Private Information Retrieval: Improved Upper Bound, Extension and Applications

Jean-François Raymond

School of Computer Science
McGill University, Montreal

December, 2000

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Master of Science.

Abstract

Private Information Retrieval (PIR), which allows users to query one (or many replicated) database(s) for the ith element, while keeping i private, has received a lot of attention in recent years. Indeed, since Chor et al. [31, 32] introduced this problem in 1995, many researchers have improved bounds and proposed extensions. The following pages continue along this path : pushing the techniques of [52] we obtain an improved upper bound and define and provide a solution to a new problem which we call private information retrieval with authentication. In addition, we motivate the study of PIRs by presenting new and useful real world applications.

Résumé

Les protocoles permettant des requêtes privées (PIR), c'est à dire des requêtes qui ne dévoilent pas quelle information est recherchée, a beaucoup été étudiés au cours des dernières années. Depuis que Chor et al. [31, 32] ont introduit ce problème en 1995, plusieurs chercheurs ont amélioré les performances et suggéré des modifications. Nous poursuivons dans cette voie: nous améliorons un protocole de Ishai et Kushilevitz [52] et suggérons un nouveau problème (ainsi qu'une solution) que nous nommons requêtes privées d'information avec authentification. Nous présentons également de nouvelles applications pratiques utilisant le PIR.

Acknowledgments

First and foremost, I would like to express my unbounded gratitude to my parents and sister without whose love, support and hard work I would not have finished this thesis and more importantly would not have had all the opportunities I've been fortunate to be presented with.

I would like to express my deep gratitude to Denis Thérien (computer science) and Isztar Zawadski (Atmospheric and Oceanic Sciences) for introducing me to research, providing guidance and giving me the opportunity to work on really interesting problems for three summers.

I would like to thank Claude Crépeau for having agreed to take me on as a M.Sc. student, providing a good environment in which we can study cryptography and helping me make this thesis cleaner, much more readable and mistake free.

I would like to thank (in no particular order) Pascal Tesson, Adam Smith, Mathias Jourdain, Frédéric Légaré, Stefan Brands, Adam Back and Anton

Stiglic for many enlightening discussions related to cryptography and theoretical computer science.

I particularily want to thank Anton Stiglic and Mathias Jourdain for having conscientiously and thoroughly read and corrected my thesis – I realize how difficult (and perhaps unpleasant `;-)`) this must have been.

I would like to thank Stefan Brands for having pointed out one of his protocols which I use in section 4.4.

I want to thank the administrative and academic staff of the School of Computer Science of McGill University. Through their work and dedication, they provide a great environment for study and research.

I also want to thank NSERC (National Science and Engineering Research Counsil of Canada) without whose financial support (in the form of a PGSA scholarship) I certainly would not have completed this work.

MERCI!

Contents

Chapter 1

Introduction

As the Internet expands, privacy issues are becoming more and more of a problem[1]. In fact, some would argue that it is one of the main stumbling blocks to a fully online society. Fortunately, many of these issues can be solved by technical means. Moreover, these methods are so effective that, theoretically at least, many online/digital processes are more secure than their real world counterparts. For example, protecting message privacy without the help of computing devices is a challenging problem (e.g. it might involve storing a message in a safe). With computers however, we can quite easily protect message privacy by using encryption.

[1]See http://www.freedom.net for examples.

1.1 On Cryptography

The most fundamental problem in cryptography[2] is that of sending secret information. Precisely, one wants to send some secret information over an insecure communication channel. For about four thousand years, the state of the art protocols consisted of more or less ad-hoc methods. Two major efforts by Shannon [66] in the late 40's and by Diffie and Hellman [38] in the 70's drastically changed this sorry state of affairs. Shannon, by inventing information theory, provided a framework for rigorously quantifying how successful protocols are at keeping sensitive information secret. Diffie and Hellman, in their 1976 seminal paper "New direction in cryptography" [38], develop a protocol for public secure key exchange and provide blue prints for public key cryptography. The foundation was thus laid so that cryptography could evolve from an art to a science.

From the initial problem of sending a secret message, cryptography has grown to encompass many entities such as zero-knowledge proofs [46], digital signatures [8], bit commitments [22], oblivious transfers [62], etc. Perhaps the most significant contribution of Diffie and Hellman [38] has been the link they created between cryptography and complexity theory. Both fields have

[2] We refer the interested reader to [56, 67] for excellent introductions to cryptography.

greatly benefited from this association. The relationship has grown so strong that some research topics, zero-knowledge proofs for example, are hard to classify as belonging to either cryptography or complexity.

1.2 Communication Complexity

Complexity theorists are usually interested in quantifying problems with respect to the time and space required to solve them. There are however other ways of analyzing problems such as interactive proofs [46], circuits [15], branching programs [69], Arthur-Merlin games [11], etc.

The rise of telecommunications motivated the study of algorithms with respect to a different resource : the amount of bits exchanged, that is, communication complexity. This approach has proved very successful as many fundamental results [12, 13, 48, 53, 68] have been obtained with its utilization.

1.3 Private Information Retrieval

This work will be concerned with techniques related to private information retrieval (PIR). In this problem, a user wants to retrieve the ith element (bit) from one or many replicated (there are k databases), non communicating database(s), which we model as an n bit string, without revealing i. We can think of many real world applications where such a mechanism would be useful. For example, the retrieval of stock prices, medical information and patent descriptions. Furthermore, it appears that PIRs would be extremely useful in privacy protecting mix networks based infrastructures [25, 43].

The obvious solution to the problem of query privacy is to have the user download the entire database. Unfortunately, the communication complexity $(n+1)$ is too large for most applications. And so, PIRs are defined as having sub-linear communication complexity.

The first indication that we could do better than $O(n)$ communication complexity came from complexity theorists [9, 18, 60, 61] who constructed protocols (with other results in mind) that could easily be used to obtain efficient PIRs. In 1995 Chor et al. [31, 32] introduced the model and provided some new, more efficient, solutions (communication complexity : $O(n^{1/k})$). New bounds and extensions were soon to follow ...

1.4 New Bounds and Extensions

Ambainis [10] got the ball rolling by presenting an improved protocol (communication complexity : $O(n^{1/(2k-1)})$). Several years later, Ishai and Kushilevitz [52] developed a linear algebraic framework that enabled them to obtain yet another more efficient construction (communication complexity : $O(n^{1/(2k-1)})$, with smaller multiplicative constants than [10]).

Probably the most important extension is what is called computational private information retrieval (cPIR). In this model, the database(s) is (are) assumed to have limited computational power. Using pseudorandom number generators [65] allowed Chor and Gilboa [29] to present a *two* database cPIR with communication complexity substantially smaller than the best known PIR (communication complexity : $O(n^{\epsilon})$, for any $\epsilon > 0$).

In addition to having lower communication complexity, cPIRs, unlike PIRs, allow private information retrieval with just one database. This advantage was first demonstrated by Kushilevitz and Ostrovsky [55] who accomplished this using the quadratic residuosity assumption [47] (communication complexity : $O(n^{\epsilon})$, for any $\epsilon > 0$). An astounding poly-logarithmic upper bound for a one database cPIR was obtained by Cachin et al. [24]. The only caveat to this result is that a new computational assumption, the ϕ-hiding assumption[3], is used.

The main drawback of cPIRs in comparison to PIRs is that they have significantly higher computational complexity. The PIR's computations are

[3] Informally, this assumption states that it is computationally intractable to decide whether a given small prime divides $\phi(m)$.

usually limited to a few exclusive-ors whereas cPIRs need expensive procedures such as prime number generators. Recently, Beimel et al. [20] have attempted to address this problem by showing how the databases can carry out pre-computations.

Ostrovsky and Shoup [59], extending the methods used in oblivious RAM protocols [44, 45, 58], provided an efficient solution for the problem of private information storage and retrieval. In this model, the user can, not only read privately, but also write privately.

Oblivious transfer [23, 39, 62], a flavor of which is essentially the same as a one database cPIR in which the user gains no knowledge other than the bit he requested, motivated the study of symmetric private information retrieval (SPIR). SPIRs can be seen as a distributed, information theoretic oblivious transfer with emphasis placed on communication complexity. In [42], Gertner et al. introduced the model and showed how to transform any k-database PIR to a $(k+1)$-database SPIR. Recently Naor et al. [57] gave a similar construction that does not require an extra database; unfortunately, their scheme works only in the computational setting.

Using ideas from Beaver's "Commodity Based Cryptography" paper [16], Di Crescenso et al. [36] gave a scheme using commodity servers which help in decreasing the communication between the database and the user to $\log n + 1$. (Most of the communication is done offline)

In certain situations it is not desirable to have all the databases keep a copy of the data. For example when the data is private/sensitive. Gertner et al. [41] provided a solution to this problem which uses databases containing random strings.

In most real world settings, the user does not know the index and wants to search the database for keywords. Chor et al. [30] provided clean methods in which one can modify a PIR to make it searchable.

Researchers have also been interested in uncovering fundamental properties of PIR. It is known that A) one-way functions are essential for one database PIRs [19] B) One database PIR implies oblivious transfer[4] [37] and C) one-way trapdoor permutations are sufficient for non-trivial PIR [54].

1.5 Relation With Other Models

Private information retrieval has many similarities with other models. For example, the hiding instance from an oracle model (HIO) [7, 17, 18], resem-

[4]

- Note that in view of [50] it is unlikely (since it would yield a proof of P not equal NP) that *one database* PIRs can be implemented from one-way functions *only*.
- Also note that in view of [49], this result implies A).

bles PIR. In this model, a user wants to evaluate a function $f()$ on some private input y. The catch is that he cannot compute $f(y)$ by himself, and so needs the help of an(many) oracle(s). Furthermore, as y is private, he needs to formulate his query(ies) in such a way that : the oracle(s) gains *no* knowledge about y and replies(y) with data allowing him to determine $f(y)$. There are important differences between HIO and PIR. First of all retrieving data from a database is just a special case of HIO and many results have been obtained for computable functions. Secondly, y has length n which implies that there are 2^n possible inputs, compared to n for PIRs. It is thus not surprising that PIR is mostly interested in the case where the number of databases is constant whereas HIO concentrates on the non-constant number of oracle case.

As mentioned previously, there are obvious links between oblivious transfer [23, 39, 62] and PIR, in fact SPIRs are essentially one out of n oblivious transfers with sub-linear communication complexity.

1.6 Overview of Thesis

In chapter 2 we present a formal definition, common assumptions used in cryptography (and their relation to PIR) and a simple PIR protocol. Ishai and Kushilevitz's linear algebraic method [52] as well as our improvement can be found in chapter 3. A new problem which we call private information retrieval with authentication is given in chapter 4. We present some new applications for which PIR is useful in chapter 5. And finally, we conclude in chapter 6.

Chapter 2

Basic PIR Protocols

In this Chapter we present some notation and conventions, formally define PIR, discuss adversary models and expose a simple PIR found in [31, 32].

2.1 Notation and Convention

In this section we present some new notation and give conventions.

