Private Data Base Access Schemes
Avoiding Data Distribution

by

Yael Gertner

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Masters of Engineering

and

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1997

Author .....
Department of Electrical Engineering and Computer Science
May 19, 1997

Certified by .....
Shafi Goldwasser
RSA Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Private Data Base Access Schemes

Avoiding Data Distribution

by

Yael Gertner

Submitted to the Department of Electrical Engineering and Computer Science

on May 19, 1997, in partial fulfillment of the

requirements for the degrees of

Masters of Engineering

and

Bachelor of Science in Computer Science and Engineering

Abstract

In this thesis we introduce and solve three privacy problems in Secure Database Access protocols: Database Privacy, the Data Replication Problem, and the Secure user Identity problem. Database Privacy is concerned with keeping the databases information secure from the user. The Data Replication problem (DRP) deals with a new security concern for databases that emanates from the need to replicate and distribute their contents in order to achieve security for the user. The Secure user ID problem is concerned with keeping private the user's identity, so that no information can be associated with or learned about that identity.

Our results rely on an existing Private Information Retrieval scheme which achieves privacy for the user's query by relying on the multiple database model. This model allows for information theoretic results and sublinear communication complexity in the size of the database. We present two schemes which solve, in addition to what was achieved previously, Database Privacy and the Data Replication problem.

We achieve two different degrees of security for DRP. The first one is *private-data-distribution* which means that all the databases in the scheme are k-wise independent for some constant k. The second is *no-data-distribution* security which means that the database's in the scheme contain data that is completely independent. The user's security in our scheme relies on the Private Information Retrieval scheme introduced in [14] which guaranties that the message the user sends to a database is uniformly distributed over all possible queries.

We show two reductions:

Theorem: For any $k \geq 2$ given any Private Information Retrieval k-database scheme for n data bits with communication complexity $R(k, n)$ there exists a *private-data-distribution* and *database private* $2k$-database scheme with communication com-

plexity $O(R(k,n)\log(n))$ where each database holds $O(n)$ bits.

Theorem: For any $k \geq 2$ given any retrieval k-database scheme for n data bits with communication complexity $R(k,n)$ there exists a *no-data-distribution* and *database private* $2k$-database scheme with communication complexity $O(R(k,n)\log(n))$ where each database holds $O(n)$ bits.

Secure ID

In addition, we solve the Secure ID problem by presenting a protocol for a network of a user U, n databases of size m with an additional server S. A database in the network does not know whether U asked him a query or asked a query from another database. Therefore, we say that he does not know the identity of the users that are querying him. The communication complexity of that scheme is $O(\log(n)R(n,k1) + \log(m)R(m,k2))$ for constants $k1$ and $k2$.

Thesis Supervisor: Shafi Goldwasser

Title: RSA Professor

Acknowledgments

I wish to express my immense gratitude to my advisor, Prof. Shafi Goldwasser, for her invaluable input, guidance, and for her helpful suggestions, comments, and pointers throughout the various stages of this work.

I would like to thank Prof. Abelson, Dr. Stephen Adams, Prof. Sussman, and Prof. Yip who were my UROP advisors and who formed an integral part of my education.

I would like to thanks Tal Malkin with whom joint work lead to the current research and for being a great person to work with.

Contents

Chapter 1

Introduction

In this thesis we introduce and solve three privacy problems in secure database access protocols: *Database Privacy*, the *Data Replication Problem* (*DRP*), and the *Secure User Identity* problem. Database Privacy is concerned with keeping the databases information secure from the user. The Data Replication problem deals with a new security concern for databases that emanates from the need to replicate and distribute their contents in order to achieve security for the user. The Secure User ID problem is concerned with keeping private the user's identity, so that no information can be associated with or learned about that identity. But before we are able to discuss these problems further, let us give some background and motivation as to what brings forth these database security problems.

Since the propagation of the World Wide Web users grew to expect almost all types of data to be available on the internet. In many cases, information that cannot be found on the web is considered non existent. Therefore, even databases that contain sensitive information find it necessary to make their data available on the web. Access to this sensitive information introduces a privacy risk not only to the database, but to the user as well, because the database can electronically record the user's identity and the type and contents of the user's query. This is not the case for data retrieved by users from books in libraries, for example, because there is no electronic trace or record of the page the user accessed, or of the user's identity. Therefore, this new internet scenario brings forth the problem of Secure Data Access,

which is concerned with keeping private the users's query, the user's identity, and the database's information other than the value of a particular query that is offered to the user.

Previously, work was done on protecting databases from the access of malicious users who destroy contents [21, 2]. This work, however, was not concerned with the user's security at all. Recently, a solution to protect only the user's privacy was proposed and called Private Information Retrieval [14]. Private Information Retrieval protects the user from the database by giving the database no knowledge about the type or value of the user's query. Those results are achieved using a multiple database model, in which a user obtains the value of his query by communicating with a few copies of the database as opposed to only one. However, the databases holding copies of the data, are not allowed to communicate amongst themselves, in order to keep the user's query private.

We would like to offer some motivation for why the Databases Security and Private Information Retrieval must be satisfied simultaneously. We observe that security for both the user and the database is crucial for the emerging applications of today's world. For example, consider an investor who decides on a stock based on information he receives from a database containing stock information. In this scenario, it is likely, that the user wishes to keep his choice of stock, or query, secret while the database itself would like to keep the stock information private to itself, except for the particular stock that the user has paid for. Therefore, the security of both should not be compromised.

The first goal of this thesis was to simultaneously satisfy these two problems in the multiple database model. Let us examine the plausibility of the multiple database model. In general it contains many databases that are not communicating with each other, and only the user talks to each one separately. In the Private Information Retrieval scheme [14] the contents of all the databases are identical and are created by replicating the original database. Later, the auxiliary databases holding the distributed copies are assumed not to communicate with the original database. This assumption seems implausible to us because according to it the database, in a sense,

gives its information to parties which can use it to create their own database black market service and sell the data independently. There is little that can be done by the database to prevent this from happening since those other parties must be separate entities with which it is not allowed to communicate. If the database cannot be secure about its data then it is not likely to agree to participate in a Private Information Retrieval protocol and then the user is at risk. Thus we are faced with a new problem: how to prevent the databases from being copied to non communicating entities in the multiple database model which is essential for achieving user privacy. We call it the "Data Replication Problem".

In this thesis we present schemes that solve the "Data Replication Problem" and "Database Privacy" while maintaining all the other properties accomplished by a Private Information Retrieval scheme. The novel idea in those schemes is to construct auxiliary databases that contain data which is independent of the original data yet gives the user the appropriate value that will help construct his query. In other words, we propose a protocol for communication between a user, a database and auxiliary databases whose contents are not dependent on the original database. This protocol allows the user to keep his query secret from the databases, and allows the database to keep its data secret from the auxiliary databases and from the user (except for the value of the particular query). Then we will show how to extend that scheme into a protocol for a network of databases that solves the Secure ID problem. Those protocols rely on no cryptographic assumptions and maintain sublinear communication complexity in the size of the database, where by communication complexity we mean the total bits sents between all the parties per query.

We note that the Database Privacy concern was addressed previously in [7], but this work did not allow for a constant number of databases, and one round of computation. No work was previously done towards the Data Replication Problem. All information theoretic multi database schemes never required nor achieved that the databases will be independent. The general multi party computation protocols can be used toward solving the problem with auxiliary databases with independent contents, but then the communication complexity is very high. In addition, those schemes re-

quire many rounds of computation, and a non constant number of auxiliary databases. Our scheme is the first that is explicitly concerned with the Data Replication problem and that solves it with information theoretic security, sublinear communications complexity, a constant number of auxiliary databases and 1 round of communication.

The third problem is the Secure User ID problem which is concerned with keeping private the user's identity, so that no information can be associated with or learned about that identity. Not much work has been done in order to solve the Secure ID problem. There is, however, a sight in Carnegie Mellon University called anonymizer which allows one to "surf the web without revealing any personal information". It is currently restricted to Carnegie Mellon Computer Science sites. The URL for this page is Http://anonymizer.cs.cmu.edu:8080/ maintained by Justin Boyan (jab@cs.cmu.edu). Still, the privacy of the user on this site is conditional on the fact that the user trusts this particular anonymizer server since this server itself can obtain information about the user when the user surfs through it.

1.1 Our Results:

We present two schemes which solve Database Privacy, the Data Replication problem, while also achieving User Privacy, no cryptographic assumptions, a constant number of databases, and sublinear communication complexity.

More specifically, we present two schemes which solve the Data Replication Problem (DRP) achieving two different degrees of security. The first one achieves *private-data-distribution* which means that all the databases in the scheme are k-wise independent for some constant k. The second scheme achieves *no-data-distribution* security which means that the databases in the scheme contain data that is completely independent. Both achieve *Database Privacy* as well which means that after the user interacts with all the databases, he does not get any information about the database, except for its value at a single location. The user's security in our schemes relies on the Private Information Retrieval scheme introduced in [14] which guaranties that the message the user sends to a database is uniformly distributed over all possible

queries.

We show two reductions:

Theorem: For any $k \geq 2$ given any Private Information Retrieval k-database scheme for n data bits with communication complexity $R(k,n)$ there exists a *private-data-distribution* and *database private* $2k$-database scheme with communication complexity $O(R(k,n)\log(n))$ where each database holds $O(n)$ bits.

Theorem: For any $k \geq 2$ given any retrieval k-database scheme for n data bits with communication complexity $R(k,n)$ there exists a *no-data-distribution* and *database private* $2k$-database scheme with communication complexity $O(R(k,n)\log(n))$ where each database holds $O(n)$ bits.

Remarks:

In all the schemes there is a trade off between increasing the number of databases and the number of faulty databases who in a coalition could break security.

If we use the [3] scheme, then $R(n,k) = n^{1/2k-1}$.

Secure ID

In addition, we solve the Secure ID problem by presenting a protocol for a network which consists of a user U, n databases of size m with an additional server S. A database in the network does not know whether U asked him a query or asked a query from another database. Therefore, we say that he does not know the identity of the users that are querying him. The communication complexity of that scheme is $O(\log(n)R(n,k1) + \log(m)R(m,k2))$ for constants $k1$ and $k2$.

1.2 Outline of Thesis.

Since our results for the Secure ID problem largely rely on the results of the Data Replication Problem, we begin by describing previous work related to the Data Replication problem in chapter 2. Then, chapter 3 will describe the security notions, assumptions, and protocols used in our schemes. In chapters 4 and 5, we describe two schemes that together achieve Database Privacy and Private Data Distribution. In chapter 6, we describe a scheme that satisfies an even stronger security requirement,

No Data Distribution, and protect the database from distributing its data. Then, we discuss the similarity between our problem and the Oblivious Transfer problem, in chapter 7. In chapter 8, we present a scheme which solves the Secure ID problem. Finally we offer some suggestions for future work in chapter 9.

Chapter 2

Previous Work

In this chapter we concentrate on the database's security, meaning Database Privacy and the Data Replication Problem, since our solution to it is a subprotocol of the Secure ID solution. In addition, there is not much work done on the Secure ID problem other than the site we mentioned in the introduction.

The new problem of Data Replication (DRP) originates in a multiple database model and is concerned with the distribution of original data to non communicating parties. If this problem arises in this model than why use this model at all, or what are the properties of this model that make it so desirable? In order to explain that, let us first state the properties we would like our model to satisfy. Having this as a goal in mind, we will go through various models explored in previous work and find the best match with our properties. One of the models that we examine is the multiple database model with databases of feasible size introduced in the PIR scheme [14] which satisfies all of our desired properties except for Database Privacy and DRP. In the rest of the thesis we will present various schemes that solve Database Privacy and DRP in this model.

2.1 Our Goals

The main goal of our work is to provide privacy for the user while not compromising the database's security. This means that the user is the only one who knows the value

of his query, and that the database is the only one who knows the data. In order to make the solution stronger we require no cryptographic assumptions. In addition, we insist on low communication complexity and a constant number of databases otherwise the solution would be too costly and make no sense in the context of our application.

