Design and Analysis of Privacy-Preserving Protocols

Russell Craig Paulet

This thesis is presented in fulfilment
of the requirements for the degree of Doctor of Philosophy

College of Engineering and Science
Victoria University

March 2013

Abstract

More and more of our daily activities are using the Internet to provide an easy way to get access to instant information. The equipment enabling these interactions is also storing information such as: access time, where you are, and what you plan to do. The ability to store this information is very convenient but is also the source of a major concern: once data are stored, it must be protected. If the data was left unprotected, then people would feel reluctant to use the service. The aim of this thesis is to remove the need to store such data, while still maintaining overall utility, by designing and analysing privacy preserving protocols.

The theory for creating privacy preserving protocols began with the notion of oblivious transfer, which was first published by Rabin in 1981. Oblivious transfer is defined for two players Alice and Bob, where initially Alice owns an index and Bob owns many secrets. When a round oblivious transfer concludes, Alice learns only one of Bob's secrets and Bob learns nothing. A closely relative primitive known as private information retrieval has a similar definition, except the communication cost is strictly less than the trivial solution of downloading the entire database and inspecting it locally. Also, there is no privacy requirement for the database.

The consequence of requiring the communication cost to be less than the transfer of the entire database in private information retrieval is that the computational cost can be prohibitively large. This is especially true for large databases, where the response times can be up to 30 minutes or more. In this thesis, we explore and revise this requirement and show that if we base a private information retrieval scheme on simple (fast) operations, such as those based on linear operations, and then we can achieve a more practical implementation.

Using the combined theory of private information retrieval and oblivious transfer, we develop practical privacy preserving protocols. These protocols are developed with respect to the constraints and requirements of an application domain. The first application domain that is considered is about performing private location based queries. In simple terms a location query is where a user queries a location database to find the answers to questions like: where is the nearest shop? And where is the nearest cinema? Using a combination of oblivious transfer and private information retrieval, we construct a practical solution that protects the owner of the query and the owner of the database.

The next application domain that is considered is in the context of data mining. A privacy preserving protocol is developed to satisfy the requirements of an association rule mining application for two parties. In this scenario, the two parties cannot simply transfer their data to the opposing site because of legal or competitive reasons. In this thesis, a protocol is developed that removes this requirement and allows both parties to obtain the result.

Overall, this thesis makes significant contributions in both theory and practice. It presents useful and practical applications. At the same time, the claims for correctness, security and security are thoroughly explored and evaluated.

Declaration

“I, Russell Craig Paulet, declare that the PhD thesis entitled *Design and Analysis of Privacy-Preserving Protocols* is no more than 100,000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work”.

Signature Date

Acknowledgements

I wish to thank my humble principal supervisor Dr. Xun Yi. I am always amazed by his breadth of knowledge and his tenacity to find the best solution. Not only is he one of the most intelligent people I have met, he is also one of the nicest. I feel blessed to have been given the opportunity to learn and work with him. Most of all, I thank him for the many hours of helpful discussions in his office. I must also thank my associate supervisor Dr. Alasdair McAndrew, who provided a great deal of encouragement and support when I needed it the most. And, at times, he provided me with a new perspective on various topics of discussion.

I make special mention of Elisa Bertino of Purdue University in West Lafayette, Indiana, United States of America. Her support with the location based query publications is greatly appreciated and recognised. In particular, her feedback on these publications helped produce higher-quality results. I also thank Raylin Tso, who joined our team briefly, and stimulated many useful discussions.

I thank the students in my local community, of which I have shared many insightful conversations. Furthermore, I appreciate the collaboration with fellow student Md. Golam Kaosar. I also acknowledge and thank the academic support staff for providing me with a fantastic environment for this project. Moreover, this project was made possible thanks to the Australian Research Council Discovery Project *Private Data Warehouse Query*, with project code DP0988411.

Last, but most of all, I owe a deep debt of gratitude to my family. Their support does not go unnoticed or unappreciated. Thank you for being there for me.

Papers published/submitted during the author's candidature

Journal

- M. Golam Kaosar, **R. Paulet**, X. Yi, 'Fully homomorphic encryption based two-party association rule mining', Data & Knowledge Engineering, Elsevier, Available online 31 March 2012
- X. Yi, M. Golam Kaosar, **R. Paulet**, E. Bertino, 'Single-Database Private Information Retrieval from Fully Homomorphic Encryption', IEEE Transactions on Knowledge and Data Engineering (TKDE), 2012
- **R. Paulet**, M. Golam Kaosar, X. Yi, E. Bertino, 'Privacy-Preserving and Content-Protecting Location Based Queries', IEEE Transactions on Knowledge and Data Engineering (TKDE), 2012, (Accepted 14/02/2013)

Conference

- M. Golam Kaosar, **R. Paulet**, X. Yi, 'Secure Two-Party Association Rule Mining', Australasian Information Security Conference (AISC 2011), 17-20 January 2011, Perth Australia

- **R. Paulet**, Md. Golam Kaosar, Xun Yi, 'k-Anonymous Private Query Based on Blind Signature and Oblivious Transfer', 2nd International Cyber Resilience Conference, 1-2 August 2011, Perth Australia
- M. Golam Kaosar, **R. Paulet**, X. Yi, 'Optimized two Party Privacy Preserving Association Rule Mining using Fully Homomorphic Encryption', 11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2011), 24-26 October 2011, Melbourne Australia
- **R. Paulet**, M. Golam Kaosar, X. Yi, E. Bertino, 'Privacy-Preserving and Content-Protecting Location Based Queries', 28th IEEE International Conference on Data Engineering (ICDE), 1-5 April 2012, Washington, DC, USA
- **R. Paulet**, X. Yi, 'Cryptanalysis of Brenner et al.'s Somewhat Homomorphic Encryption Scheme', Australasian Information Security Conference (AISC 2013), January 29-February 1 2013, Adelaide, Australia
- X. Yi, **R. Paulet**, G. Xu, E. Bertino, 'Private Data Warehouse Queries', 18th ACM Symposium on Access Control Models and Technologies (SACMAT), June 12-14 2013, Amsterdam, The Netherlands (Accepted 10/03/2013)

Contents

General Notation and Abbreviations

Notation

$\boldsymbol{K}$ Key generation algorithm of a cryptosystem

$\boldsymbol{E}$ Encryption where the parameters and key are implied from the context

$\boldsymbol{D}$ Decryption where the parameters and key are implied from the context

$\boldsymbol{G}$ General group which is based on a set S and an operation $\star$ that satisfies certain mathematical laws with respect to the operation

$\boldsymbol{\mathcal{A}}$ General adversary when analysing the security of a protocol

$\boldsymbol{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1}$ A *symmetric* bilinear map

$\boldsymbol{OT_1^2}$ 1-out-of-2 oblivious transfer

$\boldsymbol{OT_1^n}$ 1-out-of-n oblivious transfer

$\boldsymbol{OT_{k \times 1}^N}$ An adaptive oblivious transfer scheme

$\boldsymbol{O_{Enc}}$ Encryption oracle that encrypts upon request

$\oplus$ Exclusive-OR

$\boxplus$ Addition of ciphertexts

$\boxtimes$ Multiplication of ciphertexts

Abbreviations

BGN Boneh-Goh-Nissim cryptosystem that allows for arbitrary addition and one multiplication

DB A general Database

CR Cloaking Region

GCD Greatest Common Divisor: the number which evenly divides into two or more numbers

ElGamal Public key cryptosystem based on the difficulty of finding discrete logs

LBS Location Based Service

LS Location Server

LWE Learning With Errors

IEEE Institute of Electrical and Electronic Engineers

IND-CPA Indistinguishability under a chosen plaintext attack

FHE Fully Homomorphic Encryption

Paillier Paillier encryption system where security is based on the difficulty of finding composite residues

PBR Private Block Retrieval: a PIR scheme that retrieves a block of data with each execution

PIR Private Information Retrieval

POI Points Of Interest

SIMD Single Input Multiple Data

QG Query Generation (in the context of PIR)

RG Response Generation (in the context of PIR)

RR Response Retrieval (in the context of PIR)

RSA Public key cryptosystem based on on the difficulty of factoring

List of figures

List of tables

"Imagination is more important than knowledge..."
— Albert Einstein (1879 - 1955)

Chapter 1

Introduction

Cryptography has got to be one of the most intriguing fields of applied mathematics. The definitions, although complex, can be reduced to a sequence of steps one might find when playing a game. As in games, there is a clear starting point, some rules, some players, and many ending points. The players in the game negotiate the starting point (system setup), such that they feel that the game is fair and no one can cheat.

In these cryptographic games, we usually have to make some assumptions to ensure that the game is 'fair'. Two well-known assumptions in cryptography are the discrete logarithm problem [29] and the integer factorisation problem [87]. Other assumptions have been introduced, namely the knapsack problem [70]. But systems based on this idea have been shown to be insecure [90].

The role of cryptography continues to expand in our society. Historically, it began as a means of sending secret messages using insecure mediums. Examples include simple ciphers such as the Caesar Cipher or the Hill Cipher. From these simple ciphers came methods for securely establishing secret keys between one or more parties who have never met, known collectively as *public key cryptography*. This one of the most important breakthroughs of cryptography, since it has enabled convenient e-commerce.

Our world has become more complex since the early days of cryptography. Information is being collected at an exponential rate, and it showing no signs of slowing down. Since data collection and data storing methods are advancing at a fast rate, we need advanced methods for protecting the privacy of those the data represents. Before we examine these methods, we will explore what privacy means and under what conditions a privacy-preserving protocol can be constructed.

1.1 Privacy and Privacy-Preserving Protocols

The concept of privacy is nothing new for the average human. What is new, however, is how this concept applies to electronic circuits in computers. In this case we are dealing with data and the owners of the data want to protect the privacy of what the data represents by limiting the disclosure to others. In the ideal case, there is some globally trusted entity where each party submits their input and receives the corresponding output. In this model, the trusted entity provides privacy as they are *trusted* not to reveal the inputs to anyone.

This ideal case is hardly realistic because of the ease at which data can be transmitted and intercepted on the Internet. Consider the two-party case (following the exposition by Goldreich [48]). In this scenario, there are two parties called Alice and Bob who own inputs x and y respectively, and they want to compute some function specified as $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$ as an interactive protocol. For security, this function is required to be random. Despite this random nature Alice and Bob still want to compute some non-trivial function g using function f, as $f(x,y) \overset{\text{def}}{=} (g(x,y), g(x,y))$. Privacy of the inputs of Alice and Bob are preserved if at the conclusion of the protocol, Alice and Bob can learn nothing more than the output of function g and their initial inputs.

With respect to privacy-preserving protocols, there are two privacy models of particular interest. One is called the semi-honest (or honest, but curious) model, where participants are required to follow the protocol, but are permitted to record all data collected during the execution of the protocol. At the conclusion of the protocol, the participants can analyse the data collected in attempt to extract more information than just their inputs and the output of the function. A more permissive protocol is allowed in a model called the malicious model. This adversarial model is the same as the semi-honest model, except that the participants can do anything while the protocol is executing, even abort the protocol (aborting can be used as a technique to attempt to learn the opposing party's input). Given this relationship, it makes sense to first construct a secure semi-honest protocol and then transform it into a protocol that is secure in the malicious model. This transformation usually requires a zero-knowledge proof [4] sub-protocol to ensure the messages that are communicated are correctly constructed.

Conceptually, the two-party case can be expanded to the multi-party case with the appropriate modifications. In particular, a set of t participants $p_1, p_2, ..., p_t$, with respective inputs $x_1, x_2, ..., x_t$, desire to compute some functionality $f : (\{0,1\}^*)^m \mapsto (\{0,1\}^*)^m$. In other words, the i-th participant wants to obtain the i-th value of $f(x_1, x_2, ..., x_t)$. As in the two-party model, we consider two kinds of adversarial models: semi-honest and malicious models. Obviously, these definitions must take into account that an adversary may corrupt one or many of the parties engaged in the protocol. When there are more than half participants behaving according to the semi-honest model, then this is called honest-majority. Additionally, this presents the concept of *static* and *dynamic* adversarial behaviour. In the case of multi-party computations, static adversarial behaviour means that an adversary has corrupted a set of participants at the beginning of the protocol and cannot corrupt any more during the execution of the protocol. By contrast, dynamic adversarial (sometimes called adaptive adversary) behaviour permits the adversary to corrupt a set of participants at the beginning of the protocol and selectively corrupt participants during the execution of the protocol.

Although the ideal model (as outlined above) is difficult to realise in practice, it is an effective tool for secure protocol design by providing a formal proof. The logic follows that a protocol must be secure in the ideal model, by definition. Hence, if it can be proven that a real-world implementation of a protocol is statistically indistinguishable from the ideal implementation, then it can be claimed that the protocol is secure as it behaves exactly the same.

1.2 Motivation and Problem Statement

In general, privacy-preserving protocols aim to provide solutions for the user's privacy, the server's privacy, or both. We first consider what this means at a low level. During the execution of a regular protocol, many messages are exchanged between the parties. This enables the parties to construct various applications using their combined data as input. When we add the requirement for privacy for the client or server, it places restrictions on what messages are exchanged.

The privacy requirements of a solution depend on the application domain or platform we are considering. Let us look at the two logical ways in which privacy can be enforced in applications. In the context of this exposition, when we say public or private, this is with respect to only private parties engaged in the protocol. For simplicity, we will consider a simple client-server model. Obviously, this can be extended to consider many clients and many servers under appropriate assumptions. This includes peer-to-peer models, where participants have the same class or rank.

1. **Private Client/Public Server** In this setup we have the requirement that the client's query cannot be revealed to the server. This is a common scenario and appears in many applications as we will show below. This is important because if the server can determine the contents of the client's query, then they can build a profile of the client and make predictions about their behaviour. This relationship is illustrated by Figure 1.1.

2. **Private Client/Private Server** This is the strongest type of privacy protocol. In this type of protocol, both the privacy of the client and server are protected. As you could imagine, this is a combination of the previous two models, which includes many of the same issues. This construction is shown by Figure 1.2.

With respect to the privacy definitions given above, this thesis will develop and examine *privacy-preserving* protocols for various real-world applications. The overall problem, and hence thesis question, for this thesis is: is it possible to create protocols that both gives the correct outcome and preserves the privacy of the respective parties? This seems to be a paradoxical question. How could we possibly hide what we want to obtain, and still be able to obtain it? In this thesis we will explore

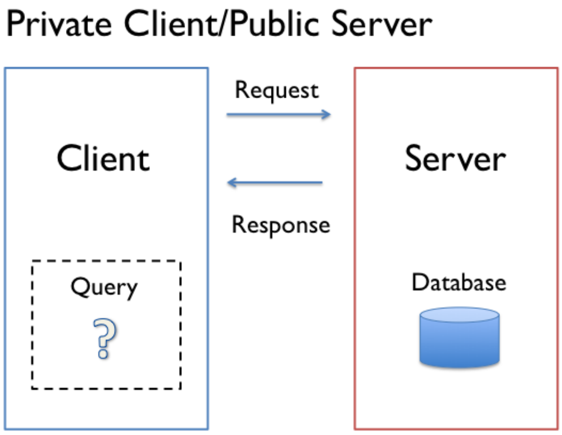


Figure 1.1: Protocol for private query and public database

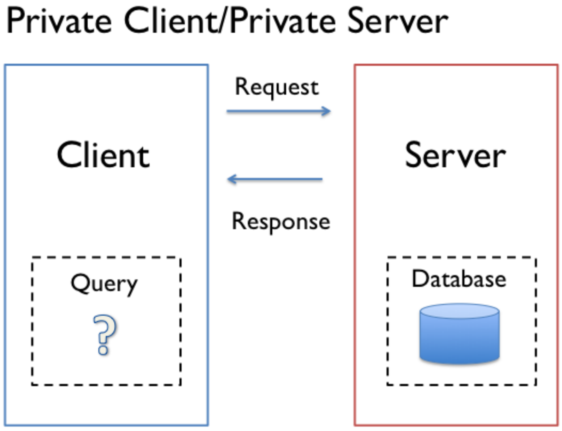


Figure 1.2: Protocol for private query and private database

this seemingly contradictory question. We will achieve this by looking at 3 privacy problems as follows.

1. **Problem 1** *Private query* This is the most general privacy-preserving problem. In the most simplest case, there is a client who owns a query and a server who owns a database. The client submits their query to the server, and the server responds with the data that is consistent with the client's query. With respect to privacy, both parties may consider their inputs to be sensitive. In the case of the client, it is the query, while in the case of the server, it is the database. In the extreme case, the inputs from both parties must be protected from unnecessary disclosure. Sometimes, the server's database can be public.

   To make this more concrete, we offer the following example. Consider a scenario where server that owns a patent database. We consider the patent database to be a public repository. The client, who wants to propose a new patent, must make sure that the patent does not exist already. The client also does not want to reveal their great idea to the patent database because someone else may steal their idea. Therefore, they must construct a private query to submit to the database.

   Another example where the inputs of both parties would have to be protected is where the server owns a database that contains information about medicine products. As in the patent database example, the client does not want to reveal the intention of creating a new drug based on the ingredients in the database. Furthermore, the server wants to earn revenue from the use of the database, so they require the client to pay for obtaining only one record. This requires a check on the query such that it has been formed correctly.

2. **Problem 2** *Location-based query* The specifications for this problem is identical to the requirements for private queries, except the database that the server owns has geographical meaning. That is, there the data are organised according to a geometric structure. In this case a 2-dimensional grid (see Figure 1.3).

   For example, consider a scenario where a client wants to learn where the nearest restaurant is. In this case, the client owns indices i and j, and the server owns a database containing some location-based records. The client, concerned for their privacy, does not want to give the actual location to the server. Likewise,

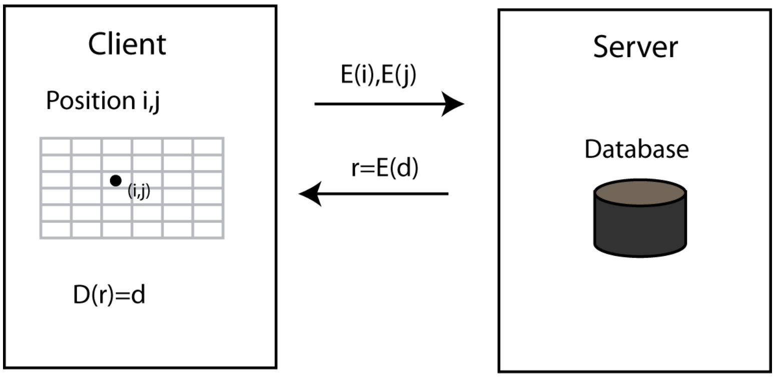


Figure 1.3: Protocol for private query and private database

the server wants to protect their records by requiring that a limited number of records are revealed per query.

3. **Problem 3** *Data mining* An important problem in recent years is *privacy-preserving data mining.* One classic example is where there are two hospitals that have data about their patients. They would like to discover knowledge from the joined database, but they cannot betray the confidentially of the records.

   For example, imagine two supermarkets that own respective transactional database about their shoppers. For legal and competitive reasons, they cannot reveal the contents of these databases to outsiders without modification. However, they would like to learn some useful rules from the data. One approach is to use a semi-trusted mixer [106]. The two sites submit their encrypted data to this mixer and the mixer would broadcast the results to the two parties. This is illustrated in Figure 1.4.

   A better approach would be for the two parties to perform the private computation between themselves that would remove the need for a third party mixer. This is important because we are trusting the mixer not to collude with one of the data sites.

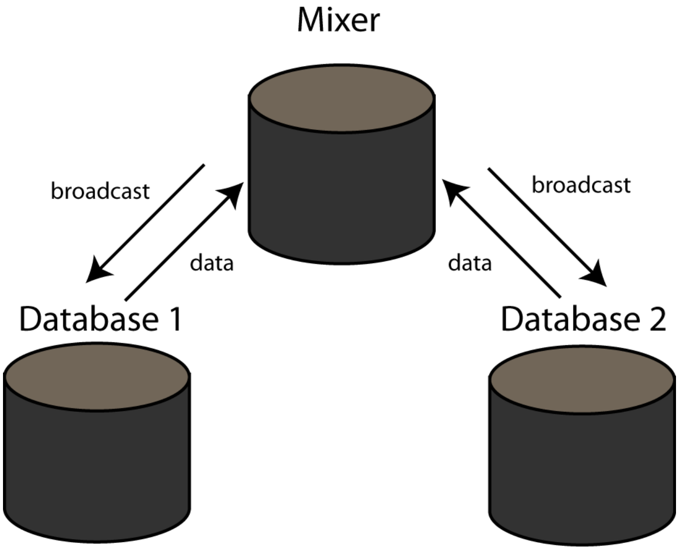


Figure 1.4: A general mixer approach for two parties

1.3 Research Outcomes

With the research problems defined we now present an overview of the research outcomes of this dissertation. Not surprisingly, for each problem listed, there is a corresponding outcome. These are very general descriptions for these methods. Consult later chapters for more details regarding these outcomes.

1. **Outcome 1** *Private query* With respect to the private query problem (Problem 1), this thesis presents contributions at both the fundamental and protocol levels of private query construction.

   a) *Fundamental* At the fundamental level we examine the security of a recent somewhat homomorphic encryption scheme. Briefly, a somewhat homomorphic encryption scheme allows any efficiently computable function according to a polynomial time Turing Machine, but eventually fails to decrypt properly due to the amount of accumulated noise inside a ciphertext. Since this concept has only been introduced recently, the underlying assumptions are as extensively covered as classical hard problems as factoring or discrete logarithms. We evaluate the security claims proposed by the authors and constructs an experiment to break the system under the prescribed parameters.

b) *Protocol* At the protocol level we present the construction of a private information retrieval scheme from the homomorphic properties introduced by the notion of fully homomorphic encryption. Previous private information retrieval schemes have been designed such that they are communicationally efficient. In other words, they download strictly less than the entire database. This has resulted in these schemes to be computationally expensive. We review and revise the feasibility of this constraint, by providing a computationally efficient private information retrieval scheme, which is built on simpler operations. We find that if the communication-computation trade-off is reversed, we result in a more practical scheme. We also compare this scheme with a similar scheme based on matrices that achieves similar results.

2. **Outcome 2** *Location-based query* We present a solution to one of the location query problems, as defined above by Problem 2. Previous solutions have used anonymity and perturbation to try and protect a user's location. These have been found to be vulnerable to attack because the coordinates are related geographically. More recently, researchers have applied private information retrieval to this problem to protect a user's query. We augment private information retrieval with a modified oblivious transfer scheme that is secure for both the user and the server.

3. **Outcome 3** *Data mining* With respect to the general data mining privacy problem given by Problem 3, we present a 2-party privacy-preserving association rule mining application. We present a solution that is based on the concepts of fully homomorphic encryption, and is thus secure if the underlying encryption scheme has semantic security. We provide the results of a working prototype implementation, and contrast our approach with another solution known as Yao's garbled circuits.

1.4 Contributions

The contributions made by this dissertation can be described as one of the following key components, which are: analysis of cryptographic primitives, analysis of privacy-preserving protocols, and implementation of privacy-preserving protocols[1]. These key pieces are elaborated as follows.

Analysis of cryptographic primitives Analyses the cryptographic assumptions of recently introduced fully homomorphic encryption techniques.

Analysis of privacy-preserving protocols Applies cryptographic techniques to construct secure privacy-preserving protocols. Once a construction is defined, the protocol is analysed according to the standard definitions of security.

Implementation of privacy-preserving protocols This thesis will also demonstrate the practical significance of the protocols that are presented using real-world hardware. The implementations of the protocols will show that the theoretical constructions are realisable in practice, under certain reasonable assumptions.

1.5 Thesis Organisation

The thesis is designed to be as self contained as possible by providing enough background knowledge to enable the reader to understand and appreciate the contributions of this thesis. With this in mind, the thesis is organised as follows.

Chapter 2 presents an overview of the necessary literature for creating private queries. The chapter begins by reviewing the theory for homomorphic encryption. Basically a homomorphic encryption scheme allows someone to modify data without being to decrypt it. With this in mind, we review of the classical homomorphic encryption schemes. It continues with an explanation of recent developments in fully homomorphic schemes that have been recently introduced. Contrasting with classical homomorphic schemes, like RSA or ElGamal, fully homomorphic encryption schemes

[1] The contributions outlined here are with respect to the thesis declaration.

promote the ability to operate on encrypted data as you would on non-encrypted data. From the theory of homomorphic encryption we review some private information retrieval schemes, which are designed to protect a client's query from a server. We then explore a stronger variation of private information retrieval, known as oblivious transfer, where there is a stronger requirement for privacy: both the client and server are to be protected.

Chapter 3 extends on the homomorphic encryption exposition in Chapter 2 by examining the security of a recent somewhat homomorphic encryption scheme (see Outcome 1a). It starts with a review of the definitions about fully homomorphic encryption and somewhat homomorphic encryption. It then describes a recent somewhat homomorphic encryption scheme and its associated security claims. An experiment is designed to evaluate the claims of the proposed system and validate its security. Following from this discussion of homomorphic properties, we proceed with protocol construction in Chapter 4, where the concepts of fully homomorphic encryption are used to create a private information retrieval protocol (see Outcome 1b). In this chapter we design a new private information retrieval scheme that is based on the concepts of fully homomorphic encryption. We judge its security and compare it with a scheme with similar performance results.

Chapter 5 presents the results for private location-based queries (see Outcome 2). The chapter is organised as follows. It begins with a review of the recent methods for protecting privacy in the location context. Next we define the system model, and subsequent protocol. Then we perform performance and security analysis of the proposed protocol and show that it is efficient. We conclude the chapter with a couple of experiments and present some recommendations.

Chapter 6 introduces data mining concepts in the privacy-preserving context (see Outcome 3). It first gives an overview of data mining in general before focussing on a privacy preserving association rule mining application. Within this application domain, we review previous methods for preserving privacy. We continue with defining the model and symbols. Next we analyse the performance and security, and then provide results from a working prototype. We conclude this chapter with recommendations about the future of privacy-preserving data mining.

Chapter 7 concludes the thesis by summarising the main contributions and presenting a logical connection. It starts by reviewing the problem statement given in this chapter. With respect to this problem statement, it then provides an overview of the contributions, highlighting a common theme. The chapter concludes with a discussion for future directions.

Chapter 2

Cryptography Background and Review

This chapter contains the necessary background knowledge to enable the reader to understand the contributions of this dissertation. The chapter begins with the notion of homomorphic encryption. It then builds on this idea to give an overview of the standard techniques for creating privacy-preserving protocols.

2.1 Homomorphic Encryption

We introduce an interesting concept known as homomorphic encryption using some well known cryptographic schemes available in the literature. Before we look at the cryptographic schemes, we will define some general terms that will be used throughout the chapter. We begin with the definition of generic encryption scheme.

Definition 1. *An encryption scheme is a tuple $\langle E, D, \mathcal{M}, \mathcal{C}, \mathcal{K} \rangle$, where E is the encryption function, D is the decryption function, $\mathcal{M}$ is the message space, $\mathcal{C}$ is the ciphertext space and $\mathcal{K}$ is the keyspace. E is described as a mapping $E : \mathcal{M} \to \mathcal{C}$, which takes a message $m \in \mathcal{M}$ and a key $k \in \mathcal{K}$ and outputs a ciphertext $c \in \mathcal{C}$ as $c = E(m, k)$. Whereas D is defined as a mapping $D : \mathcal{C} \to \mathcal{M}$, which takes a ciphertext $c \in \mathcal{C}$ and a key $k \in \mathcal{K}$, and outputs a message $m \in \mathcal{C}$ as $m = D(c, K)$.*

A encryption scheme is considered correct if $D(c, k) = D(E(m, k), k) = m$ *for all* $m \in \mathcal{M}, c \in \mathcal{C}, k \in \mathcal{K}$.

This is a very abstract definition of what an encryption scheme aims to do. With this simple definition it is possible to describe many encryption schemes where the encryption method is the same or *symmetric*. Such schemes are useful when you can establish a common secret key in order to communicate in the presence of eavesdroppers. If such an assumption cannot be achieved then we must use a different kind of encryption called *public key cryptography* [29]. This is also called asymmetric cryptography. This is where the keys used for encryption and decryption are different, but related. This definition is given next.

Definition 2. *A public key encryption scheme is where there exists two keys* e *and* d*, where* e *is used for encryption whereas* d *is used for decryption. While the keys* e, d *are related, an adversary should not be able to determine* d *when given* e*. As with symmetric cryptographic schemes, correctness must still follow.*

We now come to an important property of an encryption scheme that is called *semantic security*. Informally, this is when it is computationally infeasible to learn anything about a message from the encryption of the message. Put another way, it should be computationally infeasible to distinguish between a pair of ciphertexts. If this was not the case, then there would be something about the ciphertexts that does not appear random, and therefore it would be possible to learn some or all of an encrypted message. Understandably, the notion of semantic security for a public key encryption scheme must be stronger than that of a symmetric encryption scheme. This is because in a public key encryption scheme we are able to encrypt messages, by definition. Whereas, in symmetric encryption schemes, this may not be the case.

One very interesting property of public key cryptography is the concept of homomorphism. Literally, this word means that we have the same thing (homo) when it is changed (morph). This property was known since of the introduction of public key cryptography, under the term *privacy homomorphisms* [86]. This property is given by the following abstract definition.

Definition 3. *Mathematically, homomorphism is where we have a function* f *and it is applied to an operation on elements* a, b *as* $f(a \star b) = f(a) \cdot f(b)$*, where* $\star$ *and* $\cdot$

can be the same or different. This means the function f preserves the underlying structure of a group or field.

With these definitions in mind, we will explore some popular public key cyrptosystems and examine their homomorphic properties.

2.1.1 Well-known Homomorphic Schemes

Rivest-Shamir-Adleman Cryptosystem

We first give an explanation of one of the oldest and one of the most well known public key cryptosystems: RSA [87]. Shortly after the introduction of public key cryptography [29], this scheme was introduced that allowed the creation of both an encryption system and a signature scheme. While very simple, this scheme has proven to be very secure and reliable under reasonable conditions.

Construction

Keygen The key generation begins with choosing two distinct primes p, q, and computing their product $n = pq$. Next, we compute $\phi(n) = (p-1)(q-1)$ and choose a random number e such that $1 \leqslant e \leqslant \phi(n)$ and $gcd(e, \phi(n) = 1)$ (e and $\phi(n)$ are relatively prime and hence e will have an inverse $mod\ \phi(n)$). Find a d such that $d = e^{-1}\ (mod\ \phi(n))$ or equivalently a d that satisfies $de - 1 = 0\ (mod\ \phi(n))$. Publish (e, n) as the public key and keep d as the secret key.

Encryption Given a message m compute the ciphertext c as $E(m) = m^e\ (mod\ n)$.

Decryption Given a ciphertext c compute the message m as $D(c) = c^d\ (mod\ n)$.

This is correct because of Euler's Theorem given by $a^{\phi(n)} = 1\ (mod\ n)$, where $\phi(n)$ (ϕ is called Euler's totient function) is the cardinality of numbers that less than n and relatively prime to n. See the following equation.

$$c^d = (m^e)^d = m^{ed} = m^{\phi(n)+1} = m^{\phi(n)}m = m \ (mod\ n) \tag{2.1}$$

Homomorphic Properties This scheme is multiplicatively homomorphic. That is, given two ciphertexts that encrypt two messages m_1, m_2 as $E(m_1)$ and $E(m_2)$. Then we have the following property as shown in the next equation.

$$E(m_1)E(m_2) = m_1^e m_2^e = (m_1 m_2)^e = E(m_1 m_2) \tag{2.2}$$

ElGamal Cryptosystem

This scheme is built on the property that when we raise a number g to the power of a and then to the power of b is the same as the reverse. More precisely, $(g^a)^b = (g^b)^a$ for all $a, b \in G$, where G is a group. This is possible over any group; we will use a multiplicative group of a finite field for this exposition.

Construction

Keygen Choose a random $x \in G$ and compute $h = g^x \in G$, where g is a generator of G of order q. The public key is h and the private key x.

Encryption Compute the encryption of message m as $E(m) = [g^k, m \cdot h^k]$

Decryption Compute the decryption of ciphertext $c = [c_1, c_2]$ as $D(c) = (c_1^x)^{-1} \cdot c_2$

Correctness immediately follows from the commutativity of exponentiation. That is, $(c_1^x)^{-1} \cdot c_2 = ((g^k)^x)^{-1} \cdot (m \cdot h^k) = (g^{kx})^{-1} \cdot m \cdot (g^x k) = m$. This scheme has semantic security, which means two ciphertexts can encrypt the same message.

Homomorphic Properties Like RSA, the Elgamal scheme is multiplicatively homomorphic. When we have two messages m_1, m_2 as $E(m_1)$ and $E(m_2)$ (encrypted under the same public key $h = g^k$), we can compute the encryption of the product $m_1 m_2$ as Equation 2.3 shows.

$$E(m_1)E(m_2) = [g^{r_1}, m_1 \cdot h^{r_1}][g^{r_2}, m_2 \cdot h^{r_2}] = [g^{r_1+r_2}, (m_1 \cdot m_2)h^{r_1+r_2}] = E(m_1 m_2) \tag{2.3}$$

Goldwasser-Micali Cryptosystem

This cryptosystem was the earliest example of what is known as probabilistic encryption [49], which in this context means that there are many ciphertexts that encode the same message. In this case, we are encoding bit by bit, so the ciphertext space is larger than the message space. For the scheme's construction we define a function $Q(x)$ such that it returns 1 if x is a quadratic residue and 0 otherwise. A number q is a quadratic residue with respect to a modulus n, if $x^2 = q \ (mod\ n)$, otherwise it is a nonresidue.

