 < \rho < q$. Let χ be the uniform distribution over $\{-1, 0, 1\}$, let λ be the security parameter, and use $\texttt{PRG}(\mu, x, y, q)$ to denote a pseudorandom generator that uses a seed $\mu \leftarrow \{0,1\}^\lambda$ in the matrix $\mathbb{Z}_q^{x \times y}$. Note that $\omega = \lceil w/log(\rho) \rceil$

A.1.2 Preprocessing phase

1. **Server Setup:** The server constructs the database with m elements and samples the seed $\mu \in \{0,1\}^\lambda$
2. **Server Preprocessing:** Here the server derives the matrix

$$\boldsymbol{A} \leftarrow PRG(\mu, n, m, q),$$

and returns

$$\boldsymbol{D} \leftarrow parse(DB, \rho),$$

where the parse function encodes the database into a matrix

$$\boldsymbol{D} \in \mathbb{Z}_\rho^{m \times \omega}.$$

To generate the public parameters, the server runs

$$\boldsymbol{M} \leftarrow \boldsymbol{A} \cdot \boldsymbol{D}$$

and publishes the pair $(\mu, \boldsymbol{M})$.

3. **Client Preprocessing:** Each client downloads the pair $(\mu, \boldsymbol{M})$ and runs

$$\boldsymbol{A} \leftarrow PRG(\mu, n, m, q).$$

The client then samples c vectors $\boldsymbol{s_j} \leftarrow (\chi)^n$ and $\boldsymbol{e_j} \leftarrow (\chi)^m$.

Finally, it performs:

$$\boldsymbol{b_j} \leftarrow \boldsymbol{s_j^T} \cdot \boldsymbol{A} + \boldsymbol{e_j^T} \in \mathbb{Z}_q^m$$

and

$$\boldsymbol{c_j} \leftarrow \boldsymbol{s_j}^T \cdot \boldsymbol{M} \in \mathbb{Z}_q^\omega$$

and stores the pair $\boldsymbol{X} = (\boldsymbol{b_j}, \boldsymbol{c_j})$

A.1.3 Online phase

1. **Client query generation (FPIR.query)**: For the index i that the client wants to query, the client generates a vector

   $$\boldsymbol{f_i} = (0, \ldots, 0, q/p, 0, \ldots, 0)$$

   that is all zeros, except where $\boldsymbol{f_i}[i] = q/p$. The client then extracts a pair $(\boldsymbol{b}, \boldsymbol{c})$ from the internal state st and computes

   $$\tilde{\boldsymbol{b}} \leftarrow \boldsymbol{b} + \boldsymbol{f_i}$$

   and sends this $\tilde{\boldsymbol{b}}$ to the server

2. **Server response (FPIR.respond)**: The server receives $\tilde{\boldsymbol{b}}$ from the client and responds with

   $$\tilde{\boldsymbol{c}} \leftarrow \tilde{\boldsymbol{b}}^T \cdot \boldsymbol{D}$$

3. **Client postprocessing (FPIR.process)**: The client receives $\tilde{\boldsymbol{c}}$ and extracts

   $$\boldsymbol{x} \leftarrow \lfloor \tilde{\boldsymbol{c}} - \boldsymbol{c} \rceil_\rho$$

Annex B

Pseudocodes for FoldingFrodoPIR

In this annex, we present the pseudocodes for the *FoldingFrodoPIR* protocol.

Program B.1 Server Preprocessing.

```
1  FUNCTION server_preprocessing(n, sqrt_m, q, D, mu)
2    A ← PRG(mu, n, sqrt_m, q)  ▷ Generate matrix A pseudorandomly
3    M ← zero_matrix(n, sqrt_m)  ▷ Initialize M with zeros
4    for i ← 1 to n do
5      for j ← 1 to sqrt_m do
6        for k ← 1 to sqrt_m do
7          M[i][j] ← (M[i][j] + A[i][k] * D[k][j]) mod q
8        end
9      end
10   end
11   return M, mu
12 end
```

Program B.1: *"Código"*

Program B.2 Client Preprocessing.

```
1  FUNCTION client_preprocessing(n, sqrt_m, q, C)
2    precomputed_data ← empty_list()
3    for query ← 1 to C do
4      s ← generate_secret(n)  ▷ Generate secret vector
5      e ← generate_error(sqrt_m)  ▷ Generate error vector
6      b ← (transpose(s) * A + transpose(e)) mod q
7      c ← (transpose(s) * M) mod q
8      append (b, c) to precomputed_data
9    end
10   return precomputed_data
```

cont ⟶

```
⟶ cont
end
```

Program B.2: *"Código"*

Program B.3 Query Generation.

```
FUNCTION query_generation(i, j, sqrt_m, q, t)
    f_i ← zero_vector(sqrt_m)
    f_j ← zero_vector(sqrt_m)
    f_i[i] ← floor(q / t)
    f_j[j] ← floor(q / t)
    hat_b ← empty_list()
    for k ← 1 to sqrt_m do
        hat_b[k] ← Paillier.enc(f_j[k])   ▷ Encrypt using Paillier
    end
    return (f_i, hat_b)
end
```