- By $[n]$ we mean the set $\{1, 2, \ldots, n\}$.
- Elements in a vector (a.k.a. string) are usually referred to by subscripting the vector identifier. For example x_j refers to the jth element in the vector x.
- It is extremely important to note and remember that, in the following, when we say "chosen at random", we mean, chosen at random from a uniform distribution.

The exclusive-or of a set and an element, which we call set-xor, is defined in a natural way:

Definition: let S be a set and x an element.

$$S \bigoplus x = \begin{cases} S \cup \{x\} & \text{if} \quad x \notin S \\ S \setminus \{x\} & \text{otherwise} \end{cases}$$

2.2 A Formal Definition

A PIR is a series of processes in which a user ($\mathcal{U}$) queries k, non communicating, databases ($\mathcal{DB}_1, \mathcal{DB}_2, \ldots, \mathcal{DB}_k$), which all store the same n bit string x, for the ith element (x_i – the ith bit of the string x). Furthermore, the databases must gain *no* information about the index i and the user must be able to determine x_i. We also limit ourselves to protocols taking one round. For this task, we need k functions that generate queries, k functions to answer these queries and finally a function that will take all the information available to the user at the end of the protocol as input and evaluate to x_i.

Notice that, in an information theoretic setting, each query must be random. Hence the query generating functions cannot be deterministic (they must have access to l_{rand} random bits). Formally, we have:

Definition: A (one-round) PIR consists of the following functions:

- k query functions of the form:

$$\{0,1\}^{\lg n} \times \{0,1\}^{l_{rand}} \rightarrow \{0,1\}^{l_q}$$

(queries have length l_q and $\{0,1\}^{\lg n}$ is taken to be i.)

Note that : *no* information can be inferred from the output of a query function (it must be random). Precisely, we need that the probability of a query being associated with x_β equals that for x_γ, for all $\beta, \gamma \in [n]$.

- k answer functions of the form:

$$\{0,1\}^{l_q} \times \{0,1\}^{n} \rightarrow \{0,1\}^{l_a}$$

(answers have length l_a and $\{0,1\}^{n}$ is taken to be x.)

- one reconstruction function of the form:

$$\{0,1\}^{\lg n} \times (\{0,1\}^{l_a})^k \times (\{0,1\}^{l_{rand}})^k \rightarrow \{0,1\}$$

($\{0,1\}^{\lg n}$ is taken to be i.)

Where the output of the reconstruction function equals x_i (assuming the inputs are *correct*). Note that the k query values and answer values $((\{0,1\}^{l_{rand}})^k$ and $(\{0,1\}^{l_a})^k$ respectively) are dependent.

2.3 Adversary Models

One of the main challenges in cryptography is to model attackers. In this section, we briefly present common assumptions and their relation to private information retrieval.

2.3.1 Information Theoretic VS Computational Settings

We say we are in an information theoretic setting when an adversary cannot gain *any* information [66] about the secret data. The data the adversary has access to follows a probability distribution that is independent from the secret information's probability distribution.

Alternatively, we are in a computational setting when the adversary's computations are bounded. In this setting, the adversary *might* be able to obtain some information about the secret information if he could carry out some infeasible computations or if he is extremely lucky. Typical assumptions include: the adversary is limited to probabilistic polynomial time computations and the adversary cannot compute some "hard" function (e.g. discrete log). Note that these assumptions allow us to create protocols which have properties that *cannot* be obtained in an information theoretic setting. For example, it is impossible to construct a PIR with one database whereas this is possible for cPIR.

2.3.2 Semi-Honest VS Arbitrarily Malicious

A semi-honest adversary does nothing to disrupt the protocol; he follows the rules exactly like an honest participant. He may however, unlike a completely honest player, try to learn something about the secret information (e.g. by carrying out some extra computation).

Adversaries who can do *anything* in their power to gain some illegal information are called arbitrarily malicious adversaries. They can disrupt the protocol in many ways, sending arbitrary messages for example.

Note that these two types of adversaries are sometimes referred to as passive (semi-honest) and active (arbitrarily malicious).

2.3.3 Standard Assumptions in Private Information Retrieval

In the private information retrieval model, we assume that the communication channels are secure. Hence, the adversary is not some outside party as in the secret message scenario. The only party(ies) that can violate query privacy is (are) the database(s).

As mentioned in the introduction, the database(s) can be assumed computationally bounded or not. Most of the work on PIR, and its variants, has assumed that the database(s) is (are) semi-honest. Some protocols allow us to relax the assumption that the databases cannot communicate with each other (the no communication assumption) by allowing a certain number, t, of databases to do so. Note that when designing and analyzing protocols, the parameter t is important.

Note that some extensions need to use different assumptions. For example in SPIRs, the user is assumed to be arbitrarily malicious.

In this work, we will *only* study the information theoretic setting, and will assume semi-honest, non-communicating databases.

The remainder of this chapter is devoted to the presentation of a simple PIR from [31, 32].

2.4 An Inefficient Solution

Most efficient PIRs have one thing in common : they are slight modifications of simple, often inefficient, schemes. This next procedure is not an exception. We now present a simple two database PIR which will form the foundation of a more efficient solution.

The user randomly selects a set of indexes S_0 (i.e. he randomly chooses a subset of $[n]$). He sends this set to $\mathcal{DB}_1$ and sends $S_1 = S_0 \bigoplus i$ to $\mathcal{DB}_2$.

Notice that the sets received by $\mathcal{DB}_1$ and $\mathcal{DB}_2$ are random (each index has a 50% chance of being in a given set) and so leak no information about i. $\mathcal{DB}_1$ and $\mathcal{DB}_2$ then compute their answers $A_1 = \bigoplus_{j \in S_0} x_j$ and $A_2 = \bigoplus_{j \in S_1} x_j$ respectively. The exclusive-or of the two answers A_1 and A_2 equals x_i because all other indexes ($j \neq i$) appear an even number of time (i.e. 0 or 2 times). Unfortunately this protocol has linear communication complexity. However, using this idea along with a more compact index representation scheme, we can obtain better results.

2.5 A More Efficient Solution

Without loss of generality, we assume n has the form $n = l^d$. We can construct a bijection mapping elements of $[n]$ to the points in a d-dimensional hyper cube ($\mathbb{Z}_l^d$) by expressing elements of $[n]$ in their base l encoding. For example 9 is expressed as 1001 when using base $l = 2$ encoding.

Consequently any x_j can now be thought of as the point $x_{j_1 \ldots j_d}$ in a hyper cube where $j_\alpha \in [l]$ for all $\alpha \in [d]$. As usual, $\mathcal{U}$ is interested in bit x_i which can be expressed as $x_{i_1 \ldots i_d}$. Note that throughout the remainder of this work, we will use j and $j_1 \ldots j_d$ interchangeably (the base, l, will be clear from the context.)

It is also convenient to assume $k = 2^d$. Using the same mapping as for the string indexes (substituting l by 2), we can map $\mathcal{DB}_\sigma$ to $\mathcal{DB}_{\sigma_1 \ldots \sigma_d}$ ($\sigma_\alpha \in \{0, 1\}$ for all $\alpha \in [d]$). The protocol goes as follows:

1. $\mathcal{U}$ chooses independently and at random d subsets of $[l]$ which we denote by $S_0^1, S_0^2, \ldots, S_0^d$.

2. $\mathcal{U}$ defines the sets S_1^α as being equal to $S_0^\alpha \bigoplus i_\alpha$, for all $\alpha \in [d]$.

3. $\mathcal{U}$ then sends each $\mathcal{DB}_{\sigma_1 \ldots \sigma_d}$ the sets of indexes $S_{\sigma_1}^1, \ldots, S_{\sigma_d}^d$.

4. We can see the queries as representing sub-cubes. For example, the query sent to $\mathcal{DB}_{\sigma_1 \ldots \sigma_d}$ represents the sub-cube containing all points $(x_{j_1 \ldots j_d})$ such that $j_\alpha \in S_{\sigma_\alpha}^\alpha$ for all $\alpha \in [d]$.

   Each database, $\mathcal{DB}_{\sigma_1 \ldots \sigma_d}$, evaluates the exclusive-or of all the elements of the query's sub-cube and sends the result, $A_{\sigma_1 \ldots \sigma_d}$, to $\mathcal{U}$.

5. $\mathcal{U}$ then evaluates the exclusive-or of all the answers, $A_{\sigma_1 \ldots \sigma_d}$, to obtain x_i.

The message exchanges are graphically presented in figure 2.1.

Before showing the protocol actually works, it is useful to realize that $\mathcal{U}$'s reconstruction function calculates:

$$\mathcal{U} \qquad \xrightarrow{S^1_{\sigma_1},\ldots,S^d_{\sigma_d}} \qquad \mathcal{DB}_{\sigma_1,\ldots,\sigma_d}$$

$$\xleftarrow{\bigoplus_{j_1\in S^1_{\sigma_1},\ldots,j_d\in S^d_{\sigma_d}} x_{j_1\ldots j_d}}$$

Figure 2.1: A Simple PIR

$$\bigoplus_{\sigma_1\in\{0,1\},\ldots,\sigma_d\in\{0,1\}} \Big(\bigoplus_{j_1\in S^1_{\sigma_1},\ldots j_d\in S^d_{\sigma_d}} x_{j_1\ldots j_d} \Big) \tag{2.1}$$

The first property to notice is that $x_{i_1\ldots i_d}$ appears only once in this formula since all indexes, i_α, appear in exactly *one* of S_0^α and S_1^α. We will prove in lemma 2.5.1 that all other $x_{j_1\cdots j_d}$ appear an even number of times in formula 2.1. Which means they cancel out, leaving only $x_{i_1\ldots i_d}$.

Lemma 2.5.1 *All $x_{j_1\cdots j_d}$ for which $(j_1,\ldots,j_d) \neq (i_1,\ldots,i_d)$ appear an even number of times in $\mathcal{U}$'s final computation (formula 2.1).*

Proof:

First notice that if $x_{j_1\ldots j_d}$ is not used in any of the databases' computations, it obviously appears an even number of times (0) in formula 2.1.

Otherwise, if $x_{j_1\ldots j_d}$ $((j_1,\ldots,j_d) \neq (i_1,\ldots,i_d))$ then for all α_p such that $j_{\alpha_p} \neq i_{\alpha_p}$ (w.l.o.g. assume that $p \in [q]$). It is not hard to see that :

$(j_1, j_2, \ldots, j_d) \in$

$$S^1_{\sigma_1} \times \ldots \times S^{\alpha_1 - 1}_{\sigma_{\alpha_1 - 1}} \times S^{\alpha_1}_{y_1} \times S^{\alpha_1 + 1}_{\sigma_{\alpha_1 + 1}} \times \ldots \times S^{\alpha_q - 1}_{\sigma_{\alpha_q - 1}} \times S^{\alpha_q}_{y_q} \times S^{\alpha_q + 1}_{\sigma_{\alpha_q + 1}} \times \ldots \times S^d_{\sigma_d}$$

for all 2^q possible values of $(y_1, \ldots, y_q)$. This follows directly from the way the $S^{\alpha}_{\sigma_\alpha}$ are constructed. Hence, $x_{j_1 \ldots j_d}$ appears an even number of times in formula 2.1.

