uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Behzad Malek

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Efficient Private Information Retrieval

TITRE DE LA THÈSE / TITLE OF THESIS

Ali Miri

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Ahmed Karmouch

George Yee

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

EFFICIENT PRIVATE INFORMATION RETRIEVAL

A Thesis Submitted to
The Faculty of Graduate and Postdoctoral Studies
in Partial Fulfillment of The Requirements
for The Degree of
Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Behzad Malek

July 2005

Your file Votre référence
ISBN: 0-494-11336-7
Our file Notre référence
ISBN: 0-494-11336-7

Abstract

In this thesis, we study Private Information Retrieval and Oblivious Transfer, two strong cryptographic tools that are widely used in various security-related applications, such as private data-mining schemes and secure function evaluation protocols. The first non-interactive, secure dot-product protocol, widely used in private data-mining schemes, is proposed based on trace functions over finite fields. We further improve the communication overhead of the best, previously known Oblivious Transfer protocol from $\mathcal{O}((\log(n))^2)$ to $\mathcal{O}(\log(n))$, where n is the size of the database. Our communication-efficient Oblivious Transfer protocol is a non-interactive, single-database scheme that is generally built on Homomorphic Encryption Functions. We also introduce a new protocol that reduces the computational overhead of Private Information Retrieval protocols. This protocol is shown to be computationally secure for users, depending on the security of McEliece public-key cryptosystem. The total online computational overhead is the same as the case where no privacy is required. The computation-saving protocol can be implemented entirely in software, without any need for installing a secure piece of hardware, or replicating the database among servers.

Acknowledgments

I would like to thank my supervisor, Professor Ali Miri, for his support and countless help to finish this work. He was always patient and thoughtful on my work, and taught me the way to evaluate any academic work. I had the honor to study at the Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE), and to learn different and wide aspects of security from the highly qualified faculty of OCIECE.

I also thank my parents, my sisters and my brother for their love and emotional support that are indeed inspiring me to move on and conquer other summits of success in life. During my stay at Ottawa, I learned the most valuable lesson on how to find my way through the life. I will never forget where I started from.

Contents

List of Tables

List of Figures

Chapter 1

Overview

In this chapter, we introduce a very important primitive in cryptology that has a wide range of applications in privacy and secure function evaluations. This thesis is dedicated to Private Information Retrieval (PIR) and Oblivious Transfer (OT) protocols, two major cryptographic tools that have a wide variety of applications in security. There are many critical applications that can be implemented if efficient PIR/OT techniques are invented. We devote this chapter to motivations and definition of the problem. In Section 1.1, we begin with an introduction to the problem being studied here along with its various applications. We outline our contributions to the field in Section 1.2.

1.1 Introduction

Today, a large amount of data is stored over distributed networks among different and usually unknown servers. The sensitivity and importance of the stored data are so different that they should be studied from point of view of both servers and users.

The primary concern of servers is usually to protect their sensitive data (such as

credit card numbers and military information) in order not to give out more information than permitted to users. This problem has been studied rather exhaustively in the literature. Passwords, cryptographic keys and access control lists are a few solutions that are usually used.

In some other applications, the user's particular interest in a subset of data may be sensitive and should be protected even if the data being requested by the user – such as geographic maps and weather forecasts – is not sensitive in nature. For example, consider the case where the ministry of defense of a country is searching for geographic maps and weather forecasts of the enemy's country. From this information, the database server can infer that there may be a military mission in that region. Another important example is in online media trading, where knowledge of users' preferences is an asset of well-recognized importance. The assumption that the server will not exploit and sell users' preferences is not always valid, so users may want to hide their preferences from the server in online transactions. Patent-database inquiry is another example in which one has to check the patent-database to ensure the novelty of the idea. However, the search through the database must not reveal any useful information about the idea to the server. In such applications, users need to get access to some data from a database while hiding their queries from the database holder.

A Private Information Retrieval (PIR) protocol is a cryptographic technique that allows users to privately receive data from a database while keeping their query hidden from the database holder. There has been an extensive research on this topic in literature [25, 4, 5, 30, 8, 19, 22, 34]. Nevertheless, the major focus of the work to date has been to minimize the communication complexity of PIR protocols. A closely related term, which is also used to privately collect data, is Oblivious Transfer (OT). In an OT protocol, not only must the user be secured but also security of the database holder should be guaranteed. The primary concern of the server is not to reveal more information than what is allowed to the end user. For instance the user of an online

music center may be permitted to download one MP3 file, but should not be allowed to access more than one song. OT protocols are usually studied in the same literature as PIR protocols, and sometimes used interchangeably [24, 32, 33, 34, 41].

As the privacy of users becomes more and more important, different applications of private data-retrieval abound and justify the need for efficient protocols. Besides direct applications in private data-collecting, PIR/OT protocols are also used as the building block of Secure Function Evaluation protocols (SFE). Informally, an SFE allows two or more parties to jointly compute a function on their inputs while keeping the inputs secret from each other – for more information on SFE protocols we refer the reader to [1, 3, 9, 7, 8, 10, 29, 16, 43, 44]. The SFE protocol prevails in many vital applications in security, and a practical SFE solution will solve many of the challenges in security, such as Digital Rights Management [38], Mobile Agents' security [20], private data-mining problems [22]. These problems will be solved if an application (function) can be protected on a remote host from local attackers who usually have full control over the execution environment [35, 18, 39]. This is still a challenging problem and an on-going research area that will be answered if an efficient SFE protocol exists.

The main component of many SFE protocols is the PIR/OT block. To illustrate how a PIR/OT protocol can be applied in an SFE scheme, assume that a software code is sent over to a remote user to run the code. If the code is represented by function $f(x, y)$, the variable x and y are the user's and server's secret inputs respectively. In the SFE protocol, the user has to compute $f(x, y)$ without learning the server's secret (y). The server, first, hard-wires his input y into the function and generates $f_y(x)$. Cachin *et al.* [20] proposed a very general method to translate any function $f_y(x)$ into a binary circuit C, which returns the encrypted output of the circuit for every possible binary input (x). The server owns the set of keys for all encrypted outputs denoted by

$$K = \{(k_{1,0}, k_{1,1}), (k_{2,0}, k_{2,1}), \cdots, (k_{n_x,0}, k_{n_x,1})\},$$

where n_x is the bit-length of x, and the key $k_{i,j}$ corresponds to the key used for the i'th bit of x, which equals j. Then, the remote user has to run an OT protocol to get the corresponding key of her input (x), without the server learning any information about x. Using an OT protocol, the user learns only the key that corresponds to her input, while the server cannot learn what key was retrieved by the user – for full details of the algorithm refer to [20, 44]. Consequently, any further progress in PIR/OT protocols proportionally improves the efficiency of SFE protocols and related work.

PIR/OT protocols are also useful in some other critical applications in security, such as privacy preserving data-mining protocols, where mining and integrating data from multiple sources have to be performed in a private manner. Du and Atallah [13] proposed a private data-mining scheme with the help of an OT protocol. In [14], a secure-dot product protocol, that is mostly used in data-mining algorithms, is proposed in which one needs to execute several OT protocols. All the examples given above show importance and possible use of PIR/OT protocols in various fields of security. There may also be some other applications built on PIR/OT protocols if a very efficient way of implementing PIR/OT protocols, which preserves both communication and computation, is invented. Thus, we dedicate this thesis to PIR/OT protocol, and further focus on achieving optimal protocols that can be used in practice.

1.2 Contributions

We summerize the main contributions of this thesis as follows:

- Designing the first non-interactive, secure dot-product protocol,
- Reducing the communication complexity in PIR/OT protocols from super-logarithmic to logarithmic in the size of the database,
- Designing a new approach to reduce the total online computation overheads in

PIR/OT protocols by pre-processing and using proxies.

In Chapter 2, we first describe the necessary cryptographic preliminaries, and formally define the security of PIR/OT protocols. In Chapter 3, we propose the first non-interactive, secure dot-product protocol in which both parties, sender and receiver, are information-theoretically secure. The proposed secure dot-product protocol is so efficient in terms of computation complexity, that it can be used in any binary implementation of SFE protocols. We further extend the secure dot-product algorithm to PIR/OT protocols, and investigate its applicability to our problem.

In Chapter 4, we address the communication complexity of PIR/OT protocols and propose an optimal protocol that requires only $\mathcal{O}(\log(n))$ communications – n is the size of the database – while the best previously-known algorithm asymptotically reaches $\mathcal{O}((\log(n))^2)$. In our protocol, we achieve computational security for the user, while the server is perfectly secure.

An equally important point to be consider in implementing PIR/OT protocols is the computation complexity of the scheme, as the computation overheads of PIR/OT protocols is at least linear in the size of the database. Since, every PIR/OT scheme should involve all data in the database each time the protocol is executed, otherwise some information about the user's query will leak to the server. This is, therefore, too expensive for servers of large databases to practice a PIR/OT protocol.

A new approach to reduce the computations in PIR/OT protocols is proposed in Chapter 5, where all heavy computations are performed in the offline phase that is run once, such that the total online-computations are reduced to $\mathcal{O}(t)$ $(t \ll n)$. Using multiple proxies that are specified to offer privacy-preserving services, we further improve this scheme to achieve a realizable PIR/OT protocol in practice. Some possible research directions to be followed in future are given in Chapter 6.

Chapter 2

Definitions and Preliminaries

In this chapter, we give formal definitions and some preliminaries that will be used throughout this thesis. In Section 2.1, the required background in cryptology is given, and formal security definitions of PIR/OT protocols will follow in Section 2.2.

2.1 Semantics Background

We first recall some of the most useful criteria in evaluating security of a protocol in this section. Protocols proposed in this thesis are assessed in either computational or unconditional security model.

Definition 2.1.1 ***(Computational security):*** *This measures the computational effort required to break a cryptosystem. A system is called computationally secure if the best algorithm for breaking it requires a huge amount of efforts that cannot be gathered in reality. Computational security is usually studied with respect to certain type of attacks (e.g. exhaustive key search). Of course, security against one specific attack does not guarantee security against other type of attacks. Therefore, a computationally secure system cannot be proven to be absolutely secure. In a computationally secure*

system, the adversary is usually assumed to have polynomial-bounded time, space and computational resources in the size of the security parameter for the encryption scheme, while the breaking algorithm requires exponential-time or super polynomial-time effort.

A computationally secure cryptosystem with the encryption algorithm $\mathcal{E}$ is called *indistinguishable* if for every probabilistic polynomial time algorithm A and polynomial $p(.)$, a sufficiently large security parameter k and all plaintexts x, y of polynomial size ($|x| = |y| \leq p(k)$), there is negligible chance of identifying x and y from their encrypted values:

$$|Pr(A(k, \mathcal{E}(x)) = 1) - Pr(A(k, \mathcal{E}(y)) = 1)| < \frac{1}{p(k)}$$

It can be shown that indistinguishability against a passive attacker results in *semantic security* that guarantees no information can be obtained about the plaintext by just looking at the ciphertext – for a detailed definition on semantics see [16].

Definition 2.1.2 ***(Unconditional security):*** *This measure concerns the security of the protocol when there is no bound placed on the computational power of the adversary. A system is defined to be unconditionally secure if it cannot be broken even with infinite computational resources. An unconditionally secure system is sometimes referred to as an information-theoretically secure or a perfectly secure system.*

Having given two major approaches to define security for cryptosystems, we introduce some common terms and definitions that will be widely used in this work.

2.2 Introduction

Here, we extend the definition of computational and unconditional security to the parties involved in PIR/OT protocols. There are usually two consistent parties running

a PIR/OT protocol; the user and the server. The user – which is sometimes called chooser – wants to hide her inputs in a function f that has to be computed by another party, which we call the server. The user's inputs are denoted by x. We are especially interested in data-retrieval functions, where the server – which is also called the database holder – owns n data denoted by

$$\vec{y} = \{y_1, y_2, \cdots, y_n\}$$

and has to protect the data from unauthorized access. The variable n denotes size of the database or size of the server's inputs to the protocol. In an PIR protocol, the user wants to receive data from the server in a communication-efficient way, while not revealing what entry from the database she is asking for. A one-out-of-n PIR (PIR_1^n) is a protocol, which allows the user to privately obtain one of the input data $\{y_1, y_2, \cdots, y_n\}$ from the server.

In a computationally-secure PIR protocol, the privacy of the chooser is assured against an adversary that is computationally-bounded in polynomial time. The server receives a function with the user's inputs embedded in it to compute. The server's view of the protocol is defined as a set, $V_S\{T(i), \vec{y}, Out(i), hist\}$, where $T(i)$ is user's input to the protocol, $\vec{y}$ the sender's inputs, $Out(i)$ the protocol's output and *hist* a history of previous interactions. It should be noted that we only provide user's security against a semi-honest server who runs the protocol properly but is curious to find the user's input (passive attacker).

Definition 2.2.1 ***(Chooser's computational security):*** *For every polynomially bounded adversary with the server's view, and a probabilistic polynomial time algorithm A that returns the chooser's input (i'), the adversary has negligible advantage (Adv) in*

guessing i over a random coin-tossing:

$$Pr[i = i' | A(V_S[T(i), \vec{y}, Out(i), hist]) = i'] = \frac{1}{n} + Adv.$$

The adversary's advantage (Adv) is negligible if there exists a polynomial $p(k)$ in the user's security parameter k, such that

$$Adv < \frac{1}{p(k)} \quad for\ all\ k.$$

In other words, the database holder should not be able to guess the chooser's choice with polynomial-time computations.

Definition 2.2.2 ***(Chooser's perfect security):*** *The chooser is perfectly (information-theoretically) secure if $T(i)$ and $Out(i)$ reveal absolutely no information about i to the server. For an adversary with the server's view V_S, all $i \in [1, n]$ should be equally likely to give the same $T(i)$ (and $Out(i)$[1]). This can be compared to the ideal case, where there exists a trusted third party that gets $\{y_1, \cdots, y_n\}$ from the server, and i from the chooser, then the trusted party returns y_i to the chooser without revealing any information about i to the server.*

We define the OT protocol as a PIR protocol in which the security of the server is also provided. This seems to be solvable by applying regular access control schemes, but it should be noted that in an OT protocol there are usually several messages interchanged between the user and server in every round of the protocol. The security of the server should not allow the user to receive more information than what is permitted from these messages. The chooser's view is defined similarly as $V_C\{T(i), i, Out(i)\}$. We deliberately exclude the history (*hist*) from what can help the user to retrieve

[1]This is a very restrictive condition, as $Out(i)$ depends on the server's inputs and it may not be possible to get a totally random-looking output, but ideally it should appear random to the server.

more data from the protocol, as we assume that (*hist*) was obtained only by having the user pay for previous transactions. So, we consider the security of the server per execution of the OT protocol. A formal definition of the server's computational and unconditional security follows.

Definition 2.2.3 ***(Server's computational security):*** *The computationally bounded user must have negligible advantage in receiving any information from Out(i) about other entries in the database, except* y_i.

Definition 2.2.4 ***(Server's unconditional security):*** *We compare security of the database holder (server) to the ideal case, where there exits a trusted third party that delivers no extra information about* $\{y_0, \cdots, y_n\}$ *than what the user has requested. The server's security requires that the output of a computationally unbounded adversary with the chooser's view* (V_C) *along with a priori information of* $\{y_0, \cdots, y_n\}$ *and an adversary in the ideal case are the same. Regardless of the computational power of the adversary, he cannot guess the sender's other inputs as long as the third party is trusted.*

It should be added that the PIR/OT protocol does not have to and cannot satisfy correctness. Even if the chooser acts honestly, the database holder may refuse to follow the protocol or return a random answer. There are no enforcements to make the database holder follow the protocol properly, nor are there any detection mechanisms that show the mismatch. Thus, we make the assumption that the server properly runs the protocols that will be presented in this thesis.