More formally our desired properties can be described as follows:

- Security for the Database from the auxiliary databases - For databases $D_1, D_2, ..., D_k$. D_1 contains the original database, and the rest of the databases D_i for $i \neq 1$ contain data whose distribution is independent of D. Therefore, a single database having only the view of D_i, without communicating with other databases, cannot get any information about D.

- Security for the Database from the user - After the user interacts with all the databases, $D_1, ..., D_k$, the user does not get any information about D, except for its value at a single location (presumably b_q). That is, the view that the user gets is independent of the values in D in indices $i \neq q$.

- Security for the user - Private Information Retrieval - No database communicating with the user can get any information about the user's query q, or the value at q. That is, all the communication between the user and the particular database is distributed uniformly, for all indices.

- No computational assumptions - Information Theoretic Security - No matter how much computational power a party has it cannot gain any knowledge from the information that it has.

- Communication complexity: The total number of bits exchanged between the user and all the databases per query is sublinear in the size of the database.

- All of the above properties can be achieved with a constant number of databases. k is a constant.

2.2 Overview Of Previous Work

Having stated our goals, we proceed by giving an overview of the different models used in previous work. The models are illustrated through the schemes that use them, where the schemes are interactive protocols between a database (or a party with information) and a user (a party that wants to get a piece of information). We examine the achievement of each protocol in light of our goals and then relate these to the properties of the model they use.

2.2.1 The One Database Model - Oblivious Transfer

We start with the simplest and most intuitive model - the one database model - which consists of a database and a user, because it immediately eliminates the Data Replication problem. This model is used in the Oblivious Transfer protocol [11, 12, 20, 15, 8, ?] where the database is a party called Alice who has the secret S and the user is a party called Bob who wants to obtain the secret. After the protocol Bob either receives the bit S with probability 1/2 or Bob gains no further information about the value of S. Alice does not know which of the events occurred. This basic Oblivious Transfer was introduced by Rabin for the purpose of exchanging secrets and was later extended to the $\binom{2}{1}$ OT in which Alice has two secrets S_1 and S_0, while Bob has a bit b. At the end of the protocol Bob has S_b and knows nothing about S_{1-b}, and Alice does not know b. Furthermore, OT was extended to an even more general form in which Alice has n secrets $S_1, ...S_n$ and bob has an index i in n. At the end of the protocol Bob has S_i and nothing more than that, while Alice does not know anything about i. This achieves security for both Alice and Bob. The extended version of the Oblivious Transfer protocol found other applications such as a subprotocol for a multi party computations which only used n as a constant of at most four. Therefore, there was no need to worry about the high communication complexity of the protocol $O(n * \text{securityparameter})$.

However in our application for the Oblivious Transfer protocol with the size of the database n being polynomial, communication complexity of order n is too large

to be reasonable. So the Oblivious Transfer protocol does not match our goals in that respect. Actually the Oblivious Transfer protocol does not match our goals in another respect as well. The OT protocol also relies on computational assumptions, whereas we are interested in information theoretic security. This property is not only a property of the protocol but it is inherent in the one database model because both parties have the whole transcripts of the conversation so that nothing is hidden from them. Therefore we will have to examine other models in order to satisfy all of our goals. But before that, let us summarize the properties of the one database model.

Model - One database and one user. The properties of the protocols that were implemented using this model so far achieve the following properties.

- User privacy.
- Database Privacy and No Data Distribution.
- Requires only 1 Database.

Note that it does not achieve:

- Information theoretic security because it relies on the existence of Trapdoor functions.
- Reasonable communication complexity: The communication complexity is $O(n* k)$ where k is a security parameter.

Non Interactive OT

An interesting variation of the OT protocol which tries to reduce the cost of communication is the Non Interactive OT [8] which introduces a variation of the one database model. This protocol reduces the cost of Oblivious Transfer not by lowering the communication complexity but by eliminating the interactive step. The traditional OT requires the following interactive step: first the user sends n trap door functions to the database knowing the trap door information to only one of them, then the database applies the function to his secrets and sends the results to the user. The

non interactive OT modifies the one databases model to include a public file as well. Using this public file the interactive step is removed by having the user choose his trapdoor functions ahead of time and place them in the public file. Having access to the public file, the database could send messages to the user at a convenient time for him. The keys placed in a public file can be used for many or different applications. This application is also useful for fractional OT where a user obtains only a fraction of the messages sent to him during the protocol. However, the Non Interactive OT still makes cryptographic assumptions which we would like to avoid.

2.2.2 Software Protection - Oblivious RAM

Another model we consider is the model used in the Oblivious RAM protocol which is proposed in [17] for the software protection problem. This problem is not concerned with the copying of software, rather it is concerned with hiding the software program's access pattern to the memory. Meaning that the access should be uniformly distributed over all locations in memory. This problem cannot be solved by encryption alone because it does not hide everything, for example accesses to the same location. The Oblivious RAM protocol solves this software protection problem and produces interaction between the CPU and the memory such that the software holder who views this interaction has no information about the software. Meaning that the interaction, or access pattern, seems independent of the software.

The model of this program consists of two parties the CPU and the software holder. (Note that the software holder can also be called a user or one who buys the software, but in this case we choose not to call him a user because in this thesis we limit the term user only to those who access databases in order to get some information and here the software holder holds the software or information and does not query it and thus he is not a user in our sense). The CPU holds the input to the software according to which the program will run. The CPU is protected by hardware therefore its size is limited and it cannot hold the contents of the software's memory. Therefore, the memory must be first encrypted and permuted by the CPU and then stored by the Software Holder. Thus, the software holder knows about all the accesses

to the memory. The software holder in a sense stores the CPU as well but since the CPU is protected by hardware, the software holder does not have any information about it. The CPU is the only one who has all the permutation and decryption keys to the memory.

Having set the model, let us describe in more detail what was achieved in the Oblivious RAM protocol and how it pertains to our application. When a memory is encrypted and permuted, it does not reveal which specific address is accessed or what was the contents of it. However, it does reveal information about the access pattern such as which address is accessed more than once and in what order that occurs. Through the Oblivious RAM protocol the CPU can communicate with the memory such that the software holder does not gain this information. In fact, the interaction is such that it is indistinguishable from an interaction of another software. This is achieved using cryptographic assumptions and amortized polylogarithmic communication complexity in the size of the program.

We do not describe how the protocol is implemented here. Rather we concentrate on examining the parallels between this model and the database model and using ideas from the Oblivious RAM protocol for solutions in our schemes.

Let us then describe the similarities between the models. In the database model there is a database with n bits of information and a user who wishes to retrieve some of that information without giving away its query. In the software protection model there is a software holder who has encrypted and permuted memory and a CPU who wants to access that memory without revealing its access pattern. Those two problems are the same, except that the software holder does not know the real contents and addresses of the memory and the Database does have the real data and thus also knows the values and real locations of each query (not only the repeated ones). In our scheme if we find a way to encrypt and permute the database without giving the database the encryption and permutation keys while not compromising the database's security, we can use the Oblivious RAM method of access to achieve privacy for the user. We will describe this in more detail when we describe the actual scheme in Chapter 6.

2.2.3 Multi Prover Scheme

All the models we examined so far were variations of the one database models which achieved their security by relying on cryptographic assumptions. Those results are actually inherent in the one database model because it is impossible to obtain an information theoretic two party protocol that does not disclose the secrets of the other party because the two parties have exactly the whole transcript of their conversation [18]. Since we want to achieve information theoretic results we proceed by examining models that involve more than one database. This way, one database alone does not have the whole transcript of the conversation, and therefore, cannot compute all the information by itself. This means that the communication between the databases must be limited, otherwise a database could obtain the whole transcript. Adding a database does increase the cost of the protocol in terms of space but it is a worthwhile price to pay for information theoretic security, especially since the distribution of data is inherent in the current systems setting of the internet.

The first scheme to introduce and use the idea of distributing one of the protocol's parties into two separate ones in order to achieve information theoretic results was the Multi Prover Scheme [9]. The motivation for this scheme comes from an interactive proof system which consists of an all powerful prover who attempts to convince a probablistic polynomial time bounded verifier of the truth of a proposition. The prover and the verifier receive a common input and can exchange upto a polynomial number of messages, at the end of which the verifier either accepts or rejects the input. In a zero-knowledge interactive proof the prover is trying to convey a proof to the verifier without giving him any more knowledge. Such computationally zero-knowledge interactive proof system were shown to exist by [16] for every NP language if non-uniform one way functions exist. In the one prover model it was not possible to achieve information theoretic results as we explain before. However, in the multi prover model in which two provers were used instead of one it was shown that a perfect zero knowledge interactive proof system exists for every NP language making no intractability assumptions.

Let us describe the multi prover model in more detail. In their model there is still

one verifier. The two provers can decide on an optimal strategy before interacting with the verifier. But once the interaction starts they can no longer send each other messages. In addition, the two provers share either a polynomially long random pad or a function which they can compute but the polynomially bounded verifier cannot. One of the Provers' function is to give the proof. The other Prover's function is to periodicly output segments of the random pad he shares with the other Prover.

Inspired by this model we proceed to examine distributed database models which can achieve information theoretic results. We cannot use the Multi Prover scheme directly because its provers specialize in proving propositions and in our case we are interested in a party that holds data and gives pieces of that data to a user upon request. Another difference between the schemes is that they show that adding more provers does not add power to the multi prover model. In our case though, we show that if we have more databases we can reduce the communication complexity further.

2.2.4 Multi Party Computation

Since the multi prover model is specificly designed for proof systems we must examine a more general model designed for computing all functions, introduced in the Multi Party Computation protocol. This model consists on n parties that are equally powerful, yet they must cooperate in a computation because none of them alone have all of the necessary information. Each party does not want to disclose its information; therefore some security must be guarantied. Again, using the fact that the parties are distributed and not communication outside of the protocols specification it is possible to obtain a multi party computations for any function with information theoretic security [10].

The model of computation is a complete synchronous network of n processors. The pairwise communication channels between players are secure. In one round of computation each of the players can do an arbitrary amount of local computation, send a message to each of the players, and read all messages that were sent to it. For simplicity, they restrict themselves to the computation probablistic functions f from n inputs to n outputs. The player i holds the i-th input at the start of computation,

and should obtain the i-th output at the end, but nothing else.

What happens when the parties do not follow the protocol and they communicate outside of the permitted messages? (This was not the problem for the multi prover scheme because there the parties do not communicate at all so it is easier to verify that they are not communicating forbidden information). In this work, they allow for two types of faulty parties. A party can either try to learn more by sharing information that it receive through extra communication additional to the specified protocol, or by using a completely different protocols altogether. It is shown in [10] that for every probabilistic function and for every $t < n/3$ there exists a protocol that allows for no more than t faulty parties of each kind that was just described. They achieve those results by using constructions based on Algebraic Coding Theory, particularly the use of general BCH codes.

This result is general for any probablistic function and for many parties that are all interested in finding some value of the function. In our application we are dealing with a special case where the function is a database and the user is the only party who gets the value of the computation. Since their result is general they are not able to achieve the number of databases, communication complexity, and rounds of computation which we desire.

2.2.5 Multi Oracle Model - Instance Hiding

A more specific scheme for multi party computation is the Instance Hiding scheme [1, 4, 5]. This scheme assumes the following model: for some function f (could be exponential) and n parties, $n-1$ of the parties (thought of as oracles) contain the same data of how to evaluate f, while the n'th party (user) on private input x wishes to compute $f(x)$ but is unable to do so without the help of the $n-1$ parties. The Oracles are not allowed to communicate with one another, but only with the user.

Based on this model, Instance Hiding gives an interactive protocol involving a computationally limited user (verifier) U and $m > 1$ powerful Oracles $D_1, ..., D_m$. For a given exponential function f accessible only to the oracles, and input $i \in \{1, ..., n\}$, accessible only to the user, outputs x_i to the user only, while requiring only polynomial

communication complexity.

The protocol allows U to obtain the value of $f(i)$ without revealing to any database any information about i, thus achieving our security demands for the user. One attempt to provide security for the database using the Instance Hiding scheme is show in [7] using a proof system. In that work they use a proof system which gives the user zero knowledge about f with only a small increase in cost. This achieves Data Privacy. However, the data replication problem is not solved because all the databases know the same information, f. In addition, another one of our goals - a constant number of databases - is not solved because this scheme requires the number of databases to be logarithmic in the size of the function in order for the communication complexity to be low.