Construction

Keygen The key generation procedure begins with selecting two k-bit primes p_1, p_2 and computing the product $n = p_1 p_2$. Then, a $y \in \mathbb{Z}_n$ is chosen such that it is a quadratic nonresidue. The public key is $[n, y]$, while the the private key is $[p_1, p_2]$.

Encryption Compute the encryption of message m as $E(m) = yx^2 \ (mod\ n)$, if $m = 1$, else $E(m) = x^2 \ (mod\ n)$, where $x \in \mathbb{Z}_n^*$ is chosen at random.

Decryption Compute the decryption of ciphertext c as $D(c) = Q(c)$.

Homomorphic Properties This scheme additively homomorphic (mod 2). Equivalently, this can be seen as the XOR operation on binary digits or bits. Given encryption of bits b_1, b_2 as $E(b_1)$ and $E(b_2)$, we have the following homomorphism demonstrated by Equation 2.4.

$$E(b_1)E(b_2) = x^{b_1} r_1^2 x^{b_2} r_2^2 = x^{b_1+b_2}(r_1 r_2)^2 = E(b_1 \oplus b_2) \tag{2.4}$$

Paillier Cryptosystem

The Paillier encryption scheme [81] can be seen as a ElGamal scheme with some of the characteristics of an RSA scheme. This was introduced considerably after the birth of public key cryptography, but it has grown with popularity ever since—especially because of its attractive homomorphic property. It is related to the Goldwasser-Micali [49] encryption scheme, and its subsequent extensions [6, 77, 73]. The most notable feature, when compared to previous schemes in its family, is the decryption algorithm that is straight-forward.

Construction

Keygen Choose two prime numbers p and q and compute the product $n = pq$. Compute $\lambda = (p-1)(q-1)$ and set $g = 1 + n$. The public key is n and the private key is λ.

Encryption Compute the encryption of message m as $E(m) = g^m \cdot r^n \ (mod\ n^2)$.

Decryption Compute the decryption of ciphertext c as $D(c) = \frac{[c^\lambda \ (mod\ n^2)]-1}{n} \cdot \lambda^{-1} \ (mod\ n)$.

The correctness of the Paillier encryption scheme can be verified using the Binomial Theorem. When we raise the ciphertext $c = g^m \cdot r^n$ to the power of λ, we get $c^\lambda = (g^m \cdot r^n)^\lambda = (g^m)^\lambda \cdot (r^n)^\lambda = g^{m\lambda} \cdot 1 = g^{m\lambda}$. It remains to see what happens when $g = 1 + n$ is raised by $m\lambda$. This is shown below in Equation 2.5 as binomial expansion, where there is an expression ϵ multiplied by n^2.

$$(n+1)^{m\lambda} = 1 + mn\lambda + \epsilon n^2 = 1 + mn\lambda \ (mod\ n^2) \tag{2.5}$$

At this point, all we need to do to recover m is to subtract 1, divide by n, and then multiply by the inverse $\lambda^{-1} \ (mod\ n)$ as $\frac{(1+mn\lambda)-1}{n} \cdot \lambda^{-1} = m$, as required.

Homomorphic Properties This scheme is additively homomorphic. That is if we have two ciphertexts as $E(m_1) = g^{m_1}r_1^n$ and $E(m_2) = g^{m_1}r_2^n$, where m_1, m_2 are two messages. Then we have the following property in Equation 2.6.

$$E(m_1)E(m_2) = (g^{m_1}r_1^n)(g^{m_1}r_2^n) = g^{m_1+m_2}(r_1r_2)^n = E(m_1 + m_2) \tag{2.6}$$

Boneh-Goh-Nissim Cryptosystem

The BGN cryptosystem [10] is essentially the Paillier scheme combined with bilinear pairings, with some modifications. The motivation is that Paillier is known to be additively homomorphic, and we would want to be able to multiply too. We can achieve this by using secure *bilinear pairings.*

A bilinear pairing is an abstract mathematical mapping of two inputs that, as its name suggests, is linear in both variables. More formally, when given the definition of two cyclic multiplicative groups $\mathbb{G}$ and $\mathbb{G}_1$ of finite order n, where g is a generator of $\mathbb{G}_1$, then we can construct a mapping denoted by e as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ that satisfies $e(u^a, v^b) = e(u, v)^{ab}$. By this definition, $e(g, g)$ is a generator of $\mathbb{G}_1$. For practical reasons, this mapping must be efficiently computable. This efficiency requirement is achieved using Miller's Algorithm [71].

There are many methods for constructing a bilinear pairing. Perhaps the most simple way is the dot product for vectors, which exhibits this property. For cryptographic purposes, we need a stronger construction because linear functions are easy to invert. It has been discovered that a certain class of elliptic curves, called supersingluar, can be used to construct a mapping with large order or size n.

We can construct a specific kind of elliptic curve pairing, known as the Weil Pairing, as follows [104]. Let p be a prime such that $p = 12q - 1$ for some prime q. Let E be an elliptic curve given by the equation $y^2 = x^3 + 1$ over F_p. Then the set of points $E(F_p) = \{(x, y) \in F_p \times F_p : (x, y) \in E\}$ forms a cyclic group of order $p + 1$. Since $p + 1 = 12q$ for some prime q, the set of points of order q in $E(F_p)$. Let $\mathbb{G}_2$

be a subgroup of the finite field F_{p^2} containing all elements of order q. Under this formulation, we have a bilinear map, as required.

The bilinear pairing was first used to construct a one-round tripartite Diffie-Hellman Key exchange [56]. This was followed by, perhaps the most famous application, the practical realisation of identity based encryption [9]. A comprehensive overview of pairing based cryptography is beyond the scope of this thesis. But we refer the interested reader to a great survey on the subject [31]. As for our purposes, we can use the bilinear mapping to create a Paillier-like encryption scheme that allows multiplication of the message.

Construction

Keygen Given a secure parameter $k \in \mathbb{Z}^+$, choose two k-bit primes q_1, q_2, and compute $N = q_1 q_2$. From a family of supersingular elliptic curves, select a curve E that has order N and generator g. Let E form the group $\mathbb{G}$ and let there be a mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the appropriate bilinear properties. Pick two random generators g, u from $\mathbb{G}$ and set $h = u^{q_2}$. Then h is a random generator of the subgroup of $\mathbb{G}$ of order q_1. The public key is $PK = \{N, \mathbb{G}, \mathbb{G}_1, e, g, h\}$. The private key $SK = q_1$.

Encryption Compute the encryption of message m as $E(m) = g^m h^r \in \mathbb{G}$, where $m < q_2$ and r is chosen from $\{1, 2, \cdots, N\}$. To encrypt a message m using the public key PK, pick a random r from $\{1, 2, \cdots, N\}$.

Decryption Compute the decryption of ciphertext c as $D(c) = log_{\hat{g}}(c^{q_1})$, where $\hat{g} = g^{q_1}$. This is correct as $c^{q_1} = (g^m h^r)^{q_1} = (g^m)^{q_1}(h^r)^{q_1} = (g^m)^{q_1} = (g^{q_1})^m$.

Homomorphic Properties It is easy to show that this encryption scheme has additively homomorphic. Given two messages m_1 and m_2 encrypted under this scheme as c_1 and c_2, then computing the product $c = c_1 c_2 h^r$ adds messages m_1, m_2 accordingly. The h^r part of this product ensures the resulting ciphertext c appears as a freshly encrypted ciphertext, such that an adversary cannot distinguish it from random. This homomorphic structure is similar to the property exhibited by the Paillier encryption scheme, as mentioned above.

Since the scheme has been constructed to include a bilinear map e, we can compute the product on two encrypted messages. This can only be done *once*, because there is no known way of mapping an element of $\mathbb{G}_1$ to an element of $\mathbb{G}$. Let $g_1 = e(g,g) \in \mathbb{G}_1$ and $h_1 = e(g,h) \in \mathbb{G}_1$, and given two ciphertexts as $c_1 = g^{m_1}h^{r_1} \in \mathbb{G}$ and $c_2 = g^{m_2}h^{r_2} \in \mathbb{G}$. Compute the product $m_1 \cdot m_2 \ mod \ n$, using only c_1 and c_2, as $c = e(c_1, c_2)h_1^r$, where $r \in \mathbb{Z}_n$ is chosen at random. The correctness is shown by Equation 2.7.

$$c = e(c_1, c_2)h_1^r = e(g^{m_1}h^{r_1}, g^{m_2}h^{r_2})h_1^r = g_1^{m_1 m_2}h_1^{\tilde{r}} \in \mathbb{G}_1 \tag{2.7}$$

As pointed out in the original paper, this scheme remains additively homomorphic in the finite field group $\mathbb{G}_1$.

2.1.2 Fully Homomorphic Encryption

The problem of constructing a fully homomorphic encryption scheme was first posed by Rivest [87], under the term *privacy homomorphism*. This problem remained unsolved until Gentry [38], who showed that constructing a scheme with the desired properties is possible. More precisely, fully homomorphic encryption means we have the following properties, for two numbers X and Y, and encryption and decryption functions E and D respectively.

$$D[E(X) + E(Y)] = D[E(X + Y)] \tag{2.8}$$

$$D[E(X) \times E(Y)] = D[E(X \times Y)] \tag{2.9}$$

The original construction posed by Gentry [38] is based on the properties of ideal lattices. Gentry's solution was constructed using a three stage approach. In the first stage, a private somewhat homomorphic encryption scheme was defined. This meant that scheme was homomorphic, but the decryption procedure failed after a certain number of operations. Essentially the noise component of the ciphertext grew too large and hence it did not decrypt properly. The next stage transformed the private somewhat homomorphic encryption scheme into a public somewhat homomorphic encryption scheme. This allowed, as with other homomorphic schemes, anyone to encrypt without access to the private key. The final step employed a clever technique that converted the public somewhat encryption scheme into a fully homomorphic encryption scheme. The noise component of the ciphertext was controlled by decrypting the ciphertext homomorphically, resulting in a new ciphertext with less noise. This would allow anyone to compute indefinitely on encrypted data.

Since Gentry's initial result, there have been various improvements and simplifications. Smart and Vercauteren presented a scheme that removed the requirement to represent the public key as a large matrix [93]. The large matrix is replaced by a two element representation. Shortly after, a fully homomorphic encryption was introduced that used only elementary number arithmetic [98]. This result greatly reduced the complexity of a fully homomorphic encryption scheme by allowing it to be described in simple terms. We summarise the asymmetric version of this simplified scheme.

$\boldsymbol{KeyGen(\lambda)}$**:** Choose a random n-bit odd integer p as the private key.
Using the private key, generate the public key as $x_i = pq_i + 2r_i$ where q_i and r_i are chosen randomly, for $i = 0, 1, ..., \tau$. Relabel so that x_0 is the largest

$\boldsymbol{Encrypt(pk, m \in \{0, 1\})}$**:** Choose a random subset $S \subseteq \{1, 2, ..., \tau\}$ and a random integer r, and output $c = (m + 2r + 2\sum_{i \in S} x_i)(mod\ x_0)$.

$\boldsymbol{Decrypt(sk, c)}$**:** Output $m = (c\ mod\ p)\ mod\ 2$

If pk and sk are the public key and private key of the asymmetric version, then we can encrypt two bits $x, y \in \{0, 1\}$ as $\alpha \leftarrow E_{pk}(x) = pq_1 + 2r_2 + x$ and $\beta \leftarrow E_{pk}(y) = pq_2 + 2r_2 + y$. The homomorphic addition and multiplication can be defined as Equation 11 and 12. Since the scheme uses integers, the + and the

× are regular addition and multiplication respectively. It is easy to see that these equations will decrypt with the correct result.

$$\alpha + \beta = (q_1 + q_2)p + 2(r_1 + r_2) + (x + y) \tag{2.10}$$

$$\alpha \times \beta = (pq_1q_2 + 2q_1r_2 + 2q_2r_1 + m_1q_2 + m_2q_1)p + 2(2r_1r_2 + m_1r_2 + m_2r_1) + xy \tag{2.11}$$

When we add or multiply the ciphertext, the message is changed accordingly. Since we limit the data to a 0 or a 1, adding and multiplying reduces to the XOR and AND gates respectively. Hence, we can take any program represented as a logical circuit and apply it to the ciphertext, without the knowledge of the data that is concealed. Since the decryption function can efficiently be represented as a combination of boolean gates, then this permits the scheme to evaluate its own decryption circuit (assuming that the noise bound is small enough for the decryption function), which allows bootstrapping.

2.2 Private Information Retrieval

We now have the tools to define and construct Private Information Retrieval (PIR) schemes. Informally, a PIR scheme allows a user to retrieve the i-th bit from an n-bit database, without the database learning the value i. A trivial solution to this problem is to simply download the entire database and perform the extraction of the i-th bit locally. This trivial solution incurs significant communication cost and thus should be avoided. Hence, we require that PIR schemes have strictly less communication complexity than downloading the whole database.

The concept of Private Information Retrieval was introduced by [20] in the information theoretic setting. This means that if there are $k \geqslant 2$ non-colluding sites, where each own a copy of a database $DB_1, ..., DB_k$, represented by $x \in \{0,1\}^k$.

Then the user can submit queries $Q_1, ..., Q_k$, appropriately to each site, and collect the query responses to reconstruct the i-th bit. Chor et al. showed that under the assumption that sites are not allowed to collude, the queries reveal absolutely nothing about the value of i.

If we consider a single-database solution, the above solution reduces to downloading n bits to achieve information theoretic security. In other words, the query reveals nothing about the index i, when the whole database is downloaded. This makes intuitive sense. The database cannot learn anything about our intention if we simply download the entire database. We notice that this is exactly the trivial solution that we already mentioned and dismissed.

The first non-trivial single database PIR scheme was introduced by [64]. Since a non-trivial single database PIR scheme does not exist in the information theoretic setting, Kushilevitz and Ostrovsky introduced a PIR scheme that uses the Quadratic Residuosity Assumption. Using this assumption they were able to construct a solution that used less communication than the trivial method, based the difficulty of a computational assumption. For this reason, these schemes are also known as computational private information retrieval (cPIR). Unless specified otherwise, single-database PIR implies cPIR. With computational private information retrieval in mind, we will now consider a formal definition and outline some concrete constructions.

2.2.1 Formal Definition

A computational private information retrieval scheme consists of three algorithms: Query Generation, Response Generation, and Response Retrieval (RR). The inputs and outputs to these functions are as follows (according to conventions used in the literature [17, 42]).

(1) Query Generation (QG): Takes as input a security parameter k, the size n of the database, and the index i of a bit in the database, outputs a query Q and a secret s, denoted as $(Q, s) = \mathsf{QG}(n, i, 1^k)$.

(2) Response Generation (RG): Takes as input the security parameter k, the query Q and the database DB, outputs a response R, denoted as $R = \mathsf{RG}(DB, Q, 1^k)$.

(3) Response Retrieval (RR): Takes as input the security parameter k, the response R, the index i of the bit, the size n of the database, the query Q, and the secret s, output a bit b', denoted as $b' = \mathsf{RR}(n, i, (Q, s), R, 1^k)$.

The two main requirements of a PIR scheme are *correctness* and *privacy*. We first must show that a scheme is correct, since it is pointless to prove privacy when the scheme does not operate as expected. More formally, we have correctness and privacy of the above private information retrieval scheme.

Definition 4 (Correctness). *A single-database PIR protocol is correct if, for any security parameter k, any database DB with any size n, and any index i for $1 \leqslant i \leqslant n$, $b_i = \mathsf{RR}(n, i, (Q, s), R, 1^k)$ holds, where $(Q, s) = \mathsf{QG}(n, i, 1^k)$ and $R = \mathsf{RG}(DB, Q, 1^k)$.*

Definition 5 (Privacy). *A single-database PIR protocol is private if, $\forall n, \forall i, j \in [1, n]$, and let there be a security parameter k. Then there exists an adversary A, represented as an algorithm, subject to the following inequality.*

$$|PROB[(q, s) \leftarrow Q(n, i, 1^k) : A(n, q, 1^k) = 1] - PROB[(q, s) \leftarrow Q(n, j, 1^k) = 1]| < 2^{-k} \tag{2.12}$$

In simple terms, the correctness definition means that for every query q, the correct bit/block is retrieved. While, the privacy definition means that for any two queries q_1, q_2, with indices i, j respectively, an adversary cannot distinguish them from one another greater than $\frac{1}{2^k}$. With this general definition of a PIR scheme in mind, we give some notable constructions from the literature.

2.2.2 Concrete Schemes

There are many PIR constructions available in the literature. For now, we provide an exposition of the ones that are fundamentally distinct from one another to give a sense of what a general scheme looks like. In particular, we look at schemes based on the Quadratic Residue Assumption, the Decisional Composite Residuosity Assumption and the Phi-hiding Assumption.

For the first two schemes, correctness follows directly from the definition of the homomorphic encryption system used. The correctness of the third scheme, which is based on the Phi-hiding assumption, is not immediately obvious. This is because it is not directly based on a fundamental homomorphic encryption property. Hence we will provide some intuitive understanding for why it works.

Quadratic Residue Assumption

We begin our exposition of concrete schemes with the first single-database private information retrieval scheme [64]. As we will see, this scheme is built upon the homomorphic properties of quadratic residues [49]. Hence its security is based on the problem of determining a quadratic residue when reduced by some hard-to-factor modulus n.

In this scheme the database is represented as 2 dimensional matrix $M_{s \times t}$ of bits. The user desires to retreive the bit x_i at position (a, b). For simplicity, we consider that $M_{s \times t}$ is a square matrix, that is $s = t$.

Query Generation The user creates a k-bit number $N = pq$, where p, q are two $k/2$-bit primes. The user then selects t numbers $y_1, ..., y_t$, where y_b is a Quardratic Nonresidue and y_j, for $j \neq b$, is a Quadratic Residue. In other words, of the numbers $y_1, ..., y_t$, only y_b is a Quadratic Non Residue. The user sends $[N, y_1, ..., y_t]$ to the server.

Response Generation The server computes, for every row r, $z_r = \prod_{j=1}^{t} w_{r,j}$, where

$$w_{r,j} = \begin{cases} y_j^2 & \text{if } M_{r,j} = 0 \\ y_j & \text{if } M_{r,j} \neq 0 \end{cases}$$

The s numbers $[z_1, ..., z_s]$ are sent to the user.

Response Retrieval The user recovers the bit by analysing the ath number z_a, and the bit at (a, b) is 0 if and only if z_a is a Quadratic Residue, and a 1 otherwise. Since the factorisation of N is known, this can be done efficiently.

Composite Residuosity Assumption

We now present a private information retrieval scheme [18] that is based on the Composite Residuosity Assumption [81]. The appproach is very similar to the previous scheme: arrange the database into a square (or a cube), and then "encrypt" the indices to retrieve the specified bit. For this exposition, we will arrange the database as a square matrix of zeros and ones, with $N = \ell^2$. Let I be an indicating function, such that $I(t, t_0) = 1$ if $t = t_0$, and $I(t, t_0) = 1$ otherwise. Also, let $\mathcal{E}(m)$ be the Paillier encryption function for message m and let $\mathcal{E}(c)$ be the corresponding decryption function for ciphertext c, where there is an implicit random input supplied to the encryption function.

Query Generation The user creates their query as $[\alpha_t, \beta_t] = [\mathcal{E}(I(t, i*)), \mathcal{E}((t, j*))]$ for $t \in \{1, ..., \ell\}$ and sends their query to the server.

Response Generation Upon receiving the user's query the server computes

$$\sigma_i = \prod_{t \in \{1,...,\ell\}} (\beta_t)^{x(i,t)} \ mod \ n^2$$

for $i \in \{1, ..., \ell\}$. Server then splits each σ_i by computing $u_i, v_i \in \mathbb{N}_n$ such that $\sigma_i = u_i n + v_i$, and sends

$$u = \prod_{t \in \{1,...,\ell\}} (\alpha_t)^{u_t} \ mod \ n^2$$

and

$$v = \prod_{t \in \{1,...,\ell\}} (\alpha_t)^{v_t} \ mod \ n^2$$

to the user.

Response Retrieval The user reconstructs as $x(i*, j*) = \mathcal{D}(\mathcal{D}(u) + \mathcal{D}(v))$.

Phi-hiding Assumption

We now describe a PIR scheme that is based on a relatively new computational hardness assumption, which is known as the phi-hiding assumption ($\phi-$hiding assumption). The first use of this assumption in PIR was by [17], where only one bit could be retrieved per round. This was extended [42], to consider blocks instead of single bits. We will explain the block version here, with the expectation that it can easily be converted into the simpler bit-based one.

Before we describe the operation of this scheme, we will provide some scope for the setup. Let $\mathcal{S} = \{\pi_i = p_1^{c_1}, ..., \pi_N = p_N^{c_N}\}$ be a set of prime powers, where N is the number of blocks and every prime power pair is coprime $GCD(\pi_i, \pi_j) = 1$ for $(1 \leqslant i, j \leqslant N)$ and $(i \neq j)$. The server divides the database into t distinct blocks $DB = C_1||C_2|| \cdots ||C_t$, where each C_i is represented by an integer and is less than the corresponding prime power $C_i < p_i^{c_i}$. The server then finds an integer e, such that $e = C_j \ (mod \ \pi_j)$ for $1 \leqslant j \leqslant t$, using the Chinese Remainder Theorem. With this in mind, we detail the steps according to the PIR definition.

Query Generation The user decides to download the block at i (C_i) and constructs their query by using the corresponding π_i as follows. The user constructs a group G and "quasi-generator" g, such that $|G| = q\pi_i$ for some $q \in \mathbb{Z}$. In other words π_i divides the order of the group. The user outputs the description of the group (G, g), but computes $h = g^q$ for later use.

Response Generation After receiving the description of the group (G, g), the server computes $g_e = g^e \in G$ and sends it to the user.

Response Retrieval Upon receiving g_e, the user computes $h_e = g_e^q$. Then the user determines C_i as the discrete logarithm of h_e base h: $C_i = log_h h_e$.

Since the database is encoded as $e = C_j \ (mod \ \pi_j)$, for all j, it implies a structure that we can recover each C_i, simply by dividing by the appropriate prime power π. This, and the fact we are working with a subgroup of G called H, allows us to reduce (mod) the integer g by manipulating the order of H.

To be able to construct a hidden group H within a larger group G, of order π_i means that we must embed π_i in the order of G. This can be achieved by computing two integers $Q_0 = 2q_0\pi + 1$ and $Q_1 = 2dq_1 + 1$: where q_0, q_1 are prime numbers, π is the prime power referencing the block at i, and d is suitable random integer. If Q_0, Q_1 are also prime numbers, we can compute a modulus $n = Q_0Q_1$, where the order of the group it represents is $\phi(n) = (Q_0 - 1)(Q_1 - 1)$. Then, by definition, the group order will contain π as a factor. When an element of group G is raised to the power of integer $q = |G|/\pi$, this reduces it to an element of group H. Thus it remains to compute the discrete logarithm of group H.

In general the discrete logarithm problem is considered hard. And in most cases, we require this problem to be computationally intractable for security reasons. In this case, we require this problem to be tractable, or computable with contemporary technology. The discrete logarithm can be accelerated using Pohlig-Hellman algorithm [83]. This algorithm is very efficient, compared with brute force, when we know the order of the group. In the case of the Gentry and Razman PIR, we know the order to be $\pi_i = p_i^{c_i}$, and we can utilise this fact to essentially find the discrete log of h_e by determining the discrete logarithm with respect to the following sequence $\{p^1, p^2, ..., p^{c_i}\}$ as $\{\ell_1, \ell_2, ..., \ell_{c_i}\}$. Then we combine to find the actual discrete log as $log_h h_e = \ell_1 + \ell_2(p_1) + ... + \ell_{c_i - 1}(p_{c_i})\ (mod\ \pi)$.

2.3 Oblivious Transfer

The concept of Oblivious Transfer [85] has an intimate relationship with private information retrieval[1]. Even though chronologically, oblivious transfer actually predates private information retrieval. Oblivious transfer can be considered a stronger version of private information retrieval [79]. In the case where there are multiple non-colluding databases this is termed Symmetric Private Information Retrieval (sPIR) [43, 74][2].

[1] The idea was independently formulated by Wiesner [100], which was built on the theory of quantum mechanics. But the basic idea is the same.

[2] In the case of single-database Symmetric Private Information Retrieval, this is equivalent to oblivious transfer, with the added requirement of being communicationally efficient.

The original probability transfer presented by Rabin was extended to provide a deterministic result [33], with the introduction of the 1-out-of-2 oblivious transfer denoted by OT_1^2. These two protocols were proven to be fundamentally the same [26]. The OT_1^2 primitive can be easily extended to 1-out-of-n oblivious transfer, denoted by a similar manner as OT_1^n [14]. In [14], an OT_1^n was described under the name *all-or-nothing disclosure of secrets* This scheme was extended by [95], where the main contribution is a zero-knowledge proof.

2.3.1 Formal Definition

Oblivious transfer is an interactive protocol between two algorithms Alice and Bob. Let us focus our attention on the most simple case of the 1-out-of-2 oblivious transfer OT_1^2. In this case, Alice has two bits b_0, b_1, while Bob has a selection bit $b \in \{0,1\}$. At the conclusion of the protocol Bob should learn b_c and nothing about $b_{\overline{c}}$. Additionally, Alice should learn absolutely nothing at the end of the protocol.

These requirements are given by the following three conditions Before we formally define these conditions, we will introduce some notations (these are simplified definitions based on [28]). Let k dentote the security parameter and let $t \leftarrow \Pi_{Alice,Bob}(\cdot,\cdot,\cdot,\cdot)$ denote a protocol Π with algorithms *Alice* and *Bob*, where the first and second inputs are Alice's values $(1^k, b_0, b_1)$ and Alice's randomness r_A, while the third and fourth inputs are Bob's values $(1^k, c)$ and Bob's randomness r_B.

(1) *Correctness* After the execution of the protocol

$$t \leftarrow \Pi_{Alice,Bob}((1^k, b_0, b_1), r_A, (1^k, c), r_B),$$

where t is a transcript, the following must hold for all sufficiently large d.

$$Prob[Bob(1^k, c, r_B, t) = b_c] \geqslant 1 - k^{-d} \tag{2.13}$$

(2) *Privacy against Alice* After the execution of the protocol

$$t \leftarrow \Pi_{Alice,Bob}((1^k, b_0, b_1), r_A, (1^k, c), r_B),$$

where t is a transcript and *Alice* is polynomial time algorithm, the following must hold for all sufficiently large d.

$$Prob[Alice(1^k, b_0, b_1, r_A, t) = c] \leqslant 1/2 + k^{-d} \tag{2.14}$$

(3) *Privacy against Bob* After the execution of the protocol

$$t \leftarrow \Pi_{Alice,Bob}((1^k, b_0, b_1), r_A, (1^k, c), r_B),$$

where t is a transcript and *Bob* is a polynomial time algorithm, the following must hold for all sufficiently large d.

$$Prob[Bob(1^k, c, r_B) = b_{\overline{c}}] \leqslant 1/2 + k^{-d}$$

Indeed, oblivious transfer and private information retrieval have a lot in common. The main differences being: the strong definition of privacy for both parties in oblivious transfer; and the strong communication requirement for private information retrieval. In fact, [74] defined an efficient OT_1^n scheme from one invocation of an efficient private information retrieval scheme and $log\ N$ invocations of a simpler OT_1^2 protocol. The work of [13] achieved a similar result, which is based on the concept of intersecting codes. Furthermore, we can construct an oblivious transfer scheme entirely based on invocations of an efficient private information retrieval scheme [28].

2.3.2 Applications and Extensions

Before we look at the applications and extensions, we will provide an example to show the operation an oblivious transfer scheme and help explain the associated conditions. This example will consider an oblivious transfer protocol between two parties: Alice and Bob. In this setup, Alice will play the role of the sender, while Bob will play the role of the receiver.

1. Alice generates an instance of the RSA cryptosystem and sends the public key e, N to Bob.

2. Alice chooses two random messages $x_0, x_1 \in \mathbb{Z}$ and sends to Bob.

3. Bob selects one of the two messages and computes $v = (x_b + k^e)\ (mod\ N)$, where $b \in \{0, 1\}$, e is the public key of Alice, and $k \in \mathbb{Z}$ is chosen randomly. Bob sends v to Alice.

4. Alice computes two possible k values as $k_0 = (v - x_0)^d\ (mod\ N)$ and $k_1 = (v - x_1)^d\ (mod\ N)$, where d is the private key of Alice. Alice sends $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$ to Bob.

5. Bob either computes $m_0 = m'_0 - k$ or $m_1 = m'_1 - k$ based on his choice of x_b.

This simple example demonstrates the two main requirements of oblivious transfer. Alice cannot determine which message Bob received. This is due to the randomly chosen k, Alice cannot distinguish which message was chosen in step 3. At the same time, Bob is unable to obtain to learn both message in one iteration of this protocol. This is because only one of k_0, k_1 is the actual k value chosen by Bob.

The work of Naor and Pinkas [74] was developed into what is known as *adaptive oblivious transfer* [75, 22, 23], which is denoted as $OT^N_{k\times 1}$. Contrasting with the original oblivious transfer, adaptive oblivious transfer enables the user to successively query the database server, where each sequential query is based on the query history. The basic structure is in two phases. In the first phase, a commitment of keys is transferred from the server to the user, which takes $O(N)$ work, where N is the number of elements. In the second phase, these commitments are adaptively queried and with each query, one (and only one is revealed). The original construction of $OT^N_{k\times 1}$ used the idea of *sum consistent synthesizers* that have two identifying properties: (1) the function S is sum consistent (e.g. for all x_1, x_2, y_1, y_2 where $x_1 + y_1 = x_2 + y_2$, S satisfies $S(x_1, y_1) = S(x_2, y_2)$); and (2) the function S is pseudo-random. They realise this component in the standard model using the Decisional Diffie Hellman assumption.

One of the most attractive reasons for creating an efficient oblivious transfer scheme [76][3] is that it permits secure multi-party computation [48], via Yao's Garbled Circuit method [103] (or some variation of this idea [62]). Briefly, a circuit is represented as a tree structure where all inputs are replaced by a key in a predefined range. Once this tree is constructed, it is evaluated by obtaining the keys at the leaf nodes using oblivious transfer.

[3] We note the distinction between computationally and communicationally efficient schemes.

Chapter 3

Cryptanalysis of a Somewhat Homomorphic Encryption Scheme

This chapter is based on content from the paper entitled ‘Cryptanalysis of Brenner et al.’s Somewhat Homomorphic Encryption Scheme’, which was published in the Australasian Information Security Conference 2013.

We begin this chapter by reviewing the performance results for fully homomorphic encryption schemes. We then turn our attention to the security of the somewhat encryption schemes, which are used as a foundation to construct fully homomorphic encryption schemes. In particular, we explore the security claims of a somewhat homomorphic encryption scheme by Brenner, Perl and Smith [15].

It is easy to see that the content in this chapter enables the creation of private queries, and hence, protection for the client. Thus, it suits the private query/public server privacy model type given in Chapter 1.

3.1 Practicality of Fully Homomorphic Schemes

Initial efforts to implement a fully homomorphic encryption scheme has shown to be impractical for anything other than theoretical interest [39]. There have been improvements, due to [94], to increase the performance of the original scheme proposed by Gentry.

Different optimisations and techniques have been introduced to improve the performance of fully homomorphic schemes. The original fully homomorphic encryption scheme was based on ideal lattices, which contained additional structure that can potentially be exploited. To overcome this potential weakness, a scheme based on the standard Learning With Errors (LWE) problem was introduced [11]. The scheme included a unique re-linearisation that was used after multiplication of two ciphertexts to transform a quadratic into a standard linear size ciphertext.

A modulus switching technique has been proposed as a better method for managing the noise associated with ciphertexts [12]. Essentially, we can choose a large q and have a series of levels down to zero, where at each level we scale back the ciphertext by (p/q), which reduces the noise without requiring the bootstrapping procedure.

A technique was explored to achieve SIMD fully homomorphic encryption [41]. Due to the noise associated with adding and multiplying, a scheme is designed whereby many messages can be packed into one ciphertext. Hence, we can add and multiply ciphertexts and affect many messages. They also describe a clever permutation trick that allows one to achieve a complete set of operations.

Certain applications, like cloud computing, require endless computation on data since we want to delegate our data processing and storage to a third party. However, it seems that a somewhat homomorphic scheme (without bootstrapping) is also useful if the function to be evaluated is simple and a known stopping point exists. In other words, we know the exact function we wish to compute, and we can set the parameters of the scheme such that it avoids decryption errors.

The security of somewhat homomorphic encryption schemes, which have been introduced recently as a stepping stone to achieve fully homomorphic encryption, are based on new hardness assumptions. These are not as well known and not as well studied as classical security assumptions like the discrete logarithm problem. A recent work by [15] aims to fill this gap by constructing a somewhat homomorphic encryption scheme that is as hard as factoring a large semiprime integer. Before we investigate the security properties of this scheme, we will explore the security problem of a previous scheme, which is also based on elementary number theory for its construction.