Chapter 3

One-Round Secure Dot-Product Protocol

The dot-product is a very important primitive in cryptography with many applications in SFE and PIR/OT protocols. In this chapter, we propose the first non-interactive dot-product protocol that is information-theoretically secure for all parties involved in the protocol. The definition of a secure dot-product is given in Section 3.1, where the reader can also find some of the related work. Our secure dot-product protocol is designed based on trace functions over finite fields that we will introduce in Section 3.2. The main protocol is given in Section 3.3, and we further investigate its application to PIR/OT schemes in Section 3.4. The conclusion is followed in Section 3.5.

3.1 Introduction

The dot-product is defined over two vectors $\vec{x} = (x_1, x_2, \cdots, x_n)$ and $\vec{y} = (y_1, y_2, \cdots, y_n)$, and the result is a scalar value given by (3.1).

$$\vec{x}.\vec{y} = (x_1, x_2, \cdots, x_n).(y_1, y_2, \cdots, y_n) = \sum_{i=1}^{n} x_i y_i. \tag{3.1}$$

Suppose that there are two parties A and B that jointly compute the dot-product two vectors. By jointly, we mean that A is holding $\vec{x}$, and B's input is $\vec{y}$. The protocol will enable A to learn the dot-product of $\vec{x}$ and $\vec{y}$, without requiring the value of $\vec{y}$ to be known explicitly by A. It should be noted that such protocol has to ensure that A's input (i.e. $\vec{x}$) and the output of the dot-product are kept secure and hidden from B. A formalized definition of this notion will be given in Section 3.3.

As mentioned earlier, secure dot-product has direct application in SFE protocols, since each function can be represented by a boolean circuit comprising AND/OR gates. Generally, the SFE scheme deals with computing a function of several inputs that are provided by separate parties. Each participant holds some parts of the input and wants to hide it from the others. All parties jointly compute the circuit on their own inputs, such that the output of the circuit does not reveal any of the inputs. There exist some solutions to build an SFE protocol in literature, where secure AND/OR gates are the main building blocks [29, 16, 44]. In this approach, the dot-product of two boolean vectors can be seen as an AND of several gates that are then XORed together.

The communication complexity is usually the key factor in efficiency of any SFE protocols. The communication overheads of an SFE protocol depends on the size of the circuit that expresses the function to be computed. A secure, one-round dot-product protocol can significantly reduce the communication overheads and make SFE protocols more practical.

Another important application of secure dot-products is in data-mining algorithms; some statistical analysis like counting the frequency of an item in the database is directly computed by calculating the dot-product of two vectors [40]. Privacy of users requires this statistical analysis be done securely without revealing their attributes, in other words the dot-product should be performed securely.

Despite their wide possible applications, there has been limited work on designing

practical secure dot-product protocols. Two such protocols are those in [13, 42], although the work done by Vaidya and Clifton [42] outperforms the one by Atallah and Du [13] in terms of communication overheads. These protocols require four rounds of interactions between the user and the server. However, the work in [42] was proven to be insecure by Goethals *et al.* [23]. Goethals *et al.* also have proposed another (computationally) secure dot-product protocol, which still requires more than one round of interactions.

Here, we propose the first non-interactive, information-theoretically secure dot-product protocol that is mainly based on trace functions over finite fields. The main characteristics of finite fields that we use in this work are given next, followed by our secure dot-product protocol.

3.2 Finite Fields

Groups, rings and fields are very basic mathematical definitions in algebra on which we refer the reader to Appendix A for some backgrounds. In this section, we briefly explain the necessary preliminaries in finite fields that are mostly taken from [37].

We denote finite fields by $\mathbb{F}$, and it can be shown that the characteristic of any field has to be prime [37]. So, we denote fields of characteristic p, where p is a prime, by $\mathbb{F}_p$. The prime field $\mathbb{F}_p$ can be further extended to $\mathbb{F}_{p^n}$ by an irreducible polynomial of degree n over the ring $\mathbb{F}_p[x]$. Here, we adopt the viewpoint of treating the finite extension $\mathbb{F}_{p^n}$ of a finite field $\mathbb{F}_p$ as a vector space over $\mathbb{F}_p$. Thus, $\mathbb{F}_{p^n}$ has the dimension n over $\mathbb{F}_p$. Should $\{\alpha_1, \cdots, \alpha_n\}$ be a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, each element $\alpha \in \mathbb{F}_{p^n}$ can be uniquely represented in the following form

$$X = x_1\alpha_1 + x_2\alpha_2 + \cdots + x_n\alpha_n,$$

where $x_i \in \mathbb{F}_p$ for $i = 1, 2, \cdots, n$. A normal basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ is a basis of the form $\bar{\alpha} = \{\alpha, \alpha^p, \cdots, \alpha^{p^{n-1}}\}$, and we say that α generates the normal basis $\bar{\alpha}$, or α is a *normal element* of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$. In the rest of this chapter, when we refer to a normal basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, we always assume that $\alpha_i = \alpha^{p^i}$ for $i = 0, 1, \cdots, n-1$. Next, we introduce an important, linear mapping from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p$ that will be the basic block in our secure dot-product protocol.

Definition 3.2.1 *For $\alpha \in \mathbb{F}_{p^n} = F$ and $\mathbb{F}_p = K$, the trace $Tr_{F/K}(\alpha)$ of α over K is defined by*

$$Tr_{F/K}(\alpha) = \alpha + \alpha^p + \alpha^{p^2} + \cdots + \alpha^{p^{n-1}}.$$

If K is the prime subfield of F, then $Tr_{F/K}(\alpha)$ is called the *absolute trace* of α and is simply denoted by $Tr_F(\alpha)$. For simplicity, we denote the trace function by Tr if the fields are clear from the context. The trace function satisfies the following properties [37]:

1. $Tr(\alpha + \beta) = Tr(\alpha) + Tr(\beta)$ for all $\alpha, \beta \in \mathbb{F}_{p^n}$;
2. $Tr(c\alpha) = cTr(\alpha)$ for all $\alpha \in \mathbb{F}_{p^n}$ and $c \in \mathbb{F}_p$;
3. Tr is a linear transformation from $\mathbb{F}_{p^n}$ onto $\mathbb{F}_p$;
4. $Tr(a) = na$ for all $a \in \mathbb{F}_p$;
5. $Tr(\alpha^p) = Tr(\alpha)$ for all $\alpha \in \mathbb{F}_{p^n}$.

In the case, where a chain of extension fields is considered, the composition of trace functions proceeds in according to a very simple rule [37].

Theorem 3.2.1 *(Transitivity of Trace). Set K to be a finite field. Let F be a finite extension of K and E a finite extension of F. Then for all $\alpha \in E$*

$$Tr_{E/K}(\alpha) = Tr_{F/K}(Tr_{E/F}(\alpha))$$

Suppose that $A = \{\alpha_1, \cdots, \alpha_n\}$ and $B = \{\beta_1, \beta_2, \cdots, \beta_n\}$ are two bases of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, the base A is referred to as the *dual basis* of B if for $1 \leq i, j \leq n$:

$$Tr(\alpha_i \beta_j) = \begin{cases} 0 \text{ for } i \neq j, \\ 1 \text{ for } i = j. \end{cases} \tag{3.2}$$

It is worth noting that for any basis A, there exists a unique dual basis [37]. If the dual basis B is the same as A, then A is called a *self-dual* basis. There are many different bases of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, and if the order of a basis element is taken into account, the number of all bases is given by

$$(p^n - 1)(p^n - p)(p^n - p^2) \cdots (p^n - p^{n-1}).$$

It is easy to show that traces can be directly used in computing the dot-product of two vectors as given in (3.1). The following theorem proves this.

Theorem 3.2.2 *Let $A = \{\alpha_1, \cdots, \alpha_n\}$ be a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, and $B = \{\beta_1, \cdots, \beta_n\}$ its dual basis satisfying (3.2). For all $X, Y \in \mathbb{F}_{p^n}$ represented in the following form*

$$X = x_1\alpha_1 + x_2\alpha_2 + \cdots + x_n\alpha_n,$$
$$Y = y_1\beta_1 + y_2\beta_2 + \cdots + y_n\beta_n,$$

where $x_i, y_i \in \mathbb{F}_p$, *then*

$$Tr(XY) = \vec{x}.\vec{y}.$$

Proof:

$$\begin{aligned}
Tr(XY) &= Tr((x_1\alpha_1 + \cdots + x_n\alpha_n)(y_1\beta_1 + \cdots + y_n\beta_n)),\\
&= Tr(x_1\alpha_1(y_1\beta_1 + \cdots + y_n\beta_n) + \cdots + x_n\alpha_n(y_1\beta_1 + \cdots + y_n\beta_n)),\\
&= Tr(x_1\alpha_1(y_1\beta_1 + \cdots + y_n\beta_n) + \cdots + Tr(x_n\alpha_n(y_1\beta_1 + \cdots + y_n\beta_n)),\\
&= Tr(x_1y_1\alpha_1\beta_1 + \cdots + x_1y_n\alpha_1\beta_n) +\\
&\cdots + Tr(x_ny_1\alpha_n\beta_1 + \cdots + x_ny_n\alpha_n\beta_n),\\
&= x_1y_1Tr(\alpha_1\beta_1) + \cdots + x_1y_nTr(\alpha_1\beta_n) +\\
&\cdots + x_ny_1Tr(\alpha_n\beta_1) + \cdots + x_ny_nTr(\alpha_n\beta_n),\\
&= x_1y_1 + x_2y_2 + \cdots + x_ny_n. \quad \square
\end{aligned}$$

As we have shown in Theorem 3.2.2, traces can be used directly in computing dot-products of two vectors, however they will not be useful in cryptology unless they can be computed securely. Therefore, the definition of secure computation of a dot-product has to be translated to a secure computation of a trace. Securely computing the trace of two elements is a challenging problem, which we will address next.

3.3 The Secure Dot-Product Protocol

There are two parties involved in the secure computation of a dot-product protocol; the user owning the vector $\vec{x} = (x_1, x_2, \cdots, x_n)$ and the server owning $\vec{y} = (y_1, y_2, \cdots, y_n)$. Let $A = \{\alpha_1, \cdots, \alpha_n\}$ be a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, and $B = \{\beta_1, \cdots, \beta_n\}$ its dual. Assume that $\vec{x}$ and $\vec{y}$ are mapped to X and Y correspondingly, which are two elements of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$.

$$X = x_1\alpha_1 + x_2\alpha_2 + \cdots + x_n\alpha_n,$$
$$Y = y_1\beta_1 + y_2\beta_2 + \cdots + y_n\beta_n.$$

In our design, the server computes the trace function of XY that equals $\vec{x}.\vec{y}$ (Theorem 3.2.2), however the server must not be able to learn the user's input (X). The user, on the other hand, learns $Tr(XY)$ but nothing about Y. The user hides her input in the following way:

$$U = aX + bK, \tag{3.3a}$$
$$V = cX + dK, \tag{3.3b}$$

where $a, b, c, d \in \mathbb{F}_p$ and $K \in \mathbb{F}_{p^n}$ are randomly selected such that $(ad - bc) \neq 0$. The user transmits $\{U, V\}$ to the server. The server, who owns Y and receives only $\{U, V\}$ from the user, computes:

$$YU = Y(aX + bK),$$
$$YV = Y(cX + dK).$$

Finally, the server (Srvr) returns the trace of the results to the user (Usr).

$$\text{Srvr: } \{Tr(YU), Tr(YV)\} \longrightarrow \text{Usr}$$

The user, who knows the random integers (a, b, c and d), retrieves the correct value of the dot-product of $\vec{x}$ and $\vec{y}$ as follows:

$$\vec{x}.\vec{y} = (ad - bc)^{-1}(dTr(YU) - bTr(YV)) \mod p. \tag{3.4}$$

The followings verify correctness of the user's output.

$$\begin{aligned} dTr(YU) - bTr(YV) &= dTr(Y(aX + bK)) - bTr(Y(cX + dK)), \\ &= Tr(dY(aX + bK) - bY(cX + dK)), \\ &= Tr(adYX + dbYK - bcYX - bdYK), \\ &= Tr((ad - bc)XY), \\ &= (ad - bc)Tr(XY). \end{aligned}$$

We further examine security of the user, and prove that the user is information-theoretically secure in this protocol.

Theorem 3.3.1 *The probability of the user's choice being equal to $X \in E$ is equally distributed over $E \subset \mathbb{F}_{p^n}$, where $|E| = 2(p^2 - 1)(p^2 - p)$.*

Proof: For a given $\{U, V\}$, there exist several X's in $\mathbb{F}_{p^n}$ that may lead to the same output with a proper choice of K.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} X \\ K \end{pmatrix} = \begin{pmatrix} U \\ V \end{pmatrix} \tag{3.5}$$

If we denote the set of all X's that lead to the same output in Equation 3.5 by E, the adversary's chance in guessing the user's input is $\frac{1}{|E|}$, since there are $|E|$ equally likely answers that give the same results in Equation 3.5. To guess X, one first has to find all invertible, 2-by-2 matrices over $\mathbb{F}_p$, and then select between X and K. The number of invertible, 2-by-2 matrices over $\mathbb{F}_p$ is $(p^2 - 1)(p^2 - p)$, thus the size of E for a given

$\{U, V\}$ is $2(p^2-1)(p^2-p)$. $\square$

Adversary's advantage in guessing user's choice can be made exponentially small by choosing large values of p. In information-theoretical security, the user's choice has to be evenly distributed over the whole field $\mathbb{F}_{p^n}$, so that all elements can be a right answer with the same probability. Nevertheless, the user's input, in our setup, is equally distributed not over the whole field ($\mathbb{F}_{p^n}$), but over a smaller subset of the field ($E \subset \mathbb{F}_{p^n}$). This can be improved further if the protocol is modified as follows. Instead of sending (3.3a) and (3.3b) to the server, the user computes a random, invertible t-by-t matrix ($A_{t\times t}$) and sends the following:

$$\begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_t \end{pmatrix} = A_{t\times t} \begin{pmatrix} X \\ K_1 \\ K_2 \\ \vdots \\ K_{t-1} \end{pmatrix}$$

where $K_i \in \mathbb{F}_{p^n}$ is selected at random for $1 \leq i \leq (t-1)$. In order to get information theoretical security, the size of $|E|$ has to be at least p^n. In other words, the adversary's advantage in guessing user's input must be at most $\frac{1}{p^n}$. The number of invertible, t-by-t matrices is at least p^{2t}. Hence, we set $t \simeq n/2$. It should be noted that this extra security is obtained at the cost of $\mathcal{O}(n^2)$ communication overheads, compared to $\mathcal{O}(n)$ communications in the initial scheme. Next, we examine whether the proposed, secure dot-product protocol can be used in PIR/OT protocols.

3.4 The PIR Protocol Using Dot-Products

Let $A = \{\alpha_1, \cdots, \alpha_n\}$ be a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, and $B = \{\beta_1, \cdots, \beta_n\}$ be its dual basis satisfying (3.2). The variable S represents the database and is defined as follows:

$$S = y_1\alpha_1 + y_2\alpha_2 + \cdots + y_n\alpha_n,$$

where $y_i \in \mathbb{F}_p$ is the i'th entry in the database for $i = 1, \cdots, n$. In order to privately retrieve y_i from the database, the user has to hide (β_i). She selects $\kappa \in \mathbb{F}_{p^n}$ and $a, b, c, d \in \mathbb{F}_P$ at random, such that $(ad - bc) \neq 0$. The user computes the followings:

$$U = a\beta_i + b\kappa, \tag{3.6a}$$

$$V = c\beta_i + d\kappa. \tag{3.6b}$$

Then, she sends $\{U, V\}$, computed in (3.6a,3.6b), to the server.

$$\text{Usr} : \{U, V\} \longrightarrow \text{Srvr}$$

On the other side, the server by knowing S and $\{U, V\}$ computes the following messages:

$$u = Tr(SU) = Tr(S(a\beta_i + b\kappa)),$$
$$v = Tr(SV) = Tr(S(c\beta_i + d\kappa)).$$

The server returns $\{u, v\}$ to the user who retrieves the correct entry of the database;

$$y_i = (ad - bc)^{-1}(du - bv) \mod p. \tag{3.7}$$

The following equations justify the correctness of (3.7):

$$\begin{aligned} du - bv &= dTr(S(a\beta_i + b\kappa)) - bTr(S(c\beta_i + d\kappa)), \\ &= Tr(dS(a\beta_i + b\kappa) - bS(c\beta_i + d\kappa)), \\ &= Tr(adS\beta_i - bcS\beta_i), \\ &= (ad - bc)Tr(S\beta_i), \\ &= (ad - bc)y_i. \end{aligned}$$

In order to clarify the security of a PIR scheme based on the secure dot-product protocol, we give a numerical example for the rest of this section.