$\square$

Note that the queries are random and so do not leak any information about i to the databases.

Communication Complexity:

Analyzing the communication complexity, we see that $\mathcal{U}$ sends dkl bits; l bits for each set[1] $S^{\alpha}_{\sigma_\alpha}$, there are d such sets to send to each of the k databases. The $\mathcal{DB}$s send k bits. Hence, the scheme's total communication complexity is $dkl + k = O(dkn^{1/d}) = O(d2^d n^{1/d})$.

Notice that almost all of the communication goes from the user to the databases. In the next chapter, we will, among other things, balance the communication which will help in obtaining an improved upper bound.

[1] We can represent any subset of $[l]$ with l bits.

Chapter 3

An Improved PIR

In this chapter, we present Ishai and Kushilevitz's PIR [52] as well as our improvement.

Formalizing the techniques of chapter 2 turns out to be very useful as it allows us to grasp the essence of the protocol. This in turn facilitates our work finding better bounds.

Schemes in this chapter use:

- a more efficient secret sharing scheme[1].
- a solid framework that allows us to prove tight bounds.

[1]see definition in section 3.2.

3.1 Some definitions

Before describing protocols, we need a few definitions and conventions. As in the sub-cube PIR of chapter 2, n is assumed to be equal to l^d. We define e_α as being a unit vector, that is a vector in which all coordinates are 0 except the αth which has value 1. The unit vector's dimension will be clear from the context.

We define the query space as being $Q \stackrel{def}{=} F^l$. Where F is some field which, in this chapter, will always be $GF(2)$. Note that all computations will be carried out in F.

Remark: The sub-cube scheme fits into this framework.

In the next sections, the following function's nice properties are used.

Definition: Define $\prod : Q^d \to F^n$ such that the jth bit of the output is obtained by calculating $q^1_{j_1} \cdot q^2_{j_2} \cdot \ldots \cdot q^d_{j_d}$ from the input which can be expressed in matrix form as:

$$\begin{bmatrix} q^1_1 & q^2_1 & \dots & q^d_1 \\ q^1_2 & q^2_2 & \dots & q^d_2 \\ \vdots & \vdots & \ddots & \vdots \\ q^1_l & q^2_l & \dots & q^d_l \end{bmatrix}$$

To calculate the jth bit of the output, we simply choose the j_αth element from the αth column for all $\alpha \in [d]$ and multiply the d values together.

This mapping turns out to be useful because of the following properties:

1. $\prod(e_{j_1}, \ldots, e_{j_d}) = e_j$

   the e_{j_α}s can be seen as column vectors.

   Proof: There is only one 1 per column and so only one bit of the output will equal 1.

   $\square$

2. $\prod(q^1, \ldots, q^{\alpha-1}, u+v, q^{\alpha+1}, \ldots, q^d) = \prod(q^1, \ldots, q^{\alpha-1}, u, q^{\alpha+1}, \ldots, q^d) + \prod(q^1, \ldots, q^{\alpha-1}, v, q^{\alpha+1}, \ldots, q^d)$

   This property is usually referred to as multilinearity.

   Proof:

   This is another easy proof, and it follows directly from the definition.

   $\prod(q^1, \ldots, q^{\alpha-1}, u+v, q^{\alpha+1}, \ldots, q^d) =$

   $= (q_1^1 \cdot \ldots q_1^{\alpha-1} \cdot (u_1+v_1) \cdot q_1^{\alpha+1} \cdot \ldots \cdot q_1^d, \ldots, q_l^1 \cdot \ldots \cdot q_l^{\alpha-1} \cdot (u_l+v_l) \cdot q_l^{\alpha+1} \cdot \ldots \cdot q_l^d)$

$$= (q_1^1 \cdot \ldots \cdot q_1^{\alpha-1} \cdot u_1 \cdot q_1^{\alpha+1} \cdot \ldots \cdot q_1^d, \ldots, q_l^1 \cdot \ldots \cdot q_l^{\alpha-1} \cdot u_l \cdot q_l^{\alpha+1} \cdot \ldots \cdot q_l^d)$$

$$+(q_1^1 \cdot \ldots \cdot q_1^{\alpha-1} \cdot v_1 \cdot q_1^{\alpha+1} \cdot \ldots \cdot q_1^d, \ldots, q_l^1 \cdot \ldots \cdot q_l^{\alpha-1} \cdot v_l \cdot q_l^{\alpha+1} \cdot \ldots \cdot q_l^d)$$

$\square$

3.2 Ishai and Kushilevitz's protocol

The **secret sharing** primitive, which was introduced by Shamir [64], allows k players to have a share of some secret. Each group of less than t shares, yields *no* information about the secret. However, with at least t shares we can determine the secret efficiently.

The basic idea of the next PIR is to perform secret sharing on the private indexes $i_1, \ldots, i_d$ and have the databases perform computations with their shares.

3.2.1 Basic Protocol

The secret sharing scheme goes as follows : for all $\alpha \in [d]$, $\mathcal{U}$ starts by randomly choosing $k-1$ random l bit vectors $a_1^\alpha, \ldots, a_{k-1}^\alpha$ and setting another one, a_k^α, such that $\sum\limits_{\beta \in [k]} a_\beta^\alpha = e_{i_\alpha}$. (Remember that i can be expressed as $i_1, i_2, \ldots, i_d$.) He then sends all databases, $\mathcal{DB}_\sigma$, their shares which consist of *all* a_β^α such that $\beta \neq \sigma$. That is, each database gets all the shares except those that are labeled (subscripted) by his identifier.

Remark: The databases have *no* information on i. To determine i_α they need k shares; with their $k-1$ shares they have *no* information. (t equals k in this secret sharing scheme.)

The following development effectively illustrates how the databases can use their shares in a useful manner.

We use the two properties defined above along with the multilinearity of the inner product denoted by $\langle w, z\rangle$.

$$\begin{aligned}
x_i &= \langle x, e_i\rangle \\
&= \langle x, \prod(e_{i_1}, \dots, e_{i_d})\rangle \\
&= \langle x, \prod(\sum_{\beta_1\in[k]} a^1_{\beta_1}, \dots, \sum_{\beta_d\in[k]} a^d_{\beta_d})\rangle \\
&= \langle x, \sum_{\beta_1,\dots,\beta_d\in[k]} \prod(a^1_{\beta_1}, \dots, a^d_{\beta_d})\rangle \\
&= \sum_{\beta_1,\dots,\beta_d\in[k]} \langle x, \prod(a^1_{\beta_1}, \dots, a^d_{\beta_d})\rangle
\end{aligned}$$

The first line follows directly from the inner product's definition; the second from $\prod$'s first property. The third line illustrates the secret sharing. Using $\prod$'s and the inner product's multilinearity, we obtain lines four and five respectively. We can see that each database can compute and add all terms, $\langle x, \prod(a^1_{\beta_1}, \dots, a^d_{\beta_d})\rangle$, for which it has all the shares $a^\alpha_{\beta_\alpha}$ (i.e. $\mathcal{DB}_\sigma$ can

$$
\begin{tikzcd}
\mathcal{U} \arrow[r, shift left, "\text{All } a^{\alpha}_{\beta_\alpha} \text{ s.t. } \beta_\alpha \neq \sigma"] & \mathcal{DB}_\sigma \arrow[l, shift left, "\text{Sum of all terms assigned to } \mathcal{DB}_\sigma"]
\end{tikzcd}
$$

Figure 3.1: Ishai and Kushilevitz's unbalanced PIR

calculate $\langle x, \prod(a^1_{\beta_1}, \ldots, a^d_{\beta_d})\rangle$ if $\sigma \notin \bigcup\limits_{\alpha\in[d]} \beta_\alpha)$.

Each database can thus send the sum of the terms it has calculated, and $\mathcal{U}$ can determine x_i by adding all the database answers.

Remark: If more than one database can compute a term, it is assumed only one does so.

The only remaining problem is to make sure that there does not exist a term that cannot be computed by any database. If all terms $\langle x, \prod(a^1_{\beta_1}, \ldots, a^d_{\beta_d})\rangle$ have the property that $\bigcup\limits_{\alpha\in[d]} a^\alpha_{\beta_\alpha} \neq [k]$ we are assured that all terms can be computed by at least one database. Fortunately, if we set $d = k - 1$, every term will have this property. There are $k-1$ slots and k elements so at least one element will not be in any of the $k - 1$ slots.

The protocol is graphically represented in figure 3.1

Communication Complexity: $\mathcal{U}$ sends each database $(k-1)$ vectors $(a^\alpha_{\beta_\alpha})$ for each of the d dimension. These vectors have l elements (bits). Hence, he sends $((k-1)dl)k$ bits. Each database can sum all terms it has computed and thus sends only one bit. The communication complexity of the scheme is thus $(k-1)dkl + k = O((k-1)^2 kn^{1/d}) = O(k^2 kn^{1/k-1}) = O(k^3 n^{1/k-1})$.

Again, notice that the communication is not balanced. As mentioned in the previous chapter, balancing the communication can help us obtain better bounds. The idea is that if a database is missing a share $a^{\alpha}_{\beta_\alpha}$ to calculate a term, it can still calculate an $l = n^{1/d}$ bit list containing all possible answers. This in turn allows us to increase d.

3.2.2 Balancing the Communication

Instead of presenting the scheme of [52], we give a modified protocol that will facilitate the presentation of our improvement. Before starting, we need to define some more notation.

Denote the term $\langle x, \prod(a^1_{\beta_1}, \ldots, a^d_{\beta_d})\rangle$ by the d-tuple $(\beta_1, \ldots, \beta_d)$. Similarily, we define the term $(\beta_1, \ldots, \beta_{\alpha-1}, \star, \beta_{\alpha+1}, \ldots, \beta_d)$ as being equal to $\langle x, \prod(a^1_{\beta_1}, \ldots, a^{\alpha-1}_{\beta_{\alpha-1}}, e_{i_\alpha}, a^{\alpha+1}_{\beta_{\alpha+1}}, \ldots, a^d_{\beta_d})\rangle$. The star ($\star$) can be thought of as a type of wildcard not unlike those used in formal languages. Generalizing the notation a bit, we see that

$$
\begin{aligned}
x_i &= \langle x, e_i \rangle \\
&= \langle x, \prod (e_{i_1}, \ldots, e_{i_d}) \rangle \\
&= (\star, \ldots, \star)
\end{aligned}
$$

Starred terms that have the property that the union of all non-starred coordinates does not equal $[k]$ are called blocks. If a block contains X stars, it is called a X-block. As an example, for $k = 5, (1, 2, 3, 4, \star)$ is a 1-block. Note that if $k = 4$, $(1, 2, 3, 4, \star)$ is *not* a block since the union of all non-starred coordinates equals $[4] = \{1, 2, 3, 4\}$.