3.2 Brenner et al.'s Homomorphic Scheme

This section reviews the symmetric somewhat encryption scheme due to [15]. We review the basic construction, homomorphic properties, parameter choice and security claims.

3.2.1 Basic Construction

The scheme has the following parameters:

- the security parameter λ
- the bit length of the initial noise η
- the modulus p, which is a large prime integer of order 2^λ
- the bit length ρ of the message space, which is $\lambda - \eta$

Based on these security parameters, the scheme is composed as a tuple $(P, C, K, E, D, \oplus, \boxtimes)$, where elements are defined as the following:

P is the plaintext and contains elements from $\mathbb{N}^+$ limited by the prime integer p of order 2^λ such that for two plaintext operands $a, b \in \mathbb{N}^P, a \cdot b < p$ and $\mathbb{N}^P := \{x|x < 2^\eta\}$

C is the ciphertext space and contains elements from $\mathbb{N}^+$.

K is the key generator. The secret key is a large prime p, with an auxiliary public compression argument d with $d \leftarrow 2s + rp$, where $r \in \mathbb{N}^+$ and $s \in \mathbb{N}^C$ with $\forall x \in \mathbb{N}^C, \forall y \in \mathbb{N}^P, 2x < y$.

E is the encryption function. A bit value b is encrypted by picking a random integer a where $a \equiv b \ (mod\ 2)$ and adding a random multiple of the prime p, as

$a' = a + (rp)$. For security, r must contain at least one large prime factor of order 2^{λ}. The noise component must belong to the set of positive natural numbers $a \in \mathbb{N}^{+}$.

D is the decryption function. The message is recovered by applying the modulus $p : a = a' \ (mod\ p)$.

$\boxplus$ and $\boxtimes$ are the addition and multiplication on the ciphertext, respectively. Since these operations are similar to the related operations on the plaintext, they are mixed additive and multiplicative. In other words, the homomorphic properties are available to those who do not have the ability to encrypt.

Parameter Choice: The scheme requires that the prime key p and factor r must be sufficiently large to make factorization infeasible. The RSA-challenges suggest that each factor be 512 bits in length, resulting in a 1024 integer. This will prevent a factorization attack on a single ciphertext given current technology. The proposed size for the initial noise of a ciphertext is 8 bits.

3.2.2 Homomorphic Properties

The correctness of the homomorphic operations $\boxplus$ and $\boxtimes$ when given two ciphetexts a' and b' are proven by Equations 3.1 and 3.2. The correctness still holds using the compression argument, under the condition that the noise is sufficiently small.

$$\begin{aligned} a' \boxplus b' &= (a + r_1 p) + (b + r_2 p) \\ &= a + b + (r_1 + r_2)p \\ &= a + b \ (mod\ p) \end{aligned} \tag{3.1}$$

$$\begin{aligned} a' \boxtimes b' &= (a + r_1 p)(b + r_2 p) \\ &= ab + a(r_2 p) + b(r_1 p) + (r_1 r_2)p^2 \\ &= ab \ (mod\ p) \end{aligned} \tag{3.2}$$

The somewhat homomorphic scheme is also correct for one ciphertext and one plaintext, which is called mixed additive and multiplicative. This is shown below in Equations 3.3 and 3.4, where b is an integer with the same parity as the message.

$$\begin{aligned} a' \boxplus b &= (a + r_1 p) + b \\ &= a + b \ (mod\ p) \end{aligned} \tag{3.3}$$

$$\begin{aligned} a' \boxtimes b &= (a + rp)b \\ &= ab + rpb \\ &= ab \ (mod\ p) \end{aligned} \tag{3.4}$$

3.2.3 Security Claims

Before we look at the security assumptions given by Brenner et al. [15], we will recall the approximate-gcd problem [55]. For this definition, assume the following parameters.

$\boldsymbol{\gamma}$ is the bit-length of the integers in the public key.

$\boldsymbol{\eta}$ is the bit-length of the secret key.

$\boldsymbol{\rho}$ is the bit-length of the noise.

$\boldsymbol{\tau}$ is the number of integers in the public key.

Definition 6 (Approximate-GCD). *The (ρ, η, γ)-approximate-gcd problem is: given polynomially many samples from*

$$\mathcal{D}_{\gamma,\rho}(p) = \left\{ \texttt{choose}\ q \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/p), r \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho}) : \texttt{output}\ x = pq + r \right\}$$

for a randomly chosen η-bit odd integer, output p.

In informal terms, when given access to the public key, which is a set of near-multiples of p, find p. When the parameters above, in particular the noise, are set large enough, this is a difficult problem. However, the parameters required to achieve strong security prohibit the practical uses of the scheme. The Brenner et al. scheme addresses this by basing their system on different assumptions.

The somewhat homomorphic encryption scheme that has been revised here is symmetric. Simply, this means that there is no public key for encryption. However, the scheme still has utility as the homomorphic properties are possible without such information. We briefly recall the Lemmas of [15], and analyse their implications for security.

Lemma 1. *Let the Parameters $(p, q) \in (N)^+$ be of order 2^λ. Any Attack $\mathcal{A}$ running in time polynomial in λ against the encryption scheme can be converted into an algorithm $\mathcal{B}$ for solving the integer factorization problem of any composite integer number (pq) running in time polynomial in λ.*

This Lemma asserts that given a ciphertext $a' = (a + rp)$, according to the encryption function, then it is difficult to find the message, where r is composed of at least one prime q of order λ. Under this scenario, even if we remove the noise component a (which has the same parity as the message), then we would have to factor pq. When the product of pq is sufficiently large and p and q are roughly the same size, this is a hard problem. The next Lemma, however, introduces a security concern.

Lemma 2. *The security of the encryption scheme is IND-CPA equivalent and the success probability $|Pr[Exp_{ind-cpa} = 1] - \frac{1}{2}|$ is negligible in λ for any $\mathcal{A}$ from PPT (probabilistic polynomial time) function.*

This Lemma builds on the previous assertion about security under the indistinguishability under a chosen plaintext attack (IND-CPA) model. This model allows an adversary $\mathcal{A}$ to sample as many ciphertexts from an encryption oracle O_{Enc} as it desires. Then it is asked to distinguish an encryption of a message from random numbers. Since all ciphertexts have a large common element p, we can show that this assumption presents a problem. We explore this issue in more detail next.

3.3 Cryptanalysis

The security goal for the homomorphic scheme presented by [15] is indistinguishability under a chosen plaintext attack (IND-CPA). That is, if an adversary $\mathcal{A}$ has access to an encryption oracle $\mathcal{O}_{Enc}$, then they cannot distinguish between an encryption of a message and a random number greater than a negligible probability. When we examine this scheme under this model we find that, due to the nature of the setup, information about the secret key p can leak.

Even if we do not allow an adversary $\mathcal{A}$ access to $\mathcal{O}_{Enc}$, partial or complete information about the secret key can be discovered from the ciphertexts alone. Intuitively, this information leakage depends on the security parameters. We will look at these two modes of attack, a Chosen Plaintext Attack and Ciphertext Only Attack, in more detail with respect to the proposed security parameters: $\lambda = 512$ and $\eta = 8$.

3.3.1 Chosen Plaintext Attack

If an adversary $\mathcal{A}$ has access to an encryption oracle $\mathcal{O}_{Enc}$, then this essentially allows $\mathcal{A}$ to encrypt without complete control over the input. In terms of this scheme, the adversary can only control the parity of a. So, under the IND-CPA model we can sample N ciphertexts and then try to find p. If μ and ν are both exact multiples of p, then it is trivial to find p due to the Euclidean algorithm (see Algorithm 1).

Algorithm 1 $GCD(\mu, \nu)$

```
**Input:** $\mu, \nu$
**Output:** the greatest common divisor
while $\nu <> 0$ do
    if $\mu > \nu$ then
        $l = \nu$
        $\nu = \mu \ (mod\ \nu)$
        $\mu = l$
    end if
end while
```

There are two good properties about this algorithm. First it runs in logarithmic time of its inputs. Second we do not need to factorise either μ or ν in order to find the greatest common divisor. In the case of the somewhat homomorphic encryption scheme, we have a existing knowledge of p. From the description of the encryption scheme we know that p is a prime number.

This scheme is very similar to the first somewhat homomorphic scheme over the integers [98], where the security is reduced to the approximate-gcd problem, but we only required that p be an odd number. This distinction gives us some more power to find p, as we are able to eliminate potential values of p using efficient prime number tests. Thus we have a new variant of the approximate-gcd problem, which is given by the following definition.

Definition (Prime-Valued Approximate-GCD Problem) 1. *Let c_i be N ciphertexts for $1 \leqslant i \leqslant N$, where $c_i = a_i + q_i p$ with a_i and q_i chosen from suitable random distributions, and each a_i has the same parity of the message m_i. The problem is to find prime p.*

From this definition we can build a procedure that can find the private key p, since we have the additional fact that p is a prime. As a sketch of a simple procedure for finding p, we outline the following steps (with respect to the IND-CPA model).

1. Sample 2 ciphertexts c_1 and c_2 from the encryption oracle O_{Enc}.

2. Generate a set of potential candidates C of p from the two ciphertexts c_1, c_2 by iterating over all possible noise values.

3. Eliminate invalid elements from the set of candidate values C that do not match the characteristics of p: a prime and an integer of λ bits.

4. Test the set of potential values of p by sampling ciphertexts c_i, for $1 \leqslant i \leqslant N$, from the encryption oracle O_{Enc} and testing whether decryption works correctly.

With this sketch in mind we present Algorithm 2 that generates a set of candidates from two ciphertexts. This algorithm iterates over the possible noise values, and generates the resulting GCD.

Algorithm 2 *GenerateCandidates*(x, y, η)

```
Input: $x, y$ as ciphertexts and $\eta$ as initial noise size
Output: $C$ set of candidate values of $p$
$C \leftarrow \varnothing$
for $i = 1 \rightarrow 2^\eta$ do
    for $j = 1 \rightarrow 2^\eta$ do
        $x' = x - i$
        $y' = y - j$
        $C \leftarrow C \cup GCD(x', y')$
    end for
end for
return $C$
```

Under ideal conditions, we perform two tests. We test that the candidate c is prime using the *is_prime* function. The function *is_prime* can be implemented by using methods such as the Miller-Rabin primality test or the Fermat primality test. We also test that p within one bit either side of λ. These two tests allow us to reduce the number of candidates to a more manageable size. The details of the process is given by Algorithm 3.

Algorithm 3 $Find_p(c_1, c_2, \lambda, \eta)$

```
Input: $c_1, c_2$ as two ciphertexts outputted by the encryption function, $\lambda$ as the bit
    length of $p$ and $\eta$ as the initial noise size
Output: $E$ as a set of potential values of $p$
$E \leftarrow \varnothing$
$C \leftarrow GenerateCandidates(c_1, c_2, \eta)$
for $c \in C$ do
    if $(is\_prime(c)) \wedge (2^{\lambda-1} \leqslant |c| \leqslant 2^{\lambda+1})$ then
        $E \leftarrow E \cup \{c\}$
    end if
end for
return $E$
```

We notice that Algorithm 3 may eliminate candidates that are very near multiples of p. For example, Algorithm 2 (GenerateCandidates) may return values like $2p$ or $4p$, which are not prime, but still may help us to find p. Using this fact, we create Algorithm 4 that factors out any ‘small’ prime factors. We remark that there is no need to test for primality, as the small prime factors that would cause the test to fail have been removed.

Algorithm 4 $Find_p^{factor}(c_1, c_2, \lambda, \tau, \eta, P_\beta)$

```
Input: $c_1, c_2$ as two ciphertexts outputted by the encryption function, $\lambda$ as the bit length of $p$, $\tau$ as the bit range for $p$, $\eta$ as the initial noise size and $P_\beta$ denotes all primes less than $\beta$
Output: $E$ as a set of potential values of $p$
$E \leftarrow \varnothing$
$C \leftarrow GenerateCandidates(c_1, c_2, \eta)$
for $c \in C$ do
    for $\alpha \in P_\beta$ do
        while $\alpha | c$ do
            $c = c/\alpha$
        end while
    end for
    if $(2^{\lambda-\tau} \leqslant |c| \leqslant 2^{\lambda+\tau})$ then
        $E \leftarrow E \cup \{c\}$
    end if
end for
return $E$
```

Once we have a filtered list of candidate values C of p, we need to verify that it decrypts correctly. We now develop an algorithm that tests a candidate value of p by using the decrypting function with σ ciphertexts, and checking whether the output matches the message. The outcome of each outcome results in either a success or failure. We only accept a value of p that correctly decrypts each time. Thus, the probability of having the correct p is $1 - \frac{1}{2^\sigma}$, where σ is the number of ciphertexts. The testing algorithm is described by Algorithm 5.

When the noise component is large (i.e. $\eta > 60$), iterating over all possible values becomes infeasible. We can only do a subset of the possible values, which are chosen at random. Hence, we can only achieve a probabilistic, instead of a deterministic, result.

Algorithm 5 $Test_p(c_1, c_2, ..., c_\sigma, m, p)$

```
Input: $c_1, c_2, ..., c_\sigma$ as ciphertexts outputted by the encryption function, $m$ as the
    message encrypted by the ciphertexts $p$ as a candidate value to test
Output: $\beta$ as the probability of correct $p$ or $\perp$ as failure
for $i = 1 \to \sigma$ do
    if $D_p(c_i) \neq m$ then
        abort
        return $\perp$
    end if
end for
return $1 - \frac{1}{2^\sigma}$
```

3.3.2 Ciphertext Only Attack

In the IND-CPA attack model described above we consider sampling two ciphertexts from the encryption oracle $\mathcal{O}_{Enc}$ to mount our attack. In the real world, however, we do not have access to the encryption oracle because the scheme is defined as a secret key system. But if we examine the encryption scheme's description we find that an adversary $\mathcal{A}$ has at least one ciphertext, which is the compression argument $d = 2s + rp$. We can consider this to be an encryption of 0, since the noise component is $2s$. Hence, given one ciphertext (which is not the compression argument), we are still able to launch our attack as outlined above.

An application of a somewhat homomorphic encryption scheme that uses a single ciphertext (i.e. a single bit) would be very limited in what you could achieve. Hence it would be reasonable to consider that there are many ciphertexts available to an adversary $\mathcal{A}$, which includes at least one ciphertext as the compression modulus. If we consider n ciphertexts then there are a total of $\frac{n(n+1)}{2}$ possible pairs. We can arrange our attack in parallel and stop the first thread that finds p.

We can also use the fact that, by definition, the ciphertext is very malleable. This means we are able to modify the message by manipulating the ciphertext. Hence, if we are given a ciphertext c with an unknown message m, we can force the message to be either 0 or 1 by adding a small amount of noise to the ciphertext.

3.4 Experimental Evaluation

We now experimentally explore the security of Brenner et al.'s symmetric somewhat homomorphic scheme. All of the experiments were programmed with Java version 7 (in a single thread model) and executed on an Intel i7-2600 3.4GHz machine with 16GB of RAM. Where possible, software from the Java standard library was used, with the GCD algorithm being one example.

3.4.1 Setup

In our experiment we consider the IND-CPA security model. This means we will construct an encryption function that will take a message m and output a ciphertext that encrypts m. For this encryption function we will fix $\lambda = 512$, but we will have five noise levels $\eta_1 = 8, \eta_2 = 9, \eta_3 = 10, \eta_4 = 11, \eta_5 = 12$ to show the relative performance between different noise values.

Using this setup we sample two ciphertexts c_1, c_2 from the encryption function and execute our two algorithms $Find_p$ and $Find_p^{factor}$ (Algorithms 3 and 4, respectively). For simplicity, we set τ (the range of p) for both algorithms as 1. We used a certainty value of 3 in the is_prime function in $Find_p$, which defines how many iterations of the Miller-Rabin primality test are executed. We set $\beta = 100$ for the small prime numbers to factor out.

3.4.2 Results

We present the results of our experiment in Figure 3.1. We first observe as the noise gets larger, the time required for $Find_p$ and $Find_p^{factor}$ grows exponentially. This makes sense because we must be prepared to search a larger space to find p. Since we are searching the entire noise space, then we are guaranteed to have a case where the noise component of each ciphertext is removed. What remains for each ciphertext is some multiple of p. Once this case has been identified, it is a simple matter of

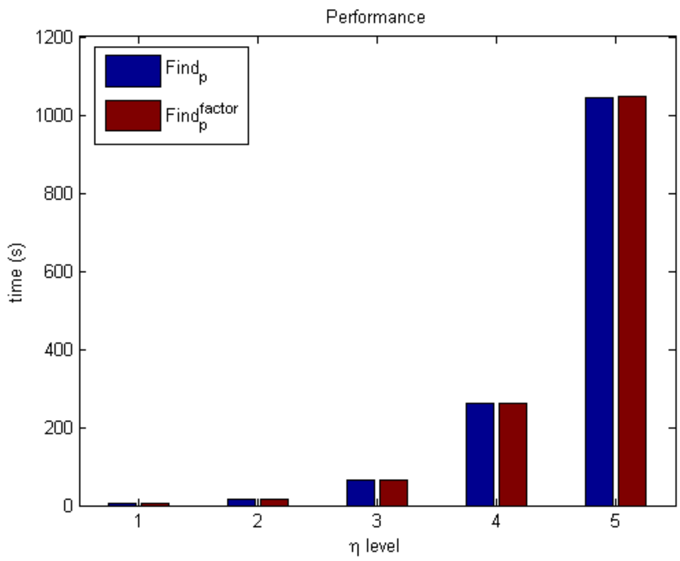


Figure 3.1: Performance when using 2 ciphertexts

finding the prime p itself, where the output of the GCD algorithm may be p times a small multiple.

In our experiment we itereated over all possible noise values, both even and odd. Since we know that the noise is in fact even ($m = 0$), we can reduce the search space by half by iterating only on even noise values. We did not use this fact in our experiment because we wanted to show the maximum time required for our approach. Even in this case, our approach is still feasible.

We also notice that there is negligible difference between the two algorithms, $Find_p$ and $Find_p^{factor}$. They seem to perform equally well, especially at the encryption scheme's proposed noise level of 8 bits.

3.5 Discussion

Obviously, increasing the noise associated with the ciphertexts greatly impacts the performance and accuracy of our method as the value of p is more difficult to uncover. Although adding noise in this way is removing the significance of the encryption scheme, which is basing new technologies on old hardness assumptions in order to reduce the size of the resulting noise component. There have been more advanced methods for discovering p based on representing the problem as a lattice, and then solving for p. In this case of this encryption scheme, a lattice approach was unnecessary as the proposed noise is small.

While keeping the same security assumption, there have been efforts for improving the practicality of somewhat homomorphic encryption schemes. One such effort reduced the size of the public key [25] by using a quadratic form in place of a linear form for the integers in the public key. In another work [102], the public key down to two integers by observing that any amount of noise can be added during encryption without access to many encryptions of zero.

3.6 Recommendations

In this chapter we explored the security characteristics of a new symmetric somewhat homomorphic encryption scheme. The scheme is able to evaluate simple circuits on encrypted data, where there was a known stopping point (the ciphertext noise did not grow too big, as to cause decryption to fail).

The scheme based its security on a well-known hardness assumption: factoring a semi-prime integer $n = pq$ where p and q are large primes. We showed theoretically and experimentally that this assumption is misleading, as we were able to break the scheme under the proposed parameters, principally using the Euclidean (GCD) algorithm. From this we designed an experiment that demonstrated the practicality of our approach. Our results suggest that we only need to break the approximate-gcd problem to find p. This suggests that, since these techniques are new, extensive research is required in order to satisfy both security and practicality issues.

The current research effort for practical fully homomorphic encryption is strong. This is mainly due to the long list of attractive applications (homomorphic spam filtering and private email). Although, fully homomorphic encryption is not at a suitable practical level for industry, somewhat homomorphic encryption seems to have good applications as [15] demonstrated. However, we must be cautious with new technologies based on new hardness assumptions, as there may be a mathematical back door that limits their potential.

Chapter 4

Computationally Efficient Private Information Retrieval

It can be shown that all of the private information retrieval schemes presented so far can be reduced to some kind of *homomorphism*. Where homomorphism means that some of the inherent structure is maintained after an operation. As already discussed, this allowed someone to change data even when it was encrypted. This allows the evaluation of some simple circuits on the data, such that a retrieval algorithm could be constructed.

This chapter explores the computational performance of private information retrieval schemes and is based on the contributions found in the publication entitled 'Single-Database Private Information Retrieval from Fully Homomorphic Encryption', which is published in IEEE Transactions on Knowledge and Data Engineering. As the title suggests, it presents a private information retrieval scheme based on an encryption scheme that permits addition and multiplication simultaneously.

In terms of the privacy model type, this chapter presents contributions mainly for the private client/public server privacy model type, as given in Chapter 1. However, it can be transformed into a private client/private server privacy model type due to contributions in [28].

Before we introduce newer techniques to implement private information retrieval, we will revisit the classical single-database private information retrieval schemes in terms of their computational performance. In particular, the fundamental primitives that permit these private information retrieval schemes will be analysed. From this analysis, we summarise two new private information retrieval schemes: one based on lattices and one based on the concepts of fully homomorphic encryption. Although, as they appear to be quite different in construction, they have one thing in common: they use only simple operations. Then, we carefully consider the performance implications of the two schemes. We conclude this chapter with some recommendations and ideas for future research.

4.1 Performance of Classical Private Information Retrieval Schemes

Initially, private information retrieval schemes were designed to be communicationally efficient. At the time, networks were slow and it seemed logical that the less communication than was absolutely necessary, the better. This has meant that the computational complexity of such protocols has been neglected [78]. In fact, [92] show that the PIR scheme based on the quadratic residuosity problem is the most computationally efficient when compared with other private information retrieval schemes [18, 17, 42]. Intuitively, this makes sense since as we increase the work done by a server, the amount of data that needs to be communicated is reduced. With the ever increasing speed of Internet access, the communication requirement is not so much of an issue.

The main source of the computational efficiency reduction is the requirement of modular exponentiations. These operations, while secure, are known to be computationally expensive. Schemes based on these operations are typically used in conjunction with symmetric encryption schemes when transferring large amounts of data securely. Subsequently, we need to develop schemes that use simpler operations and still achieve the same level of security. Simpler operations, like addition and multiplication, will promote the ability to achieve better performing systems.

A new proposal [3, 68, 69] aims to address this requirement by producing a computationally efficient solution. This proposal is based on linear algebra (i.e. *lattices*), and it has high computation throughput because the operations are simple and are efficiently executed in parallel. We also propose a private information retrieval scheme [105] based on the properties of fully homomorphic encryption [38, 98]. As in the lattice-based scheme, high computational efficiency can be achieved, when compared with previous schemes based on number theory. In this chapter we will explore the security and performance of these two schemes.

4.2 Lattice-based Private Information Retrieval

This section summarises the lattice-based scheme proposed by [3, 68, 69]. More specifically, we review the scheme in [3]. The scheme is with respect to three integer parameters: 2N, the dimension of the lattice; and special parameters p and q. As with all private information retrieval schemes, the scheme is composed of three phases: Query Generation (Algorithm 6), Response Generation (Algorithm 7), and Response Retrieval (Algorithm 8). At a high level, the user generates random matrices, where there is one matrix for each record in the database.

4.2.1 Correctness

Abstractly, this scheme works because the message is induced into the noisy matrices. The user is able to recover the noise component from the summation of all the matrices, and thus remove it to recover the message. As long as the noise does not grow too large, then this should hold true.

Stated a little more formally, each matrix has the form $M_i = [A_i|B_i + D_i\Delta]$, where the $D_i\Delta$ represents the additive noise applied to matrix B. When these matrices are multiplied by the database records, the values of the database records are induced into this noise component. For convenience, these products are added together, and then sent to the client. The client recovers the database record at i_0 by computing the (unscrambled) noise vector E.

Algorithm 6 Lattice Query Generation (Melchor2007)

```
Input: An index $i_0$.
Output: An ordered set of matrices $\{M_1, ..., M_n\}$.
Compute $l_0 = log(n \times N) + 1$, then set $q = 2^{2l_0}$ and $p \geqslant 2^{3l_0}$.
Let $M = [A|B]$, where $A$ and $B$ are two random matrices over $Z/pZ$, such that
    $A$ is invertible over $Z/pZ$.
For each $i \in [1, N]$, compute a matrix $M''_i = [A_i|B_i]$ by multiplying $M$ by a
    random invertible matrix $P_i$.
Generate a random diagonal $N \times N$ matrix $\Delta$ over $Z/pZ$.
For each $i \in \{1, ..., N\}\backslash i_0$ generate soft noise matrix $D_i$ as a $N \times N$ random matrix
    over $\{-1, 1\}$ and compute the soft disturbed matrix as $M'_i = [A_i|B_i + D_i\Delta]$.
Generate $D_{i_0}$, the hard noise matrix, by first generating a soft noise matrix.
    Then, replace each diagonal element by $q$.
Compute the hard disturbed matrix as $M_{i_0} = [A_{i_0}|B_{i_0} + B_{i_0}\Delta]$.
Construct a non-trivial permutation of columns $\mathcal{P}(\cdot)$ and compute $M_i = \mathcal{P}(M'_i)$
Transfer the ordered set $\{M_1, ..., M_n\}$ to the server.
```

Algorithm 7 Lattice Response Generation (Melchor2007)

```
Input: An ordered set of matrices $\{M_1, ..., M_n\}$.
Output: A response vector $V$.
Divide each database element $m_i$ into $N$ $l_0$-bit integers $\{m_{i1}, ..., m_{iN}\}$.
For each $i \in \{1, ..., n\}$ compute the vector $v_i = \sum_{j=1}^{N} m_{ij}M_{ij}$, where $M_{ij}$ represents
    the $j$-th row of $M_i$.
Compute $V = \sum_{i=1}^{n} v_i$
Transfer $V$ to the user.
```

Each component in this vector can be represented by the expression: $E_i = m \cdot q + \epsilon$. As long as the ϵ, also known as the soft noise satisfies $\epsilon < q$, then it can be removed. Once removed, m can be recovered by multiplying by q^{-1}, as required. Obviously, $m \cdot q$ cannot grow larger than p (the finite field size $\mathbb{Z}_p$), otherwise the process will incur information loss.

4.2.2 Security Assumptions

The security of this scheme is supported by the hardness of two different, but related, problems. The first concerns the difficulty of breaking the structure of the scheme.

Algorithm 8 Lattice Response Retrieval (Melchor2007)

```
Input: A response vector $V$.
Output: The message $m_{i_0}$.
1: Invert the permutation as $V' = \mathcal{P}^{-1}(V)$.
2: Retrieve the scrambled noise $E = V'_U - V'_D A^{-1} B$, where $V'_U$ and $V'_D$ are the
   undisturbed and disturbed halves of $V'$ respectively.
3: Compute the unscrambled noise as $E' = E\Delta^{-1}$.
4: For each $e'_j$ in $E = [e'_1, ..., e'_n]$, compute $e''_j = e'_j - \epsilon$, where $\epsilon = e'_j \ mod\ q$ if
   $e'_j \ mod\ q < q/2$, else $\epsilon = e'_j \ mod\ q - q$.
5: For $j \in \{1, ..., n\}$, compute $m_{i_0 j} = e''_j q^{-1}$.
```

Informally this means determining the Δ and $\mathcal{P}(\cdot)$ that was applied to a single matrix. The second hardness assumption is about indistinguishably among many matrices and what are SDM or HDM. We now review the formal definitions for security as defined by [3], and then we explain the meaning.

Structural Security

Here is the definition of the problem the authors of [3, 68] base their structural security on, which is called the Hidden Lattice Problem. To reiterate this is about a single matrix of the form $M_i = [A_i | B_i + D_i \Delta]$, and what transformations are applied to it. The following definition is about the ability for an adversary to determine which columns are disturbed within a matrix.

Definition 7. *Hidden Lattice Problem(Melchor2007) Let V be a k dimensional space of length n over a finite field $GF(p)$ for p a large prime number. Consider a set of r different random basis $\{V_1, ..., V_r\}$ of V with $V_i = [V_{i,1}|, ..., |V_{i,n}]$. Fix randomly a subset of s columns such that its complementary set $S = \{j_i, ..., j_{n-s}\}$ holds k independent columns. Choose randomly $i_0 \in \{1, ..., r\}$ and $q \in GF(p)$ with $1 \ll q \ll p$. For each V_i generate a set of random columns $\{R_{i,1}, ..., R_{i,n-k}\}$ such that $R_{i,j}$ is composed of elements in $\{-r_j, r_j\}$(r_j being a random element of $GF(p)$). For each $l \in \{1, ..., r\}$ multiply the l-th coordinate of $R_{i_0,l}$ by q. Disturb each V_i into V'_i by adding these random columns to $\{V_{i,j_1}, ..., V_{i,j_{n-k}}\}$. Deduce from the set of disturbed basis which are the $n - k$ disturbed columns.*

This is about how well the permutation $\mathcal{P}(\cdot)$ has reordered the columns and if there is enough noise to make them indistinguishable from each other. The authors of this scheme show that this problem is related to the Punctured Code Problem, given by [99].

Definition 8. *Punctured Code Problem (Wieschebrink2006) Let M be a $k \times n$-matrix, H a $k \times m$-matrix, $m \leqslant n$, both over finite field F. Does there exist a non-singular matrix T and a subset $S \subset \{1, ..., n\}$ with $|S| = m$ such that $(TM)_S = TM_S = H$?*

Indistinguishably

We now review the definition by [3] that takes into account indistinguishably. This defines the problem concerns an adversary's ability to distinguish between two queries, as produced by the Lattice Query Generation algorithm (see Algorithm 6).

Definition 9. *Differential Hidden Lattice Problem(Melchor2007) Let V be a k dimensional vector space of length n over a finite field $GF(p)$ for p a large prime number and T_1, T_2 two different subsets of $\{1, ..., r\}$ with t_1 and t_2 elements. Consider a set of r different random basis $\{V_1...V_r\}$ of V with $V_i = [V_{i,1}|...|V_{i,n}]$. Fix randomly a subset of s columns such that its complementary set $S = \{j_1...j_{n-s}\}$ holds k independent columns. Choose randomly $q \in GF(P)$ with $1 \ll q \ll p$, $r \in \{1, 2\}$ and set $T = T_r$. For each V_i generate a set of random columns $\{R_{i,1}...R_{i,n-k}\}$ such that $R_{i,j}$ is composed of elements in $\{-r_j, r_j\}$ (r_j being a random element of $GF(p)$). For each $l \in \{1, ..., r\}$ and each $i \in T$ multiply the l-th coordinate of $R_{i,l}$ by q. Disturb each V_i into V_i' by adding these random columns to $\{V_{i,j_i}...V_{i,jn_{n-k}}\}$. Deduce from T_1, T_2 and the set of disturbed basis the value of r.*

Overall, the security of this scheme is based on the assumption that given a set of matrices $\{M_1, ..., M_n\}$, it is hard to decide which matrix in the client's query has been disturbed by the matrix D_{i_0}, where the i_0 refers to the index corresponding to the the client's query. If an adversary is able to do this more than negligible probability, then the scheme is broken. As with other schemes that are not based on number theory, the question of difficulty is tough to answer. The best guess is to provide a scheme with parameters that resist all known attacks.

4.3 Private Information Retrieval Based on Fully Homomorphic Encryption

At the start of this chapter we stated that all private information retrieval schemes are based on some kind of homomorphic encryption. Here we investigate what kind of scheme is possible when given access to a scheme that supports addition and multiplication on ciphertexts simultaneously.

Here we present such a scheme, which is based on our result in [105]. Before we describe the main idea, we will introduce a variant of the Dijk et al. [98], which is called V-DGHV for short, that removes the need for a large public key. The reasoning for this decision is because the server does not need to encrypt data to add randomness because we are only concerned with the client's privacy, by definition of private information retrieval. For notational convenience, the homomorphic operations Add and Mult are represented by $\boxplus$ and $\boxtimes$. They are extended to a sequence in the usual way. For example: $\boxtimes_{t=1}^{\ell} \alpha_\ell = \alpha_1 \boxtimes \alpha_2 \boxtimes ... \boxtimes \alpha_\ell$.

(1) $\mathsf{KeyGen}(\lambda)$: The user takes a security parameter λ and determines a parameter set $\rho = \lambda, \eta = (\lambda + 3)\lceil \log m \rceil, \gamma = 5(\lambda + 3)\lceil \log m \rceil / 2$. Chooses a random odd η-bit integer p from $(2\mathbb{Z}+1) \cap (2^{\eta-1}, 2^\eta)$ as the secret key sk. Randomly chooses q_0 from $(2\mathbb{Z}+1) \cap [1, 2^\gamma/p)$ and sets $x_0 = q_0 p$. The public key is $pk = x_0$.