Example 3.4.1 *Let $\mathbb{F}_{p^n} = \mathbb{F}_{3^2}$ be an extension field of $\mathbb{F}_3$ generated by the irreducible polynomial $f(x) = x^2 + 1$. Let α be a root of $f(x)$, i.e. $f(\alpha) = \alpha^2 + 1 = 0 \mod 3$. The addition and multiplication tables for elements of $\mathbb{F}_{3^2}$ are given in Table 3.1 and Table 3.2 respectively.*

A normal basis of $\mathbb{F}_{3^2}$ over $\mathbb{F}_3$ is $A = \{\alpha_1 = \alpha + 1, \alpha_2 = \alpha + 2\}$, while its dual basis is $B = \{\beta_1 = 2\alpha + 1, \beta_2 = 2\alpha + 2\}$. The property (3.2) for the normal base and its dual can be easily verified.

Table 3.1: The addition table for $\mathbb{F}_3/(x^2+1)$

$+$	0	1	2	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$
0	0	1	2	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$
1	1	2	0	$\alpha+1$	$\alpha+2$	α	$2\alpha+1$	$2\alpha+2$	2α
2	2	0	1	$\alpha+2$	α	$\alpha+1$	$2\alpha+2$	2α	$2\alpha+1$
α	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$	0	1	2
$\alpha+1$	$\alpha+1$	$\alpha+2$	α	$2\alpha+1$	$2\alpha+2$	2α	1	2	0
$\alpha+2$	$\alpha+2$	α	$\alpha+1$	$2\alpha+2$	2α	$2\alpha+1$	2	0	1
2α	2α	$2\alpha+1$	$2\alpha+2$	0	1	2	α	$\alpha+1$	$\alpha+2$
$2\alpha+1$	$2\alpha+1$	$2\alpha+2$	2α	1	2	0	$\alpha+1$	$\alpha+2$	α
$2\alpha+2$	$2\alpha+2$	2α	$2\alpha+1$	2	0	1	$\alpha+2$	α	$\alpha+1$

Table 3.2: The multiplication table for $\mathbb{F}_3/(x^2+1)$

$\cdot$	0	1	2	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$
0	0	0	0	0	0	0	0	0	0
1	0	1	2	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$
2	0	2	1	2α	$2\alpha+2$	$2\alpha+1$	α	$\alpha+2$	$\alpha+1$
α	0	α	2α	2	$\alpha+2$	$2\alpha+2$	1	$\alpha+1$	$2\alpha+1$
$\alpha+1$	0	$\alpha+1$	$2\alpha+2$	$\alpha+2$	2α	1	$2\alpha+1$	2	α
$\alpha+2$	0	$\alpha+2$	$2\alpha+1$	$2\alpha+2$	1	α	$\alpha+1$	2α	2
2α	0	2α	α	1	$2\alpha+1$	$\alpha+1$	2	$2\alpha+2$	$\alpha+2$
$2\alpha+1$	0	$2\alpha+1$	$\alpha+2$	$\alpha+1$	2	2α	$2\alpha+2$	α	1
$2\alpha+2$	0	$2\alpha+2$	$\alpha+1$	$2\alpha+1$	α	2	$\alpha+2$	1	2α

$$Tr((\alpha+1)(2\alpha+1)) = Tr(2),$$

$$= 2 + 2^3 = 2 + 2 \equiv 1.$$

$$Tr((\alpha+1)(2\alpha+2)) = Tr(\alpha),$$

$$= \alpha + \alpha^3 = \alpha + 2\alpha \equiv 0.$$

$$Tr((\alpha+2)(2\alpha+1)) = Tr(2\alpha),$$

$$= 2Tr(\alpha) \equiv 0.$$

$$Tr((\alpha+2)(2\alpha+2)) = Tr(2) \equiv 1.$$

Suppose that the server owns two entries $(y_1, y_2 \in \mathbb{F}_3)$*, and represents them by* $S = y_1(\alpha+1) + y_2(\alpha+2) \in \mathbb{F}_{3^2}$*. To privately retrieve* y_1 *or* y_2*, the user randomly selects* $\kappa = \alpha$*, and sets* $a = b = c = 1, d = 2$ *in (3.6a,3.6b). If the user wants to receive* y_1 *from the database, she chooses* $\beta_1 = 2\alpha + 1$ *and computes the following messages:*

$$U = 2\alpha + 1 + \alpha \equiv 1,$$
$$V = 2\alpha + 1 + 2\alpha \equiv \alpha + 1.$$

The server, on the other side, computes:

$$\begin{aligned} u &= Tr(SU) = Tr(y_1(\alpha+1) + y_2(\alpha+2)), \\ &= 2y_1 + 2y_2. \\ v &= Tr(SV) = Tr((y_1(\alpha+1) + y_2(\alpha+2))(\alpha+1)), \\ &= y_2. \end{aligned}$$

The server returns $\{u, v\}$ *to the user who then retrieves* y_1*.*

$$\begin{aligned} y_i &= (2-1)^{-1}(2u - v), \\ &= 2(2y_1 + 2y_2) - y_2 \equiv (y_1 + y_2) - y_2, \\ &= y_1. \end{aligned}$$

In this example, the user is also able to recover y_2, since she receives two messages from the server. But in general, where there are more than two entries in the database $(n > 2)$, the user cannot receive more than one data and security of the server will be

guaranteed correspondingly. Next, we analyze security of both the user and the server in details.

3.4.1 Security Analysis

In the proposed PIR protocol using secure dot-products, the user has to choose the random number κ carefully to avoid bad choices, since a bad selection of $\kappa = \beta_i$ will compromise the user's privacy. It is not very hard for the attacker to recover the user's choice if he knows that $\kappa = \beta_i$ was selected. In order to find the user's choice, the attacker has to first subtract $x\beta_i$ from $U = a\beta_i + b\beta_j$ for all $x \in \mathbb{F}_p$, and then compare the result with $x\beta_i$'s until he finds either β_i or β_j. This attack requires at most $(pn)/2$ computations for the attacker. For small values of p and n, the user's choice can be easily recognized. Next, we show that if bad choices of κ are avoided, this can be beneficial for both the user and the server, since information-theoretical security of the server can also be satisfied.

Theorem 3.4.1 *If the random numbers in (3.6a) and (3.6b) are chosen such that* $(ad - bc) \neq 0$ *and* κ *is not equal to any of the* β_i*'s, then the user retrieves no more than one entry from the database by only receiving* $\{Tr(SU), Tr(SV)\}$.

Proof: Suppose that the user can receive more than one data point from $\{Tr(SU), Tr(SV)\}$. This implies that two choices of β_i with random parameters a_i, b_i, c_i, d_i and κ_i and β_j with random values a_j, b_j, c_j, d_j and κ_j in (3.6a) and (3.6b) lead to the same output.

$$U = a_i\beta_i + b_i\kappa_i = a_j\beta_j + b_j\kappa_j,$$
$$V = c_i\beta_i + d_i\kappa_i = c_j\beta_j + d_j\kappa_j.$$

Re-ordering the equations, we get

$$a_i\beta_i - a_j\beta_j = b_i\kappa_i - b_j\kappa_j, \tag{3.8a}$$

$$c_i\beta_i - c_j\beta_j = d_i\kappa_i - d_j\kappa_j. \tag{3.8b}$$

The random number $\kappa_i = \sum_{i=1}^{n} k_i\beta_i$ and $\kappa_j = \sum_{j=1}^{n} k_j\beta_j$ are selected such that, at least two of k_i's or k_j's are non-zero. The equations (3.8a) and (3.8b) give a non-trivial equation of β_i's that equals zero. This cannot be true, unless there exists a variable r such that $ra_i = c_i$, $ra_j = c_j$, $rb_i = d_i$ and $rb_j = d_j$, so $(a_id_i - b_ic_i) = 0$ and $(a_jd_j - b_jc_j) = 0$ correspondingly. This is a contradiction, and the user cannot receive more than one entry from the database if κ in (3.6a) and (3.6b) is not equal to any of β_i's for $i = 1, 2, \cdots, n$. □

We have already shown the information-theoretical security of the server, that is the user can only receive one entry of the database per transaction. In Section 3.3, it is proved that the user in our secure dot-product protocol is unconditionally secure, since there are approximately p^4 equally-likely answers for the user's input. Nonetheless, in the PIR/OT protocol using secure dot-products, the attacker has to solve (3.9) to find the user's choice (β_i) by just knowing $\{U, V\}$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \beta_i \\ K \end{pmatrix} = \begin{pmatrix} U \\ V \end{pmatrix} \tag{3.9}$$

This time, the exhaustive search returns a unique answer for (3.9), since Theorem 3.4.1 implies that only one β_i could be chosen to receive an entry from the database. Therefore, the unconditional security of the user is reduced to computational security, depending on how hard it is for the attacker to recover β_i from (3.9). The attacker

can find the unique answer by trying approximately all possible p^4 choices of invertible 2-by-2 matrices. In practice, it suffices to choose a 20-bit long prime p as the characteristic of the underlying field ($\mathbb{F}_p$) to make it computationally hard for the attacker – the total computation needed for the adversary to break the system is $\mathcal{O}(p^4) = 2^{80}$.

3.5 Conclusions

In this chapter, we proposed the first non-interactive (one-round), secure dot-product protocol, which was based on trace functions over a finite field $\mathbb{F}_{p^n}$. In our setting, the user was perfectly secure over $E \subset \mathbb{F}_{p^n}$, which would be extended over $\mathbb{F}_{p^n}$ at the cost of extra communication overheads of $\mathcal{O}(n^2)$. The communication complexity of the secure dot-product protocol was $\mathcal{O}(n)$, where n was the size of the input vectors. The computation cost of the protocol was also linear in the size of inputs, and required only simple arithmetic operations such as permutations, additions and multiplications.

We further examined the applicability of the proposed secure-dot product scheme in PIR/OT protocols. We proved that the server could be information-theoretically secured if the user followed the protocol properly and avoided bad choices. However, it was also showed that unconditional security of the user would reduce to computational security, when the proposed secure dot-product scheme was used in PIR/OT protocols. Additionally, we showed, by choosing appropriately large fields, the computational security of the user could be guaranteed. In conclusion, the proposed secure dot-product protocol can be very useful in SFE protocols or private data-mining schemes, but may not be suitable for PIR/OT protocols and other schemes have to be designed.

Chapter 4

An Optimal Oblivious Transfer Protocol

There are two important factors in designing any PIR/OT protocol, communication complexity and computational overheads. In this chapter, we are focusing on security of not only the user but also the server, and we will look at OT protocols generally. We propose an optimal OT protocol where the communication complexity is logarithmic in the size of the database – denoted by n – while the computational overhead is kept at a minimum (linear in the size of the database). The proposed 1-out-of-n Oblivious Transfer (OT_1^n) protocol is computationally secure for the chooser, while the server is perfectly secured. The communication complexity is $\mathcal{O}(\log(n))$ for the user side and only $\mathcal{O}(1)$ for the server. The computational overhead required for the user is $\mathcal{O}(\log n)$ which is the minimum computation needed to produce and send $\log(n)$ messages. The server has to perform $\mathcal{O}(n)$ computations, which is still the minimum amount of work needed for an OT protocol without any preprocessing.

In this chapter, the main construction of our optimal OT protocol is given in Section 4.1. This is followed by a formal definition of Homomorphic Encryption Functions in Section 4.2. We further, in Section 4.3, apply a Homomorphic Encryption Function

to secure the main data recovery function. The initial scheme is improved in terms of computation complexity in Section 4.4, and finally, conclusions and some possible extensions of this protocol are given in Section 4.5.

4.1 Introduction

The main challenge in designing OT protocols has been to minimize the communication complexity and reduce the computational overhead, in order to make such schemes realizable in practice. Some general-purpose SFE schemes, like [44, 20], can be adapted to design OT protocols. However, the communication-computation overhead of the protocol is at least linear in the size of the circuit, which would be the same as the size of the database. In [22, 34], several protocols with different approach than SFE schemes are also proposed, in which the communication complexity is still $\mathcal{O}(n)$. The first OT_1^n with polylogarithmic communications ($\mathcal{O}((\log n)^4)$) was proposed by Cachin et al. [30], where the chooser can only learn one bit each time the protocol is performed. However, this may not be suitable when large files are to be privately accessed from the database. Another protocol with super-logarithmic communications is proposed in [41], where more than one round of interactions between the chooser and the server is required. Lipmaa [24] further improved this work to design an OT protocol with $\mathcal{O}((\log n)^2)$ communications.

The most efficient PIR protocol in terms of communication and computation is proposed in [12], where there is one round of interactions between the chooser and the server. The online communication and computation complexity is $\mathcal{O}(1)$ independent of database size. However, there is a very limiting assumption that every user has a local replicate of the whole database. Updating each of these replicates when they abound in number, and distributing huge databases among thousands of users make this approach less appealing in practice. In [2], another efficient PIR protocol

with $\mathcal{O}(1)$ communication-computation complexity is given in the presence of a secure co-processor. A secure co-processor (SC) is a physical tamper-proof processing unit installed locally on each machine, so that all operations within this unit are hidden from the machine owner. Nonetheless, not only are hardware-based solutions generally more expensive than software-only solutions, but also imposing the installation of some piece of (unknown) hardware on a very diverse set of potential users may not be acceptable in a large scale.

In our setting, we are only considering non-interactive (one round of interactions between the chooser and the server), single database (the database is neither distributed nor replicated) OT protocols, where no extra secure processors or any other specific piece of hardware is required. We further assume the data-retrieving algorithm can be represented by a function (f) that indicates the whole process of data recovery. In the very basic case where the database consists of two entries, the function f can be represented by (4.1).

$$f(x, y_0, y_1) = xy_1 + \bar{x}y_0, \tag{4.1}$$

where $\{y_0, y_1\}$ are entries in the database and x is the selection bit with its opposite given by $\bar{x}$. It is straightforward to check that if $x = 0$ (or $x = 1$) is selected, then $\bar{x} = 1$ ($\bar{x} = 0$), hence the fuction f returns only y_0 (or y_1 correspondingly). This is the basis of our secure data-retrieval protocol that can be further generalized to arbitrarily large databases. If the information that the user receives from the protocol is no more than f, the chooser learns only one out of two values in the database, and the security of the database is perfectly satisfied.

In order to hide the users' input, the output of f defined in (4.1) should be hidden (encrypted) from the server, while the server can still evaluate the value of function f in an encrypted form. In other words, he has to do computations with the encrypted

values of x and $\bar{x}$ without knowing them. To achieve an encrypted but computable function, we use homomorphic cryptosystems.

4.2 Homomorphic Cryptosystems

Homomorphic cryptosystems have many important applications in cryptography and are extensively studied in the literature, e-voting, securing Mobile Agents and blind signature are some of many applications, in which homomorphic cryptosystems are widely used. We refer the interested reader to [6, 36, 39] for more information of such applications.