Model: There are $k = O(\log(n))$ databases $D_1, ..., D_k$ where n is exponential and a user U. D therefore is an oracle which helps U the user compute a value. The D's are not allowed to communicate. The properties of the protocols that were implemented using this model so far achieve:

- User privacy.
- Privacy is information theoretic.
- Communication Complexity is $O(\log(n))$.

Note that it does not solve:

- Data Privacy.
- The Data Replication problem.
- Constant number of databases. The number is $O(\log(n))$

This number of databases result is appropriate for the general goal of the Instance Hiding protocol, which is designed for a user (Verifier) which has the data necessary (i) and needs to communicate with other processors (the computationally unbounded databases) because it lacks the *power* to carry out the computation on its own. Thus the Instance Hiding scheme is also beneficial for users who want the value of an

exponential function and only have polynomial power. But our specific application of database access does not benefit from that case because our f is polynomial and can be described as $(f(i, x1, ..., xn) = i)$. Therefore, we can look at a more specific model which is not general for all functions but only applies to the database function.

2.2.6 Multi Database Model - Private Information Retrieval

A more specific variation on the multiple oracle model of Instance Hiding is the model of multiple databases of feasible size, introduced in the Private Information Retrieval protocol [14]. The PIR scheme is a protocol for communication between a user and multiple databases of feasible size which are not allowed to communicate between themselves but with which the user communicates in order to obtain his query. Based on this model the PIR protocol achieves information theoretic security for the user, sublinear communication complexity, and a constant number of databases.

In this protocol they were able to reduce the number of databases to a constant by changing the Instance Hiding model to include databases of a feasible size instead of exponential size. Thus, the Private Information Retrieval protocol is not constructed for the purpose of helping a user compute a value of any function with the help of all powerful databases, but it is constructed for querying a feasible database for the value of a special purpose type of function $(f(i, x1, ..., xn) = i)$ in a secure manner and low cost.

The Private Information Retrieval scheme copies database to other entities and forbids their communication. Therefore, we are still left with the Data Replication problem, and Data Privacy.

Model: $k \geq 2$ copies of the Database which is of a *feasible* size, and a user. The databases are not allowed to communication with one another and are not allowed to view the interaction of the user with the other databases either. The properties of the protocols that were implemented using this model so far achieve the following properties.

- User Privacy.

- Constant number of databases.
- Privacy is information theoretic.
- Communication Complexity - $O(n^{1/(1+k)})$ for $k = 2$ and $O(n^{1/k})$ for $k > 2$.

Note that it does not solve the Data Replication problem, and does not achieve Database Privacy.

In the rest of the thesis we will show how to solve the rest of our goals and use PIR as a subprotocol in order to achieve its other properties as well.

2.3 Summary - The Model We Choose

Having examined different possible models and schemes we note that the model of the Private Information Retrieval scheme achieves goals most similar to ours. In other words, a model involving a user and feasible databases which are not allowed to communicate allows for protocols which achieve all of our goals except for the Data Replication problem and Data Privacy. Therefore, in this thesis we will present schemes that solve these problems in that model. Our schemes are a reduction to the Private Information Retrieval scheme. We use PIR as a subprotocol of our schemes. In some cases we also rely on results from Oblivious RAM.

2.4 Related Work

In this section we describe work that is related to our work, but was achieved independently from our results.

Private Information Storage

Ostrovsky and Shoup [19] have extended the results of [14] and designed schemes for private information storage. Using their schemes, the user can both read and write to the database without revealing which bit is accessed. They have shown that any protocol for private information retrieval can be transformed to the protocol for

private information storage with a slight increase in the number of databases and communication.

Private Information for the user [3]:

In this work the results of the PIR scheme introduced in [14] were improved by achieving communication complexity of $O(n^{1/(2k-1)})$. This was achieved through constructing a new scheme from two PIR schemes one for $k = 2$ databases and another for $k > 2$ databases such that the $k > 2$ scheme is a subprotocol for the $k = 2$ scheme.

Computational Private Information Retrieval [13]:

In this work they show how to achieve PIR based on computational assumptions such that the communication complexity can be $O(n^e)$ for any $e > 0$.

Chapter 3

Preliminaries and Notation

3.1 Model

Multi Database Model

The model of computation of all the schemes consists of original data D which contains n bits, $D = \{b_1, \ldots, b_n\}$, were n is a feasible parameter, k multiple databases $D_1, ..., D_k$, and a user U. U has a query $q \in \{1, ..., n\}$. $D_1, ..., D_k$ consist of different values depending on the scheme, they can either consist of D, or some random data. The user, U, interacts with $D_1, ..., D_k$ while the D's are not allowed to communicate.

Network Model

The model of the network consists of n different original databases of length m, $2n$ databases of random bits, $2n$ permutation databases, a server S, and a user. Each group of 1 original database 2 random databases and 2 permutations are assigned a location by the server. The databases and the server are not allowed to communicate.

3.2 Notions of Security

We are concerned with the following notions of privacy.

User Privacy: No database communicating with the user can get any information about the user's query q, or the value b_q. That is, all the communication between the user and the particular database is distributed uniformly, for all indices.

Database Privacy: After the user interacts with all the databases, $D_1, ..., D_k$, the user does not get any information about D, except for its value at a single location (presumably b_q). That is, the view that the user gets is independent of the values in D in indices $i \neq q$.

Private Data Distribution: Only D_1 has a copy of the original data D. The rest of the databases D_i for $i \neq 1$ contain data whose distribution is independent of D. Therefore, a single database having only the view of D_i, without communicating with other databases, cannot get any information about D.

No Data Distribution: Only D_1 has a copy of the original data D. The rest of the databases D_i for $i \neq 1$ can be determined ahead of time and contain some random data that is determined independently of D. This privacy is different than the Private Data Distribution above because in Private Data Distribution even though each database D_i on its own is independent of D, a coalition of databases can be dependent on D, whereas here any coalition that does not contain D_1 is independent of D.

Secure ID: No database in the network can get any information about which database the user is querying. That is, all the communication between the user and the databases is distributed uniformly for all databases.

3.3 Assumptions

There are two types of assumptions we make about the databases. One is concerned with their faultiness. The second is concerned with the amount of communication we allow them to have with each other.

Faultiness of the Databases:

Faultiness only pertains to whether the database follows the protocol or not, and it is not concerned with extra messages that the databases might send.

Honest but curious: The databases and the user are following the specification of the protocol exactly but try to extract as much knowledge as they can from the information that they receive.

Malicious: The databases do not follow the protocol and send some other message

instead of the one that they were supposed to.

Communication:

The databases are not allowed to communicate with one another, where by communication we mean two different things: No messages, and No communication.

No messages: The databases are not allowed to send extra messages outside of the specification of the protocol to another databases. This assumption can be relaxed by increasing the number of databases and then allowing certain databases to communicate as long as a coalitions of a certain size will not form.

No communication: The databases are not allowed to talk to another database at all by any means of communication, not only through extra messages but also through pretending to be a user and following the user's protocol.

3.4 Protocols

We describe here two schemes, Oblivious Transfer [11], [15], [8] which we will show to be equivalent to our problem in chapter 7, and the Private Information Retrieval scheme [14] scheme, which we use in our schemes as a subprotocol.

3.4.1 Oblivious Transfer

$\binom{n}{1}$ **Oblivious Transfer:** In this protocol, Alice has n secret bits $S_1, \ldots, S_n$, and Bob has a selection index $i \in \{1, \ldots, n\}$. At the end of the protocol, the following three conditions hold.

1. Bob learns the i'th secret S_i.

2. Bob gains no further information about the other secrets S_j for $j \neq i$.

3. Alice learns nothing about the value of i.

A general $\binom{n}{1}$ oblivious transfer protocol based on the existence of one way functions is described in Goldreich's notes [15].

3.4.2 Private Information Retrieval

The PIR scheme is an interactive protocol between a user and multiple databases of feasible size which are not allowed to communicate between themselves but with which the user communicates in order to obtain his query. The databases obtain no information about the user's query, because the messages the user sends them are uniformly distributed over all queries.

In [14] a few schemes are described.

- A scheme for $k = 2$ databases with communication complexity of $O(n^{1/3})$.
- A scheme for k (constant) databases with communication complexity of $O(n^{1/k})$.
- A scheme for $k = O(\log(n))$ databases with communication complexity of $O(\log^2(n) \log\log(n))$.

In [3] a scheme is described that achieves communication complexity of $O(n^{1/(2k-1)})$ for k databases.

For the remainder of the thesis we will denote the communication complexity of the Private Information Retrieval scheme as $CCPIR(n, k)$.

Here we describe the simplest scheme of [14] $k = 2$ databases, which suffices for explaining the technique used to keep the user's request secret. To start with, this simpler version is not better than the trivial $O(n)$ solution (of sending the whole database to the user), but after small modifications using error correction codes it achieves communication complexity of $O(n^{\frac{1}{3}})$.

In this simple example there are two databases D_1, D_2 and a user U.

At setup time:

- The database $D = (b_1, ..., b_n)$ is duplicated into two identical copies D_1, and D_2 both contain the data values $b_1, ..., b_n$.

On Line: At this stage D_1 and D_2 are not allowed to communicate with one another.

- The user U is interested in the value of the query $q \in \{1, ..., n\}$ an index of the database.

- U chooses at random a subset S of indices $j \in \{1, .., n\}$. U then sends the subset S to $D1$ and $S' = S \oplus q$ to $D2$, where $S \oplus a = \begin{cases} S \cup \{a\} & \text{if } a \notin S \\ S \setminus \{a\} & \text{if } a \in S \end{cases}$
- D_1 sends to U the XOR of the values at the indices in S, meaning $\bigoplus_{j \in S} b_j$
- D_2 sends to U the XOR of the values at the indices in S'
- U XORs both of the values he has received from D_1 and D_2 and produces b_q, the qth value.

Neither D_1 nor D_2 can obtain any information about q from S, a uniformly distributed subset of n over all choices of indices.

Note that the databases are not allowed to talk to one another, otherwise they could find q trivially.

Again the scheme we just described with only two databases is $O(n)$ (the size of the subset S) communication complexity. However, as it is shown in [14] it is possible to restrict the subsets S that U chooses and to treat D_1 and D_2 as if they where 8 databases, thus reducing the complexity of communication to $O(n^{1/3})$.

Chapter 4

Random Pointer Scheme - Protecting the Privacy of the Database

In this chapter we present the Random Pointer scheme which guaranties Data Privacy. The security of the Random Pointer scheme is information theoretic and the communication complexity is $O(\log(n) * t * CCPIR(n,k))$ where $CCPIR(n,k)$ is the communication complexity of the PIR scheme with k databases of size n.

Before we present the scheme which achieves Data Privacy, we explain how Data Privacy is violated in a PIR scheme in order to motivate the construction of our scheme. In a PIR scheme, only the user's privacy is guarantied. Therefore, the user sends the database information that is related to its query but is uniformly distributed over all queries. On the other hand, the database sends the user information that is directly related to its data which the user uses in order to compute his query. Thus, this direct information might supply the user with more knowledge than just about one query.

In our scheme, we avoid sending the user information that is directly related to the data, yet send him something which he could use to compute his query and this way prevent the user from gaining more information about the data. In order to do so, we use additional random auxiliary databases so that D does not give the user

direct information about its data, rather it gives the user information about pointers to the random database such that the value at the pointer is the same as D's data. Using the pointers, the user only knows the data of the pointers which he followed and accessed, and he is able to follow only one pointer by accessing the random database directly. Therefore, the user only obtains the value of one data bit and privacy for the database is achieved.

It remains to show how the user accesses the random database directly without revealing the value of his query to it. This is achieved by introducing two random databases instead of just one. The database then gives the user a pair of pointers to locations in the random databases $R1$ and $R2$, such that the xor of the values at those locations is the same as the value at the related index in the original database. When the user follows the pointer directly this time $R1$ and $R2$ can't tell the value of the user's query because they do not communicate and thus do not know the value of the other bit of the xor that the user accessed from the other database.