(2) $\mathsf{Encrypt}(pk, M)$: To encrypt $M \in \{0, 1\}$, the user, who knows the secret key $sk = p$, randomly chooses q from $[1, 2^\gamma/p)$ and an integer r from $(-2^\rho, 2^\rho)$ and outputs the ciphertext

$$c = \mathsf{E}(M, pk) = (M + 2 \cdot r + q \cdot p) \ mod \ x_0.$$

(3) $\mathsf{Decrypt}(sk, c)$: With the secret key p, the user decrypts a ciphertext as the DGHV somewhat scheme, that is,

$$M = \mathsf{D}(c, sk) = (c \ mod \ p) \ mod \ 2.$$

(4) Homomorphic Addition (Add): Using the public key x_0, the database server adds two ciphertexts c_1 and c_2 as the DGHV somewhat scheme, that is,

$$\mathsf{Add}(c_1, c_2) = (c_1 + c_2) \ mod \ x_0.$$

(5) Homomorphic Multiplication (Mult): Using the public key x_0, the database server multiplies two ciphertexts c_1 and c_2 as the DGHV somewhat scheme, that is,

$$\mathsf{Mult}(c_1, c_2) = (c_1 \cdot c_2) \ mod \ x_0.$$

Like the DGHV somewhat scheme, the choice of parameters in the variant scheme achieves at least 2^λ security against all of known attacks. As with the original scheme, addition and multiplication (mod 2) are equivalent to the Boolean gates XOR and AND respectively.

Before we describe the complete solution, we will give a simple solution to help motivate the complete solution. For this naive solution, consider a database of n-bits in length $D = d_1 d_2 ... d_n$. The client constructs their query by generating a stream of bits $Q = q_1 q_2 ... q_n$ of the same length as the database and setting $q_i = 1$ for the part of the database they wish to download. They encrypt each bit with the homomorphic encryption variant and send it to the server $\hat{q}_i = E(q_i)$ for $1 \leqslant i \leqslant n$. The server multiplies the query by $z_i = \prod_{i=1}^{n} \hat{q}_i r_i$, where $r_i = d_i$ (mod 2). Subsequently, the client can decrypt and obtain the data. Correctness follows immediately from the definition of the variant homomorphic encryption scheme. Figure 4.1 illustrates this process.

This solution, while satisfying the requirements for correctness, does not satisfy the requirement communication requirement for private information retrieval. As mentioned in Chapter 2, traditional private information retrieval schemes structure the query in terms of two indices when the database is arranged as a 2 dimensional array of bits, and three indices when the database is arranged as a 3 dimensional array of bits and so on. Since now we are not restricted to one homomorphic property, let us manipulate the index (see Figure 4.2), represented in a base two string of bits

Figure 4.1: Illustrating the simple solution using the homomorphic encryption scheme variant

(similar to the query above). This is described by Algorithm 9. For simplicity no encryption is used.

Algorithm 9 Response Generation Circuit

```
Input: An index $i \in [1, n]$ and an $n$-bit database $DB = b_1 b_2 \cdots b_n$
Output: $b_i$
Write the index $i$ in the binary representation, denoted as $i = \alpha_1 \alpha_2 \cdots \alpha_\ell$, where $\ell = \lceil \log n \rceil$.
For each index $j \in [1, n]$, write $j$ in the binary representation, denoted as $j = \beta_{j,1} \beta_{j,2} \cdots \beta_{j,\ell}$. Compute
    $$\gamma_j = \prod_{t=1}^{\ell} (\alpha_t \oplus \beta_{j,t} \oplus 1), \qquad (4.1)$$
    where $\oplus$ stands for XOR operation. If $j = i$, $\gamma_j = 1$ and 0 otherwise. This means only $\gamma_i = 1$.
Output
    $$R = \bigoplus_{b_j = 1} \gamma_j. \qquad (4.2)$$
```

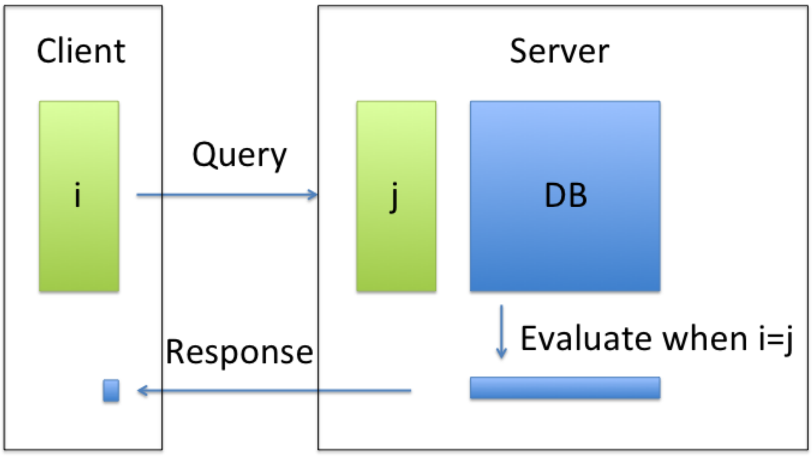


Figure 4.2: Graphically showing the indices in relation to the database

If $b_i = 1$, then $\bigoplus_{b_j=1} \gamma_j = \gamma_i = 1$. If $b_i = 0$, then $\bigoplus_{b_j=1} \gamma_j = 0$. Therefore, $R = \bigoplus_{b_j=1} \gamma_j = b_i$, as required.

The response generation circuit needs only two simple operations, addition ($\oplus$) and multiplication ($\cdot$). These are the two operations that are supported by the homomorphic encryption variant. Hence we can use the homomorphic encryption variant to enable the server to evaluate the response generation circuit while providing protection for the user's query. We first present the generic single database private information retrieval scheme, that allows the user to retrieve a single bit. Next we extend this bit-bases scheme to a block based scheme.

Generic Single-Database PIR from FHE

The generic single-database private information retrieval protocol is built on a FHE scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D}, \mathsf{Add}, \mathsf{Mult})$, as defined above. It is defined by the standard three algorithms for constructing a private information retrieval scheme (see Chapter 2): Query Generation (QG), Response Generation (RG), and Response Retrieval (RR).

As an overview of the scheme, the user generates a public and private key pair (pk, sk) according to the key generation procedure of the FHE scheme, and sends the pk to the database server, but keeps the private key sk secret. The user then constructs their query as follows. They choose the index i of interest and encrypts it using pk. We reiterate that the client is able to encrypt because they know the sk. This ciphertext is sent to the database server. Then the server computes i-th bit of the database using the response generation circuit, the public key pk, and homomorphic properties of the encryption scheme. This is sent to the client as the response. Finally, the user decrypts the response using the sk, revealing the i-th bit. This is decribed in more detail by Algorithms 10, 11, and 11.

Algorithm 10 Query Generation $\mathsf{QG}(n, i, 1^k)$

```
Input: The size $n$ of the database $DB$, an index $i \in [1, n]$, the key generation
    algorithm $\mathsf{KG}$, the encryption algorithm $\mathsf{E}$, and a security parameter $k$.
Output: A query $Q = (pk, \mathsf{E}(i, pk))$ and a secret $s = sk$, where $(pk, sk)$ is a public
    and private key pair for the FHE scheme and $\mathsf{E}(i, pk)$ is the encryption of $i$ with
    the public key $pk$.
(The user) generates a public and private key pair $(pk, sk)$ with the key generation
    algorithm ($\mathsf{KG}$) and the security parameter $k$, i.e., $(pk, sk) = \mathsf{KG}(1^k)$.
Assume that the binary representation of $i$ is $\alpha_1 \alpha_2 \cdots \alpha_\ell$, where $\alpha_i \in \{0, 1\}$ and
    $\ell = \lceil \log n \rceil$. (The user) encrypts each $a_j$ with the public key $pk$, denoted as
    $\hat{\alpha_j} = \mathsf{E}(\alpha_i, pk)$. Let $\mathsf{E}(i, pk) = (\hat{\alpha_1}, \hat{\alpha_2}, \cdots, \hat{\alpha_\ell})$.
Output the query $Q = (pk, \mathsf{E}(i, pk))$ and a secret $s = sk$.
```

Theorem 1. (Correctness) The generic single-database PIR from FHE is correct for any security parameter k, any database DB with any size n, and any index $1 \leqslant i \leqslant n$.

*P*roof: By comparing our response generation circuit and our response generation algorithm (RG), we can see that $\hat{\gamma_j}$ is an encryption of 1 when $j = i$ and an encryption of 0 otherwise, on the basis of fully homomorphic properties. Therefore, if $b_i = 1$, $R = \boxplus_{b_j=1} \hat{\gamma_j} = \hat{\gamma_i} = \hat{1}$, if $b_i = 0$, $R = \boxplus_{b_j=1} \hat{\gamma_j} = \hat{0}$. This means R is an encryption of b_i and thus $b' = \mathsf{D}(R, sk) = b_i$. □

Algorithm 11 Response Generation $\mathsf{RG}(DB, Q, 1^k)$

```
Input: An $n$-bit database $DB = b_1 b_2 \cdots b_n$, a query $Q = (pk, \mathsf{E}(i, pk))$, $\mathsf{E}$, $\mathsf{Add}$, $\mathsf{Mult}$,
    and a security parameter $k$.
Output: A response $R$.
For each index $j \in [1, n]$, (the database server) writes $j$ in the binary representation
    $\beta_{j,1}\beta_{j,2} \cdots \beta_{j,\ell}$. (The database server) encrypts each bit $\beta_{j,t}$ with the public
    key $pk$, denoted as $\widehat{\beta_{j,t}} = \mathsf{E}(\beta_{j,t}, pk)$ for $1 \leqslant t \leqslant \ell$, and computes
        $$\hat{\gamma}_j = \mathop{\boxed{\times}}\limits_{t=1}^{\ell} (\hat{\alpha}_t \boxplus \widehat{\beta_{j,t}} \boxplus \hat{1}), \qquad (4.3)$$
    where $\hat{1}$ is an encryption of 1.
(The database server) computes
        $$R = \mathop{\boxed{+}}\limits_{b_j = 1} \hat{\gamma}_j. \qquad (4.4)$$
Output the response $R$.
```

Algorithm 12 Response Retrieval $\mathsf{RR}((Q, s), R, 1^k)$

```
Input: $s = sk$, an output of $\mathsf{QG}(n, i, 1^k)$; $R$, an output of $\mathsf{RG}(DB, Q, 1^k)$, and the
    decryption algorithm $\mathsf{D}$.
Output: A bit $b$ at the position of $i \in [1, n]$.
Retrieves the bit as $b' = \mathsf{D}(R, sk)$.
```

Generic Single-Database PBR from FHE

Here we extend the single-database private information retrieval from FHE to a single-database private block retrieval. As before, this consists of three algorithms $(\mathsf{QG}, \mathsf{RG}, \mathsf{RR})$.

Assume that an n-bit database DB is equally partitioned into m blocks, denoted as $DB = B_1 \| B_2 \cdots \| B_m$, our single-database PBR by Algorithms 13, 14, and 15.

Theorem 2. (Correctness) The generic single-database PBR from FHE is correct for any security parameter k, any database DB with any size n and any number m of blocks, and any index $1 \leqslant i \leqslant m$.

Algorithm 13 Query Generation $\mathsf{QG}(m, i, 1^k)$

```
Input: The number $m$ of blocks in the database $DB$, an index $i \in [1, m]$, $\mathsf{KG}$, $\mathsf{E}$,
    and a security parameter $k$.
Output: A query $Q = (pk, \mathsf{E}(i, pk))$ and a secret $s = sk$, where $(pk, sk)$ is a public
    and private key pair for the FHE scheme.
(The user) generates a public and private key pair $(pk, sk)$ with the key generation
    algorithm ($\mathsf{KG}$) and the security parameter $k$, i.e., $(pk, sk) = \mathsf{KG}(1^k)$.
Assume that the binary representation of $i$ is $\alpha_1\alpha_2 \cdots \alpha_\ell$, where $\alpha_i \in \{0, 1\}$ and
    $\ell = \lceil \log m \rceil$. (The user) encrypts each $a_j$ with the public key $pk$, denoted as
    $\hat{\alpha_j} = \mathsf{E}(\alpha_i, pk)$. Let $\mathsf{E}(i, pk) = (\hat{\alpha_1}, \hat{\alpha_2}, \cdots, \hat{\alpha_\ell})$.
Output the query $Q = (pk, \mathsf{E}(i, pk))$ and a secret $s = sk$.
```

Algorithm 14 Response Generation $\mathsf{RG}(DB, Q, 1^k)$

```
Input: An $n$-bit database $DB = B_1 \| B_2 \cdots \| B_m$, where $B_j = (b_{j,1}, b_{j,2}, \cdots, b_{j,L})$ and
    $L = n/m$, a query $Q = (pk, \mathsf{E}(i, pk))$, $\mathsf{E}$, $\mathsf{Add}$, $\mathsf{Mult}$, and a security parameter $k$.
Output: A response $R$.
For each index $j \in [1, m]$, (the database server) writes $j$ in the binary representation
    $\beta_{j,1}\beta_{j,2} \cdots \beta_{j,\ell}$. (The database server) encrypts each bit $\beta_{j,t}$ with the public
    key $pk$, denoted as $\widehat{\beta_{j,t}} = \mathsf{E}(\beta_{j,t}, pk)$ for $1 \leqslant t \leqslant \ell$, and computes
        $\hat{\gamma_j} = \boxtimes_{t=1}^{\ell} (\hat{\alpha_t} \boxplus \widehat{\beta_{j,t}} \boxplus \hat{1}).$    (4.5)
For each $c \in [1, L]$, (the database server) computes
        $R_c = \boxplus_{b_{j,c}=1} \hat{\gamma_j}.$    (4.6)
Output the response
        $R = (R_1, R_2, \cdots, R_L).$
```

*P*roof: The generic single-database PBR can be viewed as running L generic single-database PIR protocols in parallel. In each single-database PIR, the user retrieves the i-th bit from an m-bit database $DB_c = b_{1,c}b_{2,c} \cdots b_{m,c}$ for $1 \leqslant c \leqslant L$.

Based on Theorem 1, we know each of L single-database PIR protocol is correct, that is, $\mathsf{D}(R_c, sk) = b_{i,c}$ for $1 \leqslant c \leqslant L$. Therefore, we have
$B' = (\mathsf{D}(R_1, sk), \mathsf{D}(R_2, sk), \cdots, \mathsf{D}(R_L, sk)) = B_i$. □

Algorithm 15 Response Retrieval $\mathsf{RR}((Q, s), R)$

```
Input: $s = sk$, an output of $\mathsf{QG}(n, i, 1^k)$; $R$, an output of $\mathsf{RG}(DB, Q, 1^k)$, and $\mathsf{D}$.
Output: A block $B' = (\mathsf{D}(R_1, sk), \mathsf{D}(R_2, sk), \cdots, \mathsf{D}(R_L, sk))$ at index $i \in [1, m]$.
  Retrieves the block as $B' = (\mathsf{D}(R_1, sk), \mathsf{D}(R_2, sk), \cdots, \mathsf{D}(R_L, sk))$.
```

4.3.1 Concrete Single-Database PBR from V-DGHV Scheme

The previous two schemes, the PIR and PBR protocols, have been fairly abstract in their definitions. We now make the complete scheme (private block retrieval) more concrete by actually using the functions defined by the variant of DGHV scheme as described above.

Assume that an n-bit database DB is equally partitioned into m blocks, denoted as $DB = B_1 \| B_2 \cdots \| B_m$, the practical single-database PBR from the variant of DGHV scheme is described as follows.

Algorithm 16 Query Generation $\mathsf{QG}(m, i, 1^k)$

```
Input: The number $m$ of blocks in the database $DB$, an index $i \in [1, m]$, $\mathsf{KG}$, $\mathsf{E}$,
    and a security parameter $k$.
Output: A query $Q = (x_0, \mathsf{E}(i, pk))$ and a secret $sk = p$, where $(x_0, p)$ is a public
    and private key pair for the V-DGHV scheme.
  (The user) generates a public and private key pair $(x_0, p)$ with the key generation
    algorithm ($\mathsf{KG}$) of the V-DGHV scheme and the security parameter $k$.
  Assume that the binary representation of $i$ is $\alpha_1 \alpha_2 \cdots \alpha_\ell$, where $\alpha_i \in \{0, 1\}$ and
    $\ell = \lceil \log m \rceil$. (The user) encrypts each $a_j$ with the public key $x_0$, denoted as
    $\hat{\alpha_j} = \mathsf{E}(\alpha_j, x_0) = (\alpha_j + 2 \cdot r_j + q_j \cdot p) \ mod \ x_0$, where $r_j$ and $q_j$ are randomly
    chosen on the basis of V-DGHV scheme. Let $\mathsf{E}(i, x_0) = (\hat{\alpha_1}, \hat{\alpha_2}, \cdots, \hat{\alpha_\ell})$.
  Output the query $Q = (x_0, \mathsf{E}(i, x_0))$ and a secret $sk = p$.
```

Theorem 3. *The V-DGHV scheme can correctly evaluate the response generation circuit of our practical PBR protocol.*

*P*roof: Suppose that the size of the noise in $\prod_{t=1}^{s} c_i$ is $\mathcal{N}(s)$, where $c_t = (m_t + 2r_t + q_t p) \ mod \ x_0$ is a fresh ciphertext and $r_t \in (-2^\lambda, 2^\lambda)$. According to Eq. (2), the part

Algorithm 17 Response Generation $\mathsf{RG}(DB, Q, 1^k)$

```
Input: An n-bit database $DB = B_1 \| B_2 \cdots \| B_m$, where $B_j = (b_{j,1}, b_{j,2}, \cdots, b_{j,L})$ and
    $L = n/m$, a query $Q = (x_0, \mathsf{E}(i, x_0))$, $\mathsf{E}$, $\mathsf{Add}$, $\mathsf{Mult}$, and a security parameter $k$.
Output: A response $R$.
1: For each index $j \in [1, m]$, (the database server) writes $j$ in the binary representation $\beta_{j,1}\beta_{j,2} \cdots \beta_{j,\ell}$. (The database server) computes
       $$\hat{\gamma}_j = \prod_{t=1}^{\ell} (\hat{\alpha}_t + (\beta_{j,t} \oplus 1)) \ mod\ x_0 \quad (4.7)$$
2: For each $c \in [1, L]$, (the database server) computes
       $$R_c = \sum_{b_{j,c}=1} \hat{\gamma}_j \ mod\ x_0 \quad (4.8)$$
3: Output the response
       $$R = (R_1, R_2, \cdots, R_L).$$
```

Algorithm 18 Response Retrieval $\mathsf{RR}((Q, p), R)$

```
Input: $sk = p$, an output of $\mathsf{QG}(n, i, 1^k)$; $R$, an output of $\mathsf{RG}(DB, Q, 1^k)$, and $\mathsf{D}$.
Output: A block $B'$ at index $i \in [1, m]$.
1: Retrieves a block as
       $$B' = ((R_1\ mod\ p)\ mod\ 2, (R_2\ mod\ p)\ mod\ 2, \cdots, (R_L\ mod\ p)\ mod\ 2).$$
```

of the noise in c_1c_2 is $2r_1r_2 + r_1m_2 + r_2m_1$ and

$$\mathcal{N}(2) \leqslant 2 \cdot 2^{\lambda} \cdot 2^{\lambda} + 2^{\lambda} + 2^{\lambda} < 2^{2\lambda+2}. \tag{4.9}$$

For any $s > 2$, we have

$$\begin{aligned} \mathcal{N}(s) &\leqslant 2 \cdot \mathcal{N}(s-1) \cdot 2^{\lambda} + \mathcal{N}(s-1) + 2^{\lambda} \\ &< 2^{s\lambda+2(s-1)} = 2^{(\lambda+2)s-2}. \end{aligned} \tag{4.10}$$

Therefore, for each $1 \leqslant j \leqslant m$, the size of the noise in $\hat{\gamma}_j = \prod_{t=1}^{\ell}(\hat{\alpha}_t + (\beta_{j,t} \oplus 1))$ *mod* x_0 ($\ell = \lceil \log m \rceil$) is less than $2^{(\lambda+2)\lceil \log m \rceil - 2}$ and thus the size of the noise in $R_c = \boxplus_{b_{j,c}=1} \hat{\gamma}_j$ for each c is less than $2^{(\lambda+2)\lceil \log m \rceil - 2} m \leqslant 2^{(\lambda+3)\lceil \log m \rceil}/4$, which is less than $p/2$.

In view of it, the V-DGHV scheme can correctly decrypt R_c for any $1 \leqslant c \leqslant n/m$. □

Based on the correctness of the generic PBR from FHE (Theorem 2), we can see that our practical PBR protocol is correct too.

4.3.2 Security Analysis

Since the single-database PBR protocol is a combination of the single-database PIR protocol and the practical PBR protocol from FHE is a special case of the generic PBR from FHE, we only need to analyze the security of the generic PIR protocol from FHE.

Based on the formal definition of security for single-database PIR protocol given in Chapter 2, we have the following theorem.

Theorem 4. *Assume that the underlying FHE scheme is semantically secure, then the generic single-database PIR protocol from FHE is semantically secure.*

*P*roof: We denote by $(\mathsf{KG}, \mathsf{E}, \mathsf{D}, \mathsf{Add}, \mathsf{Mult})$ the underlying FHE scheme. With reference to [89], suppose that there exists an adversary (a database server) $\mathcal{A}$ that can gain a non-negligible advantage ϵ in the semantic security game for the generic single-database PIR protocol. We prove that there exists an adversary $\mathcal{A}'$ (built on $\mathcal{A}$) who can gain a non-negligible advantage in breaking the semantic security of the underlying FHE scheme as follows.

The adversary $\mathcal{A}'$ initiates the semantic security game for the FHE scheme with some challenger $\mathcal{C}'$, which will send $\mathcal{A}'$ the public key pk for the challenge. For messages m_0 and m_1, we choose $m_0 = 0 \in \{0, 1\}$ and $m_1 = 1 \in \{0, 1\}$. After sending m_0, m_1 back to the challenger $\mathcal{C}'$, the adversary $\mathcal{A}'$ will receive $e_b = \mathsf{E}(m_b)$, an encryption of one of these two values. Next, $\mathcal{A}'$, playing a challenger $\mathcal{C}$, initiates the single-database PIR game with the adversary $\mathcal{A}$ with an n-bit database, who will give $\mathcal{A}'$ two different indices $1 \leqslant i, j \leqslant n$.

Let $x_0 = i$ and $x_1 = j$. The adversary $\mathcal{A}'$ picks a random bit q, and constructs a Q_q as follows: Assume that the binary expression of x_q is $(\alpha_{q,1}\alpha_{q,2} \cdots \alpha_{q,\ell})$ where $\ell = \log n$. The adversary $\mathcal{A}'$ constructs the encryption of x_q by replacing all zeros with $\hat{0}$ and all ones with $\hat{0} \boxplus e_b$. Note that $\hat{0}$ is the encryption of 0 with the public key *pk* and different randomness are chosen in $\hat{0}$ for different bits. We denote the result as $Y_q = (\hat{y_{q,1}}, \hat{y_{q,2}}, \cdots, \hat{y_{q\ell}})$.

Now the adversary $\mathcal{A}'$ gives a query $Q_q = (pk, Y_q)$ to the adversary $\mathcal{A}$, who then returns a guess q'. With probability 1/2, e_b is the encryption of 0, and hence Y_q is the encryption of all zeros, $\hat{\gamma_z} = \boxtimes_{t=1}^{\ell}(\hat{y_{q,t}} \boxplus \hat{\beta_{z,t}} \boxplus \hat{1}) = \hat{0}$ for all $1 \leqslant z \leqslant n$, and $R = \boxplus_{b_z=1} \hat{\gamma_z} = \hat{0}$. In this event, $\mathcal{A}$'s guess is independent of q, and hence the probability $q' = q$ is 1/2.

However, with probability 1/2, $e_b = \hat{1}$, hence Y_q is the encryption of x_q, constructed exactly as in the QG algorithm, and hence in this case with probability $1/2 + \epsilon$, the adversary $\mathcal{A}$ will guess q correctly, as the behavior of $\mathcal{A}'$ was indistinguishable for an actual challenger $\mathcal{C}$. The adversary $\mathcal{A}'$ determines his guess b' as follows: If $\mathcal{A}$ guesses $q' = q$ correctly, then $\mathcal{A}'$ will set $b' = 1$, and otherwise $\mathcal{A}'$ will set $b' = 0$. Putting it all together, we can now compute the probability that the guess of $\mathcal{A}'$ is correct:

$$\mathsf{Pr}(b' = b) = \frac{1}{2}(\frac{1}{2}) + \frac{1}{2}(\frac{1}{2} + \epsilon) = \frac{1}{2} + \frac{\epsilon}{2}$$

Therefore, the adversary $\mathcal{A}'$ has obtained a non-negligible advantage in the semantic security game for the underlying FHE scheme, a contradiction to our assumption in the theorem. Thus, the generic PIR protocol is semantically secure according to the security definition. □

Basically, this proof states that as long as the underlying cryptosystem is semantically secure, then the resulting protocol is secure. That is, a polynomial bounded adversary is unable to determine whether a ciphertext encrypts a 0 or a 1 with probability greater than half. Thus the encryptions used within the PIR protocol appear to be random and indistinguishable to the adversary.

4.4 Performance Comparison

In this section, we compare the performance between the lattice-based scheme and the scheme based on fully homomorphic encryption. For ease of this comparison, the performance of the client and server will be analysed separately. For this analysis, assume that the database is **n** bits long and **m** blocks.

The authors of the lattice-based scheme [3] suggest setting the parameters as $l_0 = 20$, $N = 50$, and $p = 2^{60} + 325$. This makes $q = 2^{2l_0} = 1099511627776$. On the other hand, the scheme based on fully homomorphic encryption, we set $\lambda = 60, \eta = 882, \gamma = 2205$. Thus, both the ciphertext size and the public key size are 2205 bits.

4.4.1 Client

In the lattice-based scheme, the user has to generate a set of matrices, one for each record in the database. Each of these matrices has dimension $[N, 2N]$, which are structured as $M = [A|B]$, where A is a random invertible matrix and $B = B' + D\Delta$. The most expensive part for producing the query is computing the matrix product $D\Delta$. Since Δ is a diagonal matrix this can be done in much less time than regular matrix multiplication. Upon receiving the response from the server, the client needs to perform one N by N matrix multiplication over $GF(p)$.

While in the case of the scheme based on fully homomorphic encryption, the user has to encrypt each bit of the index of the record they desire. This equates to encrypting $log\mathbf{m}$ bits using the somewhat encryption scheme variant. As the user has access to the secret key sk, they only require one multiplication per bit in the index. Upon receiving the database's response, the client must also perform $\mathbf{n}/\mathbf{m}$

decryptions. Since each decryption uses only a modulus reduction of an integer, we can treat this as negligible.

4.4.2 Server

Reducing the server's computational requirement is the main motivation behind this work. In the case of the lattice-based scheme, each matrix in the client's query is combined with each record in the server's database. This is done by dividing the record into N l_0-bit integers and multiplying each row in the matrix by the corresponding integer. With respect to the parameters outlined above this gives $100 * N$ multiplications.

By contrast, in the fully homomorphic based private information retrieval scheme, the database server is required to execute on the order of $\mathbf{m} \log \mathbf{m}$ multiplications and $\mathbf{n}/2$ additions[1].

4.4.3 Discussion

The main observation that can be made between the two protocols is that the one based on fully homomorphic encryption is conceptually simpler. The lattice-based scheme requires a somewhat lengthly and convoluted process. Whereas the FHE-based scheme is simply employing the homomorphic properties of the underlying scheme.

Assume that $\mathbf{m} = 10000$ and the system parameters given above for both schemes. In the lattice-based scheme, we need to store $N \times 2N$ numbers of 60 bits in length. Under the proposed parameters, this gives $50 \times 100 \times 60 = 300000$ bits. By comparison, the FHE-based scheme requires $\lceil log_2(10000) \rceil \cdot 2205 = 30870$.

It is difficult to say what parameters for either scheme is required for good security. This is because, as discussed in Chapter 3, the difficulty of the hardness assumptions are less understood than assumptions based on number theory.

[1] These are modular multiplications and additions.

4.5 Experiment

The main purpose of this section is to demonstrate the computational performance of the schemes introduced by this chapter. This will be with respect to the previous schemes based on number theory.

We will begin with the performance evaluation of the FHE-based scheme.Assume that the database DB is equally partitioned into $\mathbf{m} = 10000$ blocks, and the underlying encryption scheme is a variant of the DGHV somewhat homomorphic encryption scheme [98] with parameters $\lambda = 60, \eta = 882, \gamma = 2205$. In this setting, the somewhat encryption scheme is able to evaluate the response generation circuit of the practical PBR protocol without error and achieve the security level of 2^{60}.

To generate the response R, the database server needs to compute $\hat{\gamma_j} = \prod_{t=1}^{\ell}[\hat{\alpha_t} + (\beta_{j,t} \oplus 1)]_{x_0}$ and $R_c = [\sum_{b_{j,c}=1} \hat{\gamma_j}]_{x_0}$, where $\ell = \log m, 1 \leqslant j \leqslant m, 1 \leqslant c \leqslant n/m$. The computation complexity is about 130000 modular multiplications and $n/2$ modular additions in average, where the modulus x_0 has 2205 bits.

In this setting, we have implemented our PBR protocol with GMP version 5.0.2 [98], a highly optimized library for arbitrary precision arithmetic. In our implementation, we choose the private and public key pair (p, x_0) as follows.

p =17208191039508640929200576374999711030596601

29893779942568467253655521644657786345719504128

38051033305541156961429645296757557163794012509

14003744945293157980595942640451243898446460567

72770258422144622675796928192080365842783838721

17798711637935783321639095583031 51

x_0 =26995994051987058340696791927386621148530680

22878576811588518529882283935651584401459488934 0

87296688435128519763918588539827986380073047672 8

500299068286237357728208568513451075996848153990

688348444845602260155330522967623003731186027028

535988354635989225653431149930417553648090078778

840027799178806324435233078876516764950304624764

471258266695226539642564422402303542362725774410

839335882064377945737808337545320750967712766283

153865143838298357131422030563031417949370771316

488837892274387753389005582029398287713625948118

026786533890132497964987851958713620334994269130

133868714597009184901835011945026618675551476060

70523977230371667356941297787012675390603483

On an Intel(R) Core(TM)2 Duo CPU E4600 with clock speed of 2.40GHz, our encryption of one bit by $c = [M + 2r + qp]_{x_0}$ (including randomly generating r and q) takes about 0.00001 seconds, and the modular addition and multiplication of two ciphertexts take about 0.000001 seconds and 0.00006 seconds, respectively. The addition of two ciphertext without modulo x_0 takes about 0.0000001 seconds. The total time for the user to generate a query Q is about 0.00015 second, the total time for the database server to generate a response R is about 2 minute, when the database size is 2×10^9 bits (equally divided into 10000 blocks, each of which has 200kbits). Given that our PBR protocol allows parallel computation then this would mean that the time to compute a response (based on this experiment) will be reduced. For instance, if the database server runs 20 processors in parallel, it takes about 6 seconds to generate a response. In addition, the total communication overhead is about $(15 + 200000) \times 2205$ bits. Over a line speed of 100Mbits per second, the transmission time is about 4.5 second, which is negligible in comparison with the computation time.

A similar order of magnitude of work is also required by the lattice-based scheme based on what we know about the new hardness assumption. In this case the large numbers are broken into smaller numbers to do the same amount of work. What this results in is that this scheme achieves better results when executed in parallel. This is especially true when using graphical processing chips or GPUs [68].

4.6 Conclusion

In this section, we explored the computational efficiency of single-database private information retrieval schemes. We first reviewed the performance of previous private information retrieval by examining the fundamental primitives, the homomorphic operations, that composed them. We found that as the communication complexity went down, the amount of work the server had to do increased substantially. Thus, we needed schemes that are more computationally efficient because they only require simple operations. We reviewed and summarised two such schemes: one based on lattices and one based on ideas from fully homomorphic encryption. We compared the two schemes and found that they are roughly equivalent, depending on what parameters are deemed to be sufficient for security.

We conducted a simple experiment to show the performance of the simple operations that compose these schemes. The results show that the PIR schemes based on these simple operations are within practical limits. Although both schemes have higher communicational complexity, the computational complexity is significantly reduced. This is significant because computers[2] are reaching the limit of possible computing power, even after adding more cores. On the other hand, network speeds continue to rise.

Future work will include examining the difficulty of these new hardness assumptions to better be able to reduce the overall complexity of their performance.

[2] By computers, we mean computers based on Boolean circuits. Quantum computers are not considered in this dissertation.

Chapter 5

Private Location Based Queries

We now turn our attention to the application of location based queries. A general problem in this application domain is the user's desire to learn more about the surrounding businesses in close proximity to their location. In this problem, a server is assumed to hold a database of location records and responds to queries made by the user who supplies their location. This obviously presents a privacy concern as a user's location is very personal, and can be easily exploited.

In this chapter we address this privacy problem by presenting a private location based solution, which was presented in our paper entitled 'Privacy-Preserving and Content-Protecting Location Based Queries' and published in the International Conference on Data Engineering 2012 (ICDE2012) proceedings. This work was subsequently enhanced and published, under the same title, in the IEEE Transactions on Knowledge and Data Engineering (TKDE) journal.

If we were to characterise the privacy model type, as defined in Chapter 1, we would claim that it was a private client/private server solution. This is because it simultaneously provides protection for the client's query and for the server's database. This chapter will explore and justify these claims.

This chapter is organised as follows. First we provide some basic definitions and present a literature review. We then give the protocol model, which is followed by the the protocol description. We highlight a problem with this construction, under the name Repeated Key Problem, and present a corrected solution. Next we present a security and performance analysis. Then we evaluate the performance of our solution by giving the results from two experiments: one entirely on a desktop machine, and one that gives a full implementation, which includes a mobile implementation. We finish the chapter by giving a conclusion and some recommendations for future research.