A Homomorphic cryptosystem is generally a public-key cryptosystem (PKC). Any PKC consists of the tuple $\{\mathcal{E}, \mathcal{D}, \mathcal{P}, \mathcal{C}, \mathcal{K}\}$, where $\mathcal{E}$ is the encryption and $\mathcal{D}$ the decryption algorithm, $\mathcal{P}, \mathcal{C}$ and $\mathcal{K}$ are the plaintext, the ciphertext and the key space, respectively. In addition to common conditions on encryption functions in a PKC, the Homomorphic Encryption Function (HEF) has the following extra properties:

- *Additively homomorphic:* For all $x, y \in \mathcal{P}$, we have $\mathcal{E}(x + y) = \mathcal{E}(x) \times \mathcal{E}(y)$. In other words, there is a special operation $(\times)$ that results in the addition of plaintext elements when applied to the corresponding ciphertext elements. This operation can be performed without decrypting the ciphertext.

- *Multiplicatively homomorphic:* For all $x, y \in \mathcal{P}$, there is an operation $*$, such that $\mathcal{E}(xy) = \mathcal{E}(x) * \mathcal{E}(y)$. In this way $\mathcal{E}(xy)$ can be found directly from $\mathcal{E}(x)$ and $\mathcal{E}(y)$ without knowledge of x or y.

It has been proven that all such deterministic cryptosystems can be broken in sub-exponential time [6], thus the *probabilistic* property, that is defined next, has to be added to the definition of any secure homomorphic cryptosystem.

- *Probabilistic:* Informally, each element of the plaintext is randomly mapped into a different ciphertext every time it is encrypted (see [17] for more details).

The homomorphic multiplicative property is usually replaced with a more relaxed definition, the *mix*-multiplicative property:

- *Mix-multiplicatively homomorphic:* For $x, y \in \mathcal{P}$, we can find $\mathcal{E}(xy) = \mathcal{E}(x)^y$ without knowledge of x only. In other words, the variable y is known in computing $\mathcal{E}(xy)$.

With the mix-multiplicative definition, we can design homomorphic cryptosystems defined over groups, such as RSA, Goldwasser-Micali's PKC [17], and Paillier's PKC [36]). In this work, we base our protocol on Paillier's cryptosystem or its variants, since it is a probabilistic cryptosystem with both additively and mix-multiplicatively homomorphic properties. For completeness, we also provide a brief overview of Paillier's public-key cryptosystem.

4.2.1 Paillier's public-key cryptosystem

Let g be an element in $\mathbb{Z}^*_{N^2}$, the order of which is a multiple of N. Let the variable $N = PQ$ be an RSA moduli, where P and Q are two random large primes. In Paillier's PKC, the encryption function $\mathcal{E}$ is defined as in (4.2).

$$\mathcal{E} : \mathbb{Z}_N \times \mathbb{Z}^*_N \longrightarrow \mathbb{Z}^*_{N^2} \tag{4.2a}$$

$$\mathcal{E}(m, r) = g^m r^N \mod N^2 \tag{4.2b}$$

The encryption function $\mathcal{E}$ in (4.2) is bijective from $\mathbb{Z}_N \times \mathbb{Z}^*_N$ to $\mathbb{Z}^*_{N^2}$. To prove this, we notice that the number of elements in $\mathbb{Z}_N \times \mathbb{Z}^*_N$ is the same as the number of elements in $\mathbb{Z}^*_{N^2}$; since $\phi(N^2) = N\phi(N)$, where ϕ is the Euler's function and it suffices to show

that $\mathcal{E}$ is injective. Suppose that there are two identical ciphertexts, that is

$$g^{m_1}{r_1}^N = g^{m_2}{r_2}^N \mod N^2.$$

If both sides are first raised to the power of $\lambda(N)$, where $\lambda = \text{lcm}(P-1, Q-1)$ is the Carmichael function, and then divided by the right hand-side, we get the following:

$$g^{(m_1-m_2)}(r_1/r_2)^N = 1 \Rightarrow g^{\lambda(N)(m_1-m_2)} = 1 \mod N^2.$$

We conclude that $\lambda(N)(m_1 - m_2)$ is a multiple of order of g, which is already a multiple of N. But $\gcd(N, \lambda(N)) = 1$, so $N|(m_1 - m_2)$ and therefore $m_1 = m_2 \mod N$. Consequently, $(r_1/r_2)^N = 1 \mod N^2$ leads to $r_1/r_2 = 1 + kN$, that is $r_1 = r_2$ over $\mathbb{Z}_N^*$. We showed that $\mathcal{E}$ is one-to-one and onto, thus $\mathcal{E}$ is bijective.

It is proved that Paillier's PKC is semantically secure under the Decisional Composite Residuosity Assumption (DCRA). A number c is said to be an N-th residue modulo N^2 if there exists a number $y \in \mathbb{Z}_{N^2}^*$ such that $c = y^N \mod N^2$. Under DCRA, there exists no polynomial time distinguisher for N-th residues modulo N^2. Similarly under Composite Residuosity Assumption (CRA), it is computationally intractable to compute m given only $c = g^m r^N$, N and g. Therefore, this cryptosystem cannot be broken in sub-exponential time [36]. However, if the factorization of N is known, we can decrypt the message m using the decryption algorithm $\mathcal{D}$ as follows:

$$m = \mathcal{D}(c) = \frac{L(c^{\lambda(N)} \mod N^2)}{L((1+N)^{\lambda(N)} \mod N^2)} \mod N,$$

where $L(u) = \dfrac{u-1}{N}$ for all $u \in \mathbb{Z}_{N^2}^*$. In this scheme of Paillier's cryptosystem, the user's public-key pair is (N, g) and corresponding private-key pair would be (P, Q). Note also that Paillier's cryptosystem is probabilistic; every time the message m is encrypted, a randomly selected integer r maps the message to a different ciphertext.

Homomorphic properties of Paillier's cryptosystem can be shown as follows:

- *Additively homomorphic:*

$$\mathcal{D}(\mathcal{E}(m_1, r_1)\mathcal{E}(m_2, r_2)) = m_1 + m_2 \mod N,$$

- *Mix-multiplicatively homomorphic:*

$$\mathcal{D}(\mathcal{E}(m, r)^c) = cm \mod N.$$

There are also some technical characteristics of Paillier's cryptosystem that make it suitable for our construction. We notice that the plaintext space contains both zero and one that are the most common messages being sent from the user to the server in our setup (see Section 4.4). Moreover, zero and one in Paillier's PKC can be effectively encrypted, since number of exponentiations in (4.2) will be reduced.

4.3 The Initial OT_1^2 Scheme

Having given an explanation of a homomorphic PKC, we are now ready to combine an HEF with our data-retrieval algorithm to construct a secure OT protocol. We start with the simple case where there are only two entries in the database, so the function f is given as in (4.1). In order to securely compute f, the user and the server jointly run the following protocol.

The user encrypts the pair $\{x, \bar{x}\}$, where $x \in \{0, 1\}$ is her choice and $\bar{x}$ is its opposite. She then sends $\{\mathcal{E}(x), \mathcal{E}(\bar{x})\}$ to the server. It should be noted that user's own public-key is being used to encrypt her choice and its opposite. For simplicity of notations, we do not show the key subscript, and always assume that the user's public/private keys are used for encryption/decryption every where in this chapter.

When the server receives $\{\mathcal{E}(x), \mathcal{E}(\bar{x})\}$, he uses the mix-multiplicative homomorphism to compute $\mathcal{E}(xy_1)$ and $\mathcal{E}(\bar{x}y_0)$.

$$\begin{aligned}\mathcal{E}(xy_1) &= \mathcal{E}(x)^{y_1},\\ \mathcal{E}(\bar{x}y_0) &= \mathcal{E}(\bar{x})^{y_0}.\end{aligned}$$

He then computes the encrypted value of f using the additively homomorphic property:

$$\begin{aligned}\mathcal{E}(f(x, y_0, y_1)) &= \mathcal{E}(xy_1) \times \mathcal{E}(\bar{x}y_0),\\ &= \mathcal{E}(xy_1 + \bar{x}y_0).\end{aligned}$$

The user finally gets the result, and decrypts it to find f:

$$\begin{aligned}\mathcal{D}(\mathcal{E}(f(x, y_0, y_1))) &= xy_1 + \bar{x}y_0,\\ &= \begin{cases} y_0 & \text{if } x = 0 \text{ was selected,}\\ y_1 & \text{if } x = 1 \text{ was selected.}\end{cases}\end{aligned}$$

So far, we have constructed a very basic OT_1^2 protocol, where the user receives the y_i of her choice without getting any information about $y_{i\oplus 1}$, where $\oplus$ is the binary XOR function. On the other hand, the server does not learn anything about the user's choice if the underlying cryptosystem is computationally secure. Nontheless, the challenge here is to extend this basic construction efficiently to large databases.

4.4 The Main OT_1^n Protocol

Let the size of the database be n, and $m = \log n$. The vector $\vec{x} = (x_{m-1}, \cdots, x_0)$ denotes the user's inputs, and $\vec{y} = (y_{n-1}, \cdots, y_0)$ denotes the database. The variable y_i is the $(i+1)$'th entry from the database. For this database of size n, the data-retrieval function is generalized to f given in (4.3).

$$\begin{aligned} f(\vec{x}, \vec{y}) =& (\bar{x}_{m-1} \cdots \bar{x}_1 \bar{x}_0) y_0 \\ &+ (\bar{x}_{m-1} \cdots \bar{x}_1 x_0) y_1 \\ &\vdots \\ &+ (x_{m-1} \cdots x_1 x_0) y_{n-1}, \end{aligned}$$

$$f(\vec{x}, \vec{y}) = \vec{x}_0 y_0 + \vec{x}_1 y_1 + \cdots + \vec{x}_{n-1} y_{n-1}. \tag{4.3}$$

A naive way to securely compute f is to encrypt the coefficients $\vec{x}_i$ in (4.3), and pass them to the server (Srvr):

$$Usr : \{\mathcal{E}(\vec{x}_0), \cdots, \mathcal{E}(\vec{x}_{n-1})\} \longrightarrow \text{Srvr}$$

However, this requires $\mathcal{O}(n)$ communications and computations for the user, which is usually computationally limited. A more communication-preserving method is to transmit $\vec{x} = (x_{m-1}, \cdots, x_0)$ and their opposites $\vec{\bar{x}} = (\bar{x}_{m-1}, \cdots, \bar{x}_0)$ in *encrypted form* to the server. Then, the server has to compute the coefficients $\vec{x}_i$ of f in (4.3). It should be noted that this process must not reveal any information about $\vec{x}$ or $\vec{\bar{x}}$ to the database holder. In this way, we achieve a communication-logarithmic OT_1^n protocol, since the user is just transmitting $\vec{x}$ and $\vec{\bar{x}}$ to the other side.

This, however, impose some extra computations for the database holder. To evaluate the encrypted coefficients $\vec{x}_i$ of f in (4.3), the database holder has to perform m homomorphic multiplications for each coefficient, and totally, mn homomorphic multiplications have to be performed to calculate all the coefficients. The most expensive operation in homomorphic cryptosystems is usually the homomorphic multiplication, since it requires several exponentiations. Therefore, the computation complexity of this protocol is roughly $\mathcal{O}(n \log n)$ for the server. We further reduce the computation complexity to $\mathcal{O}(n)$ by using a binary tree expansion to compute f.

As an example, for an eight-entry database ($n = 8$ and $m = \log(8) = 3$), the data-retrieval function is given as follows:

$$f(\vec{x}, \vec{y}) = (\bar{x}_2\bar{x}_1\bar{x}_0)y_0 + (\bar{x}_2\bar{x}_1x_0)y_1 + (\bar{x}_2x_1\bar{x}_0) + y_2 + (\bar{x}_2x_1x_0)y_3 + (x_2\bar{x}_1\bar{x}_0)y_4 + (x_2\bar{x}_1x_0)y_5 + (x_2x_1\bar{x}_0)y_6 + (x_2x_1x_0)y_7.$$

In this case, the number of multiplications to compute f from (x_2, x_1, x_0), $(\bar{x}_2, \bar{x}_1, \bar{x}_0)$ and $(y_7, \cdots, y_0)$ is 24.

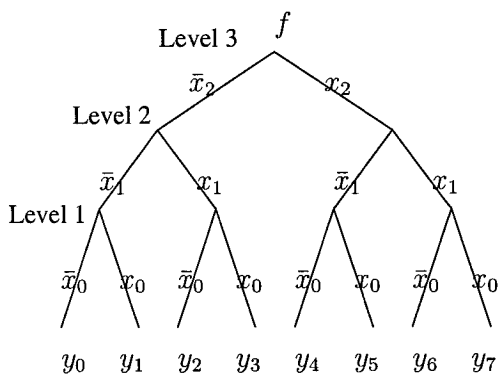

Figure 4.1: Efficiently computing f for a database of eight entries

Figure 4.1 shows how the function f for an eight-entry database should be computed in a multiplication-efficient way. One has to start by multiplying the y_i values at the bottom of graph to the values shown on each branch to which y_i is connected. Then, each pair connected to the same node has to be added together to complete Level 1.

$$\begin{aligned} \text{Level 1:} \quad & (\bar{x}_0 y_0 + x_0 y_1) \quad (\bar{x}_0 y_2 + x_0 y_3) \\ & (\bar{x}_0 y_4 + x_0 y_5) \quad (\bar{x}_0 y_6 + x_0 y_7) \end{aligned}$$

Values obtained from Level 1 are multiplied by the values on each branch to which nodes are connected. In Level 2, each pair connected to the same node has to be added together again.

$$\begin{aligned} \text{Level 2:} \quad & (\bar{x}_1(\bar{x}_0 y_0 + x_0 y_1) + x_1(\bar{x}_0 y_2 + x_0 y_3)) \\ & (\bar{x}_1(\bar{x}_0 y_4 + x_0 y_5) + x_1(\bar{x}_0 y_6 + x_0 y_7)) \end{aligned}$$

This multiplication-then-addition procedure is repeated in Level 3 with values obtained from Level 2, until the data-retrieving function is fully computed. From Figure 4.1, the function f for $n = 8$ is modified as follows:

$$\begin{aligned} f(\vec{x}, \vec{y}) = \; & \bar{x}_2(\bar{x}_1(\bar{x}_0 y_0 + x_0 y_1) + x_1(\bar{x}_0 y_2 + x_0 y_3)) \\ & + x_2(\bar{x}_1(\bar{x}_0 y_4 + x_0 y_5) + x_1(\bar{x}_0 y_6 + x_0 y_7)). \end{aligned}$$

A simple calculation shows that the total number of multiplications in the modified f is reduced from 24 in the naive form to 14 in the binary tree expansion. Next, we will show how to build a secure data-retrieval scheme using the above setup and the example given.