4.1 The Random Pointer Scheme

This scheme consists of a user U with query q, original data of n bits $D = b_1, ..., b_n$, k databases D_i for all $i \in \{1, ..., k\}$, and two random databases $R1$ and $R2$. R_1 and R_2 each consist of a random string with an equal number of zeros and ones. $D_1, \ldots, D_k$ contain the original data, D, and a copy of R_1 and R_2. The user interacts with the D_is and the R's and obtains b_q. At the end of this protocol, no D_i or R knows q, and U does not know more than b_q about D. The D_i are not allowed to communicate with each other and with $R1$ and $R2$ after the setup time.

4.1.1 Overview

We give here an informal overview of the scheme, which allows us to achieve Data Privacy. We start with the last stage of the protocol and go backwards. During the final stage of the protocol the user asks R_1 and R_2 for their values at indices j and l, $R_1(j)$ and $R_2(l)$, respectively (where j and l are pointers to R's contents that the

user obtained by communicating with the D_is). Using the values of those pointers the user can compute the value of his query

$$R_1(j) \oplus R_2(l) = b_q \tag{4.1}$$

The values of these pointers are chosen by D_i's in such a way that a pair of pointers only gives information about at most one data bit.

The rest of the interaction between the user and $D_1, \ldots, D_k$ serves the purpose of allowing the user to obtain an appropriate pair of indices (j, l) that satisfy (5.1), without revealing any information about his query q. This is done by running a PIR subprotocol in which $D_1, \ldots, D_k$ use n pairs of the form $(j_1, l_1), (j_2, l_2), \ldots, (j_n, l_n)$ for the n pieces of data, and q as the query of the user.

In order for the user to receive the correct value b_q in (5.1), the pairs used as data in the subprotocol must satisfy

$$R_1(j_r) \oplus R_2(l_r) = b_r \quad \forall r \in \{1, \ldots, n\} \tag{4.2}$$

These data pairs cannot be chosen deterministically, because (j_q, l_q) will be sent to R_1 and R_2 respectively in the clear by the user, so it should not reveal any information about his interest q. Thus, $D_1, \ldots, D_k$ need to share some randomness (in our case, they share a few random permutations on n bits).

We now turn to describing the scheme formally.

4.1.2 Setup Stage

In this stage the databases get their contents.

- R_1 consists of a random string, chosen uniformly from all strings of **n** bits, with equal number of 0's and 1's.
- R_2 consists of a random string, chosen uniformly from all strings of **2n** bits, with equal number of 0's and 1's.

- Every D_i have the bits $b_1, \ldots, b_n$, the contents of R_1, R_2, and three random permutations $\pi_1, \pi_2^0, \pi_2^1 : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$. (The subscripts indicate whether the permutation will be used to find a location in R_1 or R_2, and the superscripts indicate the value of the bit that should be found in that location).

We make another assumption here that the D's and R's are honest and do not agree on any other protocol, or do not give the R's their permutations.

4.1.3 On Line Stage

During the online stage the databases $D_1, \ldots, D_k$ are not allowed to communicate with each other and with R_1 and R_2 according to the no messages assumptions, and the R's are not allowed to communicate with each other and with the D's according to the no communication assumption. The user obtains his desired information b_q through communicating with all of them.

- Each D_i computes n pairs $(j_1, l_1), (j_2, l_2), \ldots, (j_n, l_n)$ from $\pi_1, \pi_2^0, \pi_2^1, \{b_1 \ldots, b_n\}$, and the content of R_1, R_2, as follows:
  - $j_r = \pi_1(r)$ for $r = 1, \ldots, n$, hence all the j's are chosen completely randomly.
  - l_r $(r = 1, \ldots, n)$ are chosen randomly so that the contents in the j locations and the l locations will xor to the data bits. To do that, start by letting $b = R_1(j_r) \oplus b_r$ and $m = \pi_2^b(r)$. Note that in order to satisfy (5.2) we need to choose l_r such that $R_2(l_r) = b$. Thus, we let $l_r =$ the index of the m'th b in R_2. That is, if $b = 0$ we choose l_r to be the index of the m'th 0 in R_2, and similarly for $b = 1$. (Note that R_2 has $2n$ bits, consisting of n 0's and n 1's. Thus, for any $b \in \{0, 1\}$ and $m \in \{1, \ldots, n\}$, l_r is well defined).
- $D_1, \ldots, D_k$ and U run the subprotocol PIR with $(j_1, l_1), (j_2, l_2), \ldots, (j_n, l_n)$ as the data of D_i's, and q as the selection index of the User. At the end of the subprotocol, the User has the pair $(j, l) = (j_q, l_q)$.

- The uses sends j to R_1, and l to R_2.
- R_1 sends the user the bit $R_1(j)$, and R_2 sends the user $R_2(l)$.
- The user computes the exclusive-or of these two values, yielding $b_q = R_1(j) \oplus R_2(l)$.

Note that we chose 2 random databases $R1$ and $R2$. This number of random databases can be increased in order to allow for the possibility that a coalition of adversary R can be formed without gaining any information about the user's query.

4.2 Proofs of the RP scheme

Claim 1 (Correctness) *If the underlying PIR scheme is correct then the RP scheme is correct.*

Proof: By reduction from the correctness of PIR, after running PIR with all the D_i, the User receives the pair $(j, l) = (j_q, l_q)$ corresponding to his selection index q. From the way l_q was constructed, it is a location in which R_2 has the bit $b = R_1(j) \oplus b_q$. Thus, $R_1(j) \oplus R_2(l) = b_q$ and the user receives the correct value b_q.

according to the way (j_q, l_q) is chosen

Claim 2 (User Privacy) *If PIR is user private, then the random pointer scheme is also user private.*

Proof: Since the user communicates with the D_i only through PIR, by reduction from PIR, none of the D_i's gets any information about the user's query from their communication with the user. The extra information the databases have on $R1$, $R2$, π_1, and π_2 was created before the user asks his query, and they give not information about the user's query either.

The only communication R_1 gets is the index $j = \pi_1(q)$ j is a uniformly distributed index in $\{1, \ldots, n\}$, independent of q. Thus, R_1 cannot get any information about q.

The only communication R_2 gets is the index l, which is the location of the m'th b-bit in R_2, where $b = R_1(j) \oplus b_q$, and $m = \pi_2^b(q)$. Since we showed above j is uniformly

distributed, and since R_1 has half 0's and half 1's, it follows that $R_1(j) \in_U \{0,1\}$, and therefore $b \in_U \{0,1\}$, independent of q. m is uniformly distributed in $\{1,\ldots,n\}$ by randomness as above. We showed that b and m are both distributed independent of q, in fact uniformly, and thus l is also uniformly distributed (in $\{1,\ldots,2n\}$), independent of q.

4.2.1 Privacy of Database

Theorem 2 (informal statement) *For any strategy the user has (possibly cheating), if all D_i and R's follow the protocol, the user cannot get any information about more than one bit of data b_q of his choice.*

To state the theorem formally and prove it, we define the view of the user (for any strategy), and prove that its distribution is independent of all but one bit of data.

Let U be any strategy for the recipient. U runs a PIR subprotocol with the D_i's and the data $(j_1,l_1),\ldots,(j_n,l_n)$, at the end of which he receives (j_q,l_q) and possibly additional information about these data bits which the subprotocol leaks. We assume a worst case in which U receives the full information about all the data bits, namely he gets $(j_1,l_1),(j_2,l_2),\ldots,(j_n,l_n)$, and we show that even in this worst case, U cannot obtain any information about the real bits $b_1,\ldots,b_n$ other than a single bit b_q of his choice.

Let $V(j,l) = [(j_1,l_1),\ldots,(j_n,l_n),R_1(j),R_2(l)]$, $V(j,l)$ is the view received by a U sends queries j,l to R_1,R_2 respectively. (This is the assumption mentioned above. In reality, the view of U can be derived from $V(j,l)$, but is possibly much smaller). Note that an honest U should set $j = j_q, l = l_q$, but we allow a possibly cheating U, who may choose arbitrary j,l.

Consider a partial view $V^- = [(j_1,l_1),\ldots,(j_n,l_n),R_1(j)]$ where the last answer (from R_2) is omitted. Let M be the domain of all possible partial views V^-. Thus, $|M| = 2n!\binom{2n}{n}$. We will prove that the partial view V^- is uniformly distributed over M, and from this we will be able to prove that the distribution of the complete view V depends only on one bit of data.

In what follows, the notation $X \sim U[M]$ means that the random variable X is

distributed uniformly over the domain M.

Theorem 1 *$\forall j, l$, the distribution of $V(j,l)$ may depend on at most one bit of data. More specifically, for any possible view $V(j_r, l_{r'}) \in M \times \{0,1\}$,*

$$Prob[V(j_r, l_{r'})] = \begin{cases} \frac{\epsilon}{|M|} & \text{if } R_1(j_r) \oplus R_2(l_{r'}) = b_{r'} \\ \frac{1-\epsilon}{|M|} & \text{otherwise} \end{cases}$$

where $\epsilon = 1$ if $r = r'$, and $\epsilon = \frac{1}{2} - \frac{1}{2(n-1)}$ if $r \neq r'$, and probabilities are taken over the choices of $\pi_1, \pi_2^0, \pi_2^1, R_1, R_2$.

Note that from this theorem, if j, l correspond to a pair (j_r, l_r) (as in the honest user case), then the view provides complete information about b_r (since $\epsilon = 1$, so $b_r = R_1(j_r) \oplus R_2(l_r)$), whereas if j, l do not correspond to such a pair, only partial information about $b_{r'}$ is provided (since there is a positive probability for both $b_{r'} = 0$ and $b_{r'} = 1$).

In either case, the last two components of the view contain information about the bit $b_{r'}$, but the view does not depend on any other bit.

We proceed with a sequence of lemmas that will prove the theorem, by gradually adding components to the view, while maintaining its independence of all data bits except $b_{r'}$. The first three lemmas will establish the uniform distribution of the V^-, and lemma 4 will complete the calculation for the last component in the view.

Lemma 1 $\forall j,\ R_1(j) \sim U[\{0,1\}]$ *(probability is taken over choice of R_1).*

Proof: Obvious, since R_1 is chosen uniformly from all strings of length n with half 0's and half 1's, and thus for any particular location j, $R_1(j)$ is 0 or 1 with equal probability. □

Lemma 2 $\forall j,\ [j_1, \ldots, j_n \mid R_1(j)] \sim U[\text{all permutations on } \{1, \ldots, n\}]$ *(probability is taken over choice of π_1).*

Proof: π_1 is a uniformly distributed permutation

This is true independent of $R_1(j)$, and thus $[j_1, \ldots, j_n \mid R_1(j)] = [j_1, \ldots, j_n]$ is also uniformly distributed. □

Lemma 3 $\forall j$, $[l_1, \ldots, l_n \mid R_1(j), j_1, \ldots, j_n] \sim U$ *over all sequences of* n *distinct locations in* $\{1, \ldots, 2n\}$ *(probability is taken over choices of* $R_1, R_2, \pi_2^0, \pi_2^1$*).*

Proof: Given values $R_1(j), j_1, \ldots, j_n$, we want to prove that every sequence $l_1, \ldots, l_n$ is equally likely (i.e. uniform distribution). Fix an arbitrary R_1 with a suitable $R_1(j)$. This defines a sequence of bits $\{t_r = R_1(r) \oplus b_r\}_{r=1}^n$. Then, for $r \in \{1, \ldots, n\}$, l_r is chosen to be the index of the m_r'th bit with value t_r in R_2, where $m_r = \pi_2^{t_r}(r)$. Thus, for any particular sequence $l_1, \ldots, l_n$, $Prob[l_1, \ldots, l_n \mid R_1, R_1(j), j_1, \ldots, j_n] = Prob[\forall r : R_2(r) = t_r \ \wedge \ \pi_2^{t_r}(r) = m_r$ if l_r is the m_r'th bit with value t_r in $R_2]$. This probability (for a fixed R_1) is taken over R_2 and π_2^0, π_2^1. It is not necessary to calculate the exact probability to see that it is the same for each sequence $l_1, \ldots, l_n$, since π_2^0 and π_2^1 are both uniformly distributed permutations. We have some number k of restrictions on the values of π_2^0 and $n - k$ restrictions on the values of π_2^1, which yields a certain probability that these restrictions will be satisfied, regardless of the actual values $l_1, \ldots, l_n$ of the restrictions[1]. Thus for each sequence we have the same probability, and thus $[l_1, \ldots, l_n \mid R_1, R_1(j), j_1, \ldots, j_n] \sim U$ over all sequences of n distinct locations in $\{1, \ldots, 2n\}$. (where probability is taken over the choice of R_2, π_2^0, π_2^1). This is true for any fixed R_1, and thus it is also true when R_1 is chosen randomly. □

