A Random Server Model for Private Information Retrieval or Information Theoretic PIR Avoiding Database Replication[*]

Yael Gertner Shafi Goldwasser Tal Malkin

April 1997

Abstract

Private information retrieval (PIR) schemes provide a user with information from a database while keeping his query secret from the database manager. We propose a new model for PIR, utilizing auxiliary random servers providing privacy services for database access. The principal database initially engages in a preprocessing setup computation with the random servers, followed by the on-line stage with the users.

Using this model we achieve the first PIR information theoretic solutions in which the database does not need to give away its data to be replicated, and with minimal on-line computation cost for the database. This solves privacy and efficiency problems inherent to all previous solutions. Specifically, in all previously existing PIR schemes the database on-line computation for one query is at least linear in the size of the data, and all previous information theoretic schemes require multiple replications of the database which are not allowed to communicate with each other. This poses a privacy problem for the database manager, who is required to hand his data to multiple foreign entities, and to the user, who is supposed to trust the multiple copies of the database not to communicate. In contrast, in our solutions no replication is needed, and the database manager only needs to perform O(1) amount of computation to answer questions of users, while all the extra computations required on line for privacy are done by the auxiliary random servers, who contain no informatin about the data.

1 Introduction

Private Information Retrieval (PIR) [7] schemes provide a user with information from a database in a private manner. The database consists of n items (e.g. bits), and the user has a query i which is the index of the data item he is interested in. Privacy for the user in this setting involves keeping the identity of the query, namely i, secret from the database. One motivation for this problem arises, for example, in the following setting. Consider a database of investment stocks, which sells information with a certain charge per query, and a user who wants to obtain information about a particular stock. In this scenario, it is very likely that the user does not want to reveal the stock of his interest, and thus the use of a private information retrieval protocol is appropriate. *Symmetrically Private Information Retrieval (SPIR)* [8] addresses the database privacy as well by adding the requirement that the user, on the other hand, cannot obtain any information about the database in a single query except for a single physical value.

Protocols for PIR and SPIR schemes, guaranteeing information theoretic privacy, appeared in [7, 3, 10, 8]. These solutions are all based on the idea of using multiple copies of the database that are not allowed to communicate with each other. This allows the user to ask different questions from different copies of the database and combine the responses he gets to get an answer to his query, without revealing his original query to any single database (or a coalition). The recent PIR scheme of [9] uses a single database, but guarantees only computational privacy under the assumption that distinguishing quadratic residues from

[*]This work was done with the support of DARPA grant DABT63-96-C-0018

non-residues modulo composites is intractable. In fact, it can be shown that using a single database makes it impossible to achieve information theoretic privacy with sublinear communication complexity.

We note that the central focus of the PIR and SPIR works till now, has been to minimize the communication complexity between the user and the database as a function of the database size.

1.1 Problems with the Model

Unfortunately, the common paradigm behind all the solutions that guarantee information theoretic privacy — the replication of the database in multiple separated locations — introduces a serious privacy problem to the database, *the data replication problem*. Namely, the fact that the database owner is required to distribute its data among multiple foreign entities, each of which could be broken into, or could use the data and sell it to users behind the legitimate owner's back. This is particularly problematic since the database cannot communicate with any of the other copies. Since this replication is used to protect the user's interest, it is doubtful that real world commercial databases would agree to distribute their data to completely separated holders which they cannot communicate with. Viewed from the user's standpoint, it may be doubtful that users interested in privacy of their queries would trust copies of the same database not to communicate with each other.

Secondly, the paradigm used in all existing PIR works requires the database to actively participate in a complex protocol in order to achieve privacy and security. The protocol is complex both in terms of the *computation* necessary for each data base to perform in order to answer every question of the user, and in the less quantifiable *lack of simplicity*, compared to the classical lookup-the-query-and-answer approach.

In particular, in all of the existing solutions (whether using a multiple number of databases or a single one), each database does a computation which is at least linear in the size of the database in order to compute the necessary answer for *each* question of the user. In the single database case (computational privacy) the complexity of the computation is a function of both the size of n the database and the size of the security parameter underlying the cryptographic assumption made to ensure privacy. Specifically, in the single database solution of [9], the computation takes a linear number of multiplications in a group whose size depends on the security parameter chosen for the quadratic residuosity problem. For example, to achieve communication complexity $O(n^\epsilon)$ the security parameter is of size $O(n^{\epsilon^2})$ and the number of multiplications is $O(\frac{1}{\epsilon}n)$. Again, the overhead in computational complexity and lack of simplicity of existing schemes make it unlikely to be embraced as a solution by databases in practice.

1.2 New Approach

In this paper, we introduce a new paradigm for private information retrieval, which allows for information theoretic privacy without replication of data. Since it is not possible to use a single database and achieve sublinear communication complexity information theoretic results ([7]), we must still use a multiple database model. The crucial difference is that the multiple databases are *not* copies of the original principal database. Rather, they are auxiliary databases provided by, say, WWW servers for this purpose. These auxiliary servers contain strings each of which cannot be used on its own[1] to obtain any information about the original data. In particular, we will use servers containing *random* bits. Thus, an auxiliary server cannot obtain information about the data, sell it to others, or use it in any other way. They may be viewed as servers who are selling security services to ordinary current day databases. The principal database after engaging the services of some servers for the purpose of offering private and secure access to users, performs an initial setup computation with the auxiliary servers. The servers are then ready to assist users in retrieving information from the principal database efficiently and privately during the on line stage.