In the OT_1^8 protocol, the chooser first has to encrypt and send her index bits along with their opposites to the database holder:

$$Usr : \{\mathcal{E}(x_2), \mathcal{E}(x_1), \mathcal{E}(x_0)\}, \{\mathcal{E}(\bar{x}_2), \mathcal{E}(\bar{x}_1), \mathcal{E}(\bar{x}_0)\} \rightarrow \text{Srvr}$$

The server completes Level 1 to Level 3 in the same way with encrypted values of x_i's and $\bar{x}_i$'s, but the arithmetic multiplications and additions are replaced with mix-homomorphic multiplications and homomorphic additions respectively. For $i = 0, 2, 4, 6$, he first computes:

$$\mathcal{E}(\bar{x}_0 y_i) = \mathcal{E}(\bar{x}_0)^{y_i}, \tag{4.4a}$$

$$\mathcal{E}(x_0 y_{i+1}) = \mathcal{E}(x_0)^{y_{i+1}}. \tag{4.4b}$$

The two adjacent branches at the bottom of Figure 4.1 should be homomorphically added together for $i = 0, 2, 4, 6$;

$$\mathcal{E}(\bar{x}_0 y_i + x_0 y_{i+1}) = \mathcal{E}(\bar{x}_0 y_i) \times \mathcal{E}(x_0 y_{i+1}). \tag{4.5}$$

In the OT_1^8, Level 1 is replaced by (4.5), of which values are then multiplied by either $\mathcal{E}(x_1)$ or $\mathcal{E}(\bar{x}_1)$ depending on their position in the graph.

$$\mathcal{E}(\bar{x}_1 \mathcal{E}(\bar{x}_0 y_0 + x_0 y_1)) = \mathcal{E}(\bar{x}_1)^{\mathcal{E}(\bar{x}_0 y_0 + x_0 y_1)}$$

$$\mathcal{E}(x_1 \mathcal{E}(\bar{x}_0 y_2 + x_0 y_3)) = \mathcal{E}(x_1)^{\mathcal{E}(\bar{x}_0 y_2 + x_0 y_3)}$$

$$\mathcal{E}(\bar{x}_1 \mathcal{E}(\bar{x}_0 y_4 + x_0 y_5)) = \mathcal{E}(\bar{x}_1)^{\mathcal{E}(\bar{x}_0 y_4 + x_0 y_5)}$$

$$\mathcal{E}(x_1 \mathcal{E}(\bar{x}_0 y_6 + x_0 y_7)) = \mathcal{E}(x_1)^{\mathcal{E}(\bar{x}_0 y_6 + x_0 y_7)}$$

At Level 2, the following results are available:

$$\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_0 + x_0y_1) + x_1\mathcal{E}(\bar{x}_0y_2 + x_0y_3)) =$$
$$\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_0 + x_0y_1)) \times \mathcal{E}(x_1\mathcal{E}(\bar{x}_0y_2 + x_0y_3)),$$

$$\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_4 + x_0y_5) + x_1\mathcal{E}(\bar{x}_0y_6 + x_0y_7)) =$$
$$\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_4 + x_0y_5)) \times \mathcal{E}(x_1\mathcal{E}(\bar{x}_0y_6 + x_0y_7)).$$

Finally, the values from the previous steps are homomorphically multiplied by $\mathcal{E}(x_2)$ and $\mathcal{E}(\bar{x}_2)$, then homomorphically added together.

$$\mathcal{E}(f) = \mathcal{E}(\bar{x}_2)^{\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_0+x_0y_1)+x_1\mathcal{E}(\bar{x}_0y_2+x_0y_3))}$$
$$\times \mathcal{E}(x_2)^{\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_4+x_0y_5)+x_1\mathcal{E}(\bar{x}_0y_6+x_0y_7))}.$$

$$\mathcal{E}(f) =(\bar{x}_2\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_0 + x_0y_1) + x_1\mathcal{E}(\bar{x}_0y_2 + x_0y_3))$$
$$+x_2\mathcal{E}(\bar{x}_1\mathcal{E}(\bar{x}_0y_4 + x_0y_5) + x_1\mathcal{E}(\bar{x}_0y_6 + x_0y_7)).$$

Suppose that the user has chosen the fifth entry (y_4) from the database. In order to retrieve y_4, she has to send $\{\mathcal{E}(1), \mathcal{E}(0), \mathcal{E}(0)\}$ and its opposite $\{\mathcal{E}(0), \mathcal{E}(1), \mathcal{E}(1)\}$ to the server. She, in return, receives the following message:

$$\mathcal{E}(f) = \mathcal{E}(\mathcal{E}(\mathcal{E}(y_4))).$$

The user then has to decrypt $\mathcal{E}(f)$ three times (log(8)) to retrieve y_4. In the above

example, we have described the core setup of our communication-efficient OT protocol. We are now ready to extend this to arbitrarily large databases of size n, and design the first OT_1^n protocol with $(\mathcal{O}(\log(n)))$ communication complexity. The data-retrieval function (f) has to be computed as graphically shown in Figure 4.2, in order to save the number of homomorphic multiplications.

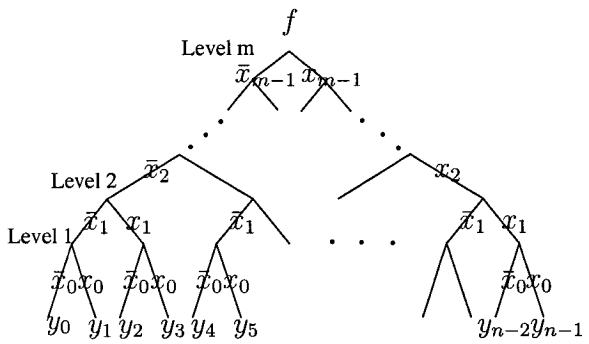

Figure 4.2: Efficiently computing f for a database of n entries

In order to privately compute the function f given in Figure 4.2, the user and the server jointly run the following protocol:

1. The user sends encrypted values of her inputs to the server;

$$Usr : \{\mathcal{E}(x_{m-1}), \cdots, \mathcal{E}(x_1), \mathcal{E}(x_0)\} \longrightarrow \text{Srvr}$$

   If the server cannot compute the opposite of $\mathcal{E}(x_i)$ without knowledge of x_i, the opposites should be transmitted separately.

$$Usr : \{\mathcal{E}(\bar{x}_{m-1}), \cdots, \mathcal{E}(\bar{x}_1), \mathcal{E}(\bar{x}_0)\} \longrightarrow \text{Srvr}$$

2. For $i = 0, 2, 4, \cdots, n-2$, the server computes $\mathcal{E}(\bar{x}_0 y_i)$ and $\mathcal{E}(x_0 y_{i+1})$;

$$\mathcal{E}(\bar{x}_0 y_i) = \mathcal{E}(\bar{x}_0)^{y_i}, \tag{4.6a}$$

$$\mathcal{E}(x_0 y_{i+1}) = \mathcal{E}(x_0)^{y_{i+1}}. \tag{4.6b}$$

Then every two values of (4.6a) and (4.6b) must be homomorphically added together. The results of Level 1 are given in (4.7), for $i = 0, 2, 4, \cdots, n-2$.

$$\mathcal{E}(\bar{x}_0 y_i + x_0 y_{i+1}) = \mathcal{E}(\bar{x}_0 y_i) \times \mathcal{E}(x_0 y_{i+1}) \tag{4.7}$$

At the next level, the server performs the homomorphic multiplication-then-addition process with the results obtained from Level 1. This procedure continues until the encrypted value of the data-retrieval function ($\mathcal{E}(f)$) is fully computed. Finally, the database holder sends back the result to the user.

3. The user decrypts the result m times to recover the entry, which she has asked for.

Note that even if a malicious user sends false values of the opposites of her input to the server, she cannot receive more information from the database. For instance, suppose that she sends all encrypted ones both as her input and its opposite to the server. She, hence, receives the sum of all entries in the database:

$$f = y_0 + y_1 + \cdots + y_{n-1}.$$

So, she can easily receive some combinations of entries in the database. But, she cannot retrieve more than one entry per transaction. Assume that she receives $y_i + y_j$ in a transaction and wants retrieve both y_i and y_j. In order for her to find exact values of both y_i and y_j, she has to obtain either y_i or y_j unless these values are not completely

independent of each other. This indeed requires the user to run the OT protocol one more time to obtain y_i (or y_j), or receive the extra information by other means, such as asking from other users.

In conclusion, the user has to run the OT_1^n protocol n times, in order to retrieve all n values in the database – we do not consider the case where the user obtains the extra information asking from other users. Since, the user cannot get more than one entry from the database per each round of the protocol, the server is totally secure against malicious users or adversaries with the users' view. On the other hand, the chooser's security has its roots in the security of the underlying homomorphic cryptosystem and is computationally secure as long as the database holder cannot break the cryptosystem with polynomially-bounded effort.

In this work, we have built our protocol on a more relaxed notion of security for the database holder, where some information about the database may leak if the chooser acts maliciously and sends wrong inputs to the server. The information leakage does not compromise the server's security if data in the database are independent of each other, and appear random to the chooser. A more acceptable solution is to get the database holder to compute the opposites of the user's inputs, in order to avoid false inputs. This also reduces the communication overhead even more, since the user does not have to send the opposites. Aside from the added the computation overhead for the server, this approach depends totally on the underlying cryptosystem, as the server has to use the additive-homomorphic property to find $\mathcal{E}(\bar{x}_i)$ from $\mathcal{E}(x_i)$:

$$\mathcal{E}(\bar{x}_i) = \mathcal{E}(x_i + 1) = \mathcal{E}(x_i) \times \mathcal{E}(1).$$

For all $x_i \in \{0, 1\}$, the homomorphic addition has to defined over $\mathbb{Z}_2$, thus we leave this option open for designers to decide how to balance the communication-computation trade-offs, and what homomorphic cryptosystem to use.

4.5 Conclusions

In this chapter, we proposed a single-database, non-interactive OT_1^n scheme, which was computationally secure for the user and information-theoretically secure for the server. The proposal was optimal in terms of both communication and computation. Our optimal OT_1^n protocol required $\mathcal{O}(\log n)$ communications on the user's side, and only $\mathcal{O}(1)$ for the server. The total computations for the user included the work to produce and send $\log(n)$ messages, and decrypt $\log(n)$ times the output of the protocol. On the other hand, the server offered private services at the cost of approximately $\mathcal{O}(n)$ computations, without any restriction for secure co-processors or multiple replicates of the database.

The actual performance and computation complexity of our protocol was mainly based on the chosen homomorphic cryptosystem, and the arithmetics to perform a probabilistic HEF. Aside from the computational overheads, security of the underlying cryptosystem would play an important role in security of the user, so it should be chosen carefully. We recommended Paillier's cryptosystem to be used in our setup, but some other variants of Paillier's PKC, that would be more efficient in number of exponentiations such as [15, 11, 31], could also be used.

A major drawback of a homomorphic cryptosystem is the malleability of the cryptosystem. Informally, this means that it is possible for an attacker controlling the communication between the user and the server to change the inputs of the protocol and launch a denial of service attack. We did not consider this type of attacks, but one may be interested in combining verifiable homomorphic functions such as [7] with the main scheme to avoid such attacks.

More improvements of this work can come from reducing the computational overhead for the server. Although the computation complexity for the server was reduce to be linear in the size of the database, this may still be too expensive for servers of large

databases to implement OT protocols. One possible solution to this problem might be to reduce the computational overhead by pre-processing some parts of it. If the communication overhead is not an issue in an application, it may also be possible to have more interactions in order to split the computation overheads.

Chapter 5

A Computation-Saving OT Protocol

The computation complexity in OT/PIR protocols is at least linear in the size of the database, so PIR/OT protocols may not be applicable to large databases. The main focus in studying PIR/OT protocols has been to reduce the communication overhead as much as possible, as there exist some protocols that achieve logarithmic communication complexity [30, 24, 26]. However, when dealing with large databases (usually of 500,000 entries or more), we should consider the computational overhead as a prime factor in implementing PIR/OT protocols. The computational overhead of PIR/OT protocols becomes more drastic in a computational security model. In the computation model large numbers, at least one thousand-bit long, have to be used to hinder attackers of computationally bounded efforts. We believe that an OT/PIR protocol will not be made practical unless both communication and computation overheads are reduced and optimized.

To our knowledge, there have been very few works that focus on reducing the computation complexity of PIR/OT protocols. In [2], an optimal OT protocol is achieved in the presence of a secure co-processor where the total amount of online computations

for the server is less than n, but it requires $\mathcal{O}(n^2)$ offline processing. Deng et al. [12] designed an OT protocol with $\mathcal{O}(1)$ computation and communication complexity under the assumption that the whole database is available locally to users. Beimel and Ishai [5] proposed several PIR protocols with sub-linear computations and communications, but more than one server with the same database has to be present.

In this chapter, we propose a computation-saving OT_1^n protocol, where the complexity of online computations required by the server is only $\mathcal{O}(1)$, while $\mathcal{O}(n)$ computations and communications are put in the offline phase that is performed once. We achieve this without further restrictions on the whole structure of the system. That is our scheme does not require either several duplicates of the database, secure co-processors, or heavy computations by the user. Our computation-saving protocol is constructed using McEliece cryptosystem, which is based on the problem of decoding arbitrary linear codes. In Section 5.1, we give an introduction to the coding theory on which the initial protocol is based (Section 5.2). We then extend the initial protocol to the case where proxies can be used to offer PIR services (Section 5.3), and finally conclude this chapter in Section 5.4.

5.1 Introduction

In this section, we only describe the main features of McEliece cryptosystem that are mainly used in our OT protocol. For a detailed description of linear block codes, we refer the reader to Appendix B.

Definition 5.1.1 ***Linear codes:*** *An algebraic expansion of length N over a finite field $\mathbb{F}_p$ is called a linear code (N, k) if and only if p^k code words form a k-dimensional subspace of the vector space of all the N-tuples over the field $\mathbb{F}_p$. In fact, a block code is linear if and only if the modulo-p sum of two code words is also a code word.*

The Hamming distance $d(a, b)$ of two code words, $a = (a_1, \cdots, a_N)$ and $b = (b_1, \cdots, b_N)$, is the number of positions where these code words are different. The minimum distance is defined as the minimum of pairwise Hamming distances between different two code words, that is:

$$d = \min_{a,b \in C; a \neq b} d(a, b).$$

When working with binary codes defined over $\mathbb{F}_2$, we define the *Hamming weight*, shown by w, of a code word as the number of non-zero positions in the vector. It is straightforward to see the minimum distance of a code C is equal to the minimum non-zero weight of all code words. A linear code with length N, dimension k and minimum distance d is usually denoted by $[N, k, d]$. Since the code is a linear subspace of dimension k, there exist k linearly independent code words that form a basis of the subspace. The $k \times k$ matrix G is called a generator matrix of the code if k independent code words form the rows of G. A vector x of length k is mapped (encoded) into a code word c of length N through the generator matrix;

$$c = xG.$$

The generator matrix is systematic if it can be represented as $G = (I_k|A)$, where I_k is the k-by-k identity matrix and A is a $k \times (N - k)$ matrix. The $(N - k) \times N$ matrix, denoted by H, is called the *parity-check matrix* of a code if it satisfies the following:

$$GH^T = 0.$$

In other words, every code word $c \in C$ can be identified by H if it returns zero, when multiplied by H;

$$cH^T = xGH^T = 0. \tag{5.1}$$

If the generator matrix is systematic ($G = (I_k|A)$), then the parity-check matrix can be easily obtained by setting $H = (A|I_{N-k})$. It can be shown that there always exists a polynomial-time algorithm to obtain G from H and vice-versa. As the parity-check matrix is mainly used in error correction mechanisms, cH^T is referred to as the *syndrome* of the vector c. If an error vector e is added to a code word c, from (5.1) we conclude that the syndrome of $y = c + e$ is zero if and only if v is a code word.

$$yH^T = cH^T + eH^T = eH^T. \tag{5.2}$$

Equation (5.2) gives a method of solving $(N - k)$ equations with N unknown variables (the error vector e) to correct the error. One can show that if the minimum distance of a code is $d = 2t + 1$, then error vectors of weight at most t can be uniquely determined from (5.2). The value t explicitly represents the maximum number of errors a code can correct.

In order to build any public-key cryptosystem, it is mandatory to use hard problems (preferably NP-complete), which have a rather wide subclass of problems of polynomial complexity. The problem of decoding in an arbitrary linear code can be used as a basis to construct a public-key cryptosystem [21], since

1. the problem of decoding is an NP-complete, and
2. there is an exponentially large class of alternate codes for which polynomial decoding algorithms exist (if the number of errors is less than half of the code distance).

There are many cryptosystems of this kind, but we only focus on McEliece public-key cryptosystem [27], which was the first cryptosystem based on error correcting codes.