Lemma 4 $\forall j = j_r$, $l = l_{r'}$,

$$Prob[R_2(l_{r'}) = 0 \mid R_1(j_r), j_1, \ldots, j_n, l_1, \ldots, l_n] = \begin{cases} \epsilon & \text{if } b_{r'} = R_1[j_r] \\ 1 - \epsilon & \text{if } b_{r'} = \overline{R_1[j_r]} \end{cases}$$

[1]For a direct calculation, it is not hard to check that the probability is

$$\frac{\binom{n}{k}}{\binom{2n}{n}} \frac{(n-k)!}{n!} \frac{k!}{n!} = \frac{1}{\binom{2n}{n} n!} = \frac{1}{(2n)(2n-1)\ldots(2n-n)}$$

which is exactly the probability of uniformly selecting a sequence of n distinct locations in $\{1, \ldots, 2n\}$, as needed.

$$= \begin{cases} \frac{1}{2} - \frac{1}{2(n-1)} & \text{if } r \neq r' \text{and } b_{r'} = R_1[j_r] \\ \frac{1}{2} + \frac{1}{2(n-1)} & \text{if } r \neq r' \text{and } b_{r'} = \overline{R_1[j_r]} \\ 1 & \text{if } r = r' \text{and } b_{r'} = R_1[j_r] \\ 0 & \text{if } r = r' \text{and } b_{r'} = \overline{R_1[j_r]} \end{cases}$$

where $\epsilon = 1$ *if* $r = r'$, *and* $\epsilon = \frac{1}{2} - \frac{1}{2(n-1)}$ *if* $r \neq r'$. *(probability is taken over choices of* R_1*)*

Proof: Given $R_1(j_r), j_1, \ldots, j_n, l_1, \ldots, l_n$, from the way the l_r's were chosen, $R_2(l_{r'}) = R_1(j_{r'}) \oplus b_{r'}$, and thus $R_2(l_{r'}) = 0 \iff R_1(j_{r'}) = b_{r'}$. Therefore,

$Prob[R_2(l_{r'}) = 0 \mid R_1(j_r), j_1, \ldots, j_n, l_1, \ldots, l_n] =$

$= Prob[R_1(j_{r'}) = b_{r'} \mid R_1(j_r), j_1, \ldots, j_n, l_1, \ldots, l_n] = Prob[R_1(j_{r'}) = b_{r'} \mid R_1(j_r)]$

$$= \begin{cases} \frac{\binom{n-2}{\frac{n}{2}}}{\binom{n-1}{\frac{n}{2}}} & \text{if } r \neq r' \text{and } b_{r'} = R_1[j_r] \\ \frac{\binom{n-2}{\frac{n}{2}-1}}{\binom{n-1}{\frac{n}{2}}} & \text{if } r \neq r' \text{and } b_{r'} = \overline{R_1[j_r]} \\ 1 & \text{if } r = r' \text{and } b_{r'} = R_1[j_r] \\ 0 & \text{if } r = r' \text{and } b_{r'} = \overline{R_1[j_r]} \end{cases}$$

For $r = r'$ this is obvious. For $r \neq r'$ this is true because R_1 is a random string of length n with $\frac{n}{2}$ O's and $\frac{n}{2}$ 1's. Given $R_1[j_r]$, there are $\binom{n-1}{\frac{n}{2}}$ possible strings for R_1, each equally probable. Out of those, the number of possibilities where $R_1[j_{r'}] = b_{r'}$ is $\binom{n-2}{\frac{n}{2}}$, if $b_{r'} = R_1[j_r]$, and $\binom{n-2}{\frac{n}{2}-1}$ otherwise.

Now it is easy to verify that $\frac{\binom{n-2}{\frac{n}{2}}}{\binom{n-1}{\frac{n}{2}}} = \frac{1}{2} - \frac{1}{2(n-1)}$ and $\frac{\binom{n-2}{\frac{n}{2}-1}}{\binom{n-1}{\frac{n}{2}}} = \frac{1}{2} + \frac{1}{2(n-1)}$, which completes the proof of the lemma. □

Proof of Theorem 2: $\forall j = j_r, l = l_{r'}, \forall v^- = [(j_1, l_1), \ldots, (j_n, l_n), R_1(j_r)], \forall v = [v^-, R_2(l_{r'})]$,

$$Prob(v^-) = Prob[R_1(j_r)] \cdot Prob[j_1, \ldots, j_n \mid R_1(j_r)] \cdot$$

$$\cdot Prob[l_1, \ldots, l_n \mid R_1(j_r), j_1, \ldots, j_n] = \frac{1}{|M|}$$

Since by lemmas 1,2,3 all three terms in the product are uniformly distributed over their domain of possible values, and therefore V^- is uniformly distributed over its

domain M. Now, from lemma 4 we have that

$$Prob[v \mid v^{-}] = \begin{cases} \epsilon & \text{if } R_1[j_r] \oplus R_2[l_{r'}] = b_{r'} \\ 1-\epsilon & \text{otherwise} \end{cases}$$

Combining these equations, we get

$$Prob[v] = Prob[v^{-}] \cdot Prob[v \mid v^{-}] = \begin{cases} \frac{\epsilon}{|M|} & \text{if } R_1(j_r) \oplus R_2(l_{r'}) = b_{r'} \\ \frac{1-\epsilon}{|M|} & \text{otherwise} \end{cases}$$

which completes the proof of the theorem. □

Claim 3 (Communication Complexity)

The communication Complexity is $O(\log(n)PIR(n,k))$ *where* $PIR(n,k)$ *is a private information scheme with k database of n bits.*

The communication complexity of this scheme is $O(\log(n))$ to communicate with $R1$ and $R2$. The complexity for communicating with the rest of the databases is the same as PIR done for $\log(n)$ bits, $O(\log(n)CCPIR(n,k))$. Thus the over all communication complexity is $O(\log(n)CCPIR(n,k))$.

Chapter 5

Randomized Approach for Secure Data Distribution

In the previous chapter we showed how to achieve Database Privacy. However, that scheme does not solve DRP. In this section we present a scheme which solves the Data Replication Problem by achieving Private Data Distribution security. By private-data-distribution security we mean that any $t+1$ auxiliary databases contain information that is t-wise independent from the original database and can be prepared ahead of time. We assume that there are no more than t faulty databases. Therefore, the auxiliary databases cannot construct the original data from their data and the problem is solved. In addition we describe how to combine it with the RP scheme of the previous section in order to achieve a scheme which guaranties Private Data Distribution, Data Privacy, and User Privacy. The results are achieved by a reduction on a PIR scheme with only a constant factor $t \geq 2$ increase in the communication complexity.

We achieve this by replacing "real" copies of the database D by t random databases (R's). These R's are constructed such that any set of up to t R's is independent of the actual data (and thus no information about the data can be extracted from it), but still all R's together can simulate real copies of the original database when interacting with users who wish to retrieve information.

5.1 The Random DB Scheme

This scheme consists of a user U with query $q \in \{1, ..., n\}$, and $t * k + 1$ Databases: the original database D of n bits $D = b_1, ..., b_n$, and $t * k$ auxiliary databases $R_{i,j}$ for all $i \in \{1, ..., t\}$ and $j \in \{1, ..., k\}$ for constant $t > 2$ $k > 1$. During the setup stage, D computes all the R's and distributed them. Then, during the runtime stage, the user interacts with D and $R_{i,j}$s in order to obtain b_q. At the end of this protocol, neither D nor the $R_{i,j}$s know q, and no coalition of $R_{i,j}$'a of size $< t$ has any information about D. The $R_{i,j}$s for different j's are not allowed to communicate with each other or with D after the setup time, no coalition of t $R_{i,j}$'s with the same j but different i's is allowed to communicate.

5.1.1 The Setup Stage:

The original database D prepares $k * t$ random databases denotes by:

$$\begin{matrix} R_{1,1} & R_{2,1} & \dots & R_{t,1} \\ & \vdots & & \\ R_{1,k} & R_{2,k} & \dots & R_{t,k} \end{matrix}$$

Such that:

- The databases $R_{1,i}, ..., R_{(t-1),i}$ for all $i \in \{1, ..., k\}$ are chosen uniformly from all possible databases of size n.
- The databases $R_{t,i}$ for all $i \in \{1, ..., k\}$ are computed by xoring D with all the random $R_{j,i}$ for all $j \in \{1, ..., t\}$. In other words, $R_{t,i}$ is chosen such that $R_{1,i} \oplus \dots \oplus R_{t,i} = D$.

Note that for each row, $i \in \{1, ..., t\}$ all but one of the R's can be prepared in advance. We suggest that D does not have to prepare them at setup time, rather D can access a special web server designed for this purpose and choose ready R's for this protocol.

5.1.2 Assumptions:

We assume that no t databases of the same row, $R_{1,j}, ...R_{(t-1),j}$ communicate with one another.

5.1.3 Proofs of Setup Stage

Claim 4 (Communication Complexity) *The communication complexity of the setup stage of the RDB scheme is $O(ktn)$ which is a constant factor t larger than the setup communication complexity of the PIR scheme (when the databases is copied to k auxiliary databases).*

Claim 5 (Private Data Distribution) *The view of a coalition of size $< t$) R's is uniformly distributed over all D's, and thus gives no information about D and thus achieves Private Data Distribution security.*

Proof: Any coalition of $(t-1)R$ from the same row is uniformly distributed over D. Fix some $t-1R$'s we compute their xor. Given this xor for any D there is a possible R_t which matches it to the xor, since all the R's are chosen randomly from a uniform distribution, $R \oplus D$ is also uniformly distributed over all R, we gain no information about D because each D is still equally likely. This is true for any subset of $t-1$ R's.

5.1.4 On Line

During the runtime stage the user interacts with D and $R_{i,j}$s in order to obtain b_q. At the end of this protocol, neither D nor the $R_{i,j}$s know q, and no coalition of $R_{i,j}$'a of size $< t$ has any information about D. The $R_{i,j}$s for different j's are not allowed to communicate with each other or with D after the setup time, no coalition of $tR_{i,j}$'s with the same j but different i's is allowed to communicate.

User U with query $q \in \{1 \ldots n\}$ and the databases execute a protocol using any Private Information retrieval PIR scheme with the restriction that $PIR(((D \oplus R) \oplus R), D, q) = PIR(D \oplus R, D, q) \oplus PIR(R, D, q)$ our example is [14], :

- U prepares random subsets of indices $S_1, \ldots, S_k$ as in [14].

- U sends $S_i \rightarrow R_{1,i}, \ldots, R_{t,i}$ for each $i \in \{1, ..., k\}$
- Each database $R_{j,i}$ for $i \in \{1, ..., k\}$ and $j \in \{1, ..., t\}$, upon receiving S_i sends the user $\bigoplus_{ind \in S_i} R_{ind}$ Where R_{ind} is the value R has in the ind index.
- U XORs the values he has received from all of the databases and produces b_q, the qth value.

5.1.5 Proofs of the Online Stage

Claim 6 (correctness) *The value obtained when using the RDB scheme with an intended query q and the underlying PIR scheme is b_q.*

Proof: From our construction, $\forall i \; R_{1,i} \oplus \ldots \oplus R_{t,i} = D$, the xor of any database row is D. Therefore, after asking each of these databases the same query and xoring the answers, the user gets the answer he would get from D to the same query (since the kinds of queries used involve xoring of subsets and other operations which are closed under xor).