5.1 Basic Definitions

A location based service (LBS) is an information, entertainment and utility service generally accessible by mobile devices such as, mobile phones, GPS devices, pocket PCs, and operates through a mobile network. A LBS can offer many services to the users based on the geographical position of their mobile device. The services provided by a LBS are typically based on a point of interest database. By retrieving the Points Of Interest (POIs) from the database server, the user can get answers to various location based queries, which include but are not limited to - discovering the nearest ATM machine, gas station, hospital, or police station. In recent years there has been a dramatic increase in the number of mobile devices querying location servers for information about POIs. Among many challenging barriers to the wide deployment of such applications, privacy assurance is a major issue. For instance, users may feel reluctant to disclose their locations to the LBS, because it may be possible for a location server to learn who is making a certain query by linking these locations with a residential phone book database, since users are likely to perform many queries from home.

The Location Server (LS), which offers some LBS, spends its resources to compile information about various interesting POIs. Hence, it is expected that the LS would not disclose any information without fees. Therefore the LBS has to ensure that LS's data is not accessed by any unauthorized user. During the process of transmission the users should not be allowed to discover any information for which they have not paid. Thus, it is crucial that solutions be devised that address the privacy of the users issuing queries, but also prevent users from accessing content to which they do not have authorization.

5.1.1 Related Work

The first solution to the problem was proposed by Beresford [7], in which the privacy of the user is maintained by constantly changing the user's name or pseudonym within some mix-zone. It can be shown that, due to the nature of the data being exchanged between the user and the server, the frequent changing of the user's name provides little protection for the user's privacy. A more recent investigation of the mix-zone approach has been applied to road networks [82]. They investigated the required number of users to satisfy the unlinkability property when there are repeated queries over an interval. This requires careful control of how many users are contained within the mix-zone, which is difficult to achieve in practice.

A complementary technique to the mix-zone approach is based on k-anonymity [50, 37, 8]. The concept of k-anonymity was introduced as a method for preserving privacy when releasing sensitive records [96]. This is achieved by generalisation and suppression algorithms to ensure that a record can not be distinguished from $(k - 1)$ other records. The solutions for LBS use a trusted anonymiser to provide anonymity for the location data, such that the location data of a user cannot be distinguished from $(k - 1)$ other users.

An enhanced trusted anonymiser approach has also been proposed, which allows the users to set their level of privacy based on the value of k [72, 67]. This means that, given the overhead of the anonymiser, a small value of k could be used to increase the efficiency. Conversely, a large value of k could be chosen to improve the privacy, if the users felt that their position data could be used maliciously. Choosing a value for k, however, seems unnatural. There have been efforts to make the process less artificial by adding the concept of feeling-based privacy [101, 66]. Instead of specifying a k, they propose that the user specifies a cloaking region that they feel will protect their privacy, and the system sets the number of cells for the region based on the popularity of the area. The popularity is computed by using historical footprint database that the server collected.

New privacy metrics have been proposed that capture the users' privacy with respect to LBSs [19]. The authors begin by analysing the shortcomings of simple k-anonymity in the context of location queries. Next, they propose privacy metrics that enables the users to specify values that better match their query privacy requirements. From these privacy metrics they also propose spatial generalisation algorithms that coincide with the user's privacy requirements.

Methods have also been proposed to confuse and distort the location data, which include path and position confusion. Path confusion was presented by Hoh and Gruteser [54]. The basic idea is to add uncertainty to the location data of the users at the points the paths of the users cross, making it hard to trace users based on raw location data that was k-anonymised. Position confusion has also been proposed as an approach to provide privacy [72, 57]. The idea is for the trusted anonymiser to group the users according to a cloaking region (CR), thus making it harder for the LS to identify an individual. A common problem with general CR techniques is that there may exist some semantic information about the geography of a location that gives away the user's location. For example, it would not make sense for a user to be on the water without some kind of boat. Also, different people may find certain places sensitive. Damiani et al. have presented a framework that consists of an obfuscation engine that takes a users profile, which contains places that the user deems sensitive, and outputs obfuscated locations based on aggregating algorithms [27].

As solutions based on the use of a central anonymiser are not practical, Hashem and Kulik presented a scheme whereby a group of trusted users construct an ad-hoc network and the task of querying the LS is delegated to a single user [53]. This idea improves on the previous work by the fact that there is no single point of failure. If a user that is querying the LS suddenly goes offline, then another candidate can be easily found. However, generating a trusted ad-hoc network in a real world scenario is not always possible.

Another method for avoiding the use of a trusted anonymiser is to use 'dummy' locations [61, 30]. The basic idea is to confuse the location of the user by sending many random other locations to the server, such that the server cannot distinguish the actual location from the fake locations. This incurs both processing and communication overheads for the user device. The user has to randomly choose a set of fake locations as well as transmitting them over a network, wasting bandwidth. We refer the interested reader to Krumm [63], for a more detailed survey in this area.

Most of the previously discussed issues are solved with the introduction of a private information retrieval (PIR) location scheme [46]. The basic idea is to employ PIR to enable the user to query the location database without compromising the privacy of the query. Generally speaking, PIR schemes allow a user to retrieve data (bit or block) from a database, without disclosing the index of the data to be retrieved to the database server [21]. Ghinita et al. used a variant of PIR which is based on the quadratic residuosity problem [64]. Basically the quadratic residuosity problem states that is computationally hard to determine whether a number is a quadratic residue of some composite modulus n ($x^2 = q \ (mod \ n)$), where the factorisation of n is unknown.

This idea was extended to provide database protection [44, 45]. This protocol consists of two stages. In the first stage, the user and server use homomorphic encryption to allow the user to privately determine whether his/her location is contained within a cell, without disclosing his/her coordinates to the server. In the second stage, PIR is used to retrieve the data contained within the appropriate cell.

The homomorphic encryption scheme used to privately compare two integers is the Paillier encryption scheme [81]. The Paillier encryption scheme is known to be additively homomorphic and multiplicatively-by-a-constant homomorphic. This means that we can add or scale numbers even when all numbers are encrypted. Both features are used to determine the sign (most significant bit) of $(b - a)$, and hence the user is able to determine the cell in which he/she is located, without disclosing his/her location.

More specifically, the client and server own two integers a and b respectively. The b owned by the server forms a boundary in their grid, and the test is used to determine on which side the client is located. Since the message space is restricted to numbers to the interval $[0, N)$, the client chooses a random N, and the client encrypts $E(N - a)$, which is sent to the server. The server then computes $c = E(b)E(N - a) = E(N + (b - a))$.

Because the client already knows a and N, it is trivial to find b. To counter this the server raises c to a random power ρ as c^{ρ}. This, when decrypted, results in $\rho \cdot (N + b - a)$. In this scenario the b must not be allowed to be changed without updating the database. Under this assumption, when given c^{ρ_1} and c^{ρ_2}, where $c = E(N + (b - a))$, then this is vulnerable to a factorisation attack. In this chapter, we design a protocol to overcome this weakness.

5.2 Protocol Model

Before describing our protocol we introduce the system model, which defines the major entities and their roles. The description of the protocol model begins with the notations and system parameters of our solution.

5.2.1 Preliminaries

Let $x \leftarrow y$ be the assignment of the value of variable y to variable x and $E \Leftarrow v$ be the transfer of the variable v to entity E. Denote the ElGamal [32] encryption of message m as $E(m) = \boldsymbol{A} = (A_1, A_2) = (g^r, g^m y^r)$, where g is a generator of group G, y is the public key of the form $y = g^x$, and r is chosen at random. Note that $\boldsymbol{A}$ is a vector, while A_1, A_2 are elements of the vector. The cyclic group G is a multiplicative subgroup of the finite field F_p, where p is a large prime number and q is a prime that divides $(p - 1)$. Let g be a generator of group G, with order q and $\langle |g| \rangle$ denote the order of generator g. We denote by $|p|$ the bit length of p, $a||b$ the concatenation of a and b, and $\oplus$ the exclusive OR operator.

We require, for security reasons, that $|p| = 1024$ and $|q| = 160$. We also require that the parameters G, g, p, q be fixed for the duration of a round of our protocol and be made publicly accessible to every entity in our protocol.

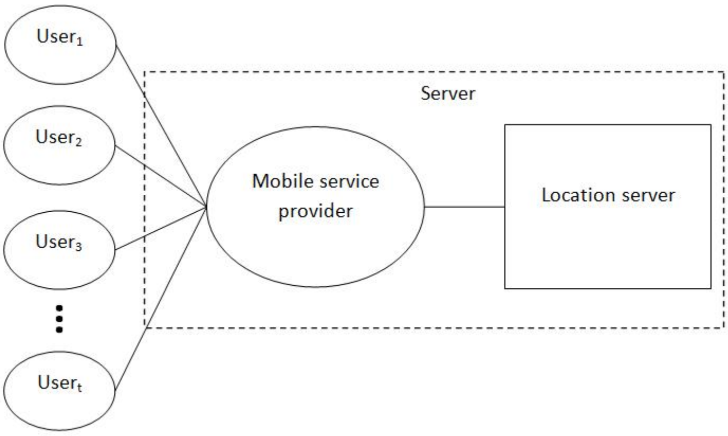


Figure 5.1: System model

5.2.2 System Model

The system model consists of three types of entities (see Figure 5.1): the set of users[1] who wish to access location data U, a mobile service provider SP, and a location server LS. From the point of view of a user, the SP and LS will compose a server, which will serve both functions. The user does not need to be concerned with the specifics of the communication.

The users in our model use some location-based service provided by the location server LS. For example, what is the nearest ATM or restaurant? The purpose of the mobile service provider SP is to establish and maintain the communication between the location server and the user. The location server LS owns a set of POI records r_i for $1 \leqslant r_i \leqslant \rho$. Each record describes a POI, giving GPS coordinates to its location (x_{gps}, y_{gps}), and a description or name about what is at the location.

We reasonably assume that the mobile service provider SP is a passive entity and is not allowed to collude with the LS. We make this assumption because the SP can determine the whereabouts of a mobile device, which, if allowed to collude with the LS, completely subverts any method for privacy. There is simply no technological method for preventing this attack. As a consequence of this assumption, the user is

[1] In this context we use the term "user" to refer to the entity issuing queries and retrieving query results. In most cases, such user is a client software executing on behalf of a human user.

able to either use GPS (Global Positioning System) or the mobile service provider to acquire his/her coordinates.

Since we are assuming that the mobile service provider SP is trusted to maintain the connection, we consider only two possible adversaries. One for each communication direction. We consider the case in which the user is the adversary and tries to obtain more than he/she is allowed. Next we consider the case in which the location server LS is the adversary, and tries to uniquely associate a user with a grid coordinate.

5.2.3 Security Model

Before we define the security of our protocol, we introduce the concept of k out of N adaptive oblivious transfer as follows.

Definition 10 ($\boldsymbol{k}$ out of $\boldsymbol{N}$ adaptive oblivious transfer ($\boldsymbol{OT^N_{k\times 1}}$) [75]). *$OT^N_{k\times 1}$ protocols contain two phases, for initialization and for transfer. The initialization phase is run by the sender (Bob) who owns the N data elements $X_1, X_2, ..., X_N$. Bob typically computes a commitment to each of the N data elements, with a total overhead of $O(N)$. He then sends the commitments to the receiver (Alice). The transfer phase is used to transmit a single data element to Alice. At the beginning of each transfer Alice has an input I, and her output at the end of the phase should be data element X_I. An $OT^N_{k\times 1}$ protocol supports up to k successive transfer phases.*

Built on the above definition, our protocol is composed of initialisation phase and transfer phase. We will now outline the steps required for the phases and then we will formally define the security of these phases.

Our initialisation phase is run by the sender (server), who owns a database of location data records and a 2-dimensional key matrix $\mathcal{K}_{m\times n}$, where m and n are rows and columns respectfully. An element in the key matrix is referenced as $k_{i,j}$. Each $k_{i,j}$ in the key matrix uniquely encrypts one record. A set of prime powers $\mathcal{S} = \{p_1^{c_1}, ..., p_N^{c_N}\}$, where N is the number of blocks, is available to the public. Each element in $\mathcal{S}$ the p_i is a prime and c_i is a small natural number such that $p_i^{c_i}$ is greater than the block size (where each block contains a number of POI records).

We require, for convenience that the elements of $\mathcal{S}$ follow a predictable pattern. In addition, the server sets up a common security parameter k for the system.

Our transfer phase is constructed using six algorithms: QG1, RG1, RR1, QG2, RG2, RR2. The first three compose the first phase (Oblivious Transfer Phase), while the last three compose the second phase (Private Information Retrieval Phase). The following six algorithms are executed sequentially and are formally described as follows.

Oblivious Transfer Phase

1. *QueryGeneration*$_1$ (*Client*) (QG1):
   Takes as input indices i, j, and the dimensions of the key matrix m, n, and outputs a query $\mathcal{Q}_1$ and secret s_1, denoted as $(\mathcal{Q}_1, s_1) = QG_1(i, j, m, n)$.

2. *ResponseGeneration*$_1$(*Server*) (RG1):
   Takes as input the key matrix $\mathcal{K}_{m\times n}$, and the query $\mathcal{Q}_1$, and outputs a response $\mathcal{R}_1$, denoted as $(\mathcal{R}_1) = RG_1(\mathcal{K}_{m\times n}, \mathcal{Q}_1)$.

3. *ResponseRetrieval*$_1$ (*Client*) (RR1):
   Takes as input indices i, j, the dimensions of the key matrix m, n, the query $\mathcal{Q}_1$ and the secret s_1, and the response $\mathcal{R}_1$, and outputs a cell-key $k_{i,j}$ and cell-id $ID_{i,j}$, denoted as $(k_{i,j}, ID_{i,j}) = RR_1(i, j, m, n, (Q_1, s_1), \mathcal{R}_1)$.

Private Information Retrieval Phase

4. *QueryGeneration*$_2$ (*Client*) (QG2):
   Takes as input the cell-id $ID_{i,j}$, and the set of prime powers $\mathcal{S}$, and outputs a query $\mathcal{Q}_2$ and secret s_2, denoted as $(\mathcal{Q}_2, s_2) = QG_2(ID_{i,j}, \mathcal{S})$.

5. *ResponseGeneration*$_2$ (*Server*) (RG2):
   Takes as input the database D, the query $\mathcal{Q}_2$, and the set of prime powers $\mathcal{S}$, and outputs a response $\mathcal{R}_2$, denoted as $(\mathcal{R}_2) = RG_2(D, \mathcal{Q}_2, \mathcal{S})$.

6. *ResponseRetrieval*$_2$ (*Client*) (RR2):
   Takes as input the cell-key $k_{i,j}$ and cell-id $ID_{i,j}$, the query $\mathcal{Q}_2$ and secret s_2,

the response $\mathcal{R}_2$, and outputs the data d, denoted as
$(d) = RR_2(k_{i,j}, ID_{i,j}, (\mathcal{Q}_2, s_2), \mathcal{R}_2)$.

Our transfer phase can be repeatedly used to retrieve points of interest from the location database. With these functions described, we can build security definitions for both the client and server [42, 75].

Definition 11 (**Client's Security (Indistinguishability)** [**75**]). *In a $OT^N_{k\times 1}$ protocol, for any step $1 \leqslant t \leqslant k$, for any previous items $I_1, ..., I_{t-1}$ that the receiver has obtained in the first t-1 transfers, for any $1 \leqslant I_t, I'_t \leqslant N$ and for any probabilistic polynomial time $\mathcal{B}'$ executing the server's part, the views that $\mathcal{B}'$ sees in case the client tries to obtain X_{I_t} and in the case the client tries to obtain $X_{I'_t}$ are computationally indistinguishable given $X_1, X_2, ..., X_N$.*

Definition 12 (**Server's Security (Comparison with Ideal Model)** [**75**]). *We compare a $OT^N_{k\times 1}$ protocol to the ideal implementation, using a trusted third party that gets the server's input $X_1, X_2, ..., X_N$ and the client's requests and gives the client the data elements he/she has requested. For every probabilistic polynomial-time machine $\mathcal{A}'$ substituting the client, there exists a probabilistic polynomial-time machine $\mathcal{A}''$ that plays the receiver's role in the ideal model such that the outputs of $\mathcal{A}'$ and $\mathcal{A}''$ are computationally indistinguishable.*

5.3 Protocol Description

We now describe our protocol. We first give a protocol summary to contextualise the proposed solution and then we will describe the solution's protocol in more detail.

5.3.1 Protocol Summary

The ultimate goal of our protocol is to obtain a set (block) of POI records from the LS, which are close to the user's position, without compromising the privacy of the user or the server. We achieve this by applying a two stage approach, which is shown in Figure 5.2. The first stage is based on a two-dimensional oblivious transfer

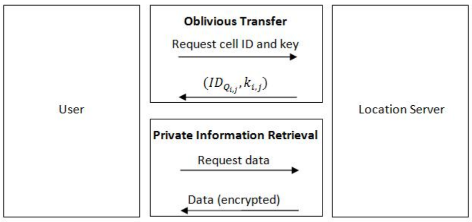


Figure 5.2: High level overview of the protocol

[75] and the second stage is based on a communicationally[2] efficient PIR [42]. The oblivious transfer based protocol is used by the user to obtain the cell ID, where the user is located, and the corresponding symmetric key. The knowledge of the cell ID and the symmetric key is then used in the PIR based protocol to obtain and decrypt the location data.

The user determines his/her location within a publicly generated grid P by using his/her GPS coordinates and forms an oblivious transfer query[3]. The minimum dimensions of the public grid are defined by the server and are made available to all users of the system. This public grid superimposes over the privately partitioned grid generated by the location server's POI records, such that there is at least one $P_{i,j}$ cell within the server's partition $Q_{i,j}$. This is illustrated in Figure 5.3.

Since PIR does not require that a user is constrained to obtain only one bit/block, the location server needs to implement some protection for its records. This is achieved by encrypting each record in the POI database with a key using a symmetric key algorithm, where the key for encryption is the same key used for decryption. This key is augmented with the cell info data retrieved by the oblivious transfer query. Hence, even if the user uses PIR to obtain more than one record, the data will be meaningless resulting in improved security for the server's database. Before we

[2] We use a communicationally efficient PIR because we assume that transferring data on a mobile network is costly and we assume that the server is powerful. Mobile networks are likely to become less expensive in the future. In which case, a computationally efficient scheme would be more appropriate.

[3] An oblivious transfer query is where a server cannot learn the user's query, while the user cannot gain more than they are entitled. This is similar to PIR, but oblivious transfer requires protection for the user and server. PIR only requires that the user is protected.

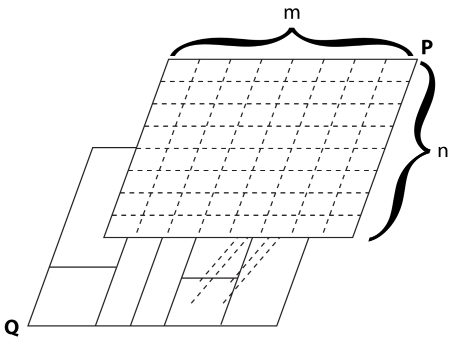


Figure 5.3: The public grid superimposed over the private grid

describe the protocol in detail, we describe some initialisation performed by both parties.

5.3.2 Global Initialisation

A user u from the set of users U initiates the protocol process by deciding a suitable square cloaking region CR, which contains his/her location. All user queries will be with respect to this cloaking region. The user also decides on the accuracy of this cloaking region by how many cells are contained within it, which is at least the minimum size defined by the server. This information is combined to form the public grid P and submitted to the location server, which partitions its records or superimposes it over pre-partitioned records (see Figure 5.3). This partition is denoted Q (note that the cells don't necessarily need to be the same size as the cells of P). Each cell in the partition Q must have the same number r_{max} of POI records. Any variation in this number could lead to the server identifying the user. If this constraint cannot be satisfied, then dummy records can be used to make sure each cell has the same amount of data. We assume that the LS does not populate the private grid with misleading or incorrect data, since such action would result in the loss of business under a payment model.

Next, the server encrypts each record r_i within each cell of Q, $Q_{i,j}$, with an associated symmetric key $k_{i,j}$. The encryption keys are stored in a small (virtual) database table that associates each cell in the public grid P, $P_{i,j}$, with both a cell in

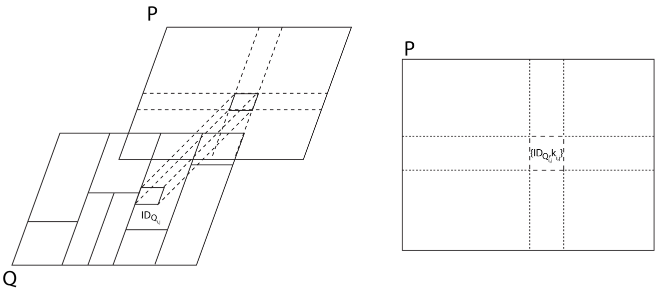


Figure 5.4: Association between the public and private grids

the private grid $Q_{i,j}$ and corresponding symmetric key $k_{i,j}$. This is shown by Figure 5.4.

The server then processes the encrypted records within each cell $Q_{i,j}$ such that the user can use an efficient PIR [42], to query the records. Using the private partition Q, the server represents each associated (encrypted) data as an integer C_i, with respect to the cloaking region. For each C_i, the server chooses a set of unique prime powers $\pi_i = p_i^{c_i}$, such that $C_i < \pi_i$. We note that the c_i in the exponent must be small for the protocol to work efficiently. We also stipulate that the unique prime powers π_i follow a predictable pattern. Finally, the server uses the Chinese Remainder Theorem to find the smallest integer e such that $e = C_i \ (mod\ \pi_i)$ for all C_i. The integer e effectively represents the database. Once the initialisation is complete, the user can proceed to query the location server for POI records.

5.3.3 Oblivious Transfer Based Protocol

The purpose of this protocol is for the user to obtain one and only one record from the cell in the public grid P, shown in Figure 5.4. We achieve this by constructing a 2-dimensional oblivious transfer, based on the ElGamal oblivious transfer [5, 76], using adaptive oblivious transfer proposed by Naor et al. [75].

The public grid P, known by both parties, has m columns and n rows. Each cell in P contains a symmetric key $k_{i,j}$ and a cell id in grid Q i.e., $(ID_{Q_{i,j}}, k_{i,j})$, which can be represented by a stream of bits $X_{i,j}$. The user determines his/her i, j coordinates in the public grid which is used to acquire the data from the cell within the grid. The protocol is initialised by the server by generating $m \times n$ keys of the form $g^{R_i}||g^{C_i}$. This initialisation is presented in Algorithm 19.

Algorithm 19 *Initialisation*

```
Input: $X_{1,1}, ..., X_{m,n}$, where $X_{i,j} = ID_{Q_{i,j}}||k_{i,j}$
Output: $Y_{1,1}, ..., Y_{m,n}$
$K_{i,j} \leftarrow K_{i,j} = g^{R_i}||g^{C_j}$, for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, where $R_i$ and $C_j$ are randomly chosen
$Y_{i,j} \leftarrow X_{i,j} \oplus H(K_{i,j})$, for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, where $H$ is a fast secure hash function
return $Y_{1,1}, ..., Y_{m,n}$ {Encryptions of $X_{1,1}, ..., X_{m,n}$ using $K_{i,j}$}
```

Algorithm 19 is executed once and the output $Y_{1,1}, ..., Y_{m,n}$ is sent to the user. At which point, the user can query this information using the indices i, and j, as input. This protocol is presented in Algorithm 20.

Algorithm 20 *Transfer*

```
Input: User:$i, j$
Output: User:$(ID_{Q_{i,j}}, k_{i,j})$
User
$y \leftarrow g^x$, where $y$ is the public key of the user and $x$ is chosen at random
$\boldsymbol{C_1} \leftarrow (A_1, B_1) = (g^{r_1}, g^{-i}y^{r_1})$
$\boldsymbol{C_2} \leftarrow (A_2, B_2) = (g^{r_2}, g^{-j}y^{r_2})$
$Server \Leftarrow \boldsymbol{C_1}, \boldsymbol{C_2}$
Server
$\boldsymbol{C'_{1,\alpha}} \leftarrow (A_1^{r'_\alpha}, g^{R_\alpha}(g^\alpha B_1)^{r'_\alpha})$ for $1 \leqslant \alpha \leqslant n$
$\boldsymbol{C'_{2,\beta}} \leftarrow (A_2^{r'_\beta}, g^{C_\beta}(g^\beta B_2)^{r'_\beta})$ for $1 \leqslant \beta \leqslant m$
$User \Leftarrow \boldsymbol{C'_{1,1}}, ..., \boldsymbol{C'_{1,n}}, \boldsymbol{C'_{2,1}}, ..., \boldsymbol{C'_{2,m}}$
User
Let $(U_{1,i}, V_{1,i}) = \boldsymbol{C'_{1,i}}$ and $(U_{2,j}, V_{2,j}) = \boldsymbol{C'_{1,j}}$
$W_1 \leftarrow V_{1,i}/(U_{1,i})^x$
$W_2 \leftarrow V_{2,j}/(U_{2,j})^x$
$K'_{i,j} \leftarrow W_1||W_2$
$X'_{i,j} \leftarrow Y_{i,j} \oplus H(K'_{i,j})$
Reconstruct $(ID_{Q_{i,j}}, k_{i,j})$ from $X'_{i,j}$
return $(ID_{Q_{i,j}}, k_{i,j})$ {Cell id of grid $Q$, with associated cell key}
```

At the conclusion of the protocol presented by Algorithm 20, the user has the information to query the location server for the associated block.

Theorem 5. *Assume that the user and server follow Algorithms 19 and 20 correctly, then* $X'_{i,j} = Y_{i,j} \oplus H(K_{i,j})$.

*P*roof: We begin this proof by showing that $K_{i,j} = K'_{i,j}$. In the initialisation algorithm (Algorithm 19) $K_{i,j}$ is calculated as $K_{i,j} = g^{R_i}||g^{C_j}$. At the end of the transfer protocol, the user computes $K'_{i,j}$ as $W_1||W_2$. We now need to prove that W_1 and W_2 equal g^{R_i} and g^{C_j} respectively. W_1 is computed as $V_{1,i}/(U_{1,i})^x$, where $U_{1,i} = A_1^{r'_\alpha} = (g^{r_1})^{r'_\alpha} = g^{r_1 r'_\alpha}$ and $V_{1,i} = g^{R_\alpha}(g^\alpha B_1)^{r'_\alpha} = g^{R_\alpha}(g^\alpha g^{-i} y^{r_1})^{r'_\alpha}$, for $1 \leqslant i \leqslant n$. When $\alpha = i$ then $V_{1,i} = g^{R_i}(y^{r_1})^{r'_i} = g^{R_i} y^{r_1 r'_i}$. Raising $U_{1,i}$ to the power x gives $(U_{1,i})^x = (g^{r_1 r'_i})^x = g^{x r_1 r'_i} = y^{r_1 r'_i}$. Therefore, $W_1 = V_{1,i}/(U_{1,i})^x = g^{R_i}$. By similar means we can prove that $W_2 = V_{2,j}/(U_{2,j})^x = g^{C_j}$. Since $W_1||W_2 = g^{R_i}||g^{C_j}$, then $K_{i,j} = K'_{i,j}$. Since $\oplus$ is self inverse and given that $Y_{i,j} = X_{i,j} \oplus H(K_{i,j})$, it follows that $X_{i,j} = Y_{i,j} \oplus H(K_{i,j})$. Using knowledge of $K_{i,j}$, the user can compute $X_{i,j}$ as desired. This completes the proof. □

5.3.4 Private Information Retrieval Based Protocol

With the knowledge about which cells are contained in the private grid, and the knowledge of the key that encrypts the data in the cell, the user can initiate a private information retrieval protocol with the location server to acquire the encrypted POI data. Assuming the server has initialised the integer e, the user u_i and LS can engage in the following private information retrieval protocol using the $ID_{Q_{i,j}}$, obtained from the execution of the previous protocol, as input. The $ID_{Q_{i,j}}$ allows the user to choose the associated prime number power π_i, which in turn allows the user to query the server. The protocol is presented in Algorithm 21.

Theorem 6. *Assume that the user and the server follow the protocol correctly, then the user successfully acquires* C_i *for his/her chosen prime index.*

*P*roof: It is easy to see that $C_i = e\ (mod\ \pi_i)$ and $h_e = g_e^{|\langle g \rangle|/\pi_i}$. Then C_i is the discrete logarithm of h_e to the base h, since $g_e^{|\langle g \rangle|/\pi_i} = g^{e|\langle g \rangle|/\pi_i} = g^{e_{\pi_i}|\langle g \rangle|/\pi_i} = h^{e_{\pi_i}}$, where e_{π_i} stands for $e\ (mod\ \pi_i)$. This completes the proof. □

Algorithm 21 *PIRProtocol*

```
Input: User:$ID_{Q_{i,j}}$
Output: User:$C_i$
User
    $\pi_0 \leftarrow \pi_i$, where $\pi_i$ is chosen based on the value of $ID_{Q_{i,j}}$
    Generate random group $G$ and group element $g$, such that $\pi_0$ divides the order of $g$
    $q \leftarrow |\langle g \rangle|/\pi_0$
    $h \leftarrow g^q$
    $Server \Leftarrow G, g$
Server
    $g_e \leftarrow g^e$
    $User \Leftarrow g_e$
User
    $h_e \leftarrow g_e^q$
    $C_i \leftarrow log_h h_e$, where $log_h$ is the discrete log base $h$
return $C_i$ {The requested (encrypted) data}
```

At the conclusion of the protocol, the user has successfully acquired the block that contain the encrypted POI records. With the knowledge of the cell key $k_{i,j}$, the user can decrypt C_i and obtain the requested data, thus concluding one round of the protocol. Using the same set-up, the user can execute several more rounds very efficiently and effectively without compromising his/her privacy. Similarly, the server's data remains protected based on the fact the user can only acquire one key per round.

5.4 Repeated Key Problem

Although the user can only obtain one key per round, the client can obtain more information. This is because the user has direct access to both the row key g^{R_i} and the column key g^{C_j}, which are subsequently concatenated and hashed. Now this is secure for one round. But if the user runs the protocol again, and obtains $g^{R_{i'}}$ and $g^{C_{j'}}$, then the user can compute four keys as $H(g^{R_i}||g^{C_j})$, $H(g^{R_i}||g^{C_{j'}})$, $H(g^{R_{i'}}||g^{C_j})$, and $H(g^{R_{i'}}||g^{C_{j'}})$. This is the same issue that was described by Naor and Pinkas [75]. Their discussion was in terms of a generic sum consistent synthesiser.

A sum consistent synthesiser S is a function with ℓ inputs is sum consistent if $S(x_1, x_2, ...m_\ell) = S(x_1, x_2, ..., x_\ell)$ when $\sum_1^\ell x_i = \sum_1^\ell y_i$. In other words, when the summation of the inputs is equal then so must be the output of S. For security, we also require this function to be pseudo random. That is, the output of the function is unpredictable without the inputs. In the original work that presented adaptive oblivious transfer, they modified the generic definition of a sum consistent synthesizer to remove the linearity in the result. Therefore it was impossible to infer new keys based on the keys already acquired.

To overcome this problem in our case, we must prevent the user from directly accessing the row and column keys. To achieve this we enclose the product $g^R g^C$ inside an outer exponentation, resulting in a key of the form $g_0^{g_1^{R_i} g_2^{C_j}}$, where g_0 generates the outer group generated by g_1 and g_2. With this in mind, we modify Algorithms 20 and 19 to use this key structure. Algorithm 22 presents the modified initialisation procedure performed by the server. The stream of bits $X_{i,j}$ is as before, representing $(ID_{Q_{i,j}}, k_{i,j})$.

Algorithm 22 *Initialisation*

```
Input: $X_{1,1}, ..., X_{m,n}$, where $X_{i,j} = ID_{Q_{i,j}}||k_{i,j}$
Output: $Y_{1,1}, ..., Y_{m,n}$
$K_{i,j} \leftarrow K_{i,j} = g_0^{g_1^{R_i} g_2^{C_j}}$, for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, where $R_i$ and $C_j$ are randomly chosen
$Y_{i,j} \leftarrow X_{i,j} \oplus H(K_{i,j})$, for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, where $H$ is a fast secure hash function
return $Y_{1,1}, ..., Y_{m,n}$ {Encryptions of $X_{1,1}, ..., X_{m,n}$ using $K_{i,j}$}
```

When Algorithm 22 concludes, the user can proceed to query the server using the new key structure. This is presented by Algorithm 23.

At the conclusion of the protocol presented by Algorithm 20, the user has the information to query the location server for the associated block.