We say a term of the form $(\beta_1, \ldots, \beta_{\alpha-1}, \beta, \beta_{\alpha+1}, \ldots, \beta_d)$ is *covered* by (or included in)$(\beta_1, \ldots, \beta_{\alpha-1}, \star, \beta_{\alpha+1}, \ldots, \beta_d)$. More generally, a term is covered by a starred term if all non-stared coordinates are equal.

$\mathcal{U}$ is interested in obtaining a linear combination of values that spans the sum of all terms. To this end, we allow the databases to compute the following values :

- **We are allowing 0-blocks:** As in the previous subsection the databases can easily compute terms $(\beta_1, \ldots, \beta_d)$ such that $\bigcup_{\alpha \in [d]} \beta_\alpha \neq [k]$. That is, at least one database possesses all the shares needed to compute the term.

- **We are allowing 1-blocks:** If a database, $\mathcal{DB}_\sigma$, is missing only one coordinate to calculate a term $(\beta_1, \ldots, \beta_{\alpha-1}, \sigma, \beta_{\alpha+1}, \ldots, \beta_d)$, that is $\sigma \notin \bigcup_{\gamma \in [d] \setminus \alpha} \beta_\gamma$, it can prepare (and send) a list containing all $n^{1/d}$ possible values for $(\beta_1, \ldots, \beta_{\alpha-1}, \star, \beta_{\alpha+1}, \ldots, \beta_d)$. The pth entry in this list will be $(\beta_1, \ldots, \beta_{\alpha-1}, e_p, \beta_{\alpha+1}, \ldots, \beta_d)$. $\mathcal{U}$, knowing e_{i_α}, can pick the correct value.

The goal is now to find a linear combination of 0-blocks and 1-blocks that equals $(\star, \ldots, \star)$. Equivalently, we need to show that all 0-blocks and all 1-blocks span $(\star, \ldots, \star)$. Remember that all computations are performed in $GF(2)$.

Lemma 3.2.1 *If all terms are either 0-blocks or are covered by at least one 1-block,* $(\star, \ldots, \star)$ *can be spanned using 0 and 1-blocks.*

Proof:

We will show that each *term* can be spanned which implies that the sum of all terms has the desired property. First note that all 0-blocks are trivially spanned. We now show that the remaining terms can also be expressed as a linear combination of 0-blocks and 1-blocks. Without loss of generality, suppose such a term, $(\beta_1, \ldots, \beta_{\alpha-1}, \beta, \beta_{\alpha+1}, \ldots, \beta_d)$, is covered by a 1-block $(\beta_1, \ldots, \beta_{\alpha-1}, \star, \beta_{\alpha+1}, \ldots, \beta_d)$. The term can be expressed as:

$$
\begin{aligned}
(\beta_1, \ldots, \beta_{\alpha-1}, \beta, \beta_{\alpha+1}, \ldots, \beta_d) = \quad & (\beta_1, \ldots, \beta_{\alpha-1}, \star, \beta_{\alpha+1}, \ldots, \beta_d) \\
- & \sum_{\phi \in [k] \setminus \beta} (\beta_1, \ldots, \beta_{\alpha-1}, \phi, \beta_{\alpha+1}, \ldots, \beta_d)
\end{aligned}
$$

But all terms $(\beta_1, \ldots, \beta_{\alpha-1}, \phi, \beta_{\alpha+1}, \ldots, \beta_d)$ such that $\phi \neq \beta$, are 0-blocks since $\beta \notin \bigcup_{\gamma \in [d] \setminus \alpha} \beta_\gamma$.

Hence, all terms can be expressed as a linear combination of 0-blocks and 1-blocks and so it follows that the sum of all terms can be spanned.

□

What is the maximum value of d which guarantees that each term is either a 0-block or included in at least one 1-block? It is not hard to see that if $d = 2k - 1$ it is impossible to have a term not included in any 1-block because at least one database index will appear less than two times. It is impossible to put k values into $2k - 1$ slots in such a way that all values appear in at least two slots.

The scheme proceeds as expected, each database computing and then

$$\mathcal{U} \qquad \underset{\xleftarrow{\text{Lists assigned to } \mathcal{DB}_\sigma}}{\xrightarrow{\text{All } a^{\alpha}_{\beta_\alpha} \text{ s.t. } \beta_\alpha \neq \sigma}} \qquad \mathcal{DB}_\sigma$$

Figure 3.2: Ishai and Kushilevitz's balanced PIR

adding the required 0-blocks and 1-blocks and sending the result to $\mathcal{U}$. Note that instead of sending each list, the databases can merge lists for 1-blocks having a star at the same position (e.g. α). This is done by simply adding elements that have assumed the same value for e_{p_α}.

Remark: The databases determine beforehand who will compute each 0-blocks and 1-blocks. (Some 0-blocks and 1-blocks can be computed by more than one database – This will be used in the next section.)

The protocol is graphically presented in figure 3.2.

Communication Complexity: $\mathcal{U}$ sends $O(k^3 n^{1/d})$ bits while each database sends at most d (one for each starred coordinate), $n^{1/d}$ bit lists, yielding a communication complexity of $O(kdn^{1/d}) = O(k^2 n^{1/d})$. Hence, the communication complexity is $O(k^3 n^{1/2k-1})$. This gives the best known *two* database PIR.

3.3 Our Improvement

The following two questions are at the heart of our improvement:

- Why do we limit ourselves to 0-blocks and 1-blocks? Can using X-blocks (X > 1) improve the previous PIR?

  Allowing databases to compute X-blocks, for X > 1, allows us to increase d. Unfortunately, the size of the lists increases which counteracts any improvement gained by having a larger d.

- Instead of sending lists, of which $\mathcal{U}$ needs only one bit, why don't we perform a PIR?

  The databases send large lists which consist mostly of useless data. If more than one database could compute the same list, it would allow us to perform a PIR to transfer the useful bit. Unfortunately, increasing the number of databases so that sets of Y databases can process the same lists does not yield an improvement over other PIRs.

Even though each idea, taken individually does not yield any improvement, combining them we can obtain an improved bound!

3.3.1 Basic Tools

In the previous section, what blocks can be computed (seen) by Y databases? This is an important question as we can perform Y database PIRs to transfer the relevant bit from the lists associated with such blocks. Observe that Y databases can compute the list associated with

$$(\beta_1, \ldots, \beta_{\alpha_1 - 1}, \star, \beta_{\alpha_1 + 1}, \ldots, \beta_{\alpha_2 - 1}, \star, \beta_{\alpha_2 + 1}, \ldots, \ldots, \beta_{\alpha_\tau - 1}, \star, \beta_{\alpha_\tau + 1}, \ldots, \beta_d)$$

if

$$(\bigcup_{\gamma \in [d] \setminus W} \beta_\gamma) \bigcap C = \varnothing$$

where $W = \{\alpha_1, \alpha_2, \ldots, \alpha_\tau\}$ and C is a subset of $[k]$ of cardinality Y.

We call τ-blocks that have this property (τ,Y)-blocks[2].

We now introduce a natural classification of terms that will be useful in the next subsection. We classify terms that are not 0-blocks with respect to which database indexes appear more than X times and where these are located. We call such classes covering classes. Covering classes are denoted the same way as their member terms except that elements appearing less than or equal to X times are replaced by $\odot$. For example, if X $= 2$ and $k =$

[2]note that $\tau \geq$ Y.

3, $(1,1,1,2,3)$ and $(1,1,1,3,2) \in (1,1,1,\odot,\odot)$. Notice that all classes are disjoint for any given X.

3.3.2 A New PIR

We start by finding a d that will guarantee that all terms are either 0-blocks or belong to a covering class that has at most $k-$Y database indexes appearing more than X times[3]. This implies that each term will either be a 0-block or be covered by an (A,B)-block with $A \geq B \geq$ Y. The easiest way to attack this problem is to find the smallest(dimension) term such that this property does not hold; d will be one less than the length of this term. We need k slots so that the term is not a 0-block. Now if $k-(Y-1)$ values appear another X times the term does not have the desired property. The minimum term is thus :

$$(1,2,\ldots,k,\underbrace{1,1,\ldots,1}_{X},\underbrace{2,2,\ldots,2}_{X},\ldots,\underbrace{k-(Y-1),\ldots,k-(Y-1)}_{X})$$

Hence, we choose $d=(k+(k-(Y-1))X)-1$.

Lemma 3.3.1 *With $d=k+(k-(Y-1))X-1$, the sum of all terms in any covering class having the property that there are at most $k-Y$ indexes*

[3]Hence there are at least Y databases appearing less than or equal to X times.

appearing more than X times, can be spanned by (A,B)-blocks and 0-blocks ($A \geq B \geq Y$).

Proof:

We proceed by induction on the number of $\odot$s.

Base case: The covering class has Y $\odot$s. Note that there are no covering classes possessing the desired property with less than Y $\odot$s. Without loss of generality, consider the class:

$$\Delta = (\underbrace{\odot, \odot, \dots, \odot}_{Y}, \underbrace{1, 1, \dots, 1}_{\lambda_1 > \mathrm{X}}, \dots, \underbrace{k - \mathrm{Y}, \dots, k - \mathrm{Y}}_{\lambda_{k-\mathrm{Y}} > \mathrm{X}})$$

In order to simplify the notation, let Ψ equal the sum of terms covered by $(\underbrace{\star, \dots, \star}_{Y}, \underbrace{1, 1, \dots, 1}_{\lambda_1 > \mathrm{X}}, \dots, \underbrace{k - \mathrm{Y}, \dots, k - \mathrm{Y}}_{\lambda_{k-\mathrm{Y}} > \mathrm{X}})$ but not in Δ.
The sum of terms in Δ equals:

$$(\underbrace{\star, \dots, \star}_{Y}, \underbrace{1, 1, \dots, 1}_{\lambda_1 > \mathrm{X}}, \dots, \underbrace{k - \mathrm{Y}, \dots, k - \mathrm{Y}}_{\lambda_{k-\mathrm{Y}} > \mathrm{X}}) - \Psi.$$

Note that the starred term in the previous equation is a (Y,Y)-block. It turns out that all terms that *need* to be summed in order to compute Ψ are 0-blocks since any term covered by a (Y, Y)-block and not in Δ has to be missing at least one database index. This can be seen by noting that one of the starred (or "bagelled") coordinates will equal an element of $\{1, 2, \ldots, k{-}\mathrm{Y}\}$. Hence, by the pigeonhole principle, an element of $\{k{-}\mathrm{Y}{+}1, \ldots, d\}$ will not appear in the term. It follows that Δ can be spanned by 0-blocks and a (Y, Y)-block.

Induction hypothesis: We assume the property holds for less than ρ $\odot$s ($\rho > \mathrm{Y}$) and show that it holds for ρ. Without loss of generality consider the class:

$$\Delta' = (\underbrace{\odot, \ldots, \odot}_{\rho}, \underbrace{1, 1, \ldots, 1}_{\lambda'_1 > \mathrm{X}}, \ldots, \underbrace{\beta, \ldots, \beta}_{\lambda'_\beta > \mathrm{X}}).$$

In order to simplify the notation, let Ψ' equal the sum of terms covered by $(\underbrace{\star, \ldots, \star}_{\rho}, \underbrace{1, 1, \ldots, 1}_{\lambda'_1 > \mathrm{X}}, \ldots, \underbrace{\beta, \ldots, \beta}_{\lambda'_\beta > \mathrm{X}}))$ but not included in Δ'.