5.1.1 McEliece Cryptosystem

McEliece cryptosystem [27] is a public-key cryptosystem based on linear error-correcting codes. Let $C[N, k, d]$ be a linear code with length N, dimension k and minimum distance d. Let the generator of the code be a k-by-N matrix denoted by $G_{k\times N}$ for which an efficient decoding algorithm exists. The public encryption matrix is defined as follows:

$$E_{k\times N} = SGP, \tag{5.3}$$

where $S_{k\times k}$ is a randomly selected invertible matrix, and $P_{N\times N}$ a random permutation matrix. The encryption function is given in (5.4), where $x \in \mathbb{F}_{q^k}$ is the message and $y \in \mathbb{F}_{q^N}$ is the corresponding ciphertext. The error vector $e \in \mathbb{F}_{q^N}$ is chosen at random, such that its weight must be less than or equal to $\frac{d-1}{2}$, in order to be correctable by the receiver.

$$y = xE + e. \tag{5.4}$$

In McEliece cryptosystem, the matrix E and the minimum distance d are the public knowledge, while the efficient decoding algorithm and the matrices S and P are the receiver's private key. The ciphertext y is decrypted by knowing the private permutation P, decoding algorithm and the inverse of the random matrix S. The receiver first calculates $yP^{-1} = xSG + eP^{-1}$ and then decodes the result. It should be noted that P^{-1} is also a permutation, and does not change the weight of the error vector ($w(eP^{-1}) = w(e)$). Thus, the error eP^{-1} is still correctable. The receiver finally

recovers the message x by multiplying the output of the decoding algorithm by S^{-1};

$$x = (xS)S^{-1}.$$

There exist several types of attack on McEliece cryptosystem that limit the number of suitable codes for such cryptosystem [21]. A direct attack searches for effective methods to decode an arbitrary linear code, without the attacker using any information about the structure of the code on which the cryptosystem is based. Another attack is entirely based on this information. Indeed, there do not exist many suitable classes of codes suitable for cryptography, as the encryption transformation must perfectly hide the characteristics of the underlying coding system in order to be resilient to the second type of attack. In addition, the size of the code should be large enough to resist any brute-force attack. McEliece has recommended the use of Goppa codes $[1024, 524, 50]$, and there has been ongoing research to find other secure codes with efficient decoding algorithms that can be realized in practice.

5.2 The Initial Computation-preserving Protocol

In this section, we describe the main construction of our computation-saving PIR protocol assuming the existence of an efficient and secure McEliece cryptosystem. The goal is to minimize online computations as much as possible, and make it comparable to the regular data-retrieval case where no privacy is provided. Without loss of generality, we assume that the database of size n can be represented by an n-by-k matrix B in

which b_{ij} denotes the j'th bit of the i'th entry in the database.

$$B_{n\times k} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nk} \end{bmatrix}$$

The variable k is fixed and defined by the underlying McEliece cryptosystem – for example, $k = 524$ for the Goppa code $[1024, 524, 50]$. For large databases of entries more than k bits, the rest of the database should be represented by separate matrices. For simplicity, we describe the initial protocol assuming that the whole database can be represented by B, and later extend it to handle larger databases.

Our approach to reduce the computations in PIR/OT protocols is to put all the processing in the setup phase that is performed only once. To achieve this, we get the user to privately permute the database in the setup phase and encrypt the output of the protocol. So, the user and the server has to find a protocol that securely computes the following function:

$$f = \pi_c BE, \tag{5.5}$$

where π_c is an arbitrary permutation known only to the user and the matrix B is the server's database. The matrix E is the McEliece encryption matrix of the user which is known to the general public. The function f is securely computed if the user learns nothing about the database B (except the size of the database), while her private permutation is hidden from the server. Such a protocol is given below.

The Initial Protocol:

The server, who owns the database B, first computes BER where $E_{k\times N}$ is the McEliece encryption matrix of the user given in (5.3), and $R_{N\times N}$ is a random invertible matrix. The matrix R has to be chosen such that each row in R^{-1} has at most m non-zero elements, where $m \leq \frac{d-1}{2}$. We later show that m is a security parameter which needs to be agreed upon by both parties (see Section 5.2.1). It is worth noting that the user cannot decrypt and access the database without knowing either B or R. The server (Srvr) sends (BER) to the user (Usr).

$$\text{Srvr}: \{BER\} \longrightarrow \text{Usr}$$

At this stage, we require the user to store (BER) temporarily. She chooses a random permutation π_c and a random k-by-N matrix R', such that there are at most $\frac{d-1}{2m}$ non-zero elements in each row of R'. Then, she computes $\pi_c(BER + R')$ on her local machine, and keeps the permutation π_c private. She can erase the stored values (except a description of her private permutation) after she has sent back the result to the server.

$$\text{Usr}: \{\pi_c(BER + R')\} \longrightarrow \text{Srvr}$$

The server finally removes his private matrix R after receiving $\pi_c(BER + R')$;

$$f' = \pi_c(BER + R')R^{-1} = \pi_c BE + R''. \tag{5.6}$$

Hereafter, the setup phase is complete and the user can access the database privately, since π_c is known only to the user. Suppose that the user is asking for the i'th entry of the database, she simply computes and sends $x_i\pi_c^{-1}$, where x_i is a vector of all zeros except in the i'th position. There is only one non-zero element in the vector $x_i\pi_c^{-1}$ and the server can immediately return the proper value from the pre-processed data, therefore the online phase requires only $\mathcal{O}(1)$ computations.

$$x_i\pi_c^{-1}(\pi_c BE + R'') = x_i\pi_c^{-1}(\pi_c BE) + x_i\pi_c^{-1}(\pi_c R'R^{-1}), = x_i BE + x_i R'R^{-1},$$

$$\mathcal{E}(y_i) = x_i BE + Err. \tag{5.7}$$

The user receives $\mathcal{E}(y_i)$ and decrypts it to retrieve $y_i = x_i B$. Since there is only one non-zero element in x_i, the maximum number of ones in Err is identical to the weight of the row in $R'R^{-1}$ to which x_i is pointing. Random matrices R^{-1} and R' are designed such that, they respectively carry m and $\frac{d-1}{2m}$ non-zero elements in each row. Therefore, the maximum number of non-zero elements in each row of $R'' = R'R^{-1}$ is $\frac{d-1}{2}$. We conclude that the vector Err, independent of x_i, is a correctable error vector and consequently f' given in (5.6) is functionally the same as f given in (5.5).

Further Generalization:

The computation-saving protocol for large databases of values larger than k bits is slightly different than the one given above. As mentioned earlier, the matrix B is an $n \times k$ matrix in the initial protocol that involves only k bits of each entry in the database. If there are more than k bits to be retrieved from the database – let's say l is the maximum bit-length of data in the database – then the whole database should be partitioned as follows:

$$\{B_1, B_2, \cdots, B_{l/k}\}$$

where B_i is the i'th k-bit block of the whole database. Consequently, the initial protocol is modified as follows.

For $j = 1, 2, \cdots, l/k$; the server encrypts B_j and multiplies the result by R_j, which

is a random matrix such that every row of R_j^{-1} has at most m non-zero elements.

$$\{B_1ER_1, B_2ER_2, \cdots, B_{l/k}ER_{l/k}\}$$

In order to increase security of the server, the server's random matrices (R_j's) should be different from each other, so that if the user can guess one of the R_j's, she cannot receive more than k bits of the database. Each R_j reveals k bits of the whole database, so the user has to find all R_j's to receive any meaningful information about the database. For $j = 1, 2, \cdots, l/k$, the user first adds a randomly selected matrix R'_j to B_jER_j and then multiplies it by her private (permutation) matrix π_c;

$$\{\pi_c(B_1ER_1 + R'_1), \cdots, \pi_c(B_{l/k}ER_{l/k} + R'_{l/k})\}$$

Each matrix R'_j has at most $\frac{d-1}{2m}$ non-zero elements. Once the user has sent the result back to the server, the setup phase is complete. Hereafter, the user simply hides her query (x_i) by applying the private (permutation) matrix π_c^{-1};

$$\text{Usr}: \ x_i\pi_c^{-1} \longrightarrow \text{Srvr}$$

The server responds to the user's query ($x_i\pi_c^{-1}$) based on the pre-processed data. For $1 \leq j \leq l/k$:

$$\begin{aligned} x_i\pi_c^{-1}\pi_c(B_jER_j + R'_j)R_j^{-1} &= x_i\pi_c^{-1}(\pi_cB_jE) \\ &\quad + x_i\pi_c^{-1}(\pi_cR'_jR_j^{-1}), \\ &= x_iBE + x_iR'R^{-1}, \end{aligned}$$

$$y_{j,i} = x_iB_jE + Err_j. \tag{5.8}$$

As shown in (5.8), the error vector Err_j is independent of π_c, therefore we can remove the restriction of π_c being a permutation matrix, and further generalize the above protocol by allowing π_c to be any invertible matrix. This generalization increases the security of the user, since the attacker, now, has to guess a random invertible function, instead of a permutation. However, this imposes more computational overheads on the server, as there are more than one non-zero elements in each row of π_c and the server has to respond to each of them separately. For example, if π_c^{-1} is a matrix with t ones in each row, the computation complexity for the server is $\mathcal{O}(t)$. The computational overheads can be made very close to the regular case with no privacy, by restricting t to small values $t \ll n$.

Although, the security of the user is based on the security of the McEliece cryptosystem, the privacy of her choice is also based on the right selection of π_c and the random matrix R'. There are other possible attacks that do not require decrypting the message to find the user's choice, as described in the next subsection.

5.2.1 Security Evaluation

As described in the initial protocol, the server first chooses a random invertible matrix (R^{-1}), every row of which has at most m non-zero elements, and then hides the encrypted values of the database by R. Having received BER, the user could access the entire database if she were able to find B by knowing E, BER and some entries of the database (a history of previous interactions). A simple calculation shows that it is impossible for the user to recover R by random guessing if she has no information about the database (information-theoretically secure). However, the user is able to recover $R_{N\times N}$ if she knows N linearly independent entries of the database (denoted by B_{hst}). Without loss of generality, we assume that the user knows the first N entries of database $B_{n\times k}$, which we represent by $B_{N\times k}$. She simply encrypts B_{hst} with her

public-key matrix and then solves for R from the following equation:

$$R = (B_{hst}E)^{-1}(B_{N\times k}ER).$$

To avoid this attack, every N block of the database should be hidden by a different random matrix. Another countermeasurement is also possible by limiting the user to N queries only. In this way, the server requires the user to repeat the protocol from the beginning after N queries have been made. In other words, the size of the database – denoted by n – that is accessible in each OT protocol is limited by the underlying coding system ($n < N$).

The security of the user is assured if the attacker cannot find the user's private transformation on the database (π_c). We examine privacy of the user against two types of attack. In the first category, the adversary tries to find π_c directly from $\{\pi_c(BER+R')\}$. It is computationally infeasible for the adversary to retrieve π_c from $\{\pi_c(BER+R')\}$ if he does not know the decoding algorithm. Since, he has to apply some cryptanalyze the encryption function given in (5.6) to obtain $\pi_c B$ from the ciphertext, and then compare it with B to find π_c.

In the second type of attack, the adversary may first try to guess $\pi_c R'$ and remove it from $\{\pi_c(BER+R')\}$ to find the permutation. For the code [1024,524,101] with $m = 5$, R' is a 1024×1024 random matrix that has $\dfrac{101-1}{10} = 10$ ones in each row. Thus, there are approximately 2^{88} (precisely $1024\binom{1024}{10}$) different choices for R', and it is computationally impossible for the attacker to randomly guess $\pi_c R'$. However, it should be noted that $\pi_c R'$ is a matrix of $\dfrac{d-1}{2m}$ ones in each row. Therefore, it is easy for the adversary to find the permutation π_c if there is any pattern in $\{\pi_c(BER+R')\}$ – each two entries in (BER) are different at least in d positions, while R' masks only m bits of the database, thus a pattern may reveal the permutation. To nullify this attack, we strongly recommend the user to apply an arbitrary invertible function instead of a

permutation to the database, as discussed in the generalized protocol. However, any invertible function other than permutation increases the computational overheads of the protocol.

We showed that the larger is the number of non-zero elements in R' (equivalently the smaller the m), the more secure will be the user. Nevertheless, for small values of m, it may be possible for the user to guess R given that there are very few ones in R. Hence, security of both parties directly depends on the variable m. Table 5.1 shows the amount of work needed in an exhaustive search to find either the server's random matrix or the user's random permutation.

If the variable $Num(N, m)$ denotes the number of invertible matrices over $\mathbb{F}_2$ with at most m ones in each row, it is formulated in (5.9).

$$Num(N,m) = \prod_{i=0}^{2^i<\binom{N}{m}} \left(\binom{N}{m} - 2^i + 1\right). \tag{5.9}$$

This formula comes from the standard method that describes how N rows (vectors) over $\mathbb{F}_p$ can be chosen such that they will form an invertible matrix. Similarly, we count the number of all choices for such a matrix. There are no restrictions on the first vector, so there are $\binom{N}{m}$ possibilities for the first row of m ones. Assume that i linearly independent rows are constructed, and the $(i+1)$-st row can be constructed in $(\binom{N}{m} - 2^i + 1)$ ways by avoiding linear combinations of all previous rows. It should be noted that a vector of all zeros is already excluded in computing $\binom{N}{m}$, therefore one should be added to 2^i when counting the number of linear combinations of i rows over $\mathbb{F}_2$. By induction, the number of $N \times N$ invertible matrices over $\mathbb{F}_2$ which have at most m ones in each row is $Num(N, m)$ given in (5.9). Clearly, $\binom{N}{m}^2 < Num(N, m)$ when N is large. In Table 5.1, we take $\binom{N}{m}^2$ as the minimum number of possible $N \times N$ invertible matrices with m ones in each row. For the case that $m = 5$, there are approximately 2^{86} choices for R, which is far beyond the computational power of any

polynomially-bounded adversary searching exhaustively for R – we have assumed that 2^{80} is computationally hard enough to resist any brute force attack.

Table 5.1: Number of possible choices for random matrices in the initial protocol

m	d =51		d =101		d =201	
	Srvr	Usr	Srvr	Usr	Srvr	Usr
2	2^{38}	2^{101}	2^{38}	2^{176}	2^{38}	2^{295}
5	2^{86}	2^{53}	2^{86}	2^{88}	2^{86}	2^{148}
10	2^{156}	2^{29}	2^{156}	2^{53}	2^{156}	2^{88}
12	2^{182}	2^{29}	2^{182}	2^{48}	2^{182}	2^{74}

As it can be observed from Table 5.1, the larger are the values of m and d, the more secure would be both parties against a brute-force attack. As an example, the amount of work required to guess the user's random matrix in a protocol with $(d = 201, m = 5)$ is 2^{95} $(2^{148}/2^{53})$ times more than a protocol with $(d = 51, m = 5)$.

The setup phase requires $\mathcal{O}(n)$ computation and communication. That is the cost to be paid for private information retrieval services at the time users subscribe to the server. This protocol enables any server to offer private services without changes to the whole structure of the system (all in software). However, the setup phase may not be suitable for servers with a large number of subscribers, since the server has to store $\mathcal{O}(n)$ messages locally for every single user asking for private services. In the following section, we rearrange the initial protocol in such a way that more users can receive private services, while reducing the storage space per user.

5.3 The Main Protocol With Proxies

In the naive way, where users work independtly and perform (the generalized form of) the initial protocol with the server, act as a proxy for other users. That is, the proxy first computes $\pi_c BE + Err$ with the server, and then users send their query (x_i) to the proxy, which they have trust on. The proxy returns the corresponding entry ($y_i = x_i B$) to the user, without revealing it to the server. If we denote the proxy by P, Figure 5.1 depicts how users can privately access the server through a trusted proxy.