Theorem 7. (Correctness) Assume that the user and server follow Algorithms 22 and 23 correctly. Let $X_{i,j}$ be the bit string encoding the pair $(ID_{Q_{i,j}}, k_{i,j})$ and

Algorithm 23 $Transfer$

```
Input: User: $i, j$
Output: User: $(ID_{Q_{i,j}}, k_{i,j})$
 1: User (QG1)
 2: $y_1 \leftarrow g_1^{x_1}$, where $y_1$ is the public key for the row and $x_1$ is chosen at random
 3: $y_2 \leftarrow g_2^{x_2}$, where $y_2$ is the public key for the column and $x_2$ is chosen at random
 4: $\boldsymbol{C_1} \leftarrow (A_1, B_1) = (g_1^{r_1}, g_1^{-i} y_1^{r_1})$
 5: $\boldsymbol{C_2} \leftarrow (A_2, B_2) = (g_2^{r_2}, g_2^{-j} y_2^{r_2})$
 6: $Server \Leftarrow \boldsymbol{C_1}, \boldsymbol{C_2}$
 7: Server (RG1)
 8: $\boldsymbol{C'_{1,\alpha}} \leftarrow (A_1^{r'_\alpha}, g_1^{R_\alpha} r_R (g_1^\alpha B_1)^{r'_\alpha})$ for $1 \leqslant \alpha \leqslant n$ and $r_R = g_1^s$, where $s$ is chosen randomly
 9: $\boldsymbol{C'_{2,\beta}} \leftarrow (A_2^{r'_\beta}, g_2^{C_\beta} r_C (g_2^\beta B_2)^{r'_\beta})$ for $1 \leqslant \beta \leqslant m$ and $r_C = g_2^t$, where $t$ is chosen randomly
10: $\gamma \leftarrow g_0^{1/r_R r_C}$
11: $User \Leftarrow \boldsymbol{C'_{1,1}}, ..., \boldsymbol{C'_{1,n}}, \boldsymbol{C'_{2,1}}, ..., \boldsymbol{C'_{2,m}}, \gamma$
12: User (RR1)
13: Let $(U_{1,i}, V_{1,i}) = \boldsymbol{C'_{1,i}}$ and $(U_{2,j}, V_{2,j}) = \boldsymbol{C'_{1,j}}$
14: $W_1 \leftarrow U_{1,i}^{-x_1}$
15: $W_2 \leftarrow U_{2,j}^{-x_2}$
16: $W_3 \leftarrow V_{1,i} W_1$
17: $W_4 \leftarrow V_{2,j} W_2$
18: $K'_{i,j} \leftarrow \gamma^{W_3 W_4}$
19: $X'_{i,j} \leftarrow Y_{i,j} \oplus H(K'_{i,j})$
20: Reconstruct $(ID_{Q_{i,j}}, k_{i,j})$ from $X'_{i,j}$
21: return $(ID_{Q_{i,j}}, k_{i,j})$ {Cell id of grid $Q$, with associated cell key}
```

let $X'_{i,j}$ the bit string generated by Algorithm 23 (Step 19) as $X'_{i,j} = Y_{i,j} \oplus H(K_{i,j})$. Then $X'_{i,j} = X_{i,j}$.

*P*roof: We begin this proof by showing that $K_{i,j} = K'_{i,j}$, where $K'_{i,j}$ is the key obtained by the user according to the Algorithm 23 (step 18). In the initialisation algorithm (Algorithm 22) $K_{i,j}$ is calculated as $K_{i,j} = g_0^{g_1^{R_i} g_2^{C_j}}$. At the end of the transfer protocol, the user computes $K'_{i,j}$ as $\gamma^{W_3 W_4}$, where W_3 can be simplified as follows when $i = \alpha$.

$$\begin{aligned}
W_3 &= V_{1,i}W_1 \\
&= g_1^{R_i}r_R(g_1^{\alpha}g_1^{-i}y_1^{r_1})^{r'_1}U_{1,i}^{-x_1} \\
&= g_1^{R_i}r_R(y_1^{r_1r'_1})U_{1,i}^{-x_1} \\
&= g_1^{R_i}r_R(g_1^{xr_1r'_1})(g_1^{r_1r'_1})^{-x_1} \\
&= g_1^{R_i}r_R(g_1^{xr_1r'_1})(g_1^{-(xr_1r'_1)}) \\
&= g_1^{R_i}r_R\ (mod\ q)
\end{aligned}$$

By similar means we can show that $W_4 = g_2^{C_j}r_C$, when $j = \beta$. So we have the following.

$$\begin{aligned}
\gamma^{W_3W_4} &= (g_0^{1/r_Rr_C})^{g_1^{R_i}r_Rg_2^{C_j}r_C} \\
&= g_0^{g_1^{R_i}g_2^{C_j}}\ (mod\ p)
\end{aligned}$$

This proves $K_{i,j} = K'_{i,j}$. Since $\oplus$ is self inverse and given that $Y_{i,j} = X_{i,j} \oplus H(K_{i,j})$, it follows that $X_{i,j} = Y_{i,j} \oplus H(K_{i,j})$. Using knowledge of $K'_{i,j}$, the user can compute $X_{i,j}$, which is the same as $X'_{i,j}$ as desired. This completes the proof. □

5.5 Security Analysis

In this section, we analyse the security of the client and the server. While the client does not want to give up the privacy of his/her location, the server does not want to disclose other records to the client. This disclosure would not make much business sense in a variety of applications. Our analysis will be with respect to the security definitions in Section 5.2.3. This analysis is with reference to our revised protocol that includes the repeated key solution.

5.5.1 Client's Security

Fundamentally, the information that is most valuable to the user is his/her location. This location is mapped to a cell $P_{i,j}$. In both phases of our protocol, the oblivious transfer based protocol and the private information retrieval based protocol, the server must not be able to distinguish two queries of the client from each other. We will now describe both cases separately.

In the oblivious transfer phase, each coordinate of the location is encrypted by the ElGamal encryption scheme, e.g., $(g_1^{r_1}, g_1^{-i} y_1^{r_1})$. It has been shown that ElGamal encryption scheme is semantically secure [32]. This means that given the encryption of one of two plaintexts m_1 and m_2 chosen by a challenger, the challenger cannot determine which plaintext is encrypted, with probability significantly greater than 1/2 (the success rate of random guessing). In view of it, the server cannot distinguish any two queries of the client from each other in this phase.

In the private information retrieval phase, the security of the client is built on the Gentry-Ramzan private information retrieval protocol, which is based on the phi-hiding (ϕ-hiding) assumption [42].

On the basis of the above security analysis, we can conclude with the following theorem.

Theorem 8. *Assume that the ElGamal encryption scheme is semantically secure and the Gentry-Ramzan PIR has client security, our protocol has client security, i.e., the server cannot distinguish any two queries of the client from each other.*

5.5.2 Server's Security

Intuitively, the server's security requires that the client can only retrieve one record in each query to the server, and the server must not disclose other records to the client in the response. Our protocol achieves the server's security in the oblivious transfer phase, which is built on the Naor-Pinkas oblivious transfer protocol [75].

Our Algorithm 1 is the same as the Naor-Pinkas oblivious transfer protocol except from the one-out-of-n oblivious transfer protocol, which is built on the ElGamal encryption scheme. In the generation of the first response (RG_1), the server computes $C'_{1,\alpha} = (A_1^{r'_\alpha}, g_1^{R_\alpha} r_R (g_1^\alpha B_1)^{r'_\alpha})$ for $1 \leqslant \alpha \leqslant n$, where $B_1 = g_1^{-i} y_1^{r_1}$, and sends $C'_{1,\alpha}$ ($1 \leqslant \alpha \leqslant n$) to the client. Only when $\alpha = i$, $C'_{1,i} = (g_1^{r_i r'_i}, g_1^{R_i} r_R y_1^{r_i r'_i})$ is the encryption of $g_1^{R_i} r_R$. When $\alpha \neq i$, $C'_{1,\alpha}$ is the encryption of $g_1^{R_\alpha} r_R g_1^{r'_\alpha}$, where r'_α is unknown to the client. Because the discrete logarithm is hard, the client cannot determine r'_α from $A_1^{r'_\alpha}$. Therefore, $g_1^{R_\alpha} r_R$ is blinded by the random factor $g_1^{r'_\alpha}$. In view of it, the client can retrieve the useful $g_1^{R_i} r_R$ only from $C'_{1,\alpha}$ $(1 \leqslant \alpha \leqslant n)$. Then following the Naor-Pinkas oblivious transfer protocol, the client can retrieve the encryption key k_{ij} only in the end of the phase.

In the private information retrieval phase, even if the client can retrieve more than one encrypted records, he/she can only decrypt one record with the encryption key k_{ij} retrieved in the first phase.

Based on the above analysis, we obtain the following result.

Theorem 9. *Assume that the discrete logarithm is hard and the Naor-Pinkas protocol is a secure oblivious transfer protocol, our protocol has server security.*

5.6 Performance Analysis

We now analyse the performance of our solution and show that it is very practical. The performance analysis consists of the computation analysis and the communication analysis. We supplement this analysis with a comparison with the protocol by Ghinita et al. [46, 45].

5.6.1 Computation

Since the most expensive operation in our protocol is the modular exponentiation, we focus on minimising the number of times it is required. We assume that some components can be precomputed, and hence we only consider the computations

		Our Solution	Ghinita et al.
Computation	User	6	$4+4(n \times m)$
	Server	$3n+3m$	$4(n \times m)$
	Total	$6+3n+3m$	$4+4(n \times m)+4(n \times m)$
Communication		$4L+2(m+n)L$	$4L+4(m \times n)2L$

Table 5.1: Stage 1 performance analysis summary

		Our Solution	Ghinita et al.
Computation	User	$O(c(\lg p^c + \sqrt{p})) + 2\lvert N\rvert$	$2(\sqrt{a \times b}) \times \frac{\lvert N\rvert}{2}$
	Server	$\lvert e\rvert$	$a \times b$
	Total	$O(c(\lg p^c + \sqrt{p})) + 2\lvert N\rvert + \lvert e\rvert$	$2(\sqrt{a \times b}) \times \frac{\lvert N\rvert}{2} + a \times b$
Communication		$2L$	$\sqrt{a \times b}L$

Table 5.2: Stage 2 performance analysis summary

needed at runtime. Furthermore, we reduce the number of exponentiations required by the PIR protocol to the number of multiplications that are required. This will make the computational comparison between our solution and the solution of Ghinita et al. easier to describe.

The transfer protocol is initiated by the user, who chooses indices i and j. According to our protocol the user needs to compute $(A_1, B_1) = (g^{r_1}, g^{-i}y^{r_1})$ and $(A_2, B_2) = (g^{r_2}, g^{-j}y^{r_2})$. Since the user knows the discrete logarithm of y (i.e. x), the user can compute (A_1, B_1) and (A_2, B_2) as $(A_1, B_1) = (g^{r_1}, g^{-i+xr_1})$ and $(A_2, B_2) = (g^{r_2}, g^{-j+xr_2})$ respectively. Hence, the user has to compute 4 exponentiations to generate his/her query.

Upon receiving the user's query, the server needs to compute $((A_1)^{r'_\alpha}, g^{R_\alpha}(g^\alpha(A_2))^{r'_\alpha})$ for $1 \leqslant \alpha \leqslant n$ and $((B_1)^{r'_\beta}, g^{C_\beta}(g^\beta(B_2))^{r'_\beta})$ for $1 \leqslant \beta \leqslant m$. Since g^α and g^β can be precomputed, the server has to compute $3n + 3m$ exponentiations.

The user requires an additional 2 more exponentiations to compute $(U_{1,i})^x$ and $(U_{2,j})^x$ to determine $K_{i,j}$. After the user has determined $K_{i,j}$, he/she can determine $X_{i,j}$ and proceed with the PIR protocol. This protocol requires 3 more exponentiations, 2 performed by the user and 1 performed by the server. In terms of multiplications, the user has to perform $2|N|$ operations and the server has to perform $|e|$ operations. The user also has to compute the discrete logarithm base h, log_h, of h_e. This process can be expedited by using the Pohlig-Hellman discrete logarithm algorithm [83]. The running time of the Pohlig-Hellman algorithm is proportional to the factorisation of the group order $O(\sum_{i=1}^{r} c_i(\lg n + \sqrt{p_i}))$, where r is the number of unique prime factors and n is the order of the group. In our case, the order of the group is $\pi_i = p_i^{c_i}$ and the number of unique factors is $r = 1$, resulting in running time $O(c(\lg p^c + \sqrt{p}))$.

When we compare our approach with the one by Ghinita et al. we find that our approach is computationally more efficient. Their protocol uses the homomorphic properties of the Paillier encryption scheme [81] in order to test whether a user is located in a cell or not. This requires the user to perform 4 exponentiations to compute the ciphertext of his/her coordinates, x and y. The server then has to compute $(4 \times (n \times m))$. The user has to decrypt at most all these ciphertexts $(4 \times (n \times m))$.

Once the user has determined his/her cell index he/she can proceed with the PIR protocol (described in [46]) to retrieve the data. The PIR is based on the Quadratic Residuosity Problem [64], which allows the user to privately query the database. Let t be the total number of bits in the database, where there are a rows and b columns. The user and server have to compute $2(\sqrt{a \times b}) \times \frac{|N|}{2}$ and $a \times b$ multiplications respectively. We remark that multiplying the whole database by a string of numbers, which is required by the PIR protocol based on the quadratic residuosity problem, is equivalent to computing g^e in our PIR protocol.

5.6.2 Communication

Since we require the discrete logarithm to be intractable for security reasons, we set the modulus p to be 1024 bits in size. Hence, one ElGamal encryption is 2048 bits. Let L be the length of an element in the ElGamal ciphertext, 1024. In our proposed solution, the user needs $4L$ communication, while the server requires $2(m + n)L$

communication in the oblivious transfer protocol. In the PIR protocol, the user and server exchange one group element each.

Since the solution by Ghinita et al. uses the Paillier encryption scheme, which has equivalent ciphertext size as ElGamal ciphertext, then the size of one ciphertext in their scheme is $2L$. Based on this parameter, the user has to submit $4L$ bits to the server as their encrypted location. Then the server has to send $4 \times n \times m \times 2L$, for the user to determine his/her location. For the PIR based on the QRA, the user and server have to send $\sqrt{a \times b} \times L$. The performance analysis for stage 1 (user location test) and stage 2 (private information retrieval) are summarised in Tables 5.1 and 5.2 respectively, where the computation in Table 5.1 is in terms of exponentiation and the computation in Table 5.2 is in terms of multiplication.

When we analyse the difference in performance between our solution and the one by Ghinita et al., we find that our solution is more efficient. The performance of the first stage of each protocol is about the same, except our solution requires $O(m + n)$ operations while the solution by Ghinita et al. requires $O(m \times n)$. In the second stage, our protocol is far more communicationally efficient, requiring the transmission of only 2 group elements whereas the Ghinita et al. solution requires the exchange of an $a \times b$ matrix.

5.7 Experimental Evaluation

We now experimentally evaluate our location-based protocol. We first present the results from a desktop implementation that was included in the ICDE2012 publication. The purpose of this experiment is to test the feasibility of our simple solution (original key structure) on a desktop machine. Next, we present an updated experiment found in TKDE, which includes the new key structure implementation. The updated experiment is conducted using both a desktop machine and a mobile device to test the real-world feasibility of our solution.

5.7.1 Desktop-only Implementation

We implemented a prototype of our simple location based query solution using the C++ programming language. We measured the time required for the oblivious transfer and private information retrieval protocols separately to test the performance of each protocol and the relative performance between the two protocols. The prototype was created on a machine with an Intel Core 2 Duo E8200 2.66GHz processor and 2GB of RAM. The prototype was written using Visual C++ under the Windows XP operating system. We used the Number Theory Library (NTL) [91] for computations requiring large integers and OpenSSL [1] to compute the SHA-1 hash. The whole solution was executed for 100 trials, where the time taken (in seconds) for each major component was recorded and the average time was calculated.

Oblivious Transfer Protocol

In our implementation experiment for the oblivious transfer protocol, we generated a modified ElGamal instance with $|p| = 1024$ and $|q| = 160$, where $q|(p-1)$. We also found a generator a, and set $g = a^q$ (g has order q). We set the public matrix P to be a 25×25 matrix of key and index information.

We first measured the time required to generate a matrix of keys according to Algorithm 19. This procedure only needs to be executed once for the lifetime of the data. There is a requirement that each hash value of the concatenation $g^{R_i}||g^{C_j}$ is unique. We use the SHA-1 to compute the hash $H(\cdot)$, and we assume that there is negligible probability that a number will repeat in the matrix.

The major three components of the oblivious transfer protocol are the user's query, server's response, and user's decode. Table 5.3 displays the average time required for each component of the protocol. The magnitude of the numbers in Table 5.3 demonstrates that our protocol is efficient at runtime.

Component	*Average Time (s)*
Initialisation	0.28829
Query	0.00484
Response	0.11495
Decode	0.00031

Table 5.3: Average time required for Oblivious Transfer protocol

Private Information Retrieval Protocol

In the PIR protocol we fixed a 15 × 15 private matrix, which contained the data owned by the server. We chose the prime set to be the first 225 primes, starting at 3. The powers for the primes were chosen to allow for at least a block size of 1024 bits $(3^{647}, 5^{442}, ..., 1429^{98})$. Random values were chosen for each prime power $e = C_i \ (mod \ \pi_i)$, and the Chinese Remainder Theorem was used to determine the smallest possible e satisfying this system of congruences.

Once the database has been initialised, the user can initiate the protocol by issuing the server his/her query. The query consists of finding a suitable group whose order is divisible by one of the prime powers π_i. We achieve this in a similar manner to Gentry and Ramzan [42]. We choose primes q_0 and q_1 and compute "semi-safe" primes $Q_0 = 2q_0\pi_i + 1$ and $Q_1 = 2q_1 + 1$. We set the modulus as $N = Q_0Q_1$ and group order as $\phi(N) = \phi(Q_0Q_1) = (Q_0 - 1)(Q_1 - 1)$. Hence, the order $\phi(N)$ has π_i as a factor. We set g to be a quasi-generator, such that the order of g also contains π_i. In our experiment, we set $|q_0| = |q_1| = 128$. This results in a modulus N which is roughly 1024 bits in length, which is equivalent to an RSA modulus.

As in the oblivious transfer based protocol there are 3 major steps: the user's query, the server's response, and the user decoding. The average time required for each of these major components are presented in Table 5.4.

Component	*Average Time (s)*
Query	9.64984
Response	4.57127
Decode	0.25451

Table 5.4: Average time required for Private Information Retrieval protocol

5.7.2 Full Implementation

For the full implementation of our protocol, with the updated key structure, we used a desktop machine (which is the same as in the previous experiment) and a mobile phone. Where the desktop machine was playing the role of the server and the mobile phone was playing the role of the client. As in the previous experiment, we measured the required time for the oblivious transfer and private information retrieval protocols separately to test the performance of each protocol and the relative performance between the two protocols.

The implementation on the mobile phone platform is programmed using the Android Development Platform, which is a Java-based programming environment. The mobile device used was a Sony Xperia S with a dual-core 1.5 GHz CPU and 1 GB of RAM. The whole solution was executed for 100 trials, where the time taken (in seconds) for each major component was recorded and the average time was calculated. The parameters for this experiment, with respect to both protocols, are described next.

Oblivious Transfer Protocol

In the full implementation we used the revised key structure and the parameters are as follows. We generated a modified ElGamal instance with $|p| = 1024$ and $|q| = 160$, where $q|(p-1)$. We also found a generator a, and set $g_0 = a^q$ (g has order q). We also set a generator g_1, which has order $q-1$. We set the public matrix P to be a 25×25 matrix of key and index information.

	Average Time (s)	
Component	**Desktop**	**Mobile**
$Initialisation_{OT}$	1.70958	—
$QueryGeneration_1$	—	0.00108
$ResponseGeneration_1$	0.00969	—
$ResponseRetrieval_1$	—	0.00004

Table 5.5: Oblivious Transfer experimental results for desktop and mobile platforms

As in the previous experiment, we measured the time required to generate a matrix of keys. Except we used the revised key structure initialisation (Algorithm 22). There is a requirement that each hash value of $g_0^{g_1^{R_i} g_1^{C_j}}$ is unique[4]. We use the SHA-1 to compute the hash $H(\cdot)$, and we assume that there is negligible probability that a number will repeat in the matrix.

Private Information Retrieval Protocol

The experimental configuration for this protocol in this implementation was the same as the desktop-only implementation. But to summarise briefly, we fix a 15 × 15 matrix, which contains data owned by the server. The matrix is populated by the first 225 primes, and the powers were chosen such that each element can represent 1024 bits ($3^{647}, 5^{442}, ..., 1429^{98}$). For more details refer to the previous subsection.

Results

In both phases of our solution, there are 3 major steps: the user's query, the server's response, and the user decoding. Table 5.5 displays the average runtime on the desktop and mobile platforms, for each component of the oblivious transfer phase. Similarly, Table 5.6 presents the average times for each component of the private information retrieval protocol.

[4] We remark that we used one generator in G_1 to simplify the experiment.

	Average Time (s)	
Component	**Desktop**	**Mobile**
$QueryGeneration_2$	—	23.90666
$ResponseGeneration_2$	4.57127	—
$ResponseRetrieval_2$	—	0.49123

Table 5.6: Private Information Retrieval experimental results for desktop and mobile platforms

5.7.3 Discussion

Based on these experimental results, most of the time is taken by the generation of the user's query (23.9 seconds to generate a query). This is due to the primality testing of Q_0 and Q_1. This requirement must be satisfied, otherwise we would not be able to compute the order as $\phi(N) = (Q_0 - 1)(Q_1 - 1)$, and the factorisation of the order would not contain π_i. The average of the response time and the decoding time are much smaller in comparison. We assume that the server has much more computational power at its disposal. Hence, if there are many users, the server can use parallel processing to increase the throughput of the protocol.

The main concern is keeping the query time for the user as low as possible, and on average the user query time is reasonable, given the amount of data that is exchanged in one round of the protocol. In addition, these results strongly agree with the theoretical performance analysis in Section 5.6.

5.8 Conclusion and Recommendations

In this chapter we presented a solution to solve one of the location-based query problems. The problem is that a client wants to learn about local businesses from a LBS, but does not want to disclose his/her location. We designed a security model and security definitions for this problem, and then we developed a solution. We evaluated the performance of our solution and demonstrated that it was within practical limits.

The protocol that is presented uses a computationally intensive PIR scheme, with respect to the server. Future work would include investigating a computationally efficient scheme, like the ones featured in Chapter 4, when the mobile network has evolved enough to support the high bandwidth.

Chapter 6

Privacy Preserving Association Rule Mining

This chapter will present contributions for privacy preserving data mining. In particular it will present the results from our paper entitled ‘Fully homomorphic encryption based two-party association rule mining’ [60]. This is an extension of our previous paper with the title ’Secure Two-Party Association Rule Mining’ [59]. Some of the results of the orignial work are included to help explain the extensions made.

This work would be classified under the private client/private server privacy model definition given in Chapter 1. Although in this instance, both parties play the role of server. But one could argue that the first player is querying the second player, where the query is constructed using the first player’s input. This does not affect the security definitions however, as the same constants carry over under different names.

We will begin this chapter by exploring the general data mining concepts. This will contextualise data mining techniques and emphasise their purpose, which is to discover knowledge. From there, we will focus on the specific case of association rule mining, and investigate the privacy issues introduced when two parties are engaged in a computation of association rules. The main contribution of this chapter is to use fully homomorphic encryption techniques to provide database privacy for both parties.

6.1 General Data Mining Concepts

The field of knowledge discovery from data, also known as *data mining*[1], has expanded greatly in the past decade [51]. It has become an essential tool in a world where enormous amounts of complex data are collected, and there is a need to find hidden meanings or patterns. Businesses use these hidden patterns to support the decision making process and promote better practices.

The task of collecting and storing data is fundamentally governed by a data model. By far, the most popular data model for storing data is the relational model (and associated relational algebra). Informally, the relational model arranges its data in a table, where each row represents some real-world entity or relationship. A natural extension of this simple construct is to augment it with object-oriented techniques. In simple terms this means adding methods to tables to create a hybrid object-relational system.

Once the data is collected, simple statistical measures can be calculated. These include: measures of central tendency, such as the mean median and mode; measures of variation or spread, such as variance; and aggregate measures, such as count, min, and max. These statistical methods do not really capture the complex information contained within the data. Therefore we need more intelligent techniques to extract this information from the data.

Data mining methods can be broadly classified into different kinds of tasks, based on what kind of knowledge we wish to discover. The main kinds of tasks are the following.

Association rule mining: This is also known as *market basket analysis*, as it looks at finding patterns among shopping transactions. Two measures, called support and confidence, are used to determine the 'goodness' of association rules. In association rule mining algorithms, these measures are used as thresholds to remove uninteresting rules.

[1] Strictly speaking, data mining forms a part of the knowledge discovery process. However in this context, they are virtually the same idea.

Clustering: This is about finding objects that are similar, usually in a coordinate space. A common method is k-means clustering, where the algorithm tries to find k centroids of k different kinds of objects. Of course, there are different ways to measure similarity. Euclidean distance being one example.

Classifying: Algorithms for classifying attempt to break a set of objects into separate categories, when supplied a finite set of attributes as input. Classification algorithms are thought of as *supervised* procedures. This means that a model is constructed by training it with data where the classes are known, and then using the model to classify unknown data.

Anomaly detection: This is where we want to know which records are so far from the center. This is commonly used in fraud detection, where we want to know which transactions are fake.

Complementing the above data mining methods is the notion of building a *data warehouse*. The main advantage for building a data warehouse is that a business is able to examine all recorded history to discover trends or patterns. This is known as OLAP, or OnLine Analytical Processing. This is contrasted with OLTP, known as OnLine Transaction Processing, which is concerned with the day-to-day analysis of data. A data warehouse is constructed by integrating all data into a single repository, where analysis is performed. The integration of the data typically includes data cleaning that fills in missing data, or converts the representation of the data so that the structure stored within the data warehouse is consistent.

6.2 Association Rule Mining

We now turn our attention to association rule mining. Association rule mining algorithms provide a means to discover interesting correlations (or implications) between different items in a transactional database. Informally, a transactional database is a collection of records, where each record represents a transaction made by a customer. Each transaction contains a subset of all items known to the system, which is called the global itemset. This will be made more formal shortly.

When a transactional database is stored at a single site, there can be no privacy concern, other than from outside attacks. When the data belongs to two or more sites, there are causes for concern. A motivating example, where privacy preserving algorithms would be needed would involve patient data belonging to two hospitals. It may be unethical or even illegal to distribute the patient data to either site. While obeying this restriction, the hospitals still wish to engage in association rule mining to determine what can be learned from the union of the data. Therefore, the challenge is for both hospitals to perform data mining on the data, without unnecessarily disclosing the individual data elements. A solution to this problem, the two party privacy preserving association rule mining problem, is proposed by this chapter.

Association rules from association rule mining algorithms are determined by the support and confidence of itemsets. An example of an itemset is {computer, keyboard, speakers}. If association rule, based on this itemset, like {computer, keyboard}⇒{speakers} is discovered and revealed to the manager of a supermarket, then the manager can consider grouping these items to increase sales. There exists a popular algorithm for determining association rules efficiently, which is Agrawal et.al.'s Apriori based solution [2]. This recursive algorithm is known to be fast and produce results in a reasonable time, but we cannot simply apply this to a data sensitive scenario like our hospital example. We need to modify the algorithm such that the privacy of the data belonging to both parties is preserved.

Generally speaking, the meaning of privacy in data mining algorithms is to prevent data misuse [24]. In the ideal setting, we assume a trusted third party will perform the data mining algorithm and then broadcast the result. Of course, this is hard to realise in practice. Hence we need tools and techniques to satisfy the privacy requirements, which is comparable to a trusted third party. The first privacy preserving data mining algorithm was introduced by Lindell and Pinkas in 2000 [65]. They present a protocol that produces a decision tree using the ID3 algorithm, proposed by Quinlan [84], whereby the entropy or information gain is computed privately. This is achieved through the use of Yao's garbled circuit [103] and 1-out-of-2 oblivious transfer [33]. The main contribution is that Yao's Garbled Circuit can be applied to provide privacy for both parties, since it enables private evaluation of a function.

With private function evaluation, there are other methods of preserving privacy in association rule mining. They include data perturbation [88, 34], and homomorphic encryption [81, 32]. Data perturbation means that random data is added to the actual record, in order to preserve privacy. The randomness alleviates the privacy concerns because the data is concealed, but it also reduces the accuracy of the final result. Homomorphic encryption, on the other hand, allows the data miner to modify the plaintext while encrypted. This provides far greater control and accuracy than that of pure data perturbation methods. Hybrid protocols, which combine both methods, have also been presented [80, 107].

The core component of association rule mining is to compare the count frequency of a particular itemset. This is the same as in the case where data is stored on two separate sites. If the database size of two databases are represented by d_1 and d_2 and the count frequency of an itemset (such as abc) possessed by both parties are c_1 and c_2, then the inequality $\frac{c_1+c_2}{d_1+d_2} \geqslant s$ tests whether the itemset is frequent or not, where s is the minimum support threshold. To preserve the privacy of the data, this test must be performed securely. Kantarcioglu et al. suggest that the computation of the form $\frac{c_1+c_2}{d_1+d_2} \geqslant s$ can be converted into the millionaire's problem [58], which can be solved by Yao's garbled circuit [103]. They also expose an inherent problem with the two party association rule mining process, which is that any rule that is supported globally and is not supported by the first party, must be supported by the second party. This inherent problem is beyond the scope of this work. Our goal is to protect both databases from unnecessary disclosure. In this chapter we build on the result by [58], by providing a more reasonable solution using a new result in homomorphic encryption.

There are many encryption schemes that have the homomorphic property [87, 32, 81]. However, up until recently, all known homomorphic encryption schemes are partially homomorphic. In other words, they are only homomorphic under one operation, either addition or multiplication. An encryption scheme that supports both operations would be considered fully homomorphic. The breakthrough work of Gentry, has provided the first secure fully homomorphic encryption scheme [38]. Basically, he created a encryption scheme that could evaluate its own decryption circuit.

Association rule mining algorithms are typically a two stage process. The first stage consists of generating a list of frequent itemsets from the set of all known items. To avoid generating a list of all possible itemsets a threshold value is chosen, which is known as the support. This support value filters out most itemsets that would lead to uninteresting results. The second stage generates association rules from the list of frequent itemsets. The association rules are chosen, based on their support value. The two values, support and confidence, define how well we should trust an association rule generated from this process.

More formally, any combination of items are known as an itemset. That is, an itemset $I_s = \{I_1 \bigcup I_2 ... \bigcup I_k\}$ where, $I_i \subseteq I$. An itemset with length k is known as a k-itemset. The general form of an association rule is $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \Phi$. The support of $X \Rightarrow Y$ is the probability of a transaction in the database to contain both X and Y. On the other hand, the confidence of $X \Rightarrow Y$ is the probability of a transaction containing X will also contain Y. If an association rule is of the form $AB \Rightarrow C$, the support and confidence is calculated as the following.

$$Support_{AB \Rightarrow C} = s = \frac{\sum_{i=1}^{sites} SupportCount_{ABC_i}}{\sum_{i=1}^{sites} DatabaseSize_i} \tag{6.1}$$

$$Support_{AB} = \frac{\sum_{i=1}^{sites} SupportCount_{AB_i}}{\sum_{i=1}^{sites} DatabaseSize_i} \tag{6.2}$$

$$Confidence_{AB \Rightarrow C} = c = \frac{Support_{AB \Rightarrow C}}{Support_{AB}} \tag{6.3}$$

The Apriori algorithm is an effective method for determining association rules [2]. It works recursively, starting with finding frequent 1-itemsets L_1, which have support greater than the threshold value s. From the 1-itemsets, the 2-itemsets L_2 are found. This repeats until L_{k+1} is empty. Then the set, $L_1 \cup L_2 \cup ... \cup L_k$ is the set of globally frequent itemsets, which is represented by L_g. Using L_g, one generates all association rules, which have confidence greater than c. We refer the interested reader to [51, 97], for a more comprehensive description of the Apriori algorithm and related algorithms.

6.3 Background

This section discusses background knowledge, terms and concepts used in the context of this chapter. Additionally, unless specified otherwise, it is assumed that Alice and Bob own a set of inputs and wish to compute some boolean function on the union of the inputs. The computation of the boolean function is performed by the party who does not have access to the private key. The notations that will be used are as follows.

6.3.1 Notations

Let us define the following notations which will be used in the context of this chapter. Let L_i : be the i-itemset, C_i be the candidate i-itemset, and L_g be the global frequent itemset. Elements of the globally frequent itemset have a support greater than the minimum threshold. Let $\oplus$ and $\otimes$ be the XOR and AND operations respectively and let $\boxplus$ and $\boxtimes$ be the respective homomorphic equivalents (these will be defined shortly). Let $\overline{X}$ be the negation of X. If X is an integer, the bit representation is inverted. Encryption and decryption is denoted by $E_{pk}(X)$ and $D_{sk}(Y)$, respectively. Again, if the input is an integer, instead of $0, 1$, then each bit in the binary representation is encrypted and the concatenation of the ciphertexts is returned. Intuitively, the decryption function does the opposite.