The sum of terms in this class (Δ') equals:

$$(\underbrace{\star, \ldots, \star}_{\rho}, \underbrace{1, 1, \ldots, 1}_{\lambda'_1 > \mathrm{X}}, \ldots, \underbrace{\beta, \ldots, \beta}_{\lambda'_\beta > \mathrm{X}}) - \Psi'$$

Note that the starred term is a (ρ, $k - \beta$) block. Now, remark that the terms that *need* to be summed in Ψ' are all either 0-blocks or included in covering classes with less than ρ $\odot$s since they cannot be included in a class with more than ρ $\odot$s. (The non-starred (or non-bagelled) coordinates are fixed.)

Furthermore, if a term summed in Ψ' belongs to some covering class, then all terms belonging to this covering class are also summed in Ψ'. Hence, by the induction hypothesis, Ψ' can be spanned which implies Δ' can also be spanned.

$\square$

Theorem 3.3.2 *With $d = k + (k-(Y-1))X - 1$, $(\star, \dots, \star)$ can be spanned by (A,B)-blocks having the property that there are at most $k - Y$ indexes appearing more than X times ($A \geq B \geq Y$).*

Proof: Since the sum of terms in every covering class having the required property can be spanned (by lemma 3.3.1) and that all terms belong to exactly one such class or are 0-blocks, the sum of all terms can be spanned.

$\square$

Remark: X and Y need to be chosen carefully as retrieving $(\star, \dots, \star)$ as a sub-problem makes no sense. Setting $d >$XY assures us that we will not have this problem since at least one database index will appear more than X times in every term. X and Y need to satisfy the following inequalities:

$$
\begin{aligned}
d &> XY \\
k + (k - (Y-1))X - 1 &> XY \\
(X+1)k - XY + X - 1 &> XY \\
(X+1)k + X - 1 &> 2XY
\end{aligned}
$$

Every list ((A-B)-block) that needs to be transferred is seen by *at least* Y databases. These lists have different lengths, for example the shortest one has length $(n^{1/d})^Y$ whereas the largest one has length $(n^{1/d})^{X(k-1)}$. The most expensive lists to transfer (via PIR) are those of length $(n^{1/d})^{Xb}$ which are seen by b databases. For all known efficient PIRs, it turns out that of all possible values for b, $b = Y$ requires the most communication. As an upper bound is sought, it will be assumed that all PIRs executed have the same communication complexity as the PIR followed for these (XY,Y)-blocks.

How many lists do we need to transfer? We need to send the same number of lists as classes whose members need to be spanned. Being loose in our analysis, there are $O(k^d)$ such classes. Again, preferring clarity for efficiency, we perform one PIR per list[4].

[4]Note that these simplifications do not effect our result in a substantial manner.

Communication Complexity: The share distribution part of the protocol takes, as usual, $O(k^3 n^{1/d})$. The sub-PIRs take $O(k^d \mathrm{PIR}_Y((n^{1/d})^{XY}))$ bits of communication where $\mathrm{PIR}_Y(m)$ is the communication complexity of a Y database PIR with a data string of length m. Our scheme needs $O(k^3 n^{1/d} + k^d \mathrm{PIR}_Y(n^{XY/d})) = O(n^{1/d} + \mathrm{PIR}_Y(n^{XY/d}))$.

In the next section, we give some explicit bounds.

3.4 Analysis

The previous PIR is not as clean and is more complicated than those in [9, 52] and so requires a more careful analysis. We need to find an optimal sub-protocol (sub-PIRs to transfer lists) and optimal values for X and Y. We now give a table that shows a few bounds for different values of X, Y, k and for different sub-PIRs.

Remember that k is taken to be a constant. Unless otherwise noted, we use section 3.2's PIR as a sub-protocol.

	Complexity	$k = 3$	$k = 6$	$k = 100$
section 3.2's PIR	$O(n^{1/2k-1})$	$O(n^{1/5})$	$O(n^{0.091})$	$O(n^{0.00503})$
X, Y = 2	$O(n^{(4/3)(1/3k-3)})$	$O(n^{2/9})$	$O(n^{0.089})$	$O(n^{0.00449})$
($\star$)X=2, Y=3	$O(n^{(6/5)(1/3k-5)})$	-	$O(n^{0.092})$	$O(n^{0.00407})$
X=3, Y=10	$O(n^{(30/18)(1/4k-28)})$	-	-	$O(n^{0.00448})$
($\odot$) X=3, Y=10	$O(n^{(180/125)(1/4k-28)})$	-	-	$O(n^{0.00387})$

$\odot$: using $\star$ as sub-PIR.

For most real world applications, n is very large and so our protocols yield a significant improvement.

Given that $k \gg X, Y$, the larger X and Y are, the more efficient the protocol is (first, second and third table entry). Furthermore, using an efficient sub-PIR decreases the communication as well (third and fourth table entry).

As k is constant, optimal values for X and Y can easily be found by inspection – simple heuristic. Furthermore, it always makes sense to use the most efficient PIRs as sub-protocols.

Unfortunately our scheme is not applicable to HIO as when k is chosen to be non-constant, k^d is super-polynomial (in the PIR scenario, k^d is a multiplicative constant and so does not appear when using the big-oh notation). An interesting open problem is to find a way to span $(\star, \ldots, \star)$ using only polynomially (in k) many (A,B)-blocks. This would yield a significant improvement over the best known HIO. This might be possible but unfortunately we have not been able to make lots of progress in this direction ... Having k not be a constant allows us to "unfold" the analysis and find a clean value for the communication complexity (i.e. one that does not include the communication complexity of the sub-PIR).

Chapter 4

Private Information Retrieval with Authentication

We define and give protocols for a new problem which we call private information retrieval with authentication.

Before delving into our new extension, we present some definitions and protocols.

4.1 Preliminaries

Symmetric Private Information Retrieval (SPIR), Private Information Retrieval of Blocks (PIRB) and PrivatE infomation Retrieval by KeYwords (PERKY) are now presented.

4.1.1 Symmetric Private Information Retrieval

In their fascinating article [42], Gertner et al. show how to construct information theoretic distributed oblivious transfers (a.k.a. Symmetric private information retrievals (SPIRs)). These protocols are like ordinary PIRs with the additional constraint that the database privacy is protected. $\mathcal{U}$ can only obtain information about one *physical* bit of data.

It is shown that in order to have an information theoretic SPIR, the databases need to have access to a common random string.

Fact 4.1.1 *There exist no (multi-round) k-database SPIR without direct interaction between different databases, even if the databases are allowed to hold private, independent, random inputs, and the user is semi-honest.*

Here are a few of the most important and useful results from their work:

Fact 4.1.2 *There exists a method in which we can transform an arbitrary k database PIR with communication complexity $C(n)$ into a $k+1$ database SPIR with communication complexity $O(C(n))$.*

Fact 4.1.3 *There exists a method in which we can transform all k database PIRs exposed in this work into k database SPIRs having the same communication complexity (up to a multiplicative constant).*

Remark: The user is assumed to be arbitrarily malicious when the database privacy is to be protected (i.e. in SPIR).

4.1.2 Private Information Retrieval of Blocks

PIRs allow $\mathcal{U}$ to retrieve a single bit; unfortunately this is not realistic as in most real world applications, the data is usually arranged in blocks (an index refers to more than one bit). The problem of privately retrieving blocks (PIRB) is studied in [31, 32] and many protocols solving this problem are given. We now present one of these.

Given any PIR, we can transform it into a PIR of blocks (PIRB). Suppose the databases have blocks of size m, the data-string can be thought of as an $m \times n$ array. Each of the n columns represents a block.

The user sends *one* query for the ith record and the databases process this query on each of the m rows and send each answer. Suppose we have a PIR in which $\mathcal{U}$ sends $\alpha(n)$ bits and the databases send $\beta(n)$ bits then the communication complexity of the PIRB is $\alpha(n) + m\beta(n)$. The protocol is graphically presented in figure 4.1. (Where $Q(i)$ is a PIR query pointing to the ith element and $A(j, Q(i))$ is the answer obtained when processing the query $Q(i)$ using jth bit of each block as the data string.)

Note that we can obtain a symmetric PIRB (SPIRB) by simply replacing

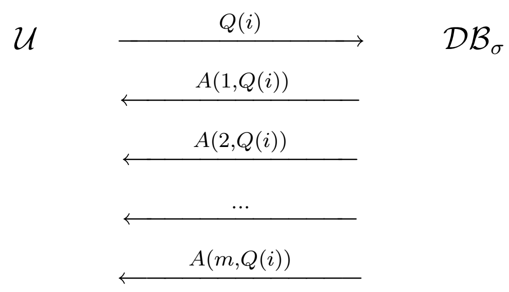

Figure 4.1: A simple PIRB.

the PIR with a SPIR. (A SPIRB is a PIRB in which the user can only obtain information about the bits in *one* block.)

4.1.3 Private Information Retrieval by Keywords

The n bit data-string used in PIRs does not properly model most databases. Indeed, most databases are queried by keyword instead of by index. Users are generally interested in searching a database for a keyword. This problem is tackled in [30] where PrivatE infoRmation by KeYwords (PERKY) protocols are defined as :

Definition: Suppose each of the k databases possess n, m bit strings $s_1, \ldots, s_n$ and that $\mathcal{U}$ holds a string $w \in \{0,1\}^m$. A solution to the PERKY problem allows the user to find out if there exists a $j \in [n]$ such that $w = s_j$, without leaking any information about w to the databases.

Figure 4.2: A binary search tree block

In addition to defining the problem, Chor et al. [30] give many interesting protocols. Some of these use classical data-structures such as tries and binary search trees [34]. We now present a protocol that uses balanced binary search trees. These trees have $\lg(n) + 1$ levels and n leaves.

The binary tree are implemented as $\lg(n){+}1$ arrays[1], the arrays containing the nodes for level i have 2^i entries $(0 \leq i \leq \lg(n))$. Each entry (block) consists of a left and a right pointer and a label. The pointers are simply addresses (indexes) of the next level's nodes. The last level's pointers are set to -1. See figure 4.2 for a graphical representation of a block. Note that we assume that there are no duplicate entries in the tree and that all keywords can be ordered in some way (e.g. alphabetically).

We can use this structure for PERKY if the leaves are labeled by $s_1, \ldots, s_n$. The user searches the tree for w and if the leaf he gets to is labeled by w he knows $w \in \{s_1, \ldots, s_n\}$ and if not, he is assured $w \notin \{s_1, \ldots, s_n\}$.

[1] one for each level.