We show two types of results. In the first (section 3), the auxiliary servers change their content after the setup time. The content is changed in such a way that none of them individually can obtain any information about the principal data, but together they contain information which makes it possible for a user to retrieve data during the on line stage. In the second type of solution (section 4), the servers do not change at all during setup time, so that even jointly all servers do not contain any information about the data. Instead,

[1]or in extented solutions, in coalition with others (the number if which is a parameter determined by the principal database)

the principal database changes its content during setup time, in a way which allows the user to utilize the servers during the on line stage.

In addition to achieving extra database privacy resulting from no replication, our model allows more practical and convenient solutions than previous works. After performing the initial setup computation, which will be proportional to the size of the principal database, the principal database only needs to make $O(1)$ on line computation to answer questions of users, while all the extra computations required on-line for privacy are done by the servers. In particular, in our first type of results, the principal database does not need to participate in the on line stage **at all**, and in our second type of results, the on line stage of the principal database consists of giving the user a bit value from a location specified by the user, recording this bit value and sending a string of bits from a fixed location –no computations involved– whose length is bounded by the communication complexity of the protocol.

Finally, our model seems more reasonable from the user's point of view as well. Previously, the user had to trust the different replicated databases, all of whom held the same data and were thus related to the same principal database, not to communicate with each other. Now the user has to trust the servers, who are not related to any single owner, whose sole functionality and livelihood is to provide privacy, who are external to the database, and may be chosen independently of it. In fact, one may alternatively view the user as the one selecting which auxiliary services he trusts to handle his queries in a private manner, rather than the database.

1.3 Our Results

We define two new kinds of privacy for the database in this setting (formal definitions are in the next section), *independence*, and *total independence*.

Independence informally means that no server can get any information about the original data of the principal database. Thus, the real data is distributed among all the servers in a private way, so that no single one (this can be generalized for any coalition of upto t holders) gets any information about it.

Total independence informally means that even *all* the auxiliary servers jointly do not have any information about the original data, and could all be prepared in advance, without any dependence on the data in the principal database. Thus, the auxiliary servers can be used by any (or multiple) principal database who buys this service.

Clearly, total independence implies independence. Indeed the solutions we propose to address the latter are simpler than the ones to address the former.

We provide general reductions, starting from any private information retrieval scheme, and achieving these new privacy requirements, while maintaining other privacy properties of the underlying starting scheme (namely user privacy and database privacy). Specifically, given any private information retrieval scheme S which requires k copies of the database, and whose communication complexity is $C(S)$, we provide schemes achieving the following (information theoretic) privacy properties (note that all terms used below are defined in section 2).

1. Adding independence and maintaining the other privacy properties of S, using $2k$ servers and communication complexity $O(C(S))$.

2. Adding t-independence (for any $t \geq 1$) and maintaining the other privacy properties of S, using $(t+1)k$ servers and communication complexity $(t+1)C(S)$.

3. Adding total independence and adding database privacy, maintaining user privacy upto equality between repeated queries[2], using $2k$ servers and the principal database, with $O(C(S)\log n)$ communication complexity.

4. Adding total independence and maintaining the other privacy properties of S (in particular complete user privacy), using $2k$ servers and the principal database, with at most $O(m + C(S)\log n)$ communication complexity, where the servers and the database need to engage in a re-setup after every m queries. For how m is chosen see remark below and section 4.4.

[2]namely, the only information that the database can compute is whether this query has been made before.

On Line Computation Time: Moreover, if the on-line computation time of the original private information retrieval scheme S was $T(S)$ for every database for every answer required by the user, then in the first two results the principal database need not participate at all whereas the auxiliary servers still require $T(S)$ computation per answer. In the second two results, the principal database does $O(1)$ computation per answer, whereas again the auxiliary servers still require $T(S)$ computation per answer.

Set Up Stage: For the first two results, the setup stage involves $O(n)$ computation and communication, and the number of servers who change their content during the setup stage is k (same as the number of copies in the original S). For the second two results, the setup stage involves $O(n \log n)$ computation and communication, and none of the servers changes their content during setup stage. Thus, all servers involved may be prepared ahead of time.

Tradeoffs for Total Independence Schemes: The third result above achieves user privacy upto equality, namely the database can detect repeated queries, but cannot gain any other information about the queries. In the fourth result total independence is achieved preserving complete user privacy. The price for eliminating detection of repeated queries is that re-setup has to be performed every m queries. The value of m, the frequency of reinitialization, is a parameter chosen to optimally trade off the frequency and the communication complexity. A suitable choice for existing schemes is to choose $m = C(S) = n^{\epsilon}$ the size of the communication complexity for a single query, so that the over all communication complexity does not increase by more than a logarithmic factor, and yet a sublinear number of queries can be made before reinitialization. Choosing between the third result and the fourth result should depend on what is more importnat for the specific application: preventing the database from detecting equal questions, or avoiding reinitialization.

To summarize, our work is the first that raises the data replication problem and solves it with information theoretic security, sublinear communication complexity, and one round of communication.

We do so by introducing the model of distinguishing between the principal database and the auxiliary databases, which can be implemented using random servers. This random server model for PIR is utilized to ensure private information retrieval without distributing any information about the database content, and without increasing the on-line computation cost of the principal database above its normal operation cost.