This is true for every row i, and thus the value obtained by combining the answers of the rows is the same value obtained by combining the answers of the copies $D_1, \ldots, D_k$ in the [14] scheme. □

Claim 7 (User Privacy) *The messages the user sends in the RDB scheme with a PIR underlying scheme satisfies User Privacy, i.e. are independent of the user's query.*

Proof: Since the messages the user sends to D and each $R_{i,j}$ are the same as the messages that he would send using PIR, his messages are independent of his query by reduction. The R's and D's do not communicate with one another and hence cannot find out the query. The only databases that can communicate are the ones from one row, but they all receive the same message so they get no information by communicating. □

Claim 8 (Communication Complexity) *The communication complexity of the Online stage of the RDB scheme is* $O(t * CCPIR(n,k))$ *where* $CCPIR(n,k)$ *is the communication complexity of a* PIR *scheme with* k *databases of size* n.

Proof: The communication complexity of the RDB scheme is exactly the same as in the PIR scheme, except that for each database in the underlying scheme we have a row of t databases here. Thus, the communication complexity is $O(t * CCPIR(n))$.

Assumptions

The $R_{i,j}$s for different j's are not allowed to communicate with each other or with D, after the setup time. In addition, no coalition of t $R_{i,j}$'s with the same j but different i's are allowed to communicate at all times.

5.2 Combining RP and RDB

In the above description of the RP scheme we treated D_i as if it was a database in a PIR scheme, instead in order to satisfy the properties of RDB we treat it as a database in the RDB scheme. In other words, in the RP scheme every D_i for $i \in \{1, ..., k\}$ we replace by R and $D_i \oplus R$ (for the simple case of $t = 2$). The user sends to R and $D_i \oplus R$ what he would have sent to D_i using a PIR scheme. When, the user gets the xors of the locations he first finds the location given to him by R, by xoring the results from all the R's. Similarly the user gets the locations given to him by $D_i \oplus R$. The user asks $R1$ and $R2$ for the values at the locations given to him by R and by $D_i \oplus R$ and then xors the results to obtain x_q. Note that the main idea here is that when the RDB scheme used the [14] it did not matter which xor it performed first. Here it is important to first ask $R1$ and $R2$ for the values in the locations and then xor the values, and not xor the locations and ask $R1$ and $R2$ in a xored location.

Chapter 6

Oblivious Database Scheme for No Data Distribution

In this section we present a scheme which solves Database Privacy and the Data Replication Problem by achieving no-data-distribution security. By no-data-distribution security we mean that all auxiliary databases contain information that is completely independent from the original database and can be prepared ahead of time. Therefore, the original data is not distributed to the auxiliary database and the problem is solved.

Before we go on to describe the scheme let us offer some motivation as to why we prefer no-data-distribution to the private-data-distribution level of security which we achieved in the previous chapter. Or in other words why completely independent databases are more desirable than pairwise independent databases. There are two immediate reasons for this. First, we would like the auxiliary database's data to be such that it will not give any information about the database even if the auxiliary databases break the rules and communicate among themselves. Otherwise, the data would enable them to regroup and form a new database which they could sell without the help of the original database. Second, we would like the auxiliary databases to be prepared ahead of time so that they will not have to compute their random contents immediately upon the request of the database, rather they can prepare it at a more convenient time, and also use it for other applications.

6.1 Overview of the Scheme

This scheme involves a user, a database, and 2 random and independent auxiliary databases. As before, the user interacts with all those databases in order to obtain his query, while all the databases are not allowed to communicate with one another. This type of interaction, however, is not immediate in our current scenario because if the auxiliary databases are completely independent then how can the user obtain relevant information from them. This difficulty is solved by assigning the auxiliary databases a new and different purpose or function than the one they had in the previous scheme. Their new purpose is to function as encryptors and permutors that are used to create an oblivious database from the original database. By an oblivious database we mean an n bit uniformly distributed string to which the database cannot find a mapping from the bits of the original data with probability greater than chance. The random auxiliary databases are used in two different settings. At setup time they encrypt and permute the original database to create an oblivious database as we mentioned. At runtime they provide the user with the decryption keys to his query to the oblivious database.

Instead of dividing the computation in our scheme into two stages, we could have used any multi party computation scheme. However, the communication complexity of multi party computation schemes is too large for our purposes (more than sublinear) and it involves many rounds of computation. By dividing our scheme into two stages, we are able to reduce the communication complexity of the runtime stage in which a multi party computation is not performed. On the other hand, the setup stage, in which a multi party computation is performed is executed only at the beginning of many queries, and we consider it no more expensive than the stage in the previous schemes in which the databases are duplicated. The runtime stage is the one that is executed as much time as there are queries. Therefore, the overall communication complexity is sublinear.

At the setup stage the oblivious database is created via a multi party computation scheme between the original database and the two random auxiliary databases. Dur-

ing the computation the Database is encrypted by one of the auxiliary database and permuted according to the contents of the other auxiliary database. The result of the multi party computation goes to the original database. Following this computation, the user, at run time, is able to privately query the oblivious database by accessing it directly. The direct access is private because the data is oblivious to the original database and thus he cannot find the original query from the index that was accessed. The user knows the new permuted location of his query through communicating with the auxiliary database, and similarly he knows how to decrypt his value based on the communication with the other auxiliary database. The communication with the auxiliary database is done through a PIR scheme so the user's query is again secure.

Before we give the detailed description of the multi party computation scheme, let us summerize what this scheme achieves. No data distribution privacy for the database is achieved because the auxiliary databases are random and prepared ahead of time. In addition, the user's privacy is achieved because the user's direct accesses is distributed uniformly for all the queries since the oblivious database does not give any information about its real query. The user's interaction with the auxiliary databases is also private because it is done through a PIR scheme. Furthermore, data privacy is achieved because the user receives the encryption of only one value, his query, and no other. The communication complexity of this scheme is a factor of $\log(n)$ times the communication complexity of the Private Information Retrieval scheme. The scheme achieves information theoretic security that depends on the assumptions we make about the faultiness of the databases. In this scheme we assume that the databases are honest in addition we assume that they follow the No Communication assumptions meaning that they do not communicate with one another via any form of communication.

6.2 Setup Stage: Creating the Oblivious Database

We will now describe the multi party computation which occurs during the setup stage and creates the oblivious database using the auxiliary databases. Later we will

describe the runtime stage.

During the setup stage the original database, D, performs a multi party computation with the auxiliary databases, R and P. The result of this computation goes to the database and it is an oblivious version of it which is uniformly distributed in $\{0,1\}^n$ over all choices of P and R.

The oblivious database is created through a xor of the original database with R and then permuting the result with a random permutation which is the value of P.

More formally:

- Database D contains the bits $b_1, \ldots, b_n$
- R contains a random sequence of n bits: $\{x_1, \ldots, x_n\}$ which we denote by R.
- P contains a permutation P of the indices of D, $(j_1, \ldots, j_n)$.
- D, R, P perform a Multi Party computation described below to compute the encryption of D which is a xor with R and permutation using P. $E(D) = P(R \oplus D)$ and is held by D at the end of the computation.

The multi party computation is done as follows:
D prepares a random database $D1$ and computes $D2$ such that $D = D1 \oplus D2$.
R prepares random $R1$ and computes $R2$ such that $R = R1 \oplus R2$.
P prepares a random permutation $P1$, and computes $P2$ such that $P(D) = P2(P1(D))$.
Note that even though we ask the parties to prepare their values for the computation now, those values are independent of one another and can also be prepared ahead of time, i.e. R can also prepare R1 and $R2$ ahead of time. The following information is sent between the parties in secure channels:

- $D \rightarrow R : D1$
- $D \rightarrow P : D2$
- $P \rightarrow R : P1$
- $R \rightarrow P : R2$

- $R \rightarrow D : P1(R1 \oplus D1)$
- $P \rightarrow D : P2$
- $P \rightarrow D : P(R2 \oplus D2)$

D computes $P2([P1(R1 \oplus D1)]) \oplus [P(R2 \oplus D2)]$ which we will show to be $P(R \oplus D)$, where the contents of [] he received. D now has $P(R \oplus D)$

P and R discard all the values that were sent to them during the multi party computation. (they discard $D1$ $D2$ $R2$ and $P1$)

After the computation

- D has $E(D)$
- P and R do not know D.
- $E(D)$ is uniformly distributed over all choices of P and R.

The communication complexity of the setup stage is high - but so it is in the PIR case when the databases are duplicated they need to send the whole data to the auxiliary database.

6.2.1 Assumptions:

For the multi party part of the protocol we assume that the databases are Honest But Curious: they follow the protocol meaning that they do not exchange some other information such as an agreement of how to act later in order to get the user's query. In addition the databases follow the no-messages assumption meaning that they do not send each other extra messages outside of what the protocol permits.

After the MPC P and R do not need the information that was sent to them during the computation. Therefore since they follow the protocol they can discurd this information right afterwards. One thing to notice here is that during the multi party computation R and P gain information that is not completely independent of D. If P and R would get together they could obtain all of D's secrets from $D1$ and $D2$. We still consider this scheme to achieve no-data-distribution security because P

and R do not need to keep this knowledge in order to function as a database. So we assume that they discard those bits before they have a chance to become malicious.

The assumption that the auxiliary databases indeed discurd the information which they do not need requires honest databases. In order to relax this assumption we can allow only less than $t > 2$ faulty auxiliary databases. Therefore, if we increase the number of auxiliary databases, such that instead of having just two R and P, we can have $R1, R2, ...Rt/2, P1, P2, ...Pt/2$ databases, then D can construct $D1, ...Dt$ instead of $D1$ and $D2$ for the multi party computation, and only a coalition of t databases will be able to compute D from it. But we are only allowing less than t faulty databases.

6.3 Proofs of the Setup Stage

Claim 9 (Correctness) *D has $P(R \oplus D)$ at the end of the multi party computation.*

Proof: D computes $P2([P1(R1 \oplus D1)]) \oplus [P(R2 \oplus D2)]$. Let $X = (R1 \oplus D1)$ and $Y = (R2 \oplus D2)$ This comes down to showing that $P(X) \oplus P(Y) = P(X \oplus Y)$ In the left side of the equation we first permute all the bits and then xor them. In the right side of the equation we first xor the bits and then permute the bits. Thus, both sides are equal. We also note trivially that $R \oplus D = (R1 \oplus D1) \oplus (R2 \oplus D2)$.

Claim 10 (Security) *We define the view of R to be $R.V$. At the end of the multi party computation $R.V$ is independent of D.*

Proof: $R.V = D1UP1$ which are random sequences of bits and are chosen randomly from all sequences $\{0,1\}^n$ independently of D.

Claim 11 (Security) *We define the view of R to be $R.V$. At the end of the multi party computation $R.V$ is independent of P.*

Proof: $R.V = P1UD1$ which are chosen randomly from all sequences $\{0,1\}^n$ independently of P.

Claim 12 (Security) *We define the view of P as $P.V$. At the end of the multi party computation $P.V$ is independent of D and R.*

Proof: $P.V = R2, D2$. $R2$ and $D2$ are uniformly distributed in all sequences $\{0,1\}^n$ taken over all choices of R and D since they are the result of a xor of a bit string with a random bit string. Therefore, the probability that P communicates with a certain R and D is equal for all R and D's so P gets no information about them and its view is independent of R and D

Claim 13 (Obliviousnes of E(D)) *At the end of the multi party computation D's view is an oblivious database: E (given D) is uniformly distributed over all sequences $\{0,1\}^n$. Taken over all possible R and P.*

Proof: D is fixed. For every possible P we can find an R that will encrypt D to become E. Since P and R are unformly distributed and they are chosen randomly from those choices. Each pair of P's and R's are equally likely and therefore all E's are equally likely and are of the same distribution.

The view of D is actually $P1(R1 \oplus D1)$, $P2$, $P(R2 \oplus D2)$. D can compute $P(R(D))$, and using $P2$, D can find $P1(R2 \oplus D2)$, $P1(R(D))$, and he has $P1(R1 \oplus D1)$. Suppose that D tries all possible permutations $P1$. For each possible permutation $P1$, D can compute a properly fitting R, $R1$, $R2$ such that $R = R1 \oplus R2$, by first permuting $E(D)$ with the inverse of $P1$ and then xoring the result with D. Since P, R and $R1$ are chosen randomly and independently , all those possibilities are of equally likely. Therefore, D is not able to get anything new about $P1$ and therefore P. Similarly, D does not learn anything new about R.

