Efficient Distributed $\binom{n}{1}$ Oblivious Transfer

Yael Gertner[*] Tal Malkin[†]