1.4 Related Work

PIR was originally introduced by [7], who were only concerned with protecting the user's (information theoretic) privacy. In particular, for a constant number k of database copies, [7] with further improvement in [3] (for the case $k > 2$), achieve information theoretic user security with communication complexity of $n^{\frac{1}{2k-1}}$, where n is the length of the data (in bits).

Recently, [8] extended PIR to SPIR (*Symmetrically private information retrieval*), where the privacy of the data (with respect to the user) is considered as well. They use a model where the multiple databases may use some shared randomness, to achieve reductions from PIR to SPIR, paying a multiplicative logarithmic factor in communication complexity.

The work in [6] considers computational privacy for the user, and achieves a 2 database scheme with communication complexity of n^{ϵ} for any $\epsilon > 0$, based on the existence of one way functions. As discussed earlier [9] relies on a stronger computational assumption – the quadratic residuosity problem – to achieve a 1-database PIR scheme with computational privacy and communication complexity of n^{ϵ} for any $\epsilon > 0$.

The work in [10] generalizes PIR for private information storage, where a user can privately read and write into the database. This model differs from ours, since in our model users do not have write access into the database. Still, some connection between the two models can be made, since one might consider a storage model where the first n operations are restricted to be private write (performed by the principal database owner), and all operations thereafter are restricted to be private reads (by users). This results in a model compatible to our model of independence (although this approach cannot lead to total independence). We note that [10] independently[3] use a basic scheme which is essentially the same as our basic RDB scheme of section 3.1. However, they use the scheme in their modified model (where users have write access), and with a different goal in mind, namely that of allowing users to privately read and write into the databases. None of the above PIR and SPIR works consider the data replication problem.

[3]our results of section 3 were actually done previously to the publication of [10]

Organization

Section 2 introduces the relevant definitions and notation used. In section 3 we describe our schemes achieving independence, and in section 4 we describe schemes achieving total independence.

2 Notation and Definitions

The Information Retrieval Model: The *data string* is a string of n bits denoted by $x = x_1, \ldots, x_n$. The user's *query* is an index $i \in \{1, \ldots, n\}$, which is the location of the bit the user is trying to retrieve. The *principal database* is denoted by D, and the *auxiliary servers* are usually denoted by $D_1, \ldots, D_k$.

An *information retrieval* scheme is a protocol consisting of a *setup stage* and an *on-line* stage. During the setup stage, the auxiliary servers are chosen, and possibly some other setup computation is performed. During the on-line stage, a user interacts with the servers and possibly also with the principal database in order to obtain his query. At the end of the interaction, the user should have the bit x_i. In all our schemes, the on line stage consists of a single round.

The Random Servers: During the setup stage, some of the auxiliary servers may be required to change their content, namely to store new information to be used during the on-line stage. Such servers are called *dynamic*. Servers who do not need to change their information are called *static*. One of the parameters for an information retrieval scheme is how many servers of each kind are required. Clearly, static servers are preferable, since they are simpler and more efficient to maintain, do not require separate space for each database if used by multiple databases, and their content may be prepared in advance, without any relation to a particular database, which is better both from the complexity and from the privacy point of view. Indeed, our strongest definition of privacy (total independence, below) will require that *all* servers involved are static. In contrast, previously existing schemes use replications of the database, which may be viewed as dynamic servers storing the data.

We require that all servers are separate, in the sense that they are not allowed to communicate with each other. We also address the case where upto t of the servers are faulty and *do* communicate with each other.

Notions of Privacy: We define the following privacy properties for an information retrieval scheme.
User privacy [7]: No single database (or server) can get any information about the user's query i from the on-line stage. That is, all the communication seen by a single database is identically distributed for every possible query. This definition can be extended to *user t-privacy*, where all communication seen by any coalitions of upto t databases (servers) is identically distributed for every possible query.
Database privacy [8]: The user cannot get any information about the data string other than its value in a single location. That is, all the communication seen by the user in the on-line stage is dependent on a single physical bit x_i (so it is identically distributed for any string x' s.t. $x_i = x'_i$).
Independence: No auxiliary server has any information about the data string x. That is, the content of the auxiliary server is identically distributed for any data string x. This definition can be extended to *t-independence*, where no coalition of upto t servers has any information about x (thus, independence is the special case of 1-independence).
Total independence: All the auxiliary servers jointly have no information about the data string x. That is, they are completely independent of the principal data, and thus may all be chosen in advance.

In all the above definitions, information theoretic privacy may be relaxed to *computational* privacy, requiring indistinguishablity instead of identical distribution.

Protocols: A private information retrieval (PIR) scheme is one that achieves user privacy, and a *symmetrically private information retrieval (SPIR)* scheme is one that achieves user privacy and database privacy.

3 Achieving Independence: The RDB Scheme

In this section we present a scheme for independent PIR (or independent SPIR), where each of the auxiliary servers do not obtain any information about the principal data. Specifically, in this section we prove the following theorem.

Theorem 1 *Given any information retrieval scheme S which requires k copies of the database and communication complexity $C(S)$, and for every $t \geq 1$, there exists an information retrieval scheme achieving t-independence and maintaining the other privacy properties (user privacy and data privacy) of S. The t-independent scheme requires $(t+1)C(S)$ communication complexity and $(t+1)k$ servers, out of which only k need to be dynamic.*

An immediate useful corollary follows, setting $t = 1$:

Corollary 1 *Given any information retrieval scheme S which requires k copies of the database, there exists an information retrieval scheme achieving independence and maintaining the other privacy properties of S, which requires a factor of 2 in communication complexity, and uses k dynamic servers and k static ones.*

The basic version of our reduction (the RDB scheme) is described in section 3.1. In section 3.2 we present another version, possessing some appealing extra properties for security and simplicity. We note however, that the starting point for the second reduction is any information retrieval scheme which has a linear reconstruction function. This is usually the case in existing PIR schemes (cf. [7, 3]). Finally, in section 3.3 we prove that the RDB construction satisfies theorem 1.