Claim 14 (Communication Complexity) *The communication complexity of the setups stage is scheme is $5n + 2n\log(n)$ for all the messages sent in the multi party computation.*

6.4 On Line - The User's Query

Having described how E, the oblivious database, is created, we will describe how it is used during runtime in order to achieve privacy for the user and privacy for the database.

The on line phase involves a user U with query $q \in \{1, \ldots, n\}$. A database D which holds the original data and the resulting oblivious database, E. In addition, there are two auxiliary database R and P as during the setup stage.

When the user wishes to get the value of his query q, from the database D, he first queries P using a PIR (Private Information Retrieval) scheme for the q'th index and gets the permuted location or the corresponding location in E, j_q. The user queries R for the decryption of the value of q, using the PIR scheme. Then, the user goes to D and asks directly for the value at j_q. He then receives an encryption of his desired value and decrypts it using the values he received from R.

More Formally: By A $\overset{PIR}{\longleftrightarrow}$ B we mean interaction between A and B trough the PIR protocol, and similarly $\overset{DIRECT}{\longrightarrow}$ means interacting directly in the clear.

- $U \overset{PIR}{\longleftrightarrow} P$: q, and $P \overset{PIR}{\rightarrow}$ U:j_q.
- $U \overset{PIR}{\longleftrightarrow} R$: q, and $R \overset{PIR}{\rightarrow}$ U:x_q.
- $U \overset{Direct}{\longrightarrow} D$: j_q.
- $D \overset{Direct}{\longrightarrow} U$: $E(D)_{j_q}$.
- U decrypts by computing $E(D)_{j_q} \oplus x_q$ in order to obtain b_q.

6.4.1 Assumptions:

We assume that the databases are honest bur curious and that they follow the protocol and do not send the user some other information then what they are supposed to. The databases also follow the No-Communication assumption which means that they are not allowed to communicate in any form, they are not allowed to send extra

messages and they are not allowed to act as a user for example, and interact with another database in that way.

6.5 Proofs of the Online

Claim 15 (User Privacy) *The oblivious database scheme is User Private: Given D and E, the user's query j_q is uniformly distributed for all indices i in D.*

Proof: $Pr[j$ is the permuted location of i for some $i] =$
$$\frac{\text{All possible permutations such that } j \to i \text{ given } D\&E}{\text{All possible permutations given } D\&E}$$
Since E is uniformly distributed over all P and R which we proven in the setup stage proofs.
$$= \frac{\text{All permutations s.t.} j \to i}{\text{All possible permutations}}$$
Because all P's are chosen randomly with the same probability from the same distribution, the number of all possible permutations with one bit fixed ($j \to i$) is $(n-1)!$. Therefore, the probability is:
$$= \frac{(n-1)!}{n!} = \frac{1}{n}$$

Therefore the probability that j is related to a particular i is the same as the probability of guessing any i. Thus one j is not related to a particular i with any more probability than chance. So the database cannot match the user's question j_q with the query q with probability greater than chance.

Claim 16 (User Privacy) *The query the user sends P and R is uniformly dirsributed.*

Proof: The user uses PIR directly to talk to P and R so therefore based on PIR schemes P and R receives information that is uniformly distributed over all queries.

Claim 17 (Data Privacy) *The oblivious database scheme satisfies Database Privacy.*

Proof: D does not give extra information to the user, since the user only gets one encrypted bit, the one of his query.

Claim 18 (No-Data-Distribution) *The oblivious database scheme satisfies No Data Distribution.*

Proof: As shown in the setup time proofs, R and P do not have any information about D, and their data was prepared ahead of time and is completely independent of D.

Claim 19 (Correctness) *If PIR is correct then this scheme gives the user the correct value b_q.*

Proof: Based on the correctness of the underlying PIR scheme the user gets the correct new location of the index q from P, and the correct x_q from R. Therefore, he is able to get the correct value from the correct location and decrypt it properly.

Claim 20 (Communication Complexity) *The communication complexity is $O(\log(n) * CCPIR(k,n))$. Where $CCPIR$ is the communication complexity of a PIR scheme with k databases of n bits.*

Proof: We analyze the communication complexity here, in terms of the communication complexity of the PIR [14] scheme, denoted by $CCPIR(n,k)$. The communication complexity is $\log(n)$ to retrieve the encrypted bit, because the user sends an index to D of size $\log(n)$. The communication complexity is $O(\log(n) * CCPIR(k,n))$ to get the permuted location of q because it is $\log(n)$ bits to represent the index to n and we do PIR for all those bits. The communication complexity is $CCPIR(k,n)$ to get the random bit from R. Over all the communication complexity is $O(\log(n) CCPIR(k,n))$

6.6 Extensions

So far the results achieving No data distribution guarantee privacy for the user for a single application of the scheme. However, when this scheme is used repeatedly, the database is able to tell whether the user is asking for the same query. This compromises the security of the user. We have not fully examined a way in which

this problem is solved, but we do suggest a scheme which can be used in future examination.

First let us examine the following trivial solution: after each query set R and P to be new random values and rerun the setup stage. Obviously this is too costly for the database. Therefore, we would like to run the scheme consecutively with the same setup stage. Another trivial solution is for the user to scan the whole database. This way the Database does not know which query the user actually wanted. But this is not efficient either.

In order to achieve a secure yet feasible solution we propose to use a method that is a compromise between the two trivial methods. This solutions is inspired by the Oblivious RAM [17] scheme. In this solution, the user does not have to scan the whole database, instead he scans a smaller section (we propose the size $n^{1/3}$) which we call the shelter. Only when the shelter is full, after some queries ($n^{1/3}$), we reinitialize the setup stage, and not after every query as suggested in the trivial solution.

The scheme works by adding two parts to the oblivious database, a shelter and a dummy section. The shelter is a separate part of memory to which the user can write, and it holds the values of indices that were already queried along with their permuted address. The dummy section consists of garbage values and its contents are interwined with those of the oblivious database such that the dummy section is indistinguishable from the other parts of the oblivious database. When the user queries the Database he first scans the shelter. If his query is in the shelter then he obtains it and accesses a dummy location. If his query is not in the shelter he accesses the permuted location of his query. Since the locations of the dummy section and the real section are indistinguishable, the Database cannot tell those two cases apart and therefore cannot tell whether the user is accessing something for the second time. Whichever value the user accessed (real or dummy) he then places its content along with its address in the shelter. This way the user never accesses the same location twice because if something was already accessed then it is in the shelter.

This gives us a method to hide the user's access pattern for multiple queries. Since the whole shelter is scanned at each query. The communication complexity is

increased according to the size of the shelter. Therefore, the size of the shelter should be as small as possible. On the other hand we can only have number of accesses as the size of the shelter before we need to reinitialize the setup stage. We would like the number of repeated steps to be as large as possible. In our case we can fix the size of the shelter to be $n^{1/3}$ which is the same as the PIR communication complexity so it would only add a constant factor in our over all communication complexity and it is a pretty large number for repeated queries without reinitialization.

We will now describe the details of how the scheme works for a shelter of size k:

Before the multi party computation:

- R adds k random bits to its data so that now it has $x_1, ..., x_{n+k}$.
- P is a new random permutation of $n+k$ locations.
  The Multi Party Computation is performed as before.
- D receives an oblivious database of size $n+k$.
- D adds a size k empty shelter to its data. This shelter is write accessible to the user.
  At runtime the user has the query q which is his m'th query to the database since the setup stage.
- The user uses PIR to get q's decryption key x_q from R as before.
- The user uses PIR to get q's new location from P, j_q, as before.
- The user scans the shelter to check whether j_q was already accessed. (In the shelter the encrypted bits are tagged with their permuted location.)
- If the user finds the tag j_q then he decrypts it using x_q and has his query.
  - Now the user needs to access something in the oblivious database so that the database will not know that the user is accessing the same thing again.

- The user ask P using PIR for the m'th dummy location, $n+m$ and gets j_{n+m} back.
- The user directly sends j_{m+n} to E. and receives the garbage value x_{m+n}.
- The user places x_{m+n} and a tag j_{m+n} in the shelter.

- If the user does not find the tag j_q in the shelter:
  - The user asks P using PIR for the location at q, and gets j_q again. (This is done because the user needs to ask P again so that P will not know that the user found his query in the shelter).
  - The user sends directly to E j_q and receives E_{j_q}.
  - The user places E_{j_q} in the shelter along with the tag j_q.

Then for everybody:

- The user decrypts the value he obtained using x_q and gets the value of his query b_q.
- P updates its public counter of how many times it has been accessed since the setup time. So that the user will have the value of m next time.

Since the user talks to P using PIR, P cannot know whether the user is asking for the address of a dummy location or of q again.

The user does not gain any information by looking at all the shelter because it cannot map the tags of the shelter to the real database bits.

6.7 Draw Back of the Oblivious Database Scheme

We wish to point out a disadvantage of this scheme. The database cannot communicate with the random databases R and P at all, not even as a user. In the previous schemes if the database queried the supplementary databases as a user he could not get information about user's queries. However, in this scheme if it queried R or P as

a user it would be able to get information about what the decryption keys are, and therefore information about the user's queries. Although, this scheme has this disadvantage we feel that using it for no data distribution security constitutes a worthwhile tradeoff since we assume that the database is trustworthy and that it will not communicate with forbidden parties as long as those parties do not hold information that is dependent on its own.

Chapter 7

Similarity To Oblivious Transfer

Database Privacy and User Privacy togetehr, as achieved in our schemes are identical to the $\binom{n}{1}$ oblivious transferproblem, since we defined security to mean the privacy of the user's query and the privacy of the database's information.

In $\binom{n}{1}$ oblivious transferthere are two parties Alice and Bob. Alice has a n secrets $S_1, ..., S_n$, and Bob has an index $i \in \{1, ..., n\}$. At the end of the protocol, Bob gets S_i, and no information about any S_j for $j \neq i$, and Alice does not know i. In the secure information retrieval problem the database is Alice and the user is Bob.

Previously, the Oblivious Transfer problem was only studies in a model with one Alice. In the one Alice model it was shown that the protocols communication complexity cannot be smaller then linear in n.

In this paper, the $\binom{n}{1}$ oblivious transferproblem was studies in a multiple Alice model. In this model, we have shown that the communication complexity can be reduced to size sublinear in n. This is a major improvement because up until now the communication complexity of $\binom{n}{1}$ oblivious transferwas thought to be linear. In addition, we achieve information theoretic security which is unattainable in the one Alice model.

Chapter 8

Hiding User's ID

8.1 Introduction

So far we showed how to achieve Database Privacy, Private Data Distribution, and User Privacy. This prevents the database from knowing which query the user is interested in. However, it allows the database to gain information about the user nontheless, because the database knows that the user is interested in one of its bits. This is important information in cases where the access to the database itself is controversial, and not so much which bit is accessed, as is the case of databases of job listings. Sometimes the user wishes to conceal the fact that he is looking for a new job but he does care to hide exactly which job he is looking for. Therefore, in this chapter we deal with preventing the database from knowing whether a certain user accessed it or not. We call this problem the Secure Identity problem not because we are hiding the user's identity but because we prevent information from being linked to the user's identity that might be used against him.

Not much work has been done in order to solve this problem. There is, however, a sight in Carnegie Mellon University called anonymizer which allows one to "surf the web without revealing any personal information". It is currently restricted to Carnegie Mellon Computer Science sites. The URL for this page is Http://anonymizer.cs.cmu.edu:8080/ maintained by Justin Boyan (jab@cs.cmu.edu). Still, the privacy of the user on this site is conditional on the fact that the user trusts

this particular anonymizer server since this server itself can obtain information about the user when the user surfs through it.