6.3.2 Summary of Yao's Garbled Circuit

Using Yao's garbled circuit [103], any circuit can be evaluated by two parties while keeping the inputs of both parties private. In the simple case there is one gate, for instance Boolean 'AND'. This gate has two inputs and one output, totalling three wires. Each wire is attached to the gate and can either be a 0 or 1, according to the

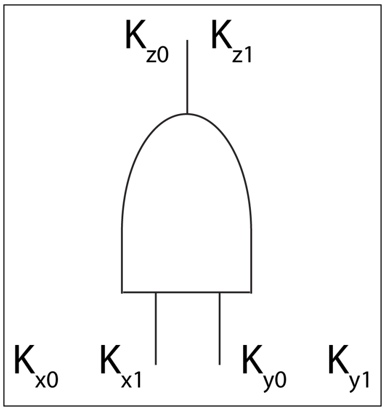


Figure 6.1: AND gate according to Yao's algorithm

Input wire x	Input wire y	Output wire z	Garbled output
K_{x_0}	K_{y_0}	K_{z_0}	$E_{K_{x_1}}(E_{K_{y_1}}(K_{z_1}))$
K_{x_0}	K_{y_1}	K_{z_0}	$E_{K_{x_1}}(E_{K_{y_1}}(K_{z_0}))$
K_{x_1}	K_{y_0}	K_{z_0}	$E_{K_{x_0}}(E_{K_{y_0}}(K_{z_0}))$
K_{x_1}	K_{y_1}	K_{z_1}	$E_{K_{x_1}}(E_{K_{y_1}}(K_{z_0}))$

Table 6.1: Garbled truth table for AND gate

Boolean domain. The first party, Alice, generates a truth table for the gate, along with randomly generated keys, and then the second party, Bob, evaluates the gate privately. Figure 6.1 illustrates the gate, with the associated keys.

Each K value is chosen by Alice and Bob does not know if it corresponds to a 0 or a 1. With these values, Alice constructs a truth table whereby both output K values are encrypted by the input K values, and then the resulting encryption is permuted to obfuscate the inputs. This is shown in Table 6.1.

Once Bob receives the gate definition and corresponding garbled circuit output, he can process the gate when he gets one key for each input wire. Alice simply sends Bob the key corresponding to her input, since this number is meaningless to Bob. Whereas Bob obtains the key for his input using 1-out-of-2 oblivious transfer

[85]. Assuming the oblivious transfer is secure, Alice cannot learn which key Bob is requesting.

Once Bob has both keys he can decrypt only one row in the table to obtain the correct output from the gate. This single gate example can easily be extended to a multiple gate scenario. This can be done by chaining many of these gates together and computing the corresponding garbled truth table for each.

6.3.3 Some Binary Operations

This section discusses how some functions such as addition or subtraction for integer numbers are built using basic binary operations. These fundamental mechanisms are used in our proposed solution to construct fully homomorphic encryption for plaintext integers. We direct the interested reader to [52], where more details can be found on constructing digital circuits.

Integer Addition

Let two integer numbers be X and Y are of ℓ bits. $X = \{X_\ell|X_{\ell-1}|...|X_2|X_1\}$ and $Y = \{Y_\ell|Y_{\ell-1}|...|Y_2|Y_1\}$ where, $X_i, Y_i \in \{0,1\}$. To add these two integers together every bit in one number is added with corresponding bit in the other number along with the carry bit from the previous stage. Initially the carry bit is 0. The following presents a simplification of the carry circuit.

$$R_i = X_i\overline{Y_iC_{i-1}} + \overline{X_i}Y_i\overline{C_{i-1}} + \overline{X_iY_i}C_{i-1} + X_iY_iC_{i-1} = X_i \oplus Y_i \oplus C_{i-1} \tag{6.4}$$

$$\begin{aligned} C_i &= \overline{X_i}Y_iC_{i-1} + X_i\overline{Y_i}Ci - 1 + X_iY_i\overline{C_{i-1}} + X_iY_iC_{i-1} \\ &= C_{i-1}(X_i \oplus Y_i) + X_iY_i \end{aligned} \tag{6.5}$$

Two's Complement Number

The Two's complement of a number is considered as the negative of the number. It can be determined by subtracting itself from a large number (must be power of two). Let X be an integer with ℓ bits then its Two's complement is equal to $2^{\ell} - X$. Two's complement of $X = \overline{X} + 1$ where $\overline{X}$ represents the binary NOT operation of X, where all bits are inverted. We direct the interested reader to [52] for more detail on Two's complement.

The Two's complement concept is used to implement subtraction, where X and Y are positive integers. More specifically, when we want to compute $X - Y$, we add X with Two's complement of Y. That is:

$$X - Y = X + (\overline{Y} + 1) \tag{6.6}$$

6.3.4 Fully Homomorphic Encryption (FHE)

We summarise the basic scheme of the fully homomorphic encryption scheme over the integers (see Chapter 2). A ciphertext is computed as the following.

$$c = pq + 2r + m \tag{6.7}$$

where p is the private key, q and r are chosen randomly, and m is the message $m \in \{0, 1\}$. The message is recovered as follows:

$$m = (c \ mod \ p) \ mod \ 2 \tag{6.8}$$

As pointed out in Chapter 2, the message space is restricted to boolean data, such that they can be made to evaluate their own decryption circuit for bootstrapping. This can easily be extended to integers, represented as binary vectors, which is useful in the association rule mining application.

Fully Homomorphic Encryption for Integers

The association rule mining algorithm deals with count values of itemsets, which are represented as integers. Hence, we need to extend the boolean nature of the underlying homomorphic encryption scheme to support plaintext integers. This is achieved by representing the integer as a binary vector and encrypting each bit separately. For instance, consider a ℓ-bit integer $X = x_\ell|x_{\ell-1}|...|x_1$ (where, $X \in \mathbb{Z}$ and $x_i \in \{0, 1\}$) can be encrypted as shown in Equation 6.9, using the encryption function of the appropriate fully homomorphic encryption scheme.

$$\alpha = E_{pk}(X) = [\alpha_\ell|\alpha_{\ell-1}|...|\alpha_1] = [E_{pk}(x_\ell)|E_{pk}(x_{\ell-1})|...|E_{pk}(x_1)] \tag{6.9}$$

Similarly decryption can be expressed as Equation 6.10

$$X = D_{sk}(\alpha) = [x_\ell|x_{\ell-1}|...|x_1] = [D_{sk}(\alpha_\ell)|D_{sk}(\alpha_{\ell-1})|...|D_{sk}(\alpha_1)] \tag{6.10}$$

Using the definition of integer encryption and decryption in the above equations, it is now possible to define some integer functions which will be used in our proposed solution. Let us consider α, β and θ are encryption of ℓ-bit integers. Their forms are $\{\alpha_\ell|\alpha_{\ell-1}|...|\alpha_1\}$, $\{\beta_\ell|\beta_{\ell-1}|...|\beta_1\}$, and $\{\theta_\ell|\theta_{\ell-1}|...|\theta_1\}$, respectively.

- Homomorphic AND operation ($\boxtimes$):
  This function computes the homomorphic AND operation between two encrypted integers and returns another encrypted integer. Then $\theta = \alpha \boxtimes \beta$. The output is computed bit-by-bit using homomorphic property for AND-ing binary digits, that is $\theta_i = \alpha_i \ AND \ \beta_i$ for $1 \leqslant i \leqslant \ell$.

- Homomorphic XOR operation ($\boxplus$):
  This function computes the homomorphic XOR operation between two encrypted integers and returns another encrypted integer. Thus $\theta = \alpha \boxplus \beta$. The output is computed bit-by-bit using homomorphic property for XOR-ing binary digit, that is $\theta_i = \alpha_i \ XOR \ \beta_i$ for $1 \leqslant i \leqslant \ell$.

- Homomorphic Addition ($\boxed{*}$):
  This function computes the homomorphic addition operation between two encrypted integers and returns another encrypted integer. Thus $\theta = \alpha \boxed{*} \beta$. This operation is performed according to the description in Section 6.3.3.

6.4 Proposed Solution

This section discusses our two party privacy preserving association rule mining protocol, which uses fully homomorphic encryption to compare the frequency counts of itemsets.

6.4.1 Motivation and Model Definition

Consider two data sites Alice (A) and Bob (B) who possess two horizontally partitioned transactional databases DB_1 and DB_2 of size $|DB_1|$ and $|DB_2|$ respectively (see Figure 6.2). A horizontally partitioned database means that data with the same attributes are on different sites. A and B desire to learn the interesting association rules from the union of their databases $DB = \{DB_1 \bigcup DB_2\}$, without disclosing individual itemset counts to each other. The interestingness of association rules is defined by s and c, which are the support and confidence thresholds respectively.

Our protocol follows a two stage process. In the first stage, the global frequent itemset L_g is produced according to the minimum support s (as defined by Equation 6.2). The second stage determines association rules from L_g, using the minimum confidence threshold c (as defined by Equation 6.2). In our protocol, the threshold comparisons are performed securely using fully homomorphic encryption.

The first stage involves determining frequent itemsets, based on inputs from A and B. The basic steps to determine whether an itemset, with counts c_1 and c_2 in A and B respectively, is frequent or not:

- Step 1: Data site A sends count c_1 and $|DB_1|$ to other party B.

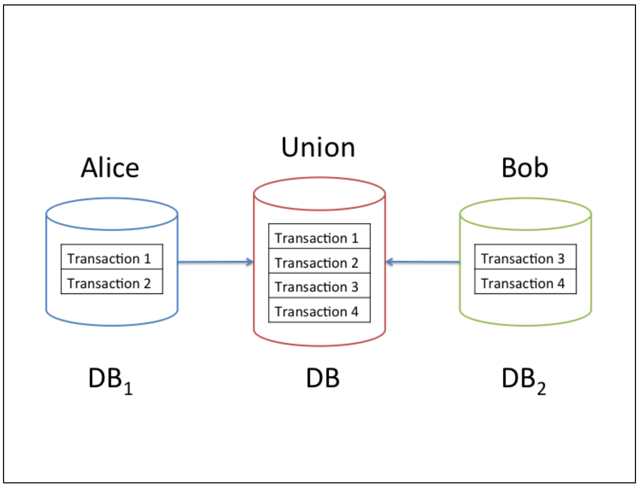


Figure 6.2: Database model for association rule mining application

- Step 2: Data site B sends count c_2 and $|DB_2|$ to other party A.
- Step 3: Both A and B compute whether $\frac{c_1+c_2}{|DB_1|+|DB_2|} \geqslant s$. If true then the itemset is frequent.

The first two steps are simply communicating values. The third step is where the comparison occurs, and is the most interesting to us. The following equation, Equation 6.11, generalises this step, such that it can support percentages.

$$\frac{c_1 + c_2}{|DB_1| + |DB_2|} \geqslant \frac{s}{100} \tag{6.11}$$

Equation 6.11 can be rewritten as:

$$c_1 \times 100 - s \times |DB_1| \geqslant s \times |DB_2| - c_2 \times 100 \Leftrightarrow \mathbb{A} \geqslant \mathbb{B} \tag{6.12}$$

The left hand side ($\mathbb{A}$) and right hand side ($\mathbb{B}$) of the Equation 6.12 is to be computed separately by party A and B respectively. To preserve the privacy of their respective databases, $\mathbb{A}$ and $\mathbb{B}$ must be compared securely.

Assume that, A and B use a fully homomorphic encryption scheme, in which A generates public and secret keys pk and sk respectively. B encrypts using public key pk without being able to decrypt any ciphertext. Let us also assume that these parties communicate through a private channel which is protected by any standard secret key cryptosystem, such as DES [36] or AES [35]. It is also assumed that A and B are semi-honest, which means that they follow the protocol correctly, but they are allowed to record all of the messages as transcripts and examine them in the future.

6.4.2 Secure Comparison of Two Integers

This section proposes a solution to compare two numbers privately. Consider two ℓ-bit long integers M and N. Our proposed technique compares M and N and determines whether M is equal or less than or greater than N without the revealing the value of M or N.

Let us first consider a simple version of the comparison algorithm, presented by Algorithm 24, where all data is in its plaintext form.

Algorithm 24 Comparison of two plaintext integers (M and N)

```
input : integers $M, N$
output : (One bit output. If $output = 0$ then $M \geqslant N$ otherwise $M < N$.)
Begin
    $Y \leftarrow M + \overline{N} + 1$ {Subtraction of $M$ and $N$ gives the clue about their relative
  size. Two's complement of a number is equivalent to the negative of the same
  number. Therefore, $Y$ is equivalent of $(M - N)$.}
    $R \leftarrow Y\ AND\ 2^{n-1}$
return    $MSB(R)$ {returns the most significant bit (MSB) of R. This is actually
is the sign bit of the subtracted result in Two's complement form. This indicates
which input is larger}
End
```

With the consideration of fully homomorphic operators, which are $\boxtimes$, $\boxplus$ and $\boxast$ as derived in Section 6.3.4, Algorithm 25 compares two ℓ-bit numbers, while encrypted. Let A and B encrypt their secret numbers as $\alpha \leftarrow E_{pk}(M)$ and $\beta \leftarrow E_{pk}(N)$ respectively. These ciphertexts, α and β, are compared in Algorithm 25.

Algorithm 25 Comparison of two ciphertext integers (α and β)

```
input : ciphertexts $\alpha, \beta$
output : ciphertext $\rho[\ell]$ (One bit encrypted output. If $R = D_{sk}(\rho[\ell]) = 0$ then $M \geqslant N$, otherwise $M < N$.)
Begin
    $\overline{\beta} \leftarrow \beta \boxplus E_{pk}(2^n - 1))$ {Binary negation of $\beta$}
    $\psi \leftarrow \alpha \boxast \overline{\beta}$ {Homomorphic addition of $\alpha$ and $\overline{\beta}$}
    $\psi \leftarrow \psi \boxast E_{pk}(1)$
    $\rho \leftarrow \psi \boxtimes E_{pk}(2^{n-1})$ {The result is encrypted and only Alice can decrypt that}
return $\rho[\ell]$ {returns the encryption of sign bit which is the most significant bit of $\rho$.}
End
```

For future reference, we define Algorithm 25 as a function: *HomComparison* (α, β). This function compares two encrypted integers and returns the encryption of one bit (MSB) result $\rho[\ell]$, which can be decrypted only by the owner of the secret key (in our case, this is Alice). If $R = D_{sk}(\rho[\ell]) = 0$ then $M \geqslant N$, else $M < N$.

6.4.3 Proposed Two Party ARM

Using the function *HomComparison*, we present Algorithm 26, where each iteration computes L_k from L_{k-1}. When used repeatedly, it enables the generation of all frequent itemsets L_g.

Figure 6.3 illustrates a flow diagram of Algorithm 26, with the assumption that counts of a particular itemset $\mathbb{I}$ in A and B are c_1 and c_2 respectively.

Once L_g has been computed by A and B, all association rules with minimum confidence c can be generated. Equations 6.1, 6.2 and 6.3 can be combined and

Algorithm 26 Global frequent itemset generation between A and B

```
input of A : $L_k$ with counts
input of B : $L_k$ with counts, $s$ as the minimum support
output : $L_{k+1}$
Begin
Both Alice(A) and Bob(B)
    $C_{k+1} \leftarrow GenerateCandidate(L_k)$
for ( All $\mathbb{I} \in C_{k+1}$) do
  Alice(A)
      $c_1 \leftarrow count(\mathbb{I})$
      $t \leftarrow \omega_1 \times 100 - s \times |DB_1|$
      $\alpha \leftarrow E_{pk}(t)$
      $SendToB(\alpha)$ {transmits encrypted left hand side of Equation 6.12}
  Bob(B)
      $c_2 \leftarrow count(\mathbb{I})$
      $t \leftarrow s \times |DB_2| - \omega_2 \times 100$
      $\beta \leftarrow E_{pk}(t)$
      $\tau \leftarrow HomComparison(\alpha, \beta)$
      $SendToA(\tau)$
  Alice(A)
      $R \leftarrow D_{sk}(\tau)$ {Only $A$ can decrypt the result}
  if    $R = 0$ then
        $L_{k+1} \leftarrow \{L_{k+1} \bigcup \mathbb{I}\}$ {itemset satisfies minimum support requirement}
        $SendToB(\mathbb{I})$
  end if
  Bob(B)
      $L_{k+1} \leftarrow \{L_{k+1} \bigcup \mathbb{I}\}$
end for
End
```

simplified as follows:

$$\begin{aligned} Confidence_{AB \Rightarrow C} = c &= \frac{Support_{AB \Rightarrow C}}{Support_{AB}} \\ &= \frac{\frac{\sum_{i=1}^{sites} SupportCount_{ABC_i}}{\sum_{i=1}^{sites} DatabaseSize_i}}{\frac{\sum_{i=1}^{sites} SupportCount_{AB_i}}{\sum_{i=1}^{sites} DatabaseSize_i}} \\ &= \frac{\sum_{i=1}^{sites} SupportCount_{ABC_i}}{\sum_{i=1}^{sites} SupportCount_{AB_i}} \end{aligned} \tag{6.13}$$

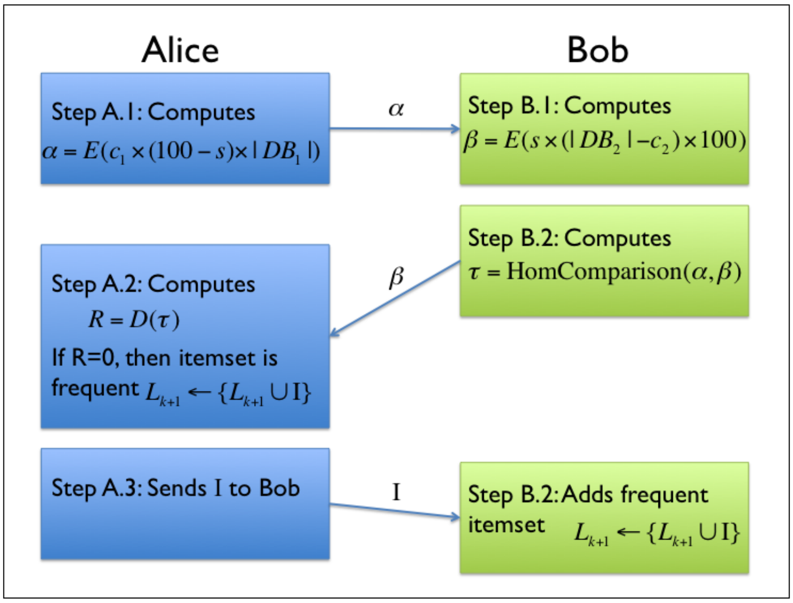

Alice
Bob
Step A.1: Computes $\alpha = E(c_1 \times (100 - s) \times |DB_1|)$
α
Step B.1: Computes $\beta = E(s \times (|DB_2| - c_2) \times 100)$
Step B.2: Computes $\tau = \mathrm{HomComparison}(\alpha, \beta)$
β
Step A.2: Computes $R = D(\tau)$
If R=0, then itemset is frequent $L_{k+1} \leftarrow \{L_{k+1} \cup \mathrm{I}\}$
Step A.3: Sends I to Bob
I
Step B.2: Adds frequent itemset $L_{k+1} \leftarrow \{L_{k+1} \cup \mathrm{I}\}$

Figure 6.3: Steps for secure itemset generation

Consider L_i as one of the frequent itemset then L_i needs to split into two itemsets to construct association rules. Let us divide L_i into $\imath_1$ and $\imath_2$ such that $L_i = \{\imath_1 \bigcup \imath_2\}$ and $\{\imath_1 \bigcap \imath_2\} = \phi$. Assume the counts of L_i and $\imath_1$ are l_1 and l_2 with A. Similarly, the counts of L_i and $\imath_1$ are $l_1\prime$ and $l_2\prime$ with B. Then for the association rule $\imath_1 \Rightarrow \imath_2$ to be selected, the following condition has to be satisfied.

$$\frac{l_1 + l_1\prime}{l_2 + l_2\prime} \geqslant \frac{c}{100} \tag{6.14}$$

which can be expressed as

$$100l_1 - cl_2 \geqslant cl_2\prime - 100l_1\prime \Rightarrow \mathbb{C} \geqslant \mathbb{D} \tag{6.15}$$

The left hand side ($\mathbb{C}$) and the right hand side ($\mathbb{D}$) of the equation are to be computed by A and B respectively. For privacy reasons, these values must be compared securely. Algorithm 27 generates all association rules from the global frequent itemset L_g.

Algorithm 27 Association rule generation from L_g

```
$input\ of\ B : L_g, c$
$output : AR$ (Set of all association rules)
Begin
Both Alice(A) and Bob(B)
for    $All(L_i \in L_g)$ do
    Split $L_i$ into all possible $\imath_1$ and $\imath_2$ such that,$L_i = \{\imath_1 \bigcup \imath_2\}$ and $\{\imath_1 \bigcap \imath_2\} = \phi$\{to generate all possible combinations of association rules from $L_i$\}
  Alice(A)
      $l_1 \leftarrow count(L_i)$
      $l_2 \leftarrow count(\imath_1)$
      $t \leftarrow 100l_1 - cl_2$
      $\alpha \leftarrow E_{pk}(t)$
      $SendToB(\alpha)$
  Bob(B)
      $l_1\prime \leftarrow count(L_i)$
      $l_2\prime \leftarrow count(\imath_1)$
      $t \leftarrow cl_2\prime - 100l_1\prime$
      $\beta \leftarrow E_{pk}(t)$
      $\tau \leftarrow HomComparison(\alpha, \beta)$
      $SendToA(\tau)$
  Alice(A)
      $R = D_{sk}(\tau)$
  if    $R = 0$ then
        $AR \leftarrow AR \bigcup \{\imath_1 \Rightarrow \imath_2\}$\{The association rule satisfies conditions and added to the final output\}
        $SendToB(\imath_1 \Rightarrow \imath_2)$
  end if
  Bob(B)
      $AR \leftarrow AR \bigcup \{\imath_1 \Rightarrow \imath_2\}$
end for
End
```

6.5 Security Analysis

The analysis of the presented protocol will assume the semi-honest or honest-but-curious model. In the semi-honest model, if either party becomes corrupted some time in the future by an adversary (we only enable an adversary to corrupt one party, not both), the adversary is unable to distinguish between a simulation of the protocol and the real protocol. Therefore, the adversary cannot obtain any further information than what the corrupted player knows. This approach is one of the standard definitions of security [47, 16].

For this analysis, we will formulate our protocol according to the real-world/ideal-world paradigm. In this paradigm, we consider an execution of protocol Π in the ideal model, where the computation is performed by a trusted third party. Then, we construct an execution of the same protocol in the real-world model. If we can show that these executions are indistinguishable, then we can claim that our solution is secure.

As our protocol is designed for two parties, we model the Π as a protocol for computing a function f, defined by $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$, for two parties. We denote $f_1(x, y)$ as the first element of f and denote $f_2(x, y)$ as the second element. With this in mind, let us define the ideal model and real world models.

Ideal-world Model Let $\mathcal{B} = \{B_1, B_2\}$ denote a pair of polynomial-time algorithms operating in the ideal model. Each algorithm has three inputs $B_i(u, v, z)$, where u is the input, v is the output and z is random input. Define the ideal execution using algorithms in $\mathcal{B}$ on inputs (x, y) (where x belongs to the first algorithm and y belongs to the second algorithm) and random input z as

$$\mathrm{IDEAL}_{f,\mathcal{B}(z)(x,y)} \overset{\mathrm{def}}{=} (f(x, y), B_1(x, f_1(x, y), z), B_2(y, f_2(x, y), z) \tag{6.16}$$

Real-world Model Let $\mathcal{A} = \{A_1, A_2\}$ be a pair of polynomial-time algorithms operating in the real-world model. Define the real-world execution using algorithms $\mathcal{A}$ (x, y) (where x belongs to the first algorithm and y belongs to the second algorithm) and random input z as

$$\mathrm{REAL}_{\Pi,\mathcal{A}(z)}(x,y) \overset{\mathrm{def}}{=} (\mathrm{OUTPUT}^{\Pi}(x,y), A_1(\mathrm{VIEW}_1^{\Pi}(x,y),z), A_2(\mathrm{VIEW}_2^{\Pi}(x,y),z)) \tag{6.17}$$

where

$$VIEW_1^{\Pi}(x,y) = (x, r_1, m_1, m_2, ..., m_t)$$

and

$$VIEW_2^{\Pi}(x,y) = (y, r_2, m_1, m_2, ..., m_t)$$

are views of algorithms A_1 and A_2, and they track each player's input (x and y), some randomness (r_1 and r_1), and all messages passed between the algorithms $(m_1, m_2, ..., m_t)$. The output is defined as

$$\mathrm{OUTPUT}^{\Pi}(x,y) = (\mathrm{OUTPUT}_1^{\Pi}(x,y), \mathrm{OUTPUT}_2^{\Pi}(x,y))$$

and is simply the output of the protocol Π for both parties.

In this model, it is assumed that at least one of the algorithms is honest and keeps only the value outputted by protocol Π. If this is the case, then we can claim that our protocol Π securely evaluates the function as defined by f in the ideal world and real-world models such that

$$\{\mathrm{IDEAL}_{f,\mathcal{B}(z)}(x,y)\}_{x,yz} \overset{c}{\equiv} \{\mathrm{REAL}_{\Pi,\mathcal{A}(z)}(x,y)\}_{x,y,z}$$

In other words, no polynomial bounded adversarial algorithm can distinguish between the real-world execution and the ideal world execution. This is another way of saying they are the same, statistically speaking.

The main goal of this framework is to formally define what is allowed in protocol in the semi-honest model. To summarise, each player who is honest follows the protocol Π correctly and only outputs the result of the protocol, and destroying all other data. While a semi-honest player will follow the protocol correctly, but at the conclusion of the protocol Π will try to infer the information based on all of the messages passed during the protocol and their own input. This is equivalent to the definition of a passive adversary.

We will now show that when our protocol executing in the real world under the semi-honest model, then an adversary who corrupts one (and only one party) is unable to attain distinguish between a real execution of the protocol and a simulated ideal world execution. If this is the case, and we have this equivalence, then the adversary cannot gain any more information, under the assumption that the ideal world execution is secure (by definition).

With respect to our protocol, we will consider two parties, Alice and Bob, who desire to compute the sign bit of the difference of their inputs x and y. In this setup, we will designate Bob to compute this function and so Alice will generate a secret key *sk* and public key *pk*. Alice sends Bob her public key. We reasonably assume that Alice does not send data to Bob, unless encrypting it first. Using the indistinguishability of the real-world and ideal world as outlined above we will show that an adversary cannot learn any more than what the protocol allows and hence is unable to distinguish between a real-world execution and an ideal world. We will consider the only two possible adversarial attacks in the semi-honest model: when Alice A is corrupted; and when Bob B is corrupted.

- *Attack 1 (A corrupted):* In our protocol, A encrypts her input and sends it to B. After computation, B sends the encrypted result τ, where τ is the encryption, under A's key, of the most significant bit of the difference of both inputs x and y. A is able to decrypt this number to reveal either a 0 or 1. But since the number τ is random, A is unable to distinguish it from an ideal world τ'. See Figure 6.4.
- *Attack 2 (B is corrupted):* Since B is computing on A's encrypted input, B can store this for future analysis. This places great reliance on the hardness

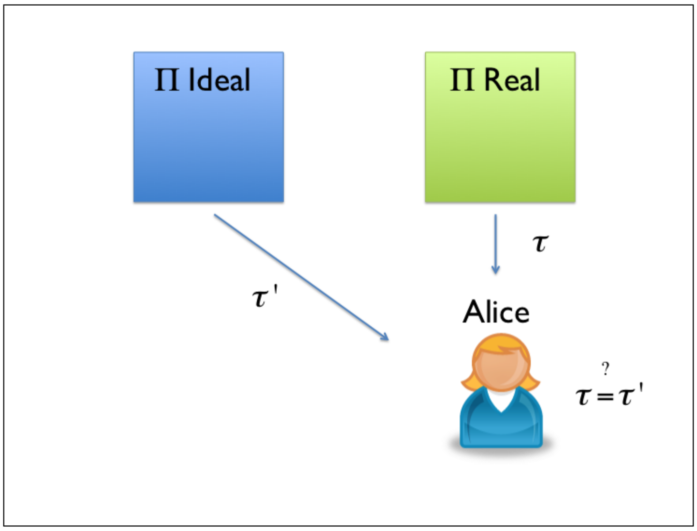


Figure 6.4: Adversary A is unable distinguish τ and τ'

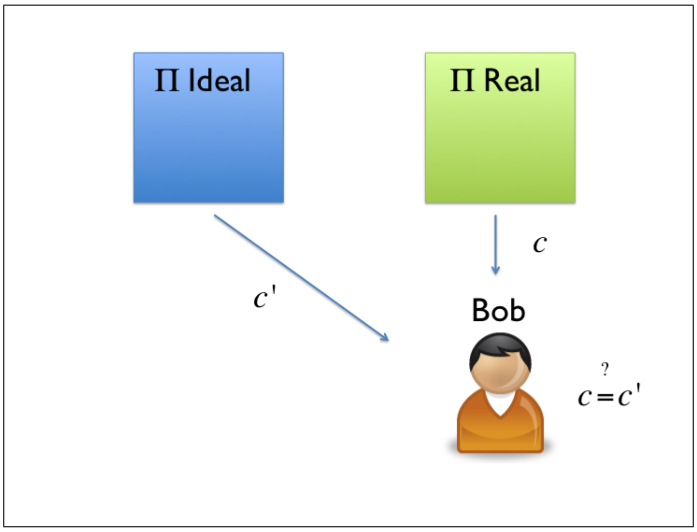


Figure 6.5: Adversary B is unable distinguish c and c'

of the underlying assumption of the homomorphic encryption scheme. If the assumption is sufficiently hard, then B will be unable to determine if the encrypted values came from a ideal world implementation or a real-world implementation. For simplicity call the encrypted value in the real-world c and c' in the ideal world. See Figure 6.5.

We conclude this section by claiming that both the A and B inputs are secure in the semi-honest model if an adversary that corrupts either A or B (but not both), cannot distinguish between a real execution of the protocol and an ideal execution of the protocol. Based on our analysis, we satisfy this condition, and hence our solution is secure in the semi-honest model.

6.6 Performance Analysis

In this section, we will compare the performance of our proposed solution with that of Yao's garbled circuit [103]. Before we analyse the performance of each solution, we will describe the operation of Yao's garbled circuit with respect to the association rule mining application.

6.6.1 Integer Comparison with Yao's Garbled Circuit

The two approaches, fully homomorphic encryption and Yao's Garbled Circuit, can be used as a fundamental building block to compute Equation 6.12. In Yao's approach we construct the full adder, for Alice's X and Bob's Y (Bob submits $\overline{Y} + 1$, which negates Y), where $X, Y \in \mathbb{Z}$. We will consider the comparison between two 4-bit integers. Figure 6.6 displays a circuit for this case. Obviously this circuit can be simplified to reduce the computational cost, but any improvement in the circuit will mean an improvement in both methods. So we will use this form to compare the two approaches of comparing two integers.

Using Yao's approach to compute this circuit requires Alice to generate a garbled truth table for every gate in the circuit. After Alice has generated the garbled truth tables for the circuit, she transfers this information to Bob. Bob then obtains the initial keys for the gate at the leaf notes of the circuit. Bob can then proceed to compute the circuit privately. Finally, Bob computes the output of the final gate in the circuit and transfers this to Alice who can decode the number as a 0 or a 1. This completes one round of Yao's garbled circuit for comparing two numbers.

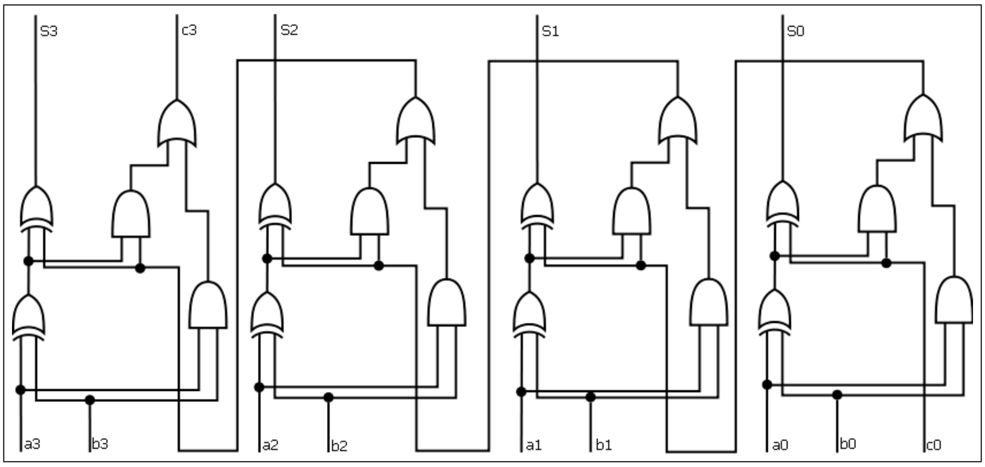


Figure 6.6: Circuit to compute the sum of two 4-bit integers

When using Yao's solution to execute the association rule mining algorithm, there are a number of weaknesses that affect performance. The following lists and describes the main weaknesses with respect to the association rule mining application.

New Garbled Table

To ensure a high degree of privacy and security when using Yao's garbled circuit to compare two numbers, a new garbled circuit must be generated. There are two main causes for this concern. Firstly, Bob may use the keys acquired in the previous round of the protocol to discover additional information in the current round. This is a major security risk, since Bob may be able to learn Alice's input. Secondly, if Bob submits the same key (at the end of the protocol) back to Alice for two comparisons, she may reach the conclusion that Bob's input was the same and make inferences. Hence, it is recommended to generate a new garbled circuit for each comparison. This is a major overhead since in the association rule mining algorithm, it is not unusual to compare thousands or tens of thousands of numbers.