Program B.3: *"Código"*

Program B.4 Server Response.

```
FUNCTION server_response(D, tilde_b, hat_b, q)
    tilde_c ← zero_vector(sqrt_m)
    for k ← 1 to sqrt_m do
        for l ← 1 to sqrt_m do
            tilde_c[k] ← (tilde_c[k] + tilde_b[l] * D[l][k]) mod q
        end
    end
    hat_s ← Paillier.enc(1)   ▷ Initialize encrypted response
    for k ← 1 to sqrt_m do
        temp ← Paillier.mul(hat_b[k], tilde_c[k])
        hat_s ← Paillier.add(hat_s, temp)
    end
    return hat_s
end
```

Program B.4: *"Código"*

Program B.5 Client Decryption.

```
FUNCTION client_decryption(hat_s, q, t)
    y ← Paillier.dec(hat_s)   ▷ Decrypt server response
    result ← round((t * y) / q)   ▷ Scale and round result
    return result
```

cont ⟶

$\longrightarrow$ *cont*

5 **end**

Program B.5: *"Código"*

References

[Aguilar-Melchor *et al.* 2014] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. *XPIR: Private Information Retrieval for Everyone*. Cryptology ePrint Archive, Paper 2014/1025. https://eprint.iacr.org/2014/1025. 2014. URL: https://eprint.iacr.org/2014/1025 (cit. on pp. x, 2, 32–35).

[Ajtai 1996] Miklós Ajtai. "Generating hard instances of lattice problems". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1996, pp. 99–108. DOI: 10.1145/237814.237838 (cit. on p. 9).

[Ali *et al.* 2019] Asra Ali *et al. Communication–Computation Trade-offs in PIR*. Cryptology ePrint Archive, Paper 2019/1483. https://eprint.iacr.org/2019/1483. 2019. URL: https://eprint.iacr.org/2019/1483 (cit. on p. 30).

[Angel *et al.* 2018] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. "Pir with compressed queries and amortized query processing". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 962–979. DOI: 10.1109/SP.2018.00062 (cit. on pp. x, 2, 30, 35, 38, 45).

[Beimel and Ishai 2001] Amos Beimel and Yuval Ishai. "Information-theoretic private information retrieval: a unified construction". In: *Automata, Languages and Programming*. Ed. by Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 912–926. ISBN: 978-3-540-48224-6 (cit. on pp. 1, 28, 29).

[Beimel, Ishai, and Malkin 2000] Amos Beimel, Yuval Ishai, and Tal Malkin. "Reducing the servers computation in private information retrieval: pir with preprocessing". In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 55–73. DOI: 10.1007/3-540-44598-6_4. URL: https://www.iacr.org/archive/crypto2000/18800056/18800056.pdf (cit. on p. 2).

[Boneh *et al.* 2005] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-dnf formulas on ciphertexts". In: *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*. Vol. 3378. Lecture Notes in Computer Science. Springer, 2005, pp. 325–341. DOI: 10.1007/978-3-540-30576-7_18. URL: https://iacr.org/archive/tcc2005/3378_325/3378_325.pdf (cit. on p. 22).

[Brakerski *et al.* 2014] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "Leveled fully homomorphic encryption without bootstrapping". *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), 13:1–13:36. DOI: 10.1145/2633600 (cit. on p. 22).

[Celi and Davidson 2024] Sofía Celi and Alex Davidson. *Call Me By My Name: Simple, Practical Private Information Retrieval for Keyword Queries*. Cryptology ePrint Archive, Paper 2024/092. https://eprint.iacr.org/2024/092. 2024. URL: https://eprint.iacr.org/2024/092 (cit. on p. 2).

[Chillotti 2018] Ilaria Chillotti. "Vers l'efficacité et la sécurité du chiffrement homomorphe et du cloud computing". Thèse de doctorat en informatique, préparée à l'Université de Versailles Saint-Quentin-en-Yvelines, École doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC). Thèse de doctorat. Versailles, France: Université Paris-Saclay, mai 2018. URL: https://theses.fr/2018SACLV020 (cit. on p. 22).

[B. Chor *et al.* 1995] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. "Private information retrieval". In: *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*. FOCS '95. USA: IEEE Computer Society, 1995, p. 41. ISBN: 0818671831 (cit. on p. 1).

[Benny Chor and Gilboa 1997] Benny Chor and Naftali Gilboa. "Computationally private information retrieval". In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1997, pp. 304–313. DOI: 10.1145/258533.258594 (cit. on pp. 1, 28, 29).