In order to solve the Secure ID problem, we consider a network model which must include more than one database or one user, otherwise since the information is sent and retrieved, it will be clear to the database that a retrieval is occurring between it and the sole user. Such a network can be modeled using multiple users or multiple databases. In this thesis, we choose to use the multiple databases approach by including in the network databases with different real data (as opposed to duplicates or random auxiliary ones that are needed for the other scheme). This prevents a database from knowing whether the user is accessing its contents or the contents of another database in the network. For example the network would consist of the following databases: a job listing, stocks information, movies in Cambridge, a university technical report listing. Using such a network the job listing database would not know whether the user is interested in its data or in any other data in this network and thus will not give the database information that is meaningful about the user.

One trivial solution to the secure ID problem is to have the user query all the databases in the network. This way each database does not know whether the user was interested in its bits or not. We assume that each database is queried using PIR so that the user's query to the database will not be revealed. The communication complexity of this result is $O(n * CCPIR(m))$, where $CCPIR(m)$ is the the communication complexity to query one database of length m using PIR. This communication complexity is too large.

Our solution achieves communication complexity of $O(\log(n)CCPIR(n)+\log(m)CCPIR(m)$ by introducing another party S which is a server who helps with the protocol. Thus our model now consists of a user, a server, and n different databases of length m. Each database also has a pair of R and P with which it creates an oblivious version of it using the oblivious database scheme described in chapter 6. However, here, instead of sending the oblivious database to the original database as in chapter 6, the oblivious database is sent directly to the server S. Then the server according to a random permutation sends it to another database in the network. The databases do

not know, and cannot determine, whose oblivious database they received. The user queries the database as in the oblivious database scheme of chapter 6, which means that he directly accesses the oblivious database. In order to find the new address of the oblivious database the user queries the server S using a PIR scheme. The user asks his query from the oblivious database in its new location in the network. Thus, the communication complexity of this scheme is $O(\log(n)CCPIR(n))$ for querying the oblivious database and $O(\log(m)CCPIR(m))$ for querying S.

8.2 Scheme

This scheme consists of n different original databases of length m, $2n$ databases of random bits, $2n$ permutation databases, a server S, and a user. Each group of 1 original database 2 random databases and 2 permutations are assigned a location by the server.

8.2.1 At Setup Time:

There are n databases. Each database Di has a location i in the network. The server S contains a random permutation, $NEW - LOC$, that maps location $i \in \{1, ..., n\}$ of each database Di to some random location l in the network. The permutation is represented in S in a way that the ith index in S's database contains the location l.

Each database Di performs a multi party computation as described in chapter 6 with the random R and P of his group. Except that now the result of the multi party computation go to S instead of to Di. So that S receives $[P1i(R1i \oplus D1i)]$, $P2i$, $[Pi(R2i \oplus D2i)]$, and computes $P2i([P1i(R1i \oplus D1i)]) \oplus [Pi(R2i \oplus D2i)]$, in order to receive $E(Di) = Pi(Ri \oplus Di)$. Once S has $E(Di)$ for each $i \in \{1, ..., n\}$ it sends $E(Di)$ to the location l in the network, $l = NEW - LOC(i)$.

At the end of this, each database in the network holds an oblivious version of another randomly chosen database in the network whose identity it cannot determine without communicating with the server S.

So at the end of the setup stage the network consists of databases Di which

hold their original value Di, and an encrypted database $E(Dj)$, such that $NEW - LOC(j) = i$, and S contains the value i in its j'th index.

Then S also relocates the R's and P's according to the same permutation.

8.2.2 At Run Time:

When the user wishes to ask his query q, from a database Di, he first queries S to find $NEW - LOC(i) = l$. Then, he goes to the lth place in the storage of P's and asks Pi's for the q'th index and gets j_q back. The user then asks the Ri in the lth location for the value of q, and receives x_q. Then, the user goes to the location l in the network and asks Dl for the value in the j_qth index from the oblivious database it holds. He then receives $E(Di)j_q$ and computes $E(Di)j_q \oplus x_q = Diq = b_q$. All the communication with P R and S are done through the PIR scheme so that P and R do not know the user's query and S does not know the user's database.

- $U \overset{PIR}{\longrightarrow}$ S: i, and $S \overset{PIR}{\longrightarrow} U$ $NEW - LOC(i) = l$.
- $U \overset{PIR}{\longrightarrow} P_l$: q, and $P_l \overset{PIR}{\longrightarrow} U$: j_q.
- $U \overset{PIR}{\longrightarrow} R_l$: q, and $R_l \overset{PIR}{\longrightarrow}$ U: x_q.
- $U \overset{Direct}{\longrightarrow} D_l$: j_q, and $D_l \overset{PIR}{\longrightarrow}$ U: $E(D_i)_{j_q} = x_q \oplus b_q$.
- U decrypts by computing $(x_q \oplus b_q) \oplus x_q = b_q$.

Since Dl does not know whose oblivious database it holds, it does not know which database the user is querying. In addition, since Di does not know which database holds its encrypted contents, it will not be able to know whether the user is asking it a question. In addition, the server does not know the user's query because the PIR scheme is their means of communication.

8.2.3 Assumptions

We hold the same assumptions about the databases as we did in the oblivious database scheme. In addition, we assume that all the databases do not communicate with the

server according to the No Communication assumption which means that they are not even allowed to talk to the server by pretending to be a user or in any other way. The sever does not communicate with the database according to the No messages assumption.

8.2.4 Proofs

Claim 21 (Security:) *Database Dj cannot decide that $E(Di)$ is the encryption of Di: $E(Di)$ is uniformly distributed over all possible D's.*

Proof: Given the view of Dj $E(Di)$ can be the encryption of each D with the same probability. The probability that a certain D is the database that is encrypted depends on the appropriate P's and R's that would compute E from that D. For each database D, Dj can produce the same number of pairs R and P that would encrypt D to $E(Di)$. The number of pairs is the same as the number of possible P's, because for every P there is exactly one R which encrypts D into E and it is possible to find an R for every P that would encrypt D to E. All the possible pairs are the same probability because each P and R are chosen independently and randomly. This is independent of D. Therefore, each D has the same probability of being the one that is encrypted. So E is uniformly distributed over all D's.

In addition to hiding the user's identity, the database Dl is not able to find which query q the user is asking because the user asks j_q and receives $x_q \oplus b_q$ which gives Dl no information about the value of b_q, or the location of q.

Claim 22 Correctness: *At the end of the scheme the user gets b_q if the underlying PIR scheme is correct.*

Proof: The correctness of this scheme is trivial. Since the server S distributes the encryptions, the user gets the correct new location from S, based on the correctness of the PIR scheme used. The user then gets the correct new location of the index q from Pi, and the correct x_q from Ri for the same reasons. Finally the user asks for the value at $E(Di)j_q$ and xors it with x_q to obtain b_q.

Claim 23 (Communication Complexity:)

The communication complexity is $O(\log(n)CCPIR(n) + \log(m)CCPIR(m))$.

Proof: The communication complexity is $\log(n)CCPIR(n)$ to get the new location (of logn bits) of the oblivious database. It is a constant to retrieve the encrypted bit. The communication complexity is $\log(m)CCPIR(m)$ to get the permuted location of q (of $\log(m)$ bits). The communication complexity is $CCPIR(m)$ to get the random bit from R. Over all the communication complexity is $O(\log(n)CCPIR(n) + \log(m)CCPIR(m))$.

At setup time the server has to send n databases which is communication complexity $n * m$.

Chapter 9

Future Work

- Repetitions: The schemes in this thesis were proven to be secure for one application of the protocol. However, for the application of those schemes it is natural that they work for multiple executions as well. Since restarting the scheme in every execution is too expensive because of the setup time, we would like to guarantee that our scheme will also work for consecutive queries.
- Removing the No Communication assumption: In order for our scheme to be secure we must assume that certain databases must not communicate with others by any means. We would like to relax this assumption and allow for the databases to communicate as a user for example, but still forbid them from sending extra messages.
- Reducing communication Complexity even further.

Chapter 10

Conclusion

In this thesis we introduced and solved three privacy problems in Secure database access protocols: Database Privacy, the Data Replication Problem, and the Secure user Identity problem. Database Privacy is concerned with keeping the databases information secure from the user. The Data Replication problem deals with a new security concern for databases that emanates from the need to replicate and distribute their contents in order to achieve security for the user. The Secure user ID problem is concerned with keeping private the user's identity, so that no information can be associated with or learned about that identity.

We solve those problems by using our scheme for communication between a user, a database and auxiliary databases whose contents are not dependent on the original database. This protocol allows the user to keep his query secret from the databases, and allows the database to keep its data secret from the auxiliary databases and from the user (except for the value of the particular query). We also showed how to extend that scheme into a protocol for a network of databases that solves the Secure ID problem. Those protocols rely on no cryptographic assumptions and maintain sublinear communication complexity in the size of the database.

We showed two reductions:

Theorem: For any $k \geq 2$ given any Private Information Retrieval k-database scheme for n data bits with communication complexity $R(k, n)$ there exists a *private-data-distribution* and *database private* $2k$-database scheme with communication com-

plexity $O(R(k,n)\log(n))$ where each database holds $O(n)$ bits.

Theorem: For any $k \geq 2$ given any retrieval k-database scheme for n data bits with communication complexity $R(k,n)$ there exists a *no-data-distribution* and *database private* $2k$-database scheme with communication complexity $O(R(k,n)\log(n))$ where each database holds $O(n)$ bits.

In addition, we solve the Secure ID problem by presenting a protocol for a network of a user U, n databases of size m with an additional server S. A database in the network does not know whether U asked him a query or asked a query from another databases. Therefore, we say that he does not know the identity of the users that are querying him. The communication complexity of that scheme is $O(\log(n)R(n,k1) + \log(m)R(m,k2))$ for constants $k1$ and $k2$.

Bibliography

[1] M. Abadi, J. Feigenbaum, J. Kilian. On Hiding Information from an Oracle. JCSS, 39:1, 1989.

[2] N. Adam, J. Wortmann. Security Control Methods for Statistical Databases: a comparative study, ACM Computing Surveys, 21(1989).

[3] A. Ambainis. Upper Bound on the Communication Complexity of Private Information Retrieval.

[4] D. Beaver, J. Feigenbaum. Hiding instances in Multioracle Queries. STACS,1990.

[5] D. Beaver, J. Feigenbaum, J. Kilian, P. Rogaway. Security with Low Communication Overhead. CRYPTO 1990.

[6] D. Beaver, J. Feigenbaum, R. Ostrovsky, V. Shoup. Instance-Hiding Proof Systems. DIMACS TR–93-65, 1993.

[7] D. Beaver, J. Feigenbaum, J. Shoup. Hiding Instances in Zero-Knowledge Proof Systems.

[8] M. Bellare, and S. Micali. Non-Interactive Oblivious Transfer and Applications. Crypto 89, pages 547-559.

[9] M.Ben-Or S.Goldwasser J.Kilian, A.Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. STOC 1988 p113-131.

[10] M.Ben-Or S. Goldwasser A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation STOC 1988 p1-10.

[11] G. Brassard, C. Crepeau, J.M. Robert. All or Nothing Disclosure of Secrets. CRYPTO 1986.

[12] G. Brassard, C. Crepeau, J.M. Robert. Information Theoretic Reductions among Disclosure Problems. FOCS 1986.

[13] B. Chor, N. Gilboa. Computationally Private Information Retrieval. STOC 1997.

[14] B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan. Private Information Retrieval. FOCS 1995.

[15] O. Goldreich. Foundations of Cryptography - Class Notes, Computer Science Dept., Technion, 1989.

[16] O. Goldreich, S. Micali,A. Wigderson. How to Play Any Mental Game", STOC 1987, 218-229.

[17] O. Goldreich, and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. 1995.

[18] J. Kilian. Founding Cryptography on Oblivious Transfer. STOC 1988 p20-31.

[19] R. Ostrovksy, V. Shoup. Private Information Storage. To appear in STOC 1997.

[20] M. Rabin. How to Exchange Secrets by Oblivious Transfer. Harvard TR-81, 1981.

[21] P. Tendick, and N. Natloff. A modified Random Perturbation Method for Database Security. ACM Transactions on Database Systems, 19:1, pp.47-63, 1994.