Another benefit of our scheme is that it does not require the participation of the primary database D after the setup stage. Instead, the servers deal with all the on-line queries and computations. Even though D is not there for the on line stage, he is guaranteed that none of the servers who are talking to users on his behalf has any information about his data x.

3.1 The Basic RDB Scheme

In the basic RDB (random data base) scheme, instead of replicating the principal database as in the underlying scheme, every copy is replaced by $t+1$ random servers whose contents xor to the contents of the principal database. The idea behind this replacement is that if these $t+1$ databases are chosen uniformly at random with this property, then any coalition of t of them are simply a random string, independent of the actual principal data string. Therefore, t-independence is achieved . We proceed with the details of the basic reduction. The communication complexity and privacy properties of this scheme will be proved in section 3.3.

Let the underlying scheme S be a PIR scheme with k copies of the database.

Setup Stage

The principal database D chooses uniformly at random $t+1$ auxiliary servers $D_1, \ldots, D_{t+1}$ in $\{0,1\}^n$, such that for every $1 \leq j \leq n$, $D_1(j) \oplus \ldots \oplus D_{t+1}(j) = D(j) = x_j$ i.e., the xor of all the servers is the original data string x. This is done by first choosing $D_1, \ldots, D_t$ containing completely random strings (static servers), and then preparing D_{t+1} (dynamic) with the appropriate content, namely $D_{t+1} = D_1 \oplus \ldots \oplus D_t \oplus x$. Each of these servers is then replicated k times, for a total of $k(t+1)$ servers.

Thus, at the end of the setup stage, the auxiliary databases are

$$D_1^1, \ldots, D_1^k, \; \ldots \; , D_{t+1}^1, \ldots, D_{t+1}^k$$

where $D_s^1 = D_s^2 = \ldots = D_s^k$ for every s $(1 \leq s \leq t+1)$, and where $D_1^r \oplus D_2^r \oplus \ldots \oplus D_{t+1}^r = x$.

On Line Stage

During the on line stage, the user executes the underlying scheme S $t+1$ times, each time with a different set of k databases. The first execution is with the k copies of D_1, which results in the user obtaining $D_1(i)$. The second execution is with the k copies of D_2, resulting in the retrieval of $D_2(i)$, and so on. Finally, the user xors all the $t+1$ values he retrieved, $D_1(i) \oplus \ldots \oplus D_{t+1}(i) = D(i) = x_i$ in order to obtain his desired value x_i.

Note that the user can perform all those $t+1$ executions of S in parallel. Also, the user may either perform all these parallel executions independently, or simply use exactly the same questions in all of them.

Our proofs will cover both these variants, but we prefer the latter since it simplifies the protocol of user-privacy against coalitions. However, in the most general case, if S is a multi round scheme with adaptive questions, we must use the first strategy of independent executions.

Remarks

Note that out of the $k(t+1)$ servers, all but k are static servers which can be prepared ahead of time, whereas the other k (copies of D_{t+1}) are dynamic.

Since servers should not know each other's contents, the strings of the k dynamic servers may be computed during the setup stage by the principal D (or some other trusted party), without leaking information to other servers. If we also wish to hide the auxiliary servers' contents from D, we can use a simple protocol that is a variation of a protocol proposed by Rabin for summing secrets, at the end of which D learns $D_{t+1} = D_1 \oplus \ldots D_t \oplus x$, but no other information, and the servers do not learn any new information.

Another thing to note is the fact that our scheme uses replication of the random servers. At first glance, this may seem to contradict our goal of solving the data replication problem. However, in contrast to replicating the original database, replicating random servers does not pose any threat to the original data string which we are trying to protect. Thus, we manage to separate the user privacy, which requires replication, from the database privacy, which requires not to replicate the data. Still, in the next section we describe a version in which there is no replication, not even of the auxiliary servers, and which subsequently provides a higher level of privacy, as discussed below.

3.2 The RDB Scheme

While the basic scheme does achieve t-independence (as no coalition of t servers has any information about x), some of the servers there are replications of each other, and thus there is not even pair-wise independence among the servers themselves.

Here, we propose an improvement to the basic scheme, in which a higher level of independence among the random servers is achieved. Specifically, we achieve t-independence among the servers, namely every combination of t servers are independent of each other (in particular, there is no replication of the servers).[4] Another benefit of this scheme over the basic one is that, while t is still the maximal size of coalition that the principal database is secure against, it is also secure against many other specific combinations of larger coalitions. This protocol works provided that the underlying PIR scheme has a linear reconstruction function (see 3.3), a quite general requirement that is satisfied by the main currently known PIR schemes. For simplicity, we assume that S consists of a single round of interaction. The protocol is easily generalized for an S with an arbitrary number of rounds.

Setup Stage

Recall that in the basic version, we created $t+1$ servers and replicated each of them k times, thereby creating $t+1$ sets, each of which consist of k identical servers. In this protocol, the k servers in every set will be independent random strings, instead of replications. Specifically, the principal D chooses *uniformly at random* $k(t+1)$ servers $D_1^1, \ldots, D_{t+1}^1, \ldots, D_1^k, \ldots, D_{t+1}^k$ with the property that $D_1^r \oplus \ldots \oplus D_{t+1}^r = x$ for every $1 \le r \le k$.