The idea behind the scheme in [30] is to have the user perform an oblivious walk on the binary search tree. This is done as follows:

1. $\mathcal{U}$ retrieves the root[2]. If w is less than the root's label, he will use the address of the left child at the next level; otherwise he will take that of the right child.

2. $\mathcal{U}$ and the databases perform a PIRB using the level one array as the data-string. The user retrieves the node he chose in the first step (of which he knows the index). He then determines which child to follow to the next level.

3. The protocol continues like this until the last level is reached and the user can determine whether $w \in \{s_1, \ldots, s_n\}$ or not.

Note that, at every level, the databases have no information on which index the user is querying and so are oblivious to the path followed.

An interesting (and perhaps useful) observation is that the data structure queried can be used for regular (non-private) queries as well.

The protocol is graphically represented in figure 4.3. $Q_j(l_j)$ is the query for the jth level of the tree. $A(l_j, Q_j(l_j))$ is the answer to the query $Q_j(l_j)$

[2]There is no need to use a PIR for this.

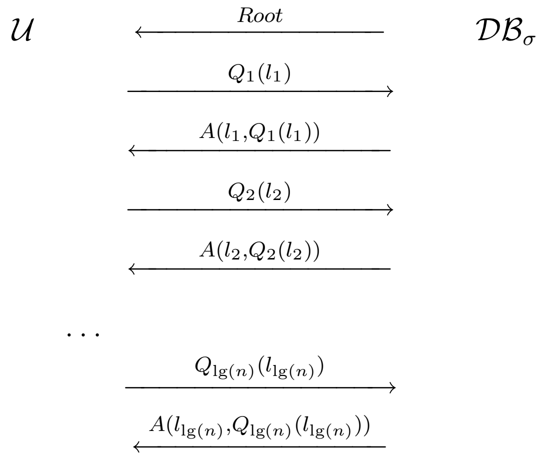

Figure 4.3: A simple PERKY

using the jth level array as the data string.

Communication Complexity: Remember that the keywords have length m and assume the labels all have length m. If the PIRBs have communication complexity $C(n, m)$, the communication complexity of the previous scheme equals $O(m + C(2, m) + C(4, m) + \ldots + C(n, m)) \subseteq O(\lg(n)C(n, m))$.

Unlike the previous retrieval schemes, this one needs more than one round of communication. It is not too complicated to see that the protocol takes $\lg(n) + 1$ rounds; one per level.

4.2 A Definition

In many application, both user privacy and user authentication are requirements. These two seemingly contradictory goals give rise to many interesting problems and solutions. For example, the digital cash schemes of Chaum et al. [28] and Brands [21], maintain user privacy *and* provide authentication (i.e. coin validation).

We are able to present a protocol for PIR which only works if the user is "allowed" to access the database entry he is querying. That is, the user proves that his query points to a database index he is allowed to access without revealing anything about the value of this index. The idea behind our protocols is to have $\mathcal{U}$ prove to the databases that his query retrieves a bit he is allowed to access.

Definition: A Private Information Retrieval with Authentication scheme (PIRA) is a SPIR in which the user cannot (with high probability) obtain information about entries he has not been authorized to access.

4.3 Protocol Phases

We present a high-level overview of the major protocols needed to construct a PIRA system.

4.3.1 Registration

In order to have access to a database entry, the user first needs to obtain the approval of some party – the *authenticator*. The information[3] the user needs to present in order to obtain authorization depends on the application and so is not discussed here. As will be seen in the following section, there are different mechanisms that can be used in order to "register" users.

4.3.2 Authenticated Private Query

The goal of this stage is to have the user send an "authenticated" SPIR query for a database entry i and *convince* the databases that he will not obtain any information about entries he has not been allowed to access.

4.4 A Protocol Based on Ideas from Electronic Cash

A straightforward method for solving this problem is to have the authenticator create the queries for the user. This might be a viable option in many settings but, unfortunately, we need to trust that the authenticator will not collude with the databases. Simple extensions to the privacy protecting electronic cash protocols of [21] can be used to eliminate this assumption.

[3] e.g. drivers license, credential, etc.

The idea is to get the authenticator to sign (authenticate) the user's queries in such a way that :

1. The *authenticator* cannot recognize his signature (i.e. it is blinded). Otherwise, he can collude with a database and reveal the user's identity and, more importantly, the index queried, i.

2. The *authenticator* should not know the queries he has signed. Otherwise, he might be able to collude with a database and reveal the user's identity and the index queried, i.

3. The probability that the user gets a signature on queries pointing to an index other than i should be small.

Now, the user can convince a database to process his query by providing a proof that the authenticator has authorized it (i.e. showing the signature).

A simple modification[4] of the protocol described in section 4.5.2 of [21] allows us satisfy the first two "requirements". The requirement that the probability that a user gets a signature on an invalid queries be small can be solved by methods specific to the underlying SPIR. We will not go into

[4]Because of space constraints, we do not describe Brands' constructs or even attempt to give some intuition. Presenting the basic notions, by themselves, would almost constitute a full thesis. We refer the uninitiated reader to [21] for more details and to [56, 67] for good introductions to fundamental cryptographic primitives.

these but, instead, we propose a general method which works for any PIR. The method is based on the "cut and choose" paradigm.

The protocols are based on the assumption that the discrete log problem is computationally intractable (see [56, 67] for details).

The user will prepare and send q "blinded" PIR query sets (pointing to i if the user is honest) and the authenticator will ask him to unblind $q-1$ of them. Brands' constructs have the useful property that the user cannot change his queries, that is, he cannot show that his blinded query set corresponds to more than *one* query set. So, in a sense, the blinded query set is a commitment to the actual queries.

It is easily seen that the user has a $1/q$ chance of obtaining a signature on an invalid query.

More rigorously, we have:

1. Given a generator g that generates a group for which the discrete log problem is difficult, the authenticator chooses k random values, $y_1, \ldots, y_k$, from $[e]$, where e is the order of the group generated by g. The authenticator then publishes/distributes the values $g_1 = g^{y_1}, g_2 = g^{y_2}, \ldots, g_k = g^{y_k}$ to *all* users (keeping the y_js private of course).

2. The user takes these values and sends $h_z = g_1^{Q_1^z} g_2^{Q_2^z} \ldots g_k^{Q_k^z}$, for all $z \in$

$[q]$, where Q_j^z is the query destined for $\mathcal{DB}_j$ in the z'th query set. Note that given the assumption that computing discrete logs is intractable, the queries are blinded (the databases cannot determine what there values are).

3. The authenticator randomly chooses $q-1$ elements from $[q]$ and asks the user to reveal all of the queries associated with these indexes.

4. The authenticator and the user then carry out the protocol in section 4.5.2 in [21] so that the user's query commitment can be authenticated. The protocol used has the property that the information that the user will present to the databases cannot be linked with any particular registration session.

We now show that the user cannot "change" the values of his queries once they have been committed[5].

Theorem 4.4.1 *The user cannot change the value of his commitments in the scheme given above. More precisely, let a commitment be of the form* $h = g_1^{Q_1} g_2^{Q_2} \dots g_k^{Q_k}$. *The user cannot find a* $(Q'_1, \dots, Q'_k)$ *not equal to* $(Q_1, \dots, Q_k)$ *such that* $h = g_1^{Q'_1} g_2^{Q'_2} \dots g_k^{Q'_k}$. *Note that the queries, taken individually,* are

[5] Note that this property is *not* used in [21].

random (otherwise the PIR scheme is not unconditionally secure).

Proof:

We show that if the user had an efficient algorithm $\mathcal{A}$ for changing his commitments, he could find $(y_1, \ldots, y_k)$ which violates the discrete log assumption.

First notice that if a user knows a $F = (Q_1, \ldots, Q_k)$ and a $F' = (Q'_1, \ldots, Q'_k)$ (with $F \neq F'$) such that $h = g_1^{Q_1} g_2^{Q_2} \ldots g_k^{Q_k} = g_1^{Q'_1} g_2^{Q'_2} \ldots g_k^{Q'_k}$ he can easily infer that $y_1 Q_1 + \ldots + y_k Q_k = y_1 Q'_1 + \ldots + y_k Q'_k$. But this is an equation with the y_j being the variable and we just need k other such equations (with different coefficients), which can be obtained using $\mathcal{A}$, in order to be able to solve a system of equation and thus find all y_js! Hence, given an algorithm $\mathcal{A}$, it is easy to compute discrete logs which contradicts our assumption.

□

It is important to note the following.

1. Firstly, the user must be computationally bounded[6] otherwise he can forge the authenticator's signatures and change his commitments.

2. The authenticator and the databases can be assumed to be computationally unbounded.

[6]He must not be able to compute discrete logs.

User $\xrightarrow{\quad q\text{, blinded query sets}\quad}$ **Authenticator**

$\xleftarrow{\quad q-1 \text{ indexes from } [q]\quad}$

$\xrightarrow{\quad \text{Queries Associated with}\quad q-1 \text{ blinded query sets}\quad}$

Figure 4.4: E-Cash Inspired Issuing Protocol

The signature issuing protocol is graphically presented in figure 4.4.

Note that it is not clear at all how we could implement this protocol using Chaumian blinding [26] as these constructs do not have the commitment property. That is, the user could "reveal" anything/query he wants, he has not *committed* to anything with the blinded query set.

4.5 A Password Based Solution

The idea behind the following construction is to have the user obtain a password for each index he can access. The databases will then need to verify that the password is consistent with the index being queried without knowing either the password nor the index.

The registration protocol is uninteresting as the user simply needs to obtain passwords for each index he is interested in from the authenticator. Hence, we focus on the authenticated query part of the system.

We separate the remainder of this section in two, the password verification part and the query validation part.

Remark: Unless otherwise noted, all computations will be carried out in $GF(2^h)$ where h is the length of the inputs which will be clear from the context.

4.5.1 Password Verification

For each index he is allowed to access, the user will have a password $\mathcal{P}$ of length m. To access the ith bit, the user has a password which we will denote by $\mathcal{P}_i$. Furthermore, assume that given $\mathcal{P}_i$, it is easy to determine i. For example, we might have that the first $\lg(n)$ bits of the password $\mathcal{P}_i$ equal i.

All the passwords will be stored in a data-structure that is compatible with PERKYs. This data-structure will be held by a second set of databases, $\mathcal{PDB}_1, \ldots, \mathcal{PDB}_e$, who will be queried by the databases possessing the data-string $\mathcal{U}$ is interested in $(\mathcal{DB}_1, \ldots, \mathcal{DB}_k)$. In order to facilitate the presentation, we will assume $k = 2$ (the number of databases equals 2).

The problem here is to have the databases query a data-structure for an element they do not know (if the user reveals his password all is lost!). This can be done as follows.

$$\mathcal{U} \quad \xrightarrow{\ p_1\ } \quad \mathcal{DB}_1 \quad \xrightarrow{\text{Shift Entries By } p_1} \quad \mathcal{PDB}_J\text{'s}$$

$$\xrightarrow{\ p_2\ } \quad \mathcal{DB}_2 \quad \xleftrightarrow{\text{PERKY for } p_2}$$

Figure 4.5: The Password Verification Protocol

1. The user randomly chooses p_1 from $\mathbb{Z}_{2^m}$ and sets $p_2 = \mathcal{P}_i - p_1$ ($\mathcal{P}_i$ is additively shared). p_1 and p_2 are then sent to $\mathcal{DB}_1$ and $\mathcal{DB}_2$ respectively.