[Benny Chor, Goldreich, *et al.* 1998] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. "Private information retrieval". *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981. DOI: 10.1145/293347.293350 (cit. on pp. 28, 29).

[Corrigan-Gibbs, Henzinger, *et al.* 2022] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. *Single-Server Private Information Retrieval with Sublinear Amortized Time*. Cryptology ePrint Archive, Paper 2022/081. https://eprint.iacr.org/2022/081. 2022. URL: https://eprint.iacr.org/2022/081 (cit. on pp. x, 2, 37–40).

[Corrigan-Gibbs and Kogan 2020] Henry Corrigan-Gibbs and Dmitry Kogan. "Private information retrieval with sublinear online time". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 44–75. isbn: 978-3-030-45721-1 (cit. on pp. 2, 28, 30, 31).

[Daemen and Rijmen 1999] Joan Daemen and Vincent Rijmen. "Aes proposal: rijndael" (1999). Available at https://csrc.nist.gov/publications/detail/fips/197/final (cit. on pp. 6, 16).

[Davidson *et al.* 2022] Alex Davidson, Gonçalo Pestana, and Sofía Celi. *FrodoPIR: Simple, Scalable, Single-Server Private Information Retrieval*. Cryptology ePrint Archive, Paper 2022/981. https://eprint.iacr.org/2022/981. 2022. url: https://eprint.iacr.org/2022/981 (cit. on pp. x, 2, 3, 32, 39, 41, 42, 45, 65).

[Dijk *et al.* 2010] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. "Fully homomorphic encryption over the integers". In: *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Vol. 6110. Lecture Notes in Computer Science (LNCS). Springer, 2010, pp. 24–43. doi: 10.1007/978-3-642-13190-5_2 (cit. on p. 24).

[Fan and Vercauteren 2012] Junfeng Fan and Frederik Vercauteren. "Somewhat practical fully homomorphic encryption". In: *Proceedings of the 11th International Conference on Cryptology in India (INDOCRYPT)*. Vol. 7668. Lecture Notes in Computer Science (LNCS). Springer, 2012, pp. 1–16. doi: 10.1007/978-3-642-34961-4_1 (cit. on p. 23).

[J. D. Feo *et al.* 2019] J. De Feo, D. Jao, and J. Plût. *Towards Quantum-Resistant Cryptosystems from Supersingular Isogeny Diffie-Hellman*. NIST Submission. 2019. url: https://sike.org (cit. on p. 17).

[Gentry 2009] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC '09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. isbn: 9781605585062. doi: 10.1145/1536414.1536440. url: https://doi.org/10.1145/1536414.1536440 (cit. on pp. 22, 25).

[Gentry and Ramzan 2005] Craig Gentry and Zulfikar Ramzan. "Single-database private information retrieval with constant communication rate". In: *Proceedings of ICALP*. Vol. 3580. Lecture Notes in Computer Science. Springer, 2005, pp. 803–815. doi: 10.1007/11523468_65 (cit. on p. 30).

[Grover 1996] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. 1996, pp. 212–219. doi: 10.1145/237814.237866 (cit. on p. 16).

[Ham 2021] Jeroen Van Der Ham. "Toward a better understanding of "cybersecurity"". *Digital Threats* 2.3 (June 2021). DOI: 10.1145/3442445. URL: https://doi.org/10.1145/3442445 (cit. on p. 5).

[Haviv and Regev 2007] Ishay Haviv and Oded Regev. "On the lattice isomorphism problem". In: *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity (CCC)*. IEEE, 2007, pp. 129–140. DOI: 10.1109/CCC.2007.11 (cit. on p. 9).

[Henzinger *et al.* 2022] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. *One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval*. Cryptology ePrint Archive, Paper 2022/949. https://eprint.iacr.org/2022/949. 2022. URL: https://eprint.iacr.org/2022/949 (cit. on p. 2).

[Henzinger *et al.* 2023] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. "One server for the price of two: simple and fast Single-Server private information retrieval". In: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 3889–3905. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger (cit. on pp. x, 32, 41, 43, 45).

[Jao and L. D. Feo 2011] David Jao and Luca De Feo. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Post-Quantum Cryptography (PQCrypto 2011)*. Vol. 7071. Lecture Notes in Computer Science. Springer, 2011, pp. 19–34. DOI: 10.1007/978-3-642-25405-5_2 (cit. on p. 17).

[Koblitz 1994] Neal Koblitz. *A Course in Number Theory and Cryptography*. Vol. 114. Graduate Texts in Mathematics. Springer, 1994 (cit. on pp. 6, 16).

[Kogan and Corrigan-Gibbs 2021] Dmitry Kogan and Henry Corrigan-Gibbs. "Private blocklist lookups with checklist". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 875–892. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/kogan (cit. on p. 1).