On-Line Stage

During the on line stage, the user sends his queries (according to the underlying S) to each of the servers, where $D_1^r, \ldots, D_{t+1}^r$ correspond to the r-th copy of the database in the underlying scheme S. After receiving the answers from all the $k(t+1)$ servers, the user xors the answers of $D_1^r, \ldots, D_{t+1}^r$ for each r to obtain the answer of the r-th copy in S, and combines these answers as in S to obtain his final value x_i.

The difference between this version and the basic version, is the following. In the basic scheme, the user first runs S to completion with each of the $t+1$ sets of servers (for example one set is $D_1^1, \ldots D_k^1$) giving the user $t+1$ values that enable him to xor them all together and obtain the value of the primary database. In

[4]Moreover, if we assume that the original data x is randomly distributed, then the servers are $2t+1$ independent.

contrast, here the user first combines answers by xoring values he received (for example from $D_1^1, \ldots D_{t+1}^1$) in the middle of the S protocol, which gives the user the intended answer of each copy of the database, and only then combines the answers as needed in S.

Thus, to succeed in this version, the underlying S must have the following closeness property under xor: *If $f_r(x, q)$ is the function used by the r-th copy of the database in S to answer the user's query q with the data string x, and given $y_1, \ldots, y_m$, then $f_r(y_1, q) \oplus \ldots \oplus f_r(y_m, q) = f_r(y_1 \oplus \ldots \oplus y_m, q)$.* This may be generalized to any underlying scheme with a linear reconstruction function. This requirement is very general, and is usually satisfied by existing PIR protocols (for example, protocols based on xoring subsets of locations in the data string, such as [7, 3], the best PIR schemes known to date).

3.3 Analysis of the RDB Scheme: Proof of Theorem 1

We now analyse the RDB scheme in terms of complexity, correctness, and privacy, to show that it satisfies the bounds given in theorem 1.

The RDB scheme requires a multiplicative factor of $(t+1)$ in communication complexity over the underlying scheme S, since S is simply executed $t+1$ times. Typically, t is a constant $t \geq 1$, which means the communication complexity of RDB is $O(C(S))$, where $C(S)$ is the communication complexity of S. The number of dynamic servers required is the same as the number of databases required in S, since all the tuples $D_1, \ldots, D_t$ can be prepared in advance, and then they can be xored with the original data to produce D_{t+1}. Thus, one dynamic server is needed per one copy of the principal data in the underlying S.

It is not hard to check that the scheme gives the user the correct value x_i, because of the way the servers were chosen, and from the correctness of S.

User privacy properties carry from S, namely if S was user-l-private (i.e. user private against coalitions of upto l databases), then so is the corresponding RDB scheme (where user privacy is protected from any coalition of l servers). This is clear for coalitions involving servers from the same set $D_s^1, \ldots, D_s^k$ for some s, since the user simply runs S with the set. This argument immediately extends to arbitrary coalitions if the user sends exactly the same questions in all sets (i.e. in every execution of S).[5] In the case of parallel independent executions and a multi round adaptive S, a little more care is needed to show that the view of any coalition is independent of i, using the l-user-privacy of S inside sets, and the independence of the executions across sets.

Database privacy of S also implies database privacy of the corresponding RDB scheme, as follows. If in the r-th parallel execution of S the user gets at most one bit of D_r, then altogether the user gets at most $(t+1)$ bits, one from each of $D_1, \ldots, D_{t+1}$. Since these are chosen uniformly at random among all strings that xor to x, it follows that if the $(t+1)$ bits are from the same location i in all servers, they are distributed uniformly over all $(t+1)$-tuples that xor to x_i, and otherwise the $(t+1)$ bits are distributed randomly among all possible tuples. In any case, the user's view depends on at most one physical bit of x, and database privacy is maintained.

Finally, the RDB scheme achieves t-independence since any coalition of up to t servers contains only t or less of the servers in $D_1^r, \ldots, D_{t+1}^r$, and thus (from the way the auxiliary databases were defined), the coalition consists of a string uniformly distributed over all strings of appropriate length, independent of x.

4 Achieving Total Independence: The Oblivious Data Scheme

In this section we present two schemes for totally independent PIR (or totally independent SPIR), where *all* auxiliary servers jointly are independent of the principal database, and are all static servers, which do not change their content during the setup stage.

Overview of Results

We first describe a basic scheme, which achieves total independence, as well as database privacy, but has the drawback that the user privacy is only "almost" maintained, with one exception: in repeated executions of

[5] This strategy is always possible in our basic version of section 3.1 unless S is a multi round adaptive scheme, and always possible in the improved version of section 3.2.

the basic scheme, the database can tell whether the questions in different executions correspond to the same query or not. We prove that no other information about the content of the queries or the relations among them is revealed to the database. We call this *user privacy upto equality between repeated queries.* Thus, we prove the following theorem.

Theorem 2 *Given any PIR scheme S which requires k copies of the database and communication complexity $C(S)$, there exists a totally independent SPIR scheme, secure for database and secure for user upto equality between repeated queries, which uses $2k$ static servers, and requires communication complexity of at most $O(\log nC(S))$.*

The scheme is described in section 4.1, and in section 4.2 we prove that it satisfies the theorem. In section 4.3 we show how to generalized our scheme so that it can maintain user privacy not only against a single server, but also against a coalition of servers as large as the underlying scheme can handle.