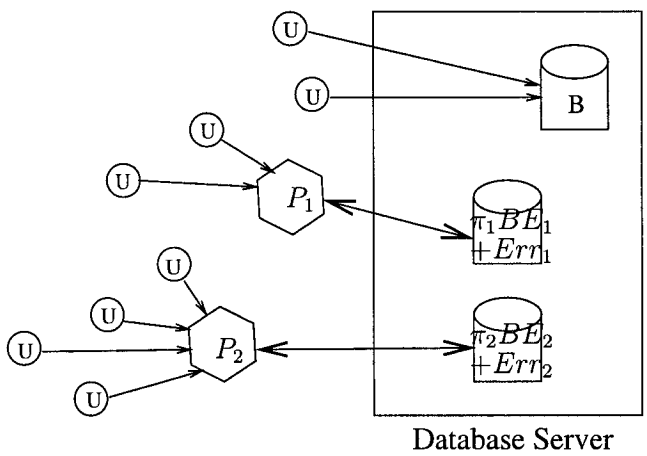

Figure 5.1: Privately accessing the server through trusted proxies

Figure 5.1 shows that some users still can contact the server directly to run the initial protocol to retrieve data privately, if they cannot trust any of proxies. In the naive OT protocol with proxies, users are secure, as long as the proxy is honest and does not reveal the queries to the server. However, security of the user may be easily compromised if the proxy acts maliciously and reveal the query to the server. Another shortcoming of the naive way of using proxies is that any pattern in proxy's retrieving data may leak some information about queries. Next, we propose another protocol with proxies that addresses these challenges.

The Protocol with Threshold Privacy:

Here, we assume that there exist several (non-trusted) proxies $(P_1, P_2, \cdots, P_s)$ that are specifically designed to provide privacy-preserving services and have enough storage to store a replicate of the database. These proxies run a variant of the initial protocol as follows:

In the setup phase, each proxy P_j receives BE_jR_j from the server, where E_j is the P_j's public-key matrix. The matrix R_j is the server's secret invertible matrix that is selected such that each row of R_j^{-1} has at most m non-zero elements ($m \leq \frac{d-1}{2}$). The server has to save (a description of) R_i's on his local machine for future use. In the online phase, we use a simplified version of [5] to provide threshold security for users. In order to privately retrieve the i'th entry of the database, denoted by y_i, each user randomly splits her query into $\{x_{i1}, x_{i2}, \cdots, x_{is}\}$. The variables x_{ij} has to be selected, such that their cumulative sum ($x_i = x_{i1} + x_{i2} + \cdots + x_{is}$) is vector of all zeros except at the i'th position. Then, the user select s proxies from a list of proxies who have done the setup phase with the server. For $1 \leq j \leq s$, the user sends each x_{ij} to P_j.

Each proxy computes $q_{ij} = x_{ij}(BE_jR_j) + r'$, where r' is a random error vector of at most $\frac{d-1}{2m}$ non-zero elements. The variable q_{ij} is sent to the server who then removes his private matrix from the message.

$$\begin{aligned} q_{ij}R_j^{-1} &= (x_{ij}(BE_jR_j) + r')R_j^{-1}, \\ &= x_{ij}(BE_jR_j)R_j^{-1} + r'R_j^{-1}, \\ \mathcal{E}(y_{ij}) &= x_{ij}BE_j + Err_j, \end{aligned}$$

The variables r' and R_j^{-1} are designed, such that Err_j is an error vector of $\frac{d-1}{2}$ non-zero elements, therefore it is correctable. Each proxy, P_j, decrypts $\mathcal{E}(y_{ij})$ and returns the result (y_{ij}) to the user. Finally, the user recovers y_i by adding up all the y_{ij}'s from proxies;

$$\begin{aligned} y_i &= y_{i1} + y_{i2} + \cdots + y_{is} \\ &= x_{i1}B + x_{i2}B + \cdots + x_{is}B, \\ &= (x_{i1} + x_{i2} + \cdots + x_{is})B, \\ &= x_iB. \end{aligned}$$

It should be noted that if the server, acting maliciously, tries to reveal the choice of the user, he has to a) find the list of proxies collaborating on behalf of the user, b) have all the proxies reveal the shares they have received from the user to him. If any of these steps fail, the user will remain private.

The model shown in Figure 5.2 represents an adaptive platform, where both non-private and private services are available for users. To privately retrieve data from the server, some users may run the initial protocol to directly access the database in a private manner. Some other users may privately access the database through a proxy they have completely trusted. But some other users, who are not willing to either go through the heavy computations and communications in the setup phase of the initial protocol, or have complete trust on one proxy, can run the threshold scheme with proxies to privately access the database. In such a scheme, the user is secure as long as not all proxies are colluding against her (threshold privacy), while the proxies are computationally secure depending on the underlying cryptosystem. The server, on the other hand, is still information-theoretically secure.

This structure is more promising than previous approaches to make OT/PIR schemes

practical. This is specifically true in the presence of several proxies that provide privacy-preserving services to their subscribers. Users can receive the privacy-preserving service quickly through a set of proxies they trusts. The combination of proxies can be changing to make it more difficult for proxies and the server to act against users. The main advantage of this approach is that the database server is still the only entity who controls the whole database even on (remote) proxies.

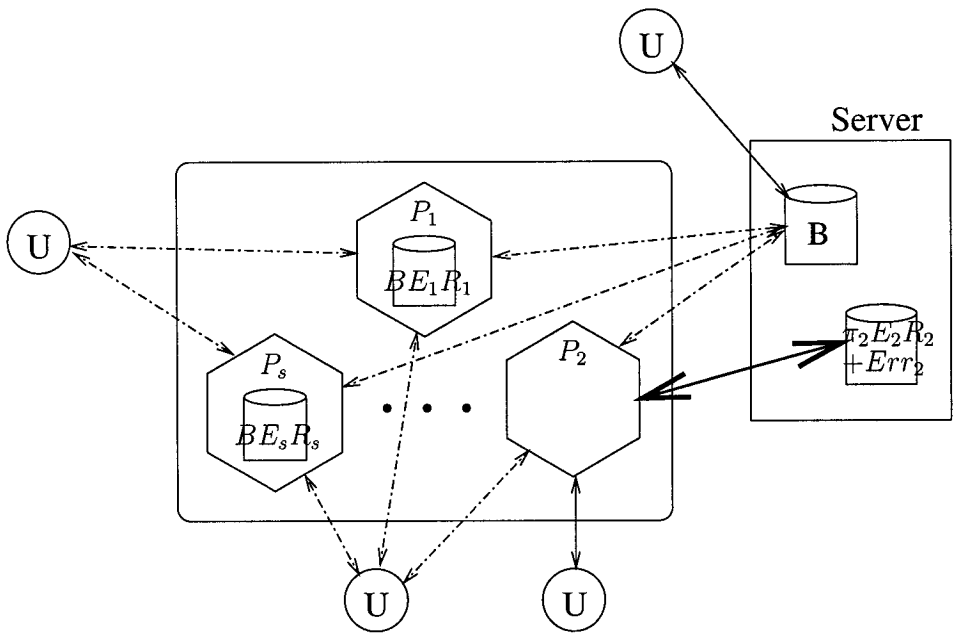

Figure 5.2: Accessing the server through multiple untrusted proxies

In this model, however, the total online computations is increased as the threshold method is applied. If we assume that the maximum number of non-zero elements in each query (x_{ij}) is k, the total online computational complexity for each proxy will be $\mathcal{O}(k)$. On the other side, the server has to perform $\mathcal{O}(sk)$ online computations if s proxies are involved in the protocol. Clearly, this protocol would not be computation-efficient PIR/OT protocol for both proxies and the server, unless $k \ll n$ and $sk \ll n$

respectively. We also require each proxy to store an encrypted replicate of the whole database locally, while updating the replicates on remote proxies is done by repeating the setup phase.

5.4 Conclusions

In this chapter, we proposed PIR/OT protocols with $\mathcal{O}(1)$ online computations after the setup phase. However, the setup phase required $\mathcal{O}(n)$ computations and communications. In the initial protocol, the user was computationally secure against an adversary, while the server was information-theoretically secure. We further improved the initial protocol and combined it with the threshold scheme of Beimel and Ishai [5], in order to balance the computation-storage tradeoffs for the server. In this setup, we suggested the use of multiple proxies that were specifically designed to offer privacy-preserving services. A few proxies could be used to provide privacy-preserving services to thousands of subscribers, without sacrificing users' security. The main protocol with s proxies required $\mathcal{O}(k)$ computations for each proxy, and $\mathcal{O}(sk)$ computations for the server. The computational overheads of the protocol would be sublinear in the size of the database (n) if $k \ll n$ and $sk \ll n$. In the computation-efficient protocol with threshold privacy, more users could privately access the database with a fixed computation-communication complexity. This proposed PIR/OT protocol is advantageous over the initial scheme as major overheads of the setup phase would be on proxies who service large number of users.

In our model, the server was perfectly secure against both users and proxies, while the server still uniquely controlled the database over proxies. The users were private unless all proxies were bribed to act against the user. Our protocol was mainly built on McEliece public-key cryptosystem, which outperforms other public-key cryptosystems like RSA more than 8-10 times in terms of number of elementary operations ([21]).

The main disadvantage of the code based cryptosystem is the size of the public-key matrix, but such a loss may be acceptable in PIR/OT protocols where storage is not an issue.

In our protocol, every time the database might be modified, replicates of the database on remote proxies had to be updated correspondingly. Refreshing the encrypted replicates remotely required the setup phase to be repeated for all proxies. A further research in this direction has to overcome this challenge, and one should find a way to avoid repeating the setup phase.

Chapter 6

Conclusions and Future Work

In this thesis, we studied Private Information Retrieval (PIR) and Oblivious Transfer (OT) protocols with three different approaches as shown in Table 6.1.

Table 6.1: Summary of the proposed protocols

Protocol	Computation		Communication	
	Online	Offline	Online	Offline
Perfectly secure dot-product	$\mathcal{O}(n)$	–	$\mathcal{O}(n^2)$	–
Computationally secure PIR/OT	$\mathcal{O}(n)$	–	$\mathcal{O}(\log n)$	–
Computationally secure PIR/OT	$\mathcal{O}(t)$	$\mathcal{O}(n)$	trivial	$\mathcal{O}(n)$

We proposed the first non-interactive, secure dot-product protocol that is widely used in secure data-mining protocols and other SFE schemes. The secure dot-product protocol is mainly based on trace functions over finite fields that can be efficiently computed. It was also shown that the computational overheads of the information-theoretically secure dot-product was $\mathcal{O}(n^2)$, with $\mathcal{O}(n)$ computations. Furthermore, we proved the proposed secure dot-product protocol to be information-theoretically secure for all contributing parties. We further examined the application of our secure dot-product protocol to PIR/OT schemes. It was suggested to improve and reduce

the communication/computation overheads of the secure dot-product protocol before applying it to PIR/OT schemes.

A separate approach was proposed to optimize the communication complexity of PIR/OT protocols for databases of size n, and to reduce it from $\mathcal{O}((\log n)^2)$ to $\mathcal{O}(\log n)$ without drastically increasing the computation overheads. It was shown that the users were computationally secure against any polynomially bounded adversary, based on any homomorphic cryptosystem. Unconditional security was also provided for the server. In the communication-saving protocol, we improved the communication complexity from $\mathcal{O}((\log(n))^2)$ to $\mathcal{O}(\log(n))$. The computational overheads in this protocol were minimized, but still linear in the size of the database.

A computation-saving protocol was designed to put the heavy computations of PIR/OT at the setup phase that was performed only once and at the beginning of the protocol. The total online computation was reduced to be sublinear in the size of the database. Linear error-correcting cryptosystems were mainly used as the basis of this computation-saving protocol. We further improved this work by using proxies that were specifically designed to provide privacy-preserving services to users. The server, in our setup, was still information-theoretically secure against proxies, while proxies were computationally secure based on the underlying McEliece cryptosystem. On the other hand, users were private, provided that not all proxies were colluding against them.

We proposed a very adaptive model in which users were allowed to choose to run the protocol with one trusted proxy, through multiple proxies or directly without any proxy. In the protocol with one proxy, security of the users was totally dependent on the proxy not to reveal the users' query. In the threshold scheme with more than one proxy, security of the user can be guaranteed if not all proxies act maliciously. For users that did not trust any of the proxies, we provided the option to run the protocol directly and perform all the required offline computations and communications with

the server.

Further improvements of this work can be in reducing the communication complexity in the setup phase of the computation-saving protocol with proxies. One may be able to combine the different approaches, we proposed here, to achieve a very efficient PIR/OT protocol in terms of both communications and computations. Reducing the amount of communications and computations required in PIR/OT protocols will definitely improve their usability in practice.

Appendix A

Finite Fields Preliminaries

A set G on which an operation '$*$' is defined is called a *group* if the following conditions are satisfied:

1. For all $a, b \in G$, $a * b \in G$.
2. The operation '$*$' is associative, that is for all $a, b, c \in G$, $(a * b) * c = a * (b * c)$.
3. The set G contains an identity e, an element such that for any $a \in G$, $a * e = e * a = a$.
4. For any element $a \in G$, there exists another element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$. The element a^{-1} is called the inverse of a.

A group is said to be *commutative* if for any $a, b \in G$, $a * b = b * a$. Now we use the group concepts to introduce another algebraic system, called a *field.*

Let $\mathbb{F}$ be set of elements on which two operations addition '$+$' and multiplication '$\cdot$' are defined. The set $\mathbb{F}$ together with two operations '$+$' and '$\cdot$' is a field if the following conditions are satisfied:

1. The set $\mathbb{F}$ is a commutative group under addition '+'. The identity element with respect to addition is called the *zero* element or the additive identity and is denoted by 0.

2. The set of non-zero elements in $\mathbb{F}$ is a commutative group under multiplication. The identity element with respect to multiplication is called the *unit* element or the multiplicative identity of $\mathbb{F}$ and is denoted by 1.

3. Multiplication is *distributive* over addition, that is for any $a, b, c \in G$, $a \cdot (b+c) = (a \cdot b + a \cdot c)$.

The number of elements in a field is called the *order* of the field, and a field with a finite number of elements in called a *finite field.* An example of a finite field is the set of integers modulo a prime number p, which is usually denoted by $\mathbb{F}_p$. The positive integer p is called the *characteristic* of a field F if for every $a \in F$, $pa = 0$. It can be proved that a finite field has always prime characteristic [37].

It is easily shown that for any positive integer n, one can extend the prime field $\mathbb{F}_p$ to a field of p^n elements [37]. The extension field of $\mathbb{F}_p$ is usually denoted by $\mathbb{F}_{p^n}$. In order to extend $\mathbb{F}_p$ to p^n, one has to choose an irreducible polynomial of degree n. The polynomial $f = a_n x^n + \cdots + a_x + a_0$, with coefficients defined over $\mathbb{F}_p$ and $a_n \neq 0$, is called irreducible if f is not divisible by any non-trivial polynomial with coefficients defined over $\mathbb{F}_p$. For instance, $f(x) = x^2 + x + 1$ is an irreducible polynomial over $\mathbb{F}_2$ that can be used to construct a field of 2^2 elements.

If f is irreducible, the residue classes of polynomials over f with coefficients in $F = \mathbb{F}_p$, shown by $F[x]/f$, form a field of characteristic p. For example, let $f(x) = x^2 + x + 1 \in F = \mathbb{F}_2[x]$. Then $F[x]/f$ has the $p^n = 2^2$ elements $0, 1, x$ and $x + 1$. Table A.1 and A.2 show the addition and multiplication operations over $\mathbb{F}_2/(x^2+x+1)$. The operation tables for the residue classes are obtained by performing the required operations with the polynomials determining the residue classes and by carrying out

reduction mod f if necessary.

Table A.1: Addition table for $\mathbb{F}_2/(x^2+x+1)$

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Table A.2: Multiplication table for $\mathbb{F}_2/(x^2+x+1)$

$\cdot$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

Clearly, there are p^n polynomials of degree less than n and coefficients in $\mathbb{F}_p$, so the order of $F[x]/f$ equals p^n. For fundamental properties of finite fields and a description of methods for constructing finite fields refer to [37, 28].