[Kushilevitz and Ostrovsky 1997] E. Kushilevitz and R. Ostrovsky. "Replication is not needed: single database, computationally-private information retrieval". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. 1997, pp. 364–373. DOI: 10.1109/SFCS.1997.646125 (cit. on p. 30).

[McEliece 1978] Robert J. McEliece. "A public-key cryptosystem based on algebraic coding theory". *DSN Progress Report* 42-44 (1978), pp. 114–116 (cit. on p. 17).

[Menon and Wu 2022] Samir Jordan Menon and David J. Wu. *Spiral: Fast, High-Rate Single-Server PIR via FHE Composition*. Cryptology ePrint Archive, Paper 2022/368. https://eprint.iacr.org/2022/368. 2022. URL: https://eprint.iacr.org/2022/368 (cit. on pp. 2, 32).

[Micciancio and Goldwasser 2002] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*. Vol. 671. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 2002 (cit. on pp. 9, 10).

[Mittal *et al.* 2011] Prateek Mittal, Femi Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. "PIR-Tor: scalable anonymous communication using private information retrieval". In: *20th USENIX Security Symposium (USENIX Security 11)*. San Francisco, CA: USENIX Association, Aug. 2011. URL: https://www.usenix.org/conference/usenix-security-11/pir-tor-scalable-anonymous-communication-using-private-information (cit. on p. 1).

[Mughees *et al.* 2021] Muhammad Haris Mughees, Hao Chen, and Ling Ren. *OnionPIR: Response Efficient Single-Server PIR*. Cryptology ePrint Archive, Paper 2021/1081. https://eprint.iacr.org/2021/1081. 2021. URL: https://eprint.iacr.org/2021/1081 (cit. on pp. 2, 31).

[Nielsen and Chuang 2010] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010 (cit. on p. 15).

[Paillier 1999] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238. ISBN: 978-3-540-48910-8 (cit. on pp. 4, 11, 20, 45, 46).

[Patel *et al.* 2018] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. *Private Stateful Information Retrieval*. Cryptology ePrint Archive, Paper 2018/1083. https://eprint.iacr.org/2018/1083. 2018. DOI: 10.1145/3243734.3243821. URL: https://eprint.iacr.org/2018/1083 (cit. on p. 32).

[Peikert 2015] Chris Peikert. *A Decade of Lattice Cryptography*. Cryptology ePrint Archive, Paper 2015/939. 2015. URL: https://eprint.iacr.org/2015/939 (cit. on pp. 7, 16, 19, 40).

[Pereira and Morais 2021] Hilder V. L. Pereira and Eduardo Morais. "Introdução à criptografia completamente homomórfica com implementação em Sage". In: *Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Porto Alegre, Brazil: Sociedade Brasileira de Computação – SBC, 2021. Chap. 1, pp. 1–50. DOI: 10.5753/sbc.7165.8 (cit. on p. 25).

[Regev 2009] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: 10.1145/1568318.1568324. URL: https://doi.org/10.1145/1568318.1568324 (cit. on pp. 20, 40).

[Rivest *et al.* 1978] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342 (cit. on pp. 6, 22).

[Robins 2021] Sinai Robins. *A friendly invitation to fourier analysis on polytopes*. Apr. 2021. DOI: 10.48550/arXiv.2104.06407 (cit. on p. 8).

[Shor 1994] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700 (cit. on pp. 5, 16).

[Sion and Carbunar 2007] Radu Sion and Bogdan Carbunar. "On the practicality of private information retrieval". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2007. URL: https://www.ndss-symposium.org/ndss2007/practicality-private-information-retrieval/ (cit. on p. 33).

[Standards and (NIST) 1977] National Institute of Standards and Technology (NIST). "Data encryption standard (des)". *Federal Information Processing Standards Publication 46* (1977) (cit. on p. 6).

[Stoneburner *et al.* 2004] G Stoneburner, Clark Hayden, and Alexis Feringa. *Engineering Principles for IT Security (A Baseline for Achieving Security), Revision A*. en. 2004-06-21 2004. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=151294 (cit. on p. 5).

[Wikipedia contributors 2024a] Wikipedia contributors. *Lattice (group)*. Accessed: 2024-12-10. 2024 (cit. on pp. vii, 7).

[Wikipedia contributors 2024b] Wikipedia contributors. *Lattice problem*. Accessed: 2024-12-10. 2024 (cit. on pp. vii, 9, 10).

[Zhou *et al.* 2023] Mingxun Zhou, Andrew Park, Elaine Shi, and Wenting Zheng. *Piano: Extremely Simple, Single-Server PIR with Sublinear Server Computation*. Cryptology ePrint Archive, Paper 2023/452. https://eprint.iacr.org/2023/452. 2023. URL: https://eprint.iacr.org/2023/452 (cit. on p. 32).