Since the information of whether two users are asking the same question or not may in some applications be an important information that violates the user privacy, we present a final scheme in section 4.4, which completely hides all information about the user queries, even after multiple executions. This scheme maintains the privacy properties of the underlying scheme, namely it transforms a PIR scheme into totally independent PIR, and a SPIR scheme into totally independent SPIR. The price we pay for eliminating the equality leakage, is that the setup stage needs to be repeated every m queries, and an additive factor of m is added to the communication complexity, where m is a parameter to the scheme (see 4.4 for how to choose m). Thus, we prove the following theorem.

Theorem 3 *Given any information retrieval scheme S which requires k copies of the database and communication complexity $C(S)$, there exists a totally independent information retrieval scheme, maintaining the privacy properties (user privacy and database privacy) of S, which uses $2k$ independent servers, and requires communication complexity of at most $O(m + \log nC(S) + \log n)$, where m is the number of queries allowed before the system needs to be reinitialized.*

The following corollary is obtained by setting $m = n^{\epsilon} \log n$ in the above theorem, where n^{ϵ} is some polynomial equal to the communication complexity of the underlying PIR scheme (it is conjectured in [7] that all information theoretic PIR schemes must have communication complexity of at least $\Omega(n^{\epsilon})$ for some ϵ).

Corollary 2 *Given any information retrieval scheme S which requires k copies of the database and has communication complexity $C(S) = O(n^{\epsilon})$, there exists a totally independent information retrieval scheme, maintaining the privacy properties (user privacy and database privacy) of S, which uses $2k$ independent servers, requires communication complexity of $O(\log nC(S))$, and has to be reinitialized after every $O(n^{\epsilon} \log n)$ number of queries.*

It is not clear which of the two schemes – the one achieving privacy upto equality, or the one achieving full privacy but with periodic setups – is better. This depends on the particular needs of the application.

The Oblivious Data Idea

To achieve total independence, all auxiliary servers must be (jointly) independent of the data. On the other hand we must use a number of multiple servers in order to achieve information theoretic results that are efficient. To accommodate these two conflicting requirements we use the following idea. During the setup stage, the principal database and the auxiliary servers create a new *oblivious* string y which depends on the content of all of them. This string must be held by the principal database D (since all others cannot hold any data dependent on the principal database). Later, during the on line stage, the user interacts with the servers to obtain information about the relation between y and x. Knowing this information the user can simply ask D for the value of y in an appropriate location, whose relation to x he knows from communication with the servers, which enables him to compute x_i. We call y an "oblivious" data string, since it should be related to the data string x, yet in a way which is oblivious to its holder D, so that D cannot relate the user's query in y to any query in x, and therefore learns nothing about the user's interests from the user's query in y.

4.1 Basic Scheme

Let the underlying scheme S be a PIR scheme with k copies of the database.

Setup Stage

The (static) auxiliary servers are k servers consisting of a copy of R and k servers consisting of a copy of P, where R contains a random string $r \in \{0,1\}^n$, and P contains a random permutation $\pi : [1..n] \rightarrow [1..n]$. D,R, and P engage in a specific multi party computation, described below, at the end of which D obtains the oblivious data string

$$y = \pi(x \oplus r)$$

but no other information about r, π. Each server does not obtain any new information about x or about the other servers.

The multi party computation is done as follows: D chooses uniformly at random two strings x^1 and x^2 such that $x^1 \oplus x^2 = x$. Similarly, R chooses uniformly at random r^1, r^2 such that $r^1 \oplus r^2 = r$. P chooses uniformly at random π^1, π^2 such that $\pi^1 \circ \pi^2 = \pi$, where $\circ$ is the composition operator (that is, $\pi^2(\pi^1(\cdot)) = \pi(\cdot)$). The following information is then sent between the parties on secure channels:

$D \rightarrow R$ **:** x^1

$D \rightarrow P$ **:** x^2

$R \rightarrow P$ **:** r^2

$P \rightarrow R$ **:** π^1

$R \rightarrow D$ **:** $v = \pi^1(r^1 \oplus x^1)$

$P \rightarrow D$ **:** π^2, $u = \pi(r^2 \oplus x^2)$

D can now compute $y = \pi^2(v) \oplus u = \pi(r^1 \oplus x^1) \oplus \pi(r^2 \oplus x^2) = \pi(r \oplus x)$ ("the oblivious string"). R and P discard all communication sent to them during the setup stage, and need to maintain only their original independent content, r and π respectively.

At the end of the setup stage the principal database D has two strings: x which is the original data string, and also y which is the oblivious data. The auxiliary servers contain the same strings as before the setup stage, and do not change or add to their content.

On Line Stage

In the on line stage the user interacts with the servers and the principal database in order to obtain his query. He runs S (the underlying PIR scheme) with the copies of P to obtain $j = \pi(i)$, and independently runs S with the copies of R to obtain r_i. Recall that r_i is the random bit with which the user's desired data bit was masked, and that j is the location of the masked bit in the oblivious data string. Finally, the user queries D for the value at the j-th location y_j. This is done by simply sending j to D on the clear, and receiving the corresponding bit y_j back. To reconstruct his desired bit, the user computes $y_j \oplus r_i = [\pi(x \oplus r)]_j \oplus r_i = (x \oplus r)_i \oplus r_i = x_i$.