2. $\mathcal{DB}_1$ sends p_1 to all $\mathcal{PDB}_j$s who then subtract it from the passwords and rearrange the data-structure to cope with the change.

3. $\mathcal{DB}_2$ acts as a user and performs a PERKY (keyword $= p_2$) with the $\mathcal{PDB}_j$s. If p_2 is in the database then the password is valid.

The graphical representation of the protocol can be seen in figure 4.5.

Notice that the password has been verified without any database gaining any information about it other than whether it is valid or not.

Communication Complexity: The communication complexity of the password verification scheme is $O(C(o,m))$ where o is the number of passwords, m is the length of the password and $C(o,m)$ is the communication complexity of the PERKY.

Remark: The user has a small probability of randomly choosing a correct password. Fortunately this probability is exponentially small in the length of the password m.

4.5.2 Query Validation

The last subsection solves an important problem by obliviously verifying the password. Unfortunately, the user can still query any bit he wants as long as he has a valid password. We need to force the query to be consistent with the password. That is, $\mathcal{P}_i$ must only be used with queries to the ith bit. Two solutions to this problem are now given.

Secure Multi-Party Computations and Conditional Disclosure of Secrets

First note that if we allow the databases to communicate with each other, they can perform a secure multi-party computation [35] to determine whether the query is consistent with the password.

Fortunately, we do not need to change our assumptions if we use an interesting primitive called conditional disclosure of secrets (CDS) first introduced in [42]. This primitive allows k players to disclose a secret to another participant (Carol) if and only if some function $f(x_1, \dots, x_k)$ evaluates to 1.

Definition: A conditional disclosure of secret (CDS) protocol has $k + 1$

participants : k players, $P_1, \ldots, P_k$ (who can only communicate with Carol), receive inputs from the other participant who is called Carol. Let x_j denote the value received by P_j. At least one of the k players possesses a secret s and the players share some random string.

After receiving their inputs, each player sends Carol some information. If $f(x_1, \ldots, x_k) = 1$, Carol can take all the data she has received and determine s. But if $f(x_1, \ldots, x_k) \neq 1$, Carol gains no information about s.

This is exactly what we need! We just have to construct a function $f()$ taking as inputs the query sent by the user (password and SPIR query) that outputs 1 if and only if the query and the password are related to the same index[7]. The secret can be taken to be the answers of the databases. It follows that the user will only gain information about the database answers if $f()$ evaluates to 1.

The construction of the function $f()$ depends on the SPIR used and so will be omitted from this work. CDS protocols are relatively efficient as can be seen from the following fact taken from [42].

Fact 4.5.1 *There exists a protocol for CDS that has communication complexity $O(z^2)$ where z is the size of the branching program[8] realizing $f()$.*

[7] Remember that it's easy given a password to find the associated index.
[8] See [51] for details.

Cut and Choose

The idea here is to have the user send many queries that will almost all be checked for validity (without revealing i). The remaining query will be used to retrieve the desired bit.

The password will be composed of two parts : the index and some random identifier, id. In order to simplify the presentation, we assume that there are two databases $(\mathcal{DB}_1, \mathcal{DB}_2)$.

The user prepares and sends β queries in the following manner[9]:

- $\mathcal{U}$ chooses two random values q, w such that $q + w = i$. q will be taken to be the shift value and w the query index.
- $\mathcal{U}$ prepares and sends SPIR queries for retrieving the wth bit.
- $\mathcal{U}$ chooses six random values $q_1, q_2, w_1, w_2, id_1, id_2$ such that $q_1 + q_2 = q$, $w_1 + w_2 = w$ and $id_1 + id_2 = id$. He sends q_1 and w_1 to $\mathcal{DB}_1$ and q_2 and w_2 to $\mathcal{DB}_2$.

All the passwords are first verified using a *slightly* modified version of techniques of subsection 4.5. That is, the password now consists of two parts

[9]The field in which the computations are carried out and the random numbers chosen will be clear from the context.

that are already additively shared and both parts can just be concatenated so that the normal protocol can be followed.

The databases then randomly choose $\beta - 1$ of the queries and check that the SPIR queries point to w (they share w_1 and w_2 and the SPIR queries). In order to verify the correctness of the $\beta - 1$ queries, the databases can communicate with each other or transmit messages via the user. This might be acceptable in some settings but it does violate the assumption that databases do not communicate with each other. We propose a simple and efficient solution that does not violate this assumption.

First note that Gertner et al. [42] provide an efficient CDS scheme[10] for functions verifying that the sum of all distributed shares equal 0. This protocol can be easily modified for our purposes, precisely, we have that :

Fact 4.5.2 *There exists a CDS scheme in which the user sends w_1, w_1' to $\mathcal{DB}_1$ and w_2 and w_2' to $\mathcal{DB}_2$ and the user obtains the secret if and only if $w_1' = w_2'$ and $w_1 + w_2 = w_1'$.*

Furthermore, the communication complexity of the scheme is the largest of the lengths of w_1, w_1', w_2, w_2' and of the secret.

If the user sends the query index, w, to each database for each of the

[10] Lemma 2 of [42].

$\beta - 1$ chosen queries, we can easily use this scheme to reveal some secret to the user if and only if he has correctly followed the protocol (i.e. $w_1 + w_2 = w$ and the same w was sent to both databases).

The trick now is to have the databases apply the selected queries on databases containing random entries (for which the databases know the value of the w'th bit) and use the response as the secret. By repeatedly applying the queries on different databases, the databases can make sure that the user's query points to the correct value (by asking the user to divulge the wth bit of each of these random databases).

This trick works because, if the user cheats and does not divulge the correct w, he will not gain any information about the database responses and so no information on the wth bit. Furthermore, the queries protect the database privacy and so if the query does not point to the w'th bit then the user has no information about it.

It is not hard to see that for every execution (for each random database) of this protocol, the user has a 1/2 chance of cheating. Hence, if we repeat the process t times, the user has a probability of success of $1/2^t$.

The communication complexity of this scheme is reasonable :
$(\beta - 1)tk$ max(length of database responses, length of passwords).

Now that this problem is solved, we see that since w is a random index, the databases gain no information about i. If the user has cheated in one of these $\beta - 1$ queries then he is assured of getting caught.

For the remaining query, the databases tell each other their share of q and cyclically shift the data-string q positions to the left. The SPIR query should now point to the correct slot ($i - q = w$). The SPIR query is then processed on the shifted data-string.

The user has one chance in β of querying a bit he is not allowed to see (he can give a correct password $\mathcal{P}_{i'}$, and use a SPIR query pointing the ith bit).

Communication Complexity: The communication complexity of this scheme is $O(\beta(C(n) + m))$ where $C(n)$ is the communication complexity of the SPIR and m is the length of the password.

Chapter 5

Applications

Although PIR is an interesting primitive from a theoretical perspective, the potential application that are mentioned in the literature are rather uninteresting. For example, retrieving stock prices or patent information from a database privately probably do not satisfy any important real world requirements, in any case, they certainly are not “killer apps”. In this chapter, we attempt to improve this deficiency by presenting more convincing applications.

5.1 Privacy Protecting Network Information Databases

Since Chaum introduced constructions for anonymous broadcast networks[1] [27] (i.e. dc-nets) and privacy protecting networks[2] (i.e. mix-nets) [25] many

[1]The identity of the sender of the broadcasted message is hidden from most attackers.
[2]Most attackers cannot determine which parties are communicating with each other.

researchers have attempted to implement them in real world settings. One of the difficulties in making the transition from theoretical constructions to real world systems is that networks are *not* stable (network configurations change, machines malfunction, etc.). This problem is usually dealt with by having network information query servers that parties can use to obtain information pertaining to the current network status.

These servers are usually fine for "normal" systems. However, for anonymous broadcast and privacy protecting networks, a simple query can violate user privacy. Indeed, queries typically reveal information about the system's clients that could violate their privacy or, at the very least, leak information that can be used in subsequent attacks. Hence, the use of these network information query servers basically obliges the user to trust the server operator.

The use of PIRs with network information query servers would effectively solve this problem as the server would not have any knowledge of what the clients are interested in. Note that ideas of this kind were first suggested in [33].

5.2 Unlinkable e-mail Addresses

In some settings (e.g. [25, 43]) individuals want their different e-mail addresses to be unlinkable. For example, we can see why one would want to hide that the e-mail address *Clark_Kent@anonymous.com* does not belong to the same person as *Superman@anonymous.com*. At first glance, it seems that this problem can be easily solved by mix-nets and/or dc-nets which were mentioned in section 5.1. However, the problem is that users typically want to retrieve all of their e-mail messages at the same time and so, the e-mail storage system administrators might be able to link e-mail addresses just by looking at the time at which they are accessed.

It's not hard to see that if users collected their e-mails with PIR, this problem could be solved. Note that PIRA is more appropriate in most settings as e-mails are usually confidential and the system administrators cannot enforce access restrictions if a simple PIR is used.

5.3 Privacy Protecting Certificate Revocation Databases

Public Key Infrastructures (PKI) [40], although they allow confidentiality and authentication, pose serious privacy risks. If governments, companies and organizations do not change their plans, PKIs will probably form the foundations for the most extensive surveillance system ever known to mankind. In view of the fact that companies specializing in PKIs [1, 2, 3] are already worth millions of dollars, it will probably be hard to prevent Big Brother from coming into existence (see [21] for more details). In addition to the fascinating techniques of [21], PIRs could help us annihilate this threat.

One of the most important construct in PKIs is the Certificate Revocation List (CRL), this list allows the Certificate Authority (CA) to revoke certificates. Large CRLs are not attractive from a privacy perspective as they cannot be distributed to all parties and so if Alice wants to communicate with Bob, she will need to query a CRL databases in order to verify that Bob's certificate is valid. But this allows the CRL database operator to link who is communicating with whom which is clearly a breach of privacy. PIR can obviously be used to solve this problem. We believe this is going to be the first real world application in which PIRs will be used as the size of the CRLs usually is not extremely large and so the resulting retrieval scheme will be efficient enough.

5.4 Privacy Protecting Distributed Information Storage and Distribution Systems

The Internet has always provided mechanisms in which users can trade, distribute and exchange information. This year, a lot of controversy has erupted because users have been using this functionality to distribute (perhaps illegally) copyrighted material[3]. Parties distributing, retrieving and facilitating the transfer of copyrighted material have even been taken to court [5, 6]. Whether these lawsuits will be successful remains to be seen but, in any case, PIR schemes can be used in order to complicate prosecutor's tasks even more *and* protect participant privacy ...

Note that even though this is a controversial example, we remark that one could think of situations in which these mechanisms are clearly desirable (e.g. political dissidents securely publishing manifestos, etc.).