Appendix B

Linear Block Codes

By definition, a block code of length n and p^k code words is called a linear (n, k) code if and only if its p^k code words form a k-dimensional subspace of the vector space of all the n-tuples over the finite field $\mathbb{F}_p$. In fact, a block code is linear if and only if the modulo-p sum of two code words is also a code word. Since an (n, k) linear code C is a k-dimensional subspace of the vector space of all the n-tuples in $\mathbb{F}_p$, it is possible to find k linearly independent code words $g_1, g_2, \cdots, g_k$ in C such that every code word $c \in C$ is a linear combination of these k code words, that is

$$c = a_1 g_1 + a_2 g_2 + \cdots + a_k g_k,$$

where $a_i \in \mathbb{F}_p$ for $1 \leq i \leq k$. Therefore, an (n, k) linear code is completely specified by the matrix G whose rows are k linearly independent code words.

$$G = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{pmatrix},$$

and the k-by-n matrix G is called a generator of the (n, k) linear code. If x is the message to be encoded, the corresponding code word can be obtained as follows:

$$c = xG.$$

The matrix H is called a parity check matrix of the (n, k) linear code C generated by the matrix G if $GH^T = 0$. Also, the $n - k$ rows of H are linearly independent. For all code words $c \in C$, $s = cH^T = 0$, where s is called the *syndrome* of the vector c.

Example B.0.1 *Suppose the* $(7, 4)$ *linear code has the following matrix as a generator matrix:*

$$G = \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

If $x = (1, 1, 0, 1)$ *is the message to be encoded, its corresponding code word would be*

$$\begin{aligned} c = xG &= (1, 1, 0, 1) \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{pmatrix} \\ &= 1.g_1 + 1.g_2 + 0.g_3 + 1.g_4 \\ &= (1, 1, 0, 1, 0, 0, 0) + (0, 1, 1, 0, 1, 0, 0) + (1, 0, 1, 0, 0, 0, 1) \\ &= (0, 0, 0, 1, 1, 0, 1). \end{aligned}$$

The corresponding parity check matrix of G is

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

It can be easily checked that

$$\begin{aligned} s = cH^T &= (0,0,0,1,1,0,1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \\ &= (1,1,0) + (0,1,1) + (1,0,1) \\ &= (0,0,0). \end{aligned}$$

Because of channel noise, the received vector y at the receiver may be different from c, the code word that was transmitted over the noisy channel.

$$y = c + e.$$

When y is received, the decoder computes the syndrome $s = yH^T$. Then $s = 0$ if and only if y is a code word, and $s \neq 0$ if and only if y is not a code word. Therefore, when $s \neq 0$, the receiver knows that y is not a code word and the presence of errors can be detected.

Another important parameter of a block code is called the *minimum distance* that

determines the random-error-detecting and random-error-correcting capabilities of a code. Let v be a binary n-tuple. The *Hamming weight* of v, denoted by $w(v)$, is defined as the number of non-zero components of v. Let u and v be two n-tuples. The *Hamming distance* between v and u, denoted by $d(u,v)$, is defined as the number of places where they differ. The hamming distance is a metric function that satisfies the *triangle inequality.* Let r, u and v be three n-tuples, then

$$d(r,u) + d(u,v) \geq d(r,v).$$

It can also be shown that for any two binary n-tuples u and v

$$d(u,v) = w(u+v).$$

Given a block code C, one can compute the Hamming weight of all non-zero code words. The minimum distance $d_{\min}$ is defined as follows:

$$d_{\min} = \min\{w(u) : u \in C, u \neq 0\}.$$

A block code with minimum distance $d_{\min}$ guarantees correction of all the error patterns of $t = [(d_{\min} - 1)/2]$ or fewer errors. Next, we introduce the standard array and syndrome decoding.

Let C be an (n,k) linear code. Any decoding scheme used by the receiver is a rule to partition the 2^n possible received vectors into 2^k disjoint subsets $D_1, D_2, \cdots, D_k$ such that the code vector c_i is contained in the subset D_i for $1 \leq i \leq 2^k$. Thus each subset D_i is one-to-one correspondence of a code vector c_i. If the received vector y is found in the subset D_i, y is decoded to be c_i. Correct decoding is made if and only if the received vector y is in the subset D_i that corresponds to the actual code vector transmitted. The table formed in this way is called a *standard array* of the given linear

code C (see Table B.1).

Table B.1: Standard array for (n, k) linear code

$c_1 = 0$	c_2	$\cdots$	c_{2^k}
e_2	$e_2 + c_2$	$\cdots$	$e_2 + c_{2^k}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$e_{2^{n-k}}$	$e_{2^{n-k}} + c_2$	$\cdots$	$e_{2^{n-k}} + c_{2^k}$

All the 2^k n-tuples of a coset (a row in the standard array) have the same syndrome. Using this one-to-one correspondence relationship, we can form a decoding table, which is much simpler to use than a standard array. The table consists of 2^{n-k} correctable error patterns and their corresponding syndromes. The decoding of a received vector y is performed in three steps:

1. Compute the syndrome of y ($s = yH^T$).
2. Locate the correctable error pattern e_i whose syndrome is equal to s. Then e_i is assumed to be the error pattern caused by the channel.
3. Decode the received vector y into the code vector $c = y - e_i$.

Table B.2: Decoding table for $(7, 4)$ linear code

Syndrome	Error patterns
$(1,0,0)$	$(1,0,0,0,0,0,0)$
$(0,1,0)$	$(0,1,0,0,0,0,0)$
$(0,0,1)$	$(0,0,1,0,0,0,0)$
$(1,1,0)$	$(0,0,0,1,0,0,0)$
$(0,1,1)$	$(0,0,0,0,1,0,0)$
$(1,1,1)$	$(0,0,0,0,0,1,0)$
$(1,0,1)$	$(0,0,0,0,0,0,1)$

Table B.2 shows the correctable error patterns and their corresponding syndromes for Example B.0.1. The decoding scheme described above is called *syndrome decoding* or

table-lookup decoding. In principle, table-lookup decoding can be applied to any (n, k) linear code. It results in minimal decoding delay and minimal error probability. However, for large codes, the implementation of this decoding scheme becomes impractical. Several decoding scheme which are variations of table-lookup decoding exist that are beyond the scope of this thesis.

Bibliography

[1] M. Abadi and J. Feigenbaum, *Secure circuit evaluation*, Journal of Cryptology, Volume 2, Pages 1-12, 1990.

[2] D. Asonov and J. C. Freytag, *Almost optimal private information retrieval*, In R. Dingledine and P. Syverson, editors, The 2nd Workshop on Privacy Enhancing Technologies, Volume 2482 of Lecture Notes in Computer Science, Pages 209-223, Springer-Verlag, April 2003.

[3] Mikhail J. Atallah and Wenliang Du, *Secure multi-party computational geometry*, In The Seventh International Workshop on Algorithms and Data Structures (WADS 2001), Pages 165-179, August 2001.

[4] A. Beimel and Y. Ishai, *Information-theoric private information retrieval: A unified construction*, In P.G. Spirakis and J. van Leeuven, editors, the 28th International Colloquium on Automata, Languages and Programming, Volume 2076 of Lecture Notes in Computer Science, Pages 912-926, Springer-Verlag, July 2001.

[5] A. Beimel and Y. Ishai, *Reducing the servers' computation in private information retrieval: PIR with preprocessing*, Journal of Cryptology, Volume 17, Pages 125-151, 2004.

[6] D. Boneh and R. J. Lipton, *Algorithms for black-box fields and their application to cryptography*, In Neal Koblitz, editor, Advances in Cryptology- Crypto'96, volume 1109 of Lecture Notes in Computer Science, Pages 283-297, Springer-Verlag, August 1996.

[7] C. Cachin and J. Camenisch, *Optimistic fair secure computation*, In Mihir Bellare, editor, Advances in Cryptology- Crypto'2000, Volume 1880 of Lecture Notes in Computer Science, Pages 93-111, Springer-Verlag, 2000.

[8] R. Canetti, Y. Ishai, R. Kumar, M. K. Reiter, R. Rubinfeld, and R. N. Wright, *Selective private function evaluation with applications to private statistics*, In Neal Koblitz, editor, the 20th ACM Symposium on Principles of Distrbuted Computing (PODC), Pages 293-304, ACM Press, 2001.

[9] I. B. Damgard, D. Chaum and J. van de Graaf, *Multiparty computations ensuring privacy of each party's input and correctness of the result*, In Carl Pomerance, editor, Advances in Cryptology- Crypto'87, Volume 293 of Lecture Notes in Computer Science, Pages 87-119, Springer-Verlag, 1987.

[10] I. Damgard, R. Cramer and U. Maurer, *General secure multi-party computation from any linear secre-sharing scheme*, In Bart Preneel, editor, Advances in Cryptology- EuroCrypto'00, Volume 1807 of Lecture Notes in Computer Science, Pages 316-334, Springer-Verlag, 2000.

[11] I. Damgard and M. Jurik, *A generalisation, simplification and some applications of paillier's probabilistic public-key system*, In Kwangjo Kim, editor, Public Key Cryptography 2001, Volume 1992 of Lecture Notes in Computer Science, Pages 119-136, Springer-Verlag, February 2001.

[12] R. H. Deng, F. Bao and P. Feng, *An efficient and practical scheme for privacy protection in the e-commerce of digital goods*, In D. Won, editor, the third International Conference on Information Security and Cryptology, Volume 2015 of Lecture Notes in Computer Science, pages 162-170, Springer-Verlag, 2001.

[13] Wenliang Du and Mikhail J. Atallah, *Privacy-preserving statistical analysis*, In the Proceedings of the 17th Annual Computer Security Applications Conference, Pages 102-110, December 2001.

[14] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas, *Efficient private matching and set intersection*, In Christian Cachin and Jan Camenisch, editors, Advances in Cryptology- EUROCRYPT 2004, Volume 3027 of Lecture Notes in Computer Science, pages 1-19, Springer-Verlag, May 2004.

[15] R. Gennaro, D. Catalano and N. H. Graham, *Paillier's cryptosystem revisited*, In ACM conference on computer and communication security 2001, Pages 206-214, ACM Press, 2001.

[16] O. Goldreich, *Secure multi-party computation*, Manuscript, Preliminary version, 1998, available at: `www.wisdom.weizmann.ac.il/~oded/pp.html`.

[17] S. Goldwasser and S. Micali, *Probabilistic encryption*, In Journal of Computer and Systems Science, Volume 28, Pages 270-299, April 1984.

[18] F. Hohl, *Time limited blackbox security: Protecting mobile agents from malicious hosts*, In G. Vigna, editor, Mobile Agents and Security, Volume 1419 of Lecture Notes in Computer Science, Pages 123-137, Springer-Verlag, 1998.

[19] Y. Ishai and E. Kushilevitz, *Improved upper bounds on information theoretic private information retrieval*, In P. G. Spirakis and J. van Leeuven, editors, the 31st

Annual ACM Symposuim on the Theory of Computing, Volume 44, Pages 79-88, ACM Press, May 1999.

[20] J. Kilian, C. Cachin, J. Camenisch and J. Muller, *One-round secure computation and secure autonomous mobile agents* In Jose P. Rolim Ugo Montanari and Emo Welzl, editors, the 27th International Colloquium on Automata, Languages and Programming (ICALP), Volume 1853 of Lecture Notes in Computer Science, Pages 512-523, Springer-Verlag, 2000.

[21] E. Krouk, G. Kabatiansky and S. Semenov, *Error Correcting Coding and Security for Data Networks*, John Wiley and Sons, West Sussex, England, 2005.

[22] Y. Lindell and B. Pinkas, *Privacy preserving data mining*, In Journal of Cryptology, Volume 15, Number 3, Pages 177-206, 2002.

[23] Helger Lipmaa, Bart Goethals, Sven Laur and Taneli Mielikainen, *On secure scalar product computation for privacy-preserving data mining*, In Choonsik Park and Seongtaek Chee, editors, In the Proceeding of the Seventh Annual International Conference in Information Security and Cryptology (ICISC 2004), Volume 3506 of Lecture Notes in Computer Science, pages 104-120, Springer- Verlag, December 2004.

[24] H. Lipmaa, *An oblivious transfer protocol with log-squared communication*, Cryptology ePrint Archive, Report 2004/063, 2004.

[25] B. Malek and S. A. Miri, *Private information retrieval with homomorphic encryption functions*, In the Proceeding of the seventh International Symposium on Communications Interworking, Pages 1-8, November 2004.

[26] B. Malek and A. Miri, *An optimal secure data retrieval using oblivious transfers, to appear* In IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, (WiMob05), August 2005.

[27] Robert J. McEliece, textitA public-key cryptosystem based on algebraic coding theory: DNS progress report, Technical report, Jet Propulsion Laboratory, 1978.

[28] Robert J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, 1987.

[29] M. Micali, O. Goldreich and A. Wigderson, *How to play any mental game*, In the ACM Symposium on Theory of Computing, Pages 218-229, ACM Press, 1987.

[30] S. Micali, C. Cachin and M. Stadler, *Computationally private information retrieval with polylogarithmitic communication*, In J. Stern, editor, Advances in Cryptology-Eurocrypt'99, Volume 1592 of Lecture Notes in Computer Science, Pages 402-414, Springer-Verlag, May 1999.

[31] P. Morillo, D. Galindo, S. Mollevi and J. Villar, *A practical public key cryptosystem from paillier and rabin schemes*, In Public Key Cryptography 2003, Volume 2567 of Lecture Notes in Computer Science, Pages 279-291, Springer-Verlag, 2003.

[32] M. Naor and P. Pinkas, *Oblivious transfers and polynomial evaluation*, In ACM Symposium on Theory of Computing, Pages 245-254, ACM Press, 1999.

[33] M. Naor and B. Pinkas, *Distributed oblivious transfer*, In T. Okamoto, editor, Advances in Cryptology- ASIACRYPT 2000, Volume 1976 of Lecture Notes in Computer Science, Pages 205-219, Springer-Verlag, 2000.

[34] M. Naor and P. Pinkas, *Efficient oblivious transfer protocols*, In the 13th ACM SIAM Symposium on Discrete Algorithms (SODA), Pages 448-457, ACM Press, 2001.

[35] B. Pareneel, J. Claessens and J. Vandewalle, *(How) can mobile agents do secure electronic transactions on un-trusted hosts? a survey of the security issues and the current solutions*, In ACM transactions on internet technology, Volume 3, pages 28-48, February 2003.

[36] Pascal Paillier, *Public key cryptosystem based on composite degree residuosity classes*, Advances in Cryptology- Eurocrypt'99, Volume 1592 of Lecture Notes in Computer Science, Pages 223-238, Springer-Verlag, 1999.

[37] Rudolf Lidl and Harald Niederreiter, *Finite Fields*, Volume 20, Cambridge University Press, New York, 1997.

[38] T. Sander, J. Feigenbaum, M. J. Freedman and A. Shostack, *Privacy engineering for digital rights management systems*, In T. Sander, editor, Digital Right Management 2001 (DRM 2001), Volume 2320 of Lecture Notes in Computer Science, Pages 76-105, Springer-Verlag, 2002.

[39] T. Sander and C. F. Tschudin, *On software protection via function hiding*, In D. Aucsmith, editor, the second International Workshop on Information Hiding, Volume 1525 of Lecture Notes in Computer Science, Pages 111-123, Springer-Verlag, 1998.

[40] K. Sivakumar, R. Chen and H. Kargupta, *Distributed web mining using bayesian networks from multiple data streams*, In the 2001 IEEE International Conference on Data Mining, Pages 75-82, IEEE, December 2001.

[41] J. P. Stern, *A new and efficient all-or-nothing disclosure of secrets protocol*, In K. Ohta and D. Pei, editors, Advances in Cryptology- ASIACRYPT'98, Volume 1514 of Lecture Notes in Computer Science, Pages 357-371, Springer-Verlag, October 1998.

[42] Jaideep Vaidya and Chris Clifton, *Privacy preserving association rule mining in vertically partitioned data*, In the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 639-644, Canada, July 2002.

[43] A. C. Yao, *Protocol for secure computations*, In the 23th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Pages 160-164, November 1982.

[44] A. C. Yao, *How to generate and exchange secrets (extended abstract)*, In Los Alamitos, editor, the 27th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Pages 162-167, October 1986.