Note that intuitively (formal proofs are in the next section), D cannot get any information about the user's query because of the obliviousness of y, but R and P cannot obtain any information about i because the user communicates with them using S.

Note also, that the computation complexity for the principal database here is minimal – $O(1)$. In fact, the only thing required from D is to send to the user a single bit from the specified location.

4.2 Analysis of the Basic Oblivious Scheme: Proof of Theorem 2

We now analyse the oblivious scheme in terms of complexity, privacy, and correctness, to show that it satisfies the bounds given in theorem 2.

It is not hard to verify that the setup stage computation is correct, namely that indeed $y = \pi(x \oplus r)$. Now the **correctness** of the scheme follows from the correctness of the underlying S: since the user uses S to obtain r_i and $j = \pi(i)$, it follows that $y_j = x_i \oplus r_i$ and thus $x_i = y_j \oplus r_i$.

The **communication complexity** of the scheme is $C(S,1) + C(S,\log n) + \log n + 1$, where $C(S,l)$ is the communication complexity that the underlying scheme S requires to retrieve a block of l bits. Note that $C(S,\log n) \leq \log nC(S,1)$, since the block may be retrieved bit by bit. Thus, the communication complexity we achieve is at most $O(\log nC(S,1){+}\log n)$, namely a logarithmic factor over S in communication complexity. This can be further decreased when methods for block retrieval are more efficient than bit by bit are available (cf. [7]). The communication complexity of the setup stage is $O(n \log n)$, which is a factor of $\log n$ over the $O(n)$ of existing PIR algorithms, where the database has to be replicated.

Total independence is clearly achieved, since the auxiliary servers may all be predetermined in advance, and do not change their content after setup stage.

Database Privacy is also guaranteed by our scheme, even if the underlying S is not database private. This is because, no matter what information the user obtains about P and R, this information is completely independent of the data x. The user gets only a single bit of information which is related to x, and this is the bit y_j at a certain location j of the user's choice. Note that since $y = \pi(x \oplus r)$, the bit y_j depends only on a single physical bit of x.

User Privacy is maintained in a single execution of the scheme, and in multiple executions upto equality between queries, as we prove below. However, if in multiple executions two users are interested in the same query i, the database will receive the same query $j = \pi(i)$, and will thus know that the two queries are the same. This problem will be fixed in section 4.4. We proceed in proving user privacy upto equality with respect to the database D. Note that user privacy with respect to P or R databases follows directly from the user privacy of the underlying PIR scheme.

Consider an arbitrary sequence $(i_1, \ldots, i_m)$ of query indices which are all distinct. We will prove that the distribution of D's view after the setup stage and m execution of the on-line stage with these indices is independent of $i_1, \ldots, i_m$.

Proposition 1 *Let V_{setup} be the view of D after performing the setup stage. For every permutation $\hat{\pi} : [1..n] \rightarrow [1..n]$, $Prob[\pi = \hat{\pi} \,|\, V_{\text{setup}}] = \frac{1}{n!}$ where probability is taken over all the random setup choices π, π^1, r, r^1. In particular, D does not get any information about the permutation π from the setup stage.*

Proof: D's view consists of $V_{\text{setup}} = [x^1, x^2, \pi^2, v = \pi^1(r^1 \oplus x^1), u = \pi(r^2 \oplus x^2)]$. Given this view, every choice for a permutation π fixes the choices of π^1, r, r^1. That is, every $\hat{\pi}$ corresponds to a single choice $(\hat{\pi}, \hat{\pi}^1, \hat{r}, \hat{r}^1)$ which generates the given view. Since all these random choices of the setup stage are done uniformly and independently of each other, each such choice is equally likely. Thus, the probability of a particular $\hat{\pi}$ is $\frac{1}{n!}$. □

Proposition 2 (User Privacy) *Let $(i_1, \ldots, i_m)$ be a tuple of distinct indices in $[1..n]$. Let V_{setup} be the view of D after the setup stage, and $V(i_1, \ldots, i_m)$ be the view of D for m executions of the on line stage with queries $i_1, \ldots, i_m$. Then for every tuple $(j_1, \ldots, j_m)$ of distinct indices, and for every setup view V_{setup},*

$$Prob[V(i_1, \ldots, i_m) = (j_1, \ldots, j_m) \,|\, V_{\text{setup}}] = \frac{(n-m)!}{n!}$$

where probability is taken over all the random choices π, π^1, r, r^1. In particular, the view is independent of the user queries.

Proof: Since after the setup stage D did not get any information about π, as proved in proposition 1, every π is equally likely, and thus the given tuple $(j_1, \ldots, j_m)$ may correspond to any original queries tuple $(i_1, \ldots, i_m)$ with equal probability. A formal derivation follows. Denote by $\Pi = \{\pi \,|\, \pi(i_1, \ldots, i_m) {=} (j_1, \ldots, j_m)\}$.

$$Prob[V(i_1, \ldots, i_m) {=} (j_1, \ldots, j_m) \,|\, V_{\text{setup}}] =$$

$$= \sum_{\pi} Prob[\pi \mid V_{\text{setup}}] Prob[V(i_1, \ldots, i_m) = (j_1, \ldots, j_m) \mid V_{\text{setup}}, \pi] =$$

$$= \sum_{\pi \in \Pi} Prob[\pi \mid V_{\text{setup}}] = \sum_{\pi \in \Pi} \frac{1}{n!} = \frac{(n-m)!}{n!}$$

$\square$

We proved that any two tuples of distinct queries $(i_1, \ldots, i_m)$ and $(i'_1, \ldots, i'_m)$ induce the same distribution over the communication $(j_1, \ldots, j_m)$ sent to D. Therefore, the basic scheme is user private upto equality.