There are many paradigms for distributing information, we limit ourselves to two : the distributed server repository and the massively distributed user repository with indexing server.

[3]See for example [4].

5.4.1 Distributed Server Repository

Here, the data files are submitted to a certain number (> 1) of repository servers where they can be retrieved by other clients. At first glance, this setup seems vulnerable to attacks against privacy. That is, an attacker can determine what the servers are distributing and what the users are retrieving.

Cryptographic techniques can however make this model more resistant.

- Firstly, the data can be distributed using a secret sharing scheme (see section 3) so that any group of less than t servers do not *know* what data is actually stored. This protects the servers as they do not know what is stored on their system.

- Secondly, the clients can access the servers using a PIR. This protects the clients as they do not reveal what information they actually obtain.

- Thirdly, for added security the participants can use mix-net [25] type communication channels.

5.4.2 Massively Distributed Repository with Indexing Server

In this setting, the data is stored on a large number of machines, typically, each user will be a repository and an indexing server is used to help in determining where the files are stored[4]. The naive implementation of this model is not resistant as all user/repositories are obviously vulnerable *and*, even the indexing server is susceptible to law suits for facilitating the transfer of dubious data files. Again, using cryptographic methods allows us to overcome these shortcomings.

- Firstly, the indexing server can be queried using a slight extension to PERKY. This protects the indexing server since it does not know what information it is distributing.
- If just storing indexes is risky, the indexes can be distributed over a large number of indexing servers using a secret sharing scheme.
- A secret sharing scheme can be used to distribute the actual data (the users do not know what they are storing).
- The users/content providers can be accessed via PIR so that they do not know what they are distributing.
- Finally, if an extra layer of security is required, all participants can communicate via a privacy protecting communication channel (e.g. mix-nets).

[4]This is the model used by napster [4].

Chapter 6

Conclusion

The improved upper bound on information theoretic PIR presented in section 3 provides a new framework for working with PIRs. We believe this is a significant contribution to the state of the art in cryptography. The result not only improves the best known upper bound but it also poses some new open problems which might pave the way for other interesting research. Indeed, it would not be surprising for future work to extend our methods and improve the tightness of our bound. We mention the two most important problems related to this result.

- Is there a way of more tightly spanning $(\star, \star, \dots, \star)$? If this can be done using only polynomially many in k different blocks, this would yield a significant improvement in the HIO model.

- It would be interesting to gain a more solid understanding of the properties exploited in this protocol; perhaps through a link with other mathematical theories (coding?).

The second contribution of this work has focussed on extending the functionality of PIR. We believe this is an important step in the path towards using PIR based protocols in practice. This work has greatly increased the range of applications in which we can foresee using PIRs.

Addressing real world requirements is not only potentially useful but can also yield interesting problems and elegant solutions. Private information retrieval with authentication nicely illustrates this point.

We mention the three most important problems related to this result.

- Are there efficient methods that allow the user to *prove*, without leaking any other information, that his queries (in the e-cash inspired model) are valid without actually revealing them. This would allow us to get rid of the cut and choose step which is clearly the weakest link of the protocol. Brands, in [21], shows that some useful (for credentials) properties can be efficiently demonstrated. Unfortunately, these do not seem to be helpful for demonstrating the validity of PIR queries.

- Are there better ways to bind password and query? Can we do better than linear security in the number of queries?

- Are there tricks that could help us in the computational setting ? We have put little effort addressing this problem even though it does seem very promising.

It is an extremely important to try and motivate research using convincing practical applications. Theoretical research without grounding in real world considerations risks being totally arbitrary and of following the path of least resistance. In chapter 5, we present some new applications which truly validate the study of PIRs. Researchers and practitioners can look at PIR and see more than just a frivolous research topic. We note an interesting open problem :

- In section 5.2 we propose the use of PIRs. We note however that they have security properties that are *not* required to solve this problem (we just need unlinkability, not query privacy). Are there schemes that are more efficient than PIR for the e-mail unlinkability problem?

In concluding, we would like to stress that this work has only tackled a small subset of interesting questions related PIR and that there is still lots of work to be done. We now mention a few of the most interesting problems which peek our curiosity :

- Can we assume more powerful adversaries? For example, can we do anything if a subset of the databases can be arbitrarily malicious?
- Can we come up with an efficient probabilistic PIR? That is, one in which the databases gain only probabilistic information such as : the query points to bit j with probability $2/n$ (more than the current $1/n$).
- What are the real world implementation (performance, pitfalls)?
- What are the other functionalities which would be useful?
- Can we integrate PIR with information dispersal algorithms [63]?
- Can we find lower bounds? There are a few results [31, 32], but they are weak.
- As noted by A. Back [14], another way of looking at the problem is to hide the meaning[1] of i to the database. There is a simple and clever way of doing this using mix nets [25], are there other?

[1] For example, the database does not know that the value stored at database index i is Alice's e-mail box.

Appendix A

List of Acronyms

1. `cPIR`: computational Private Information Retrieval
2. `HIO`: Hiding an Instance from an Oracle
3. `PERKY`: PrivatE information Retrieval by KeYword
4. `PIR`: Private Information Retrieval
5. `PIRA`: Private Information Retrieval with Authentication
6. `PIRB`: Private Information Retrieval of Blocks
7. `SPERKY`: Symmetric PrivatE information Retrieval by KeYword
8. `SPIR`: Symmetric Private Information Retrieval
9. `SPIRB`: Symmetric Private Information Retrieval of Blocks

Bibliography

[1] Baltimore Technologies, http://www.baltimore.com.

[2] Entrust, http://www.entrust.com.

[3] VeriSign, http://www.verisign.com.

[4] Napster, http://www.napster.com.

[5] http://www.stopnapster.com.

[6] http://www.cnn.com/2000/LAW/06/23/tech.napster.reut/index.html.

[7] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle (extended abstract). In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 195–203. ACM Press, 1987.

[8] L. Adleman, R. L. Rivest, and A. Shamir. A method for obtaining digital signature and public-key cryptosystems. *Communication of the ACM*, 21(2), 1978.

[9] A. Ambainis. Upper bounds on multiparty communication complexity of shifts. In *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science*, volume 1046 of *Lecture Notes in Computer Science*, pages 631–642. Springer-Verlag, 1996.

[10] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Proceedings of the 24th International Colloquium on Automata Languages and Programming*, volume 1256 of *Lecture Notes in Computer Science*. Springer, 1997.

[11] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 421–429. ACM Press, 1985.

[12] L. Babai, P. G. Kimmel, and S. V. Lokam. Simultaneous messages vs. communication. In *Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 361–372. Springer-Verlag, 1995.

[13] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols and logspace-hard pseudorandom sequences (extended abstract). In *Proceedings of*

the 21st Annual ACM Symposium on Theory of Computing, pages 1–11. ACM Press, 1989.

[14] A. Back. Personal communication.

[15] P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.

[16] D. Beaver. Commodity-based cryptography (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 446–455. ACM Press, 1997.

[17] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the 7th Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer-Verlag, 1990.

[18] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead (extended abstract). In *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 62–76. Springer-Verlag, 1991.

[19] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *Proceedings of the 31st ACM Symposium on Theory of Computing*. ACM Press, 1999.

[20] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In *Advances in Cryptology—Crypto '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 55–73. Springer-Verlag, 2000.

[21] S. Brands. Phd thesis : Rethinking public key infrastructures and digital certificates — building in privacy, September 1999. Second edition will be published by MIT press.

[22] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, Oct. 1988.

[23] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 234–238. Springer-Verlag, 1986.

[24] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[25] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, Feb. 1981.

[26] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology—CRYPTO '82*, pages 199–203, New York, 1983. Plenum Press.

[27] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[28] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer-Verlag, 1990.

[29] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 304–313. ACM Press, 1997.

[30] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Report 98-03, Theory of Cryptography Library, 1998.

[31] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of the 36th IEEE Conference on the Foundations of Computer Science*, pages 41–50. IEEE Computer Society Press, 1995.

[32] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998.

[33] D. A. Cooper and K. P. Birman. Preserving privacy in a network of mobile computers. In *1995 IEEE Symposium on Research in Security and Privacy*, pages 26–38. IEEE Computer Society Press, 1995. http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR85-1490.

[34] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[35] R. Cramer. Introduction to secure computation. In *Lectures on data security : modern cryptology in theory and practice*, volume 1561 of *Lecture Notes in Computer Science*, pages 16–62. Springer-Verlag, 1999.

[36] G. Di-Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for database private information retrieval (extended abstract). In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing*. ACM Press, 1998.

[37] G. Di-Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrieval implies oblivious transfer. In *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138. Springer-Verlag, 2000.

[38] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[39] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[40] W. Ford and M. S. Baum. *Secure electronic commerce : building the infrastructure for digital signatures and encryption.* Prentice Hall PTR, 1997.

[41] Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. Proceeding of the second RANDOM:

International Workshop on Randomization and Approximation Techniques in Computer Science, 1998.

[42] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 151–160. ACM Press, 1998.

[43] I. Goldberg and A. Shostack. Freedom network whitepapers, http://www.freedom.net.

[44] O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 182–194. ACM Press, 1987.

[45] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

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

[47] S. Goldwasswer and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1994.

[48] V. Grolmusz. Separating the communication complexity of MOD m and MOD p circuits. *Journal of Computer and System Sciences*, 51(2):307–313, 1995.

[49] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 230–235. IEEE Computer Society Press, 1989.

[50] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 44–61. ACM Press, 1989.

[51] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the 5th Israel Symposium on the Theory of Computing and Systems*, 1997.

[52] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *Proceed-*

ings of the 31st ACM Symposium on Theory of Computing. ACM Press, 1999.

[53] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 539–550. ACM Press, 1988.

[54] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single server private information retrieval. In *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 104–121. Springer-Verlag, 2000.

[55] E. Kushilevitz and R.Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 364–373. IEEE Computer Society Press, 1997.

[56] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.

[57] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation (extended abstract). In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 245–254. ACM Press, 1999.

[58] R. Ostrovsky. Efficient computation on oblivious RAMs. In *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, pages 514–523. ACM Press, 1990.

[59] R. Ostrovsky and V. Shoup. Private information storage (extended abstract). In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 294–303. ACM Press, 1997.

[60] P. Pudlák and V. Rödl. Modified ranks of tensors and the size of circuits. In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 523–531. ACM Press, 1993.

[61] P. Pudlák, V. Rödl, and J. Sgall. Boolean circuits, tensor ranks, and communication complexity. *SIAM Journal on Computing*, 26(3):605–633, 1997.

[62] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[63] M. O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[64] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[65] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38–44, Feb. 1983.

[66] C. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:656–715, 1949.

[67] D. R. Stinson. *Cryptography: theory and practice*. CRC Press, 1995.

[68] M. Szegedy. Functions with bounded symmetric communication complexity and circuits with mod m gates. In *Proceedings of the 22nd ACM Symposium on Theory of Computing*, pages 278–286. ACM Press, 1990.

[69] I. Wegener. The complexity of boolean functions. Teubner, 1987.