Chance of Error

After receiving two keys for a gate in the circuit, Bob does not know which row will result in the correct answer. Hence, Bob must decrypt all of the rows and choose the key that falls into some predefined range for keys in the garbled circuit. This means that one and only one decryption leads to the key as output. Therefore, there is a chance for error, even if the chance is negligible. In the association rule mining algorithm, it is required to compare two numbers repeatedly and this will result in a higher chance of failure in the algorithm. Most certainly, Alice can generate a garbled circuit that is free of this problem. Although this leads to a greater overhead for generating a garbled circuit.

Oblivious Transfer Overhead

Oblivious transfer is known to have high computational overhead for the result it provides [85, 33]. It is an example of a security trade-off. The higher security means greater overhead. Although it is used in Yao's garbled circuit algorithm to receive the keys at the leaf nodes, any way to remove this requirement will be beneficial for the association rule mining application, since this will reduce the communication and computation overhead.

6.6.2 Performance Comparison

We will now analyse the performance of comparing two integers using either the fully homomorphic encryption approach or Yao's garbled circuit approach. We will perform analysis with respect to communication, storage and computation.

Communication

Yao's garbled circuit requires that a garbled circuit be transmitted from Alice to Bob for each comparison in the association rule mining algorithm, and this structure has to be renewed each time to maintain privacy of both Alice and Bob's input.

Additionally, to allow the circuit to operate correctly, it is required that oblivious transfer be used to obtain the initial keys representing Bob's input. This incurs additional communication cost.

In contrast, the Fully Homomorphic Encryption approach requires Alice to generate a public key for Bob to encrypt with. This public key can be several gigabytes in size to ensure security. However, this key needs only to be generated once and sent once. It can be reused multiple times without affecting the privacy of either Alice or Bob's inputs. The fully homomorphic encryption scheme has a large plaintext expansion factor (i.e. 1 bit in the plaintext expands into many thousand bits in the ciphertext). Although, this is comparable, possibly less than (depending on the choice of parameters), the communication required by the whole Yao's garbled circuit approach.

The required number of communication rounds also differs between the two approaches. If we ignore the initial stages in both techniques, we find that the garbled circuit method requires three rounds of communication per comparison. They include: a round for transferring the circuit; a round for using oblivious transfer for the inputs; and a round to broadcast the output. In contrast, the fully homomorphic encryption approach requires two rounds of communication: one round for transferring the encrypted inputs and one round to broadcast the output. This difference means we are saving one round of communication per comparison and since we are repeatedly comparing numbers, the saving is substantial.

Storage

For security reasons, we cannot permit Bob to reuse the garbled circuit multiple times. Hence, the storage required by the garbled circuit approach is only the amount of memory that contains the current garbled circuit. Any other data pertaining to the execution of the algorithm must be discarded.

The fully homomorphic encryption on the other hand requires that Bob store a huge public key many gigabytes in size. However, since this can be used multiple times, the storing of such a large amount of data for a public key is justified.

Obviously, this public key size is dependent on the security parameter of the fully homomorphic encryption scheme.

Computation

In terms of computation, Yao's garbled circuit approach is very symmetric. Meaning both parties, Alice and Bob, are required to perform approximately the same amount of computation. Alice needs to generate a garbled circuit and Bob needs to evaluate it.

In the fully homomorphic encryption approach, Alice generates a one time public key and sends it to Bob. During execution of the protocol, Alice encrypts her data and sends it to Bob who evaluates the circuit homomorphically. This computation is very asymmetric, since encryption and decryption is very fast, while evaluating the circuit is the major bottleneck of the comparison [40]. Hence, it is possible to speed up the association rule mining algorithm, by splitting the number of comparisons in half and having each party evaluate their respective halves.

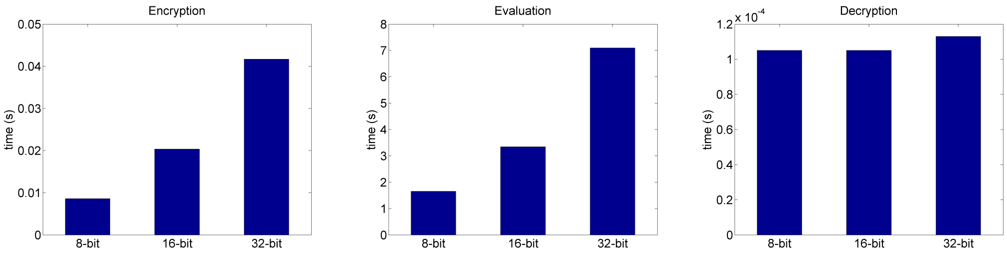
6.6.3 Summary of Performance Analysis

The main problem with applying Yao's solution in the association rule mining application is that Yao's solution was designed for a single execution of a circuit. It was not designed to repeatedly execute the same circuit with different inputs. Of course this limitation can be solved by representing the association rule mining algorithm as a very large boolean circuit. However, this overhead will make the execution very impractical, since a large volume of data would need to be computed and stored. This is compounded by the fact that the general association rule mining algorithm is very recursive. That is, the output of the current round is dependent on the previous round, leading to redundancies in the garbled circuit. Based on the analysis in this section, we conclude that the fully homomorphic encryption solution provides a more efficient solution compared with Yao's garbled solution.

6.7 Experimental Evaluation

We implemented a software prototype to test the feasibility of our approach. The prototype was executed on a machine with a 3.40GHz Intel Core i7-2600 with 16GB of memory, running the Linux (3.1.0-1.2) operating system. We used an open source library[2] of the Smart-Vercauteren fully homomorphic encryption scheme [93] to enable the cryptographic operations. Using this library we were able to measure the time required for the integer comparison method, which was required for the association rule mining application.

We separated the comparison method into three logical stages of computation, which include: encryption, evaluation and decryption. The data or count value was represented as a vector of bits, which was encrypted by the fully homomorphic encryption scheme. Two integers, represented as bit vectors, were encrypted. In the case of the second integer, the number was converted to Two's Complement before encryption. Next, the addition circuit was applied to the ciphertext that resulted in a single encrypted bit. Finally, this was decrypted to reveal the output of the circuit. We used different bit vector lengths to obtain an overall view of performance, which included 8, 16 and 32. The timings of encryption, evaluation and decryption are shown in Figures 6.7a, 6.7b, and 6.7c, respectively.



(a) Encryption time **(b)** Evaluation time **(c)** Decryption time

Figure 6.7: Execution time of fully homomorphic encryption experiment

These timing results greatly promote the significance of our solution. Apart from the key generation method, there is no other component required to make

[2] http://www.hcrypt.com/

this solution complete. Once we have the encryption and decryption keys and the encrypted data, we can evaluate the data according to our circuit and obtain the result. Contrasting this with Yao's approach, we find that generating the garbled circuit itself is not a complete solution to the privacy preserving association rule mining problem. We also need to allow for many executions of oblivious transfer, which incurs great communication cost. Plus, the garbled circuit must be refreshed each time. For more discussion about comparing the performance of our approach using FHE with Yao's garbled circuit approach, see Section 6.6.

In the current start-of-the-art fully homomorphic encryption techniques it is difficult to say how secure the cryptosystem is, since the underlying hardness assumptions are less understood than classical assumptions like RSA or ElGamal. More research is required to test the foundations of the hardness assumptions to adequately determine what system parameters are required.

6.8 Conclusion and Recommendations

In this chapter, a two party privacy preserving association rule mining algorithm was presented, which used new techniques in homomorphic encryption [38, 93, 98]. The protocol was shown to be secure under the semi-honest model of multi-party computation. The security analysis is based on the hardness assumption of the encryption scheme.

The main contribution of this chapter is the use of fully homomorphic encryption to solve the privacy preserving association rule problem. Previous efforts either used some combination of homomorphic encryption and data perturbation, or a Yao's Garbled Circuit based approach. The former approach lead to trade-offs between accuracy and privacy, while the latter had high communication cost since the generated circuit could not be reused. Our solution does not have the privacy-accuracy trade-off, and once the public key has been transferred it can be reused numerous times.

Future work regarding privacy preserving data mining would include: improving the efficiency by removing unnecessary communication; expanding on the number of parties to a multi-party computation interaction; and applying the fully homomorphic encryption system to other data mining algorithms. More fundamentally, further work is also required to improve both the efficiency and security of the underlying cryptosystem.

Chapter 7

Conclusion

Here we summarise the main contributions of this dissertation. After doing so, we will highlight the common theme that is present throughout this thesis, which is to construct privacy preserving protocols for various applications. The purpose of this chapter is to link them all together and ascertain the overall meaning.

7.1 Problem Statement Review

Before we cover the contributions, we recall the general problem statement given in the introduction. This problem statement is as follows: develop and examine privacy preserving protocols for various real-world problems. This can be expressed by the question: can we develop protocols that both give the correct result while maintaining the privacy of the respective parties? This thesis has answered this question by giving numerous applications. We now summarise these contributions and present a big picture view by linking them together.

7.2 Contributions Overview

This thesis studies the problem at three levels: theory, implementation and practice. They are arranged in a hierarchy as you would expect (see Figure 7.1). To design a

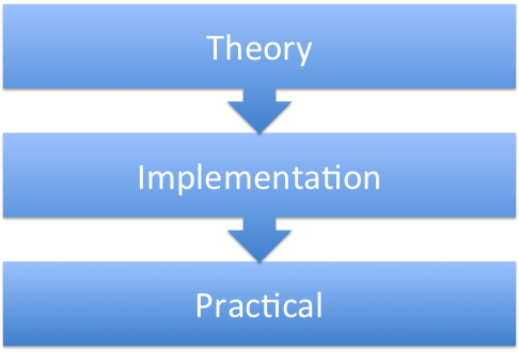


Figure 7.1: Illustrating the hierarchy of contributions

protocol, indeed a privacy preserving protocol, we must begin with theory. Without the theory, there would be no clear way to determine the correctness of a protocol. There would be no test we could apply to determine if we are right or wrong.

Next comes implementation. With the definitions clearly specified by the theory, we must deal with the implementation related requirements. These include: system hardware (CPU and RAM), and programming language and operating system. We must take into account what features are made available by the system and use them accordingly.

At the highest level we consider practical issues. At this level, the main question is whether the operation of the protocol is within acceptable limits. Essentially this means setting the system parameters such that, while ensuring security, the protocol is not rendered impractical. We will now enumerate the contributions made by this thesis, with respect to this hierarchy.

The contributions begin with looking at the security properties of a somewhat homomorphic encryption scheme (see Chapter 3). As discussed, this is a fundamental ingredient for constructing a fully homomorphic scheme according to current theory. We cannot use a cryptographic scheme if there are problems with the security reduction. Furthermore, if we do not have a secure somewhat homomorphic encryption scheme, then it is impossible to bootstrap it to realise a secure fully homomorphic scheme.

Next we consider how to construct a single-database private information retrieval scheme using the ideas introduced by pursuit of fully homomorphic encryption techniques (see Chapter 4). More specifically, what can be achieved if we are allowed both addition and multiplication on ciphertexts, instead of only one. Since the operations that permit this are simple, this leads to computationally efficient private information retrieval. This is as opposed to communicationally efficient private information retrieval, which was the original intention of the technology. The significance of this is highlighted by the fact we are reaching the limit of computational power. On the other hand, communication is ultimately limited by the speed of light.

We then apply these cryptographic techniques to provide a private location based query solution (see Chapter 5). This is realised by combining oblivious transfer with private information retrieval. The two isolated techniques are executed in tandem to produce a secure solution for both parties. The oblivious transfer protocol is used to obtain a key, while the private information retrieval is used to obtain the data that the key encrypts. Keeping in mind that the client in the protocol is using an underpowered device, like a mobile phone, we design a protocol that is both secure and practical.

Next we develop a privacy preserving data mining protocol that focuses on generating association rules for data stored on two independent sites (see Chapter 6). We use the properties of a homomorphic encryption scheme that permits addition and multiplication on ciphertext to evaluate an addition circuit. This addition circuit enables the comparison of two integers, which is necessary in the Apriori association rule mining algorithm. This leads to a better solution, compared with Yao's Garbled Circuit, because it promotes better reuse, as a garbled circuit would have to be generated each time there is an integer comparison. Additionally, the garbled circuit approach requires many executions of a oblivious transfer protocol that incurs great communication overhead.

Of the four contributions made by this dissertation, only the first one is purely theoretical, while the other three make contributions at all three levels of our hierarchy. Furthermore, for each contribution the results from an experimental implementation has been provided. The purpose of these implementation results is to demonstrate that the approaches used are within reasonable practical limits. Clearly, each approach is within these bounds. The implementation results depend on the parameter sizes used in the experiments. There is explicit control over these experimental parameters, for example the ciphertext size, such that the experiments are repeatable and therefore the results are reliable. Of course each of the results are with respect to the machine that each experiment was executed.

7.3 The Big Picture

The big picture is that secure protocols will always incur an overhead that is absolutely not required by the application. Indeed, the applications explored by this dissertation (and many others) can operate correctly without any need for privacy techniques. But as computers have become more sophisticated, the cryptographic methods for privacy have become required by many applications where the data is deemed important.

The real question is what kind of overhead will these added steps incur? Will the added functionality add computational overhead, communicational overhead, or a combination of the two? This is a really subjective question because to answer it would require a deep knowledge of the application domain, and what system features are available. Obviously, these vary considerably from application to application, and from system to system. So, we must make the best choices for what is required.

7.4 Future Directions

The breakthrough work of fully homomorphic encryption has made a huge impact on the cryptographic research community. Although, a construction of a system that permits the endless computation on encrypted data seems to be too inefficient for any practical use. For some applications, endless computation is not required because there is a known stopping point. So, there is no need to refresh the ciphertext.

This results in future research in two distinct, but not independent, areas: application and theory. We can find more applications for fully homomorphic encryption techniques. At the more fundamental level, we can explore the difficulty of the newly introduced hardness assumptions, and what is required to make them exponentially hard. However, we must keep in mind the performance implications. Otherwise, the overhead required might outweigh the benefits.

Colophon

This thesis was made in LaTeX 2$_\varepsilon$ using the “hepthesis” class.

References

[1] "Openssl," http://www.openssl.org/, 2011, [Online; accessed 7-July-2011].

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[3] C. Aguilar-Melchor and P. Gaborit, "A lattice-based computationally-efficient private information retrieval protocol," in *WEWORC 2007*, 2007.

[4] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Advances in Cryptology CRYPTO 92*, ser. Lecture Notes in Computer Science, E. Brickell, Ed. Springer Berlin Heidelberg, 1993, vol. 740, pp. 390–420.

[5] M. Bellare and S. Micali, "Non-interactive oblivious transfer and applications," in *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology.* London, UK: Springer-Verlag, 1990, pp. 547–557.

[6] J. Benaloh, "Dense probabilistic encryption," in *In Proceedings of the Workshop on Selected Areas of Cryptography*, 1994, pp. 120–128.

[7] A. Beresford and F. Stajano, "Location privacy in pervasive computing," *Pervasive Computing, IEEE*, vol. 2, no. 1, pp. 46 – 55, 2003.

[8] C. Bettini, X. Wang, and S. Jajodia, "Protecting privacy against location-based personal identification," in *Secure Data Management*, ser. Lecture Notes in Computer Science, W. Jonker and M. Petkovic, Eds. Springer Berlin / Heidelberg, 2005, vol. 3674, pp. 185–199.

[9] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Advances in Cryptology - CRYPTO 2001*, ser. Lecture Notes in Computer Science, J. Kilian, Ed. Springer Berlin / Heidelberg, 2001, vol. 2139, pp. 213–229.

[10] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-dnf formulas on ciphertexts,” in *Theory of Cryptography*, ser. Lecture Notes in Computer Science, J. Kilian, Ed. Springer Berlin / Heidelberg, 2005, vol. 3378, pp. 325–341.

[11] Z. Brakerski and V. Vaikuntanathan, “Efficient fully homomorphic encryption from (standard) lwe,” in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, oct. 2011, pp. 97 –106.

[12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS ’12. New York, NY, USA: ACM, 2012, pp. 309–325.

[13] G. Brassard, C. Crepeau, and M. Santha, “Oblivious transfers and intersecting codes,” *Information Theory, IEEE Transactions on*, vol. 42, no. 6, pp. 1769 –1780, nov 1996.

[14] G. Brassard, C. Crepeau, and J.-M. Robert, “All-or-nothing disclosure of secrets,” in *Advances in Cryptology - CRYPTO86*, ser. Lecture Notes in Computer Science, A. Odlyzko, Ed. Springer Berlin / Heidelberg, 1987, vol. 263, pp. 234–238.

[15] M. Brenner, H. Perl, and M. Smith, “Practical applications of homomorphic encryption,” in *Proceedings of the International Conference on Security and Cryptography (SECRYPT 2012)*, 2012, pp. 5–14.

[16] P. Bunn and R. Ostrovsky, “Secure two-party k-means clustering,” in *Proceedings of the 14th ACM conference on Computer and communications security*, ser. CCS ’07. New York, NY, USA: ACM, 2007, pp. 486–497.

[17] C. Cachin, S. Micali, and M. Stadler, “Computationally private information retrieval with polylogarithmic communication,” in *Advances in Cryptology*

- *EUROCRYPT99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin / Heidelberg, 1999, vol. 1592, pp. 402–414.

[18] Y.-C. Chang, "Single database private information retrieval with logarithmic communication," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science, H. Wang, J. Pieprzyk, and V. Varadharajan, Eds. Springer Berlin / Heidelberg, 2004, vol. 3108, pp. 50–61.

[19] X. Chen and J. Pang, "Measuring query privacy in location-based services," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, ser. CODASPY '12. New York, NY, USA: ACM, 2012, pp. 49–60.

[20] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, Oct. 1995, pp. 41 –50.

[21] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[22] C.-K. Chu and W.-G. Tzeng, "Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries," in *Public Key Cryptography - PKC 2005*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed. Springer Berlin Heidelberg, 2005, vol. 3386, pp. 172–183.

[23] ——, "Efficient k-out-of-n oblivious transfer schemes," *Journal of Universal Computer Science*, vol. 14, no. 3, pp. 397–415, feb 2008.

[24] C. Clifton, M. Kantarcioglu, and J. Vaidya, "Defining privacy for data mining," in *in National Science Foundation Workshop on Next Generation Data Mining*, 2002, pp. 126–133.

[25] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology CRYPTO 2011*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed. Springer Berlin / Heidelberg, 2011, vol. 6841, pp. 487–504.

[26] C. Crepeau, "Equivalence between two flavours of oblivious transfers," in *Advances in Cryptology CRYPTO 87*, ser. Lecture Notes in Computer Science, C. Pomerance, Ed. Springer Berlin Heidelberg, 1987, vol. 293, pp. 350–354.

[27] M. L. Damiani, E. Bertino, and C. Silvestri, "The probe framework for the personalized cloaking of private locations," *Trans. Data Privacy*, vol. 3, no. 2, pp. 123–148, Aug. 2010.

[28] G. Di Crescenzo, T. Malkin, and R. Ostrovsky, "Single database private information retrieval implies oblivious transfer," in *Advances in Cryptology - EUROCRYPT 2000*, ser. Lecture Notes in Computer Science, B. Preneel, Ed. Springer Berlin / Heidelberg, 2000, vol. 1807, pp. 122–138.

[29] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644 – 654, nov 1976.

[30] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Pervasive Computing*, ser. Lecture Notes in Computer Science, H. Gellersen, R. Want, and A. Schmidt, Eds. Springer Berlin / Heidelberg, 2005, vol. 3468, pp. 243–251.

[31] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptographic protocols : A survey," 2004.

[32] T. ELGAMAL, "A public key cryptosystem and a signature scheme based on discrete logarithms," New York, NY, ETATS-UNIS, 1985.

[33] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[34] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," *Information Systems*, vol. 29, no. 4, pp. 343 – 364, 2004, knowledge Discovery and Data Mining (KDD 2002).

[35] FIPS-PUB.197, "Advanced encryption standard," *Federal Information Processing Standards Publications, US Department of Commerce/N.I.S.T., National Technical Information Service*, 2001.

[36] FIPS-PUB.46, “Data encryption standard,” *National Bureau of Standards, US Department of Commerce*, 1977.

[37] B. Gedik and L. Liu, “Location privacy in mobile systems: A personalized anonymization model,” in *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, 2005, pp. 620 –629.

[38] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *STOC ’09: Proceedings of the 41st annual ACM symposium on Theory of computing.* New York, NY, USA: ACM, 2009, pp. 169–178.

[39] ——, “Computing arbitrary functions of encrypted data,” *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[40] C. Gentry and S. Halevi, “Implementing gentrys fully-homomorphic encryption scheme,” in *Advances in Cryptology EUROCRYPT 2011*, ser. Lecture Notes in Computer Science, K. Paterson, Ed. Springer Berlin / Heidelberg, 2011, vol. 6632, pp. 129–148.

[41] C. Gentry, S. Halevi, and N. Smart, “Fully homomorphic encryption with polylog overhead,” in *Advances in Cryptology EUROCRYPT 2012*, ser. Lecture Notes in Computer Science, D. Pointcheval and T. Johansson, Eds. Springer Berlin / Heidelberg, 2012, vol. 7237, pp. 465–482.

[42] C. Gentry and Z. Ramzan, “Single-database private information retrieval with constant communication rate,” in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, L. Caires, G. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds. Springer Berlin / Heidelberg, 2005, vol. 3580, pp. 803–815.

[43] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, “Protecting data privacy in private information retrieval schemes,” in *STOC ’98: Proceedings of the thirtieth annual ACM symposium on Theory of computing.* New York, NY, USA: ACM, 1998, pp. 151–160.

[44] G. Ghinita, P. Kalnis, M. Kantarcioglu, and E. Bertino, “A hybrid technique for private location-based queries with database protection,” in *Advances in*

Spatial and Temporal Databases, ser. Lecture Notes in Computer Science, N. Mamoulis, T. Seidl, T. Pedersen, K. Torp, and I. Assent, Eds. Springer Berlin / Heidelberg, 2009, vol. 5644, pp. 98–116.

[45] ——, “Approximate and exact hybrid algorithms for private nearest-neighbor queries with database protection,” *GeoInformatica*, pp. 1–28, 2010.

[46] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, “Private queries in location based services: anonymizers are not necessary,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD ’08. New York, NY, USA: ACM, 2008, pp. 121–132.

[47] O. Goldreich, *The Foundations of Cryptography, Basic Applications*, 1st ed. Cambridge University Press, 2004.

[48] ——, “Secure multi-party computation,” 1998.

[49] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270 – 299, 1984.

[50] M. Gruteser and D. Grunwald, “Anonymous usage of location-based services through spatial and temporal cloaking,” in *Proceedings of the 1st international conference on Mobile systems, applications and services*, ser. MobiSys ’03. New York, NY, USA: ACM, 2003, pp. 31–42.

[51] J. Han, *Data Mining: Concepts and Techniques.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[52] Harris and David, *Digital Design and Computer Architecture : From Gates to Processors.* Elsevier, Burlington 2007, 2007.

[53] T. Hashem and L. Kulik, “Safeguarding location privacy in wireless ad-hoc networks,” in *UbiComp 2007: Ubiquitous Computing*, ser. Lecture Notes in Computer Science, J. Krumm, G. Abowd, A. Seneviratne, and T. Strang, Eds. Springer Berlin / Heidelberg, 2007, vol. 4717, pp. 372–390.

[54] B. Hoh and M. Gruteser, “Protecting location privacy through path confusion,” in *Security and Privacy for Emerging Areas in Communications Networks,*

2005. SecureComm 2005. First International Conference on, 2005, pp. 194 – 205.

[55] N. Howgrave-Graham, “Approximate integer common divisors,” in *Cryptography and Lattices*, ser. Lecture Notes in Computer Science, J. Silverman, Ed. Springer Berlin / Heidelberg, 2001, vol. 2146, pp. 51–66.

[56] A. Joux, “A one round protocol for tripartite diffiehellman,” in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science, W. Bosma, Ed. Springer Berlin / Heidelberg, 2000, vol. 1838, pp. 385–393.

[57] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, “Preventing location-based identity inference in anonymous spatial queries,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 12, pp. 1719 –1733, 2007.

[58] M. Kantarcioglu and C. Clifton, “Privacy-preserving distributed mining of association rules on horizontally partitioned data,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 9, pp. 1026–1037, 2004.

[59] M. G. Kaosar, R. Paulet, and X. Yi, “Secure two-party association rule mining,” in *Australasian Information Security Conference (AISC 2011)*, ser. CRPIT, C. Boyd and J. Pieprzyk, Eds., vol. 116. Perth, Australia: ACS, 2011, pp. 15–22.

[60] ——, “Fully homomorphic encryption based two-party association rule mining,” *Data & Knowledge Engineering*, vol. 7678, no. 0, pp. 1 – 15, 2012.

[61] H. Kido, Y. Yanagisawa, and T. Satoh, “An anonymous communication technique using dummies for location-based services,” in *Pervasive Services, 2005. ICPS ’05. Proceedings. International Conference on*, 2005, pp. 88 – 97.

[62] J. Kilian, “Founding crytpography on oblivious transfer,” in *STOC ’88: Proceedings of the twentieth annual ACM symposium on Theory of computing.* New York, NY, USA: ACM, 1988, pp. 20–31.

[63] J. Krumm, “A survey of computational location privacy,” *Personal and Ubiquitous Computing*, vol. 13, pp. 391–399, 2009.

[64] E. Kushilevitz and R. Ostrovsky, “Replication is not needed: single database, computationally-private information retrieval,” in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, oct 1997, pp. 364 –373.

[65] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” in *Advances in Cryptology CRYPTO 2000*, ser. Lecture Notes in Computer Science, M. Bellare, Ed. Springer Berlin Heidelberg, 2000, vol. 1880, pp. 36–54.

[66] L. Marconi, R. Di Pietro, B. Crispo, and M. Conti, “Time warp: How time affects privacy in lbss,” in *Information and Communications Security*, ser. Lecture Notes in Computer Science, M. Soriano, S. Qing, and J. Lpez, Eds. Springer Berlin / Heidelberg, 2010, vol. 6476, pp. 325–339.

[67] S. Mascetti and C. Bettini, “A comparison of spatial generalization algorithms for lbs privacy preservation,” in *Mobile Data Management, 2007 International Conference on*, May 2007, pp. 258 –262.

[68] C. Melchor, B. Crespin, P. Gaborit, V. Jolivet, and P. Rousseau, “High-speed private information retrieval computation on gpu,” in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE ’08. Second International Conference on*, aug. 2008, pp. 263 –272.

[69] C. Melchor and P. Gaborit, “A fast private information retrieval protocol,” in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, july 2008, pp. 1848 –1852.

[70] R. Merkle and M. Hellman, “Hiding information and signatures in trapdoor knapsacks,” *Information Theory, IEEE Transactions on*, vol. 24, no. 5, pp. 525–530, 1978.

[71] V. S. Miller, “The weil pairing, and its efficient calculation,” *Journal of Cryptology*, vol. 17, pp. 235–261, 2004.

[72] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, “The new casper: query processing for location services without compromising privacy,” in *Proceedings of the 32nd international conference on Very large data bases*, ser. VLDB ’06. VLDB

Endowment, 2006, pp. 763–774.

[73] D. Naccache and J. Stern, “A new public key cryptosystem based on higher residues,” in *Proceedings of the 5th ACM conference on Computer and communications security*, ser. CCS ’98. New York, NY, USA: ACM, 1998, pp. 59–66.

[74] M. Naor and B. Pinkas, “Oblivious transfer and polynomial evaluation,” in *STOC ’99: Proceedings of the thirty-first annual ACM symposium on Theory of computing.* New York, NY, USA: ACM, 1999, pp. 245–254.

[75] ——, “Oblivious transfer with adaptive queries,” in *Advances in Cryptology CRYPTO 99*, ser. Lecture Notes in Computer Science, M. Wiener, Ed. Springer Berlin / Heidelberg, 1999, vol. 1666, pp. 791–791.

[76] ——, “Efficient oblivious transfer protocols,” in *SODA ’01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001, pp. 448–457.

[77] T. Okamoto and S. Uchiyama, “A new public-key cryptosystem as secure as factoring,” in *Advances in Cryptology EUROCRYPT’98*, ser. Lecture Notes in Computer Science, K. Nyberg, Ed. Springer Berlin Heidelberg, 1998, vol. 1403, pp. 308–318.

[78] F. Olumofin and I. Goldberg, “Revisiting the computational practicality of private information retrieval,” in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, G. Danezis, Ed. Springer Berlin Heidelberg, 2012, vol. 7035, pp. 158–172.

[79] R. Ostrovsky and W. Skeith, “A survey of single-database private information retrieval: Techniques and applications,” in *Public Key Cryptography PKC 2007*, ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds. Springer Berlin / Heidelberg, 2007, vol. 4450, pp. 393–411.

[80] W. Ouyang and Q. Huang, “Privacy preserving association rules mining based on secure two-party computation,” in *Intelligent Control and Automation*, ser. Lecture Notes in Control and Information Sciences, D.-S. Huang, K. Li, and

G. Irwin, Eds. Springer Berlin / Heidelberg, 2006, vol. 344, pp. 969–975.

[81] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Advances in Cryptology - EUROCRYPT99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin / Heidelberg, 1999, vol. 1592, pp. 223–238.

[82] B. Palanisamy and L. Liu, “Mobimix: Protecting location privacy with mix-zones over road networks,” in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, april 2011, pp. 494 –505.

[83] S. Pohlig and M. Hellman, “An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.),” *Information Theory, IEEE Transactions on*, vol. 24, no. 1, pp. 106 – 110, jan 1978.

[84] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[85] M. O. Rabin, “How to exchange secrets with oblivious transfer,” Cryptology ePrint Archive, Report 2005/187, 1981.

[86] R. Rivest, L. Adleman, and M. Dertouzos, “On data banks and privacy homomorphisms.” Academic Press, 1978, pp. 169–177.

[87] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[88] S. J. Rizvi and J. R. Haritsa, “Maintaining data privacy in association rule mining,” in *Proceedings of the 28th international conference on Very Large Data Bases*, ser. VLDB ’02. VLDB Endowment, 2002, pp. 682–693.

[89] T. Sander, A. Young, and M. Yung, “Non-interactive cryptocomputing for nc1,” in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999, pp. 554 –566.

[90] A. Shamir, A. Shamir, A. Shamir, and A. Shamir, “A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem,” in *Foundations*

of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on, 1982, pp. 145–152.

[91] V. Shoup, "Number thoery library," http://www.shoup.net/ntl/, [Online; accessed 7-July-2011].

[92] R. Sion and B. Carbunar, "On the computational practicality of private information retrieval," in *NDSS Symposium*, 2007.

[93] N. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Public Key Cryptography PKC 2010*, ser. Lecture Notes in Computer Science, P. Nguyen and D. Pointcheval, Eds. Springer Berlin / Heidelberg, 2010, vol. 6056, pp. 420–443.

[94] D. Stehle and R. Steinfeld, "Faster fully homomorphic encryption," in *Advances in Cryptology - ASIACRYPT 2010*, ser. Lecture Notes in Computer Science, M. Abe, Ed. Springer Berlin / Heidelberg, 2010, vol. 6477, pp. 377–394.

[95] J. Stern, "A new and efficient all-or-nothing disclosure of secrets protocol," in *Advances in Cryptology - ASIACRYPT98*, ser. Lecture Notes in Computer Science, K. Ohta and D. Pei, Eds. Springer Berlin / Heidelberg, 1998, vol. 1514, pp. 357–371.

[96] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, pp. 557–570, October 2002.

[97] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining.* Pearson Education, Inc., 2006.

[98] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology - EUROCRYPT 2010*, ser. Lecture Notes in Computer Science, H. Gilbert, Ed. Springer Berlin / Heidelberg, 2010, vol. 6110, pp. 24–43.

[99] C. Wieschebrink, "Two np-complete problems in coding theory with an application in code based cryptography," in *Information Theory, 2006 IEEE International Symposium on*, july 2006, pp. 1733 –1737.

[100] S. Wiesner, “Conjugate coding,” *SIGACT News*, vol. 15, no. 1, pp. 78–88, Jan. 1983.

[101] T. Xu and Y. Cai, “Feeling-based location privacy protection for location-based services,” in *Proceedings of the 16th ACM conference on Computer and communications security*, ser. CCS ’09. New York, NY, USA: ACM, 2009, pp. 348–357.

[102] H.-M. Yang, Q. Xia, X. fen Wang, and D. hua Tang, “A new somewhat homomorphic encryption scheme over integers,” in *Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference on*, march 2012, pp. 61 –64.

[103] A. C. Yao, “Protocols for secure computations,” in *Foundations of Computer Science, 1982. SFCS ’08. 23rd Annual Symposium on*, 3-5 1982, pp. 160 –164.

[104] X. Yi, “An identity-based signature scheme from the weil pairing,” *Communications Letters, IEEE*, vol. 7, no. 2, pp. 76 – 78, feb 2003.

[105] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, “Single-database private information retrieval from fully homomorphic encryption,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. PrePrints, 2012.

[106] X. Yi and Y. Zhang, “Privacy-preserving distributed association rule mining via semi-trusted mixer,” *Data & Knowledge Engineering*, vol. 63, no. 2, pp. 550 – 567, 2007.

[107] J. Zhan, S. Matwin, and L. Chang, “Privacy-preserving collaborative association rule mining,” *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 1216 – 1227, 2007.