4.3 Adding User Privacy Against Coalitions

The basic scheme, described above, provides user privacy against each individual server (or the principal database), namely the view of a single server (database) does not give him any information about the identity of the user's query. Here, we describe how to extend the basic scheme in order to achieve user privacy against a coalition of servers (or a coalition of the principal database with other servers).

Obviously, if the underlying S is not secure against coalitions, we cannot hope that our scheme, which relies on S's security for the user with respect to the servers, will be. Thus, we want to generalize the basic scheme so that if S is secure against coalitions of size l ("user-l-private"), then so is the new scheme.

First we observe that in the basic scheme a coalition of D and one of the P servers may find information about the user's query i, because P knows the permutation π, and D knows $j = \pi(i)$. A coalition of D with one of the R's may also extract some information about the answer to the user's query (depending on the relations between r and x), which in turn reveals some information about the query i. Therefore, we propose that instead of using one R and P, l of them will be used: $R_1, \ldots, R_l$ containing random strings $r_1, \ldots, r_l$, and $P_1, \ldots, P_l$ containing random permutations $\pi_1, \ldots, \pi_l$. We define the string $r = r_1 \oplus \ldots \oplus r_l$ and the permutation $\pi = \pi_1 \circ \ldots \circ \pi_l$. The multi party computation of the setup stage is extended so that the database can obtain the oblivious string $y = \pi(x \oplus r)$ from the auxiliary servers. The on line stage is changed as follows. The user runs S with each of $R_1, \ldots R_l$ with the index i and computes $r(i) = r_1(i) \oplus \ldots \oplus r_l(i)$. The user also runs S with P_1 and index i to obtain $j_1 = \pi_1(i)$, with P_2 with index j_1 to obtain $j_2 = \pi_2(j_1) = \pi_2(\pi_1(i))$ and so on, until the last stage where he gets $j_l = \pi_l(j_{l-1}) = \pi(i) = j$. As in the basic scheme, the user sends this last j to D on the clear, and gets y_j, and can now compute his desired bit $x_i = y_j \oplus r_i$.

4.4 Eliminating Detection of Repeated Queries: Proof of Theorem 3

In order for the oblivious database scheme to be complete we need to generalize the basic scheme so that it will guarantee user privacy completely and not only upto equality between repeated executions. To extend this to full privacy we need to ensure that no two executions will ever ask for the same location j, even if the two users are interested in the same index i. To achieve this, we use a buffer of some size m, in which all (question,answer) pairs (j, y_j) that have been asked so far are recorded. This buffer is held by the principal database. The on line stage is changed as follows: the user who is interested in index $\hat{i}$, first obtains the corresponding $r_{\hat{i}}, \hat{j}$ from R and P as before. Then, the user scans the buffer. If the pair $(\hat{j}, y_{\hat{j}})$ is not there, the user asks D for $y_{\hat{j}}$ (as in the basic scheme). If the desired pair is there, the user asks D for y_j in some random location j not appearing in the buffer so far. In any case, the pair (j, y_j) which was asked from D is added to the buffer.

Clearly, a buffer size m results in an additive factor of m in the communication complexity over the basic scheme. On the other hand, after m executions the buffer is full, and the system has to be reinitialized (i.e. the setup stage is repeated, with new R and P). Thus, we want to choose m as big as possible without increasing the communication complexity much. A suitable choice for existing schemes will therefore be $m = n^{\epsilon}$ the same as the communication complexity of the underlying S (or $m = n^{\epsilon} \log n$). This only increases communication complexity by a constant factor, and still allows for polynomial number of executions before reinitialization is needed. We note that in many practical situations, reinitialization after m steps is likely to be needed anyway, as the database itself changes and needs to be updated.

The database privacy in this case depends on the underlying scheme S: If S is database private (a SPIR scheme), then so is our scheme. This is because, when running S with R, the user gets only a single physical bit r_i. Now, no matter how much information the user obtains about y (either from direct queries or from scanning the buffer), the data x is masked by r (namely $y = \pi(x \oplus r)$), and thus the user may only obtain information depending on a single physical bit of x.

The other privacy and correctness properties are easy to verify, similarly to the basic scheme proofs of section 4.2.

5 Conclusion

In this paper we introduce a new model for PIR which allows us to obtain the first efficient solution achieving information theoretic privacy without database replication. Instead of replicating the data, the database engages the services of auxiliary random servers, providing privacy services for database access. The principal database can utilize the servers to ensure private information retrieval, without giving away its information, and without increasing on line computation cost. The database manager only needs to perform O(1) amount of computation to answer questions of users, while all the extra computations required on line for privacy are done by the auxiliary random servers who contain no informatin about the database. Our model is more reasonable from the user's point of view as well, since the servers are external to the database, and their sole functionality is to provide security services. In fact, since the servers may be chosen independently of the database, one may view the user as the one selecting whose services he wants to use.

References

[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. ICALP 97.

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

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

[8] Y. Gertner, Y. Ishai, E. Kushilevitz, T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. Manuscript

[9] E. Kushilevitz, R. Ostrovsky. Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. FOCS 1997

[10] R. Ostrovksy, V. Shoup. Private Information Storage. STOC 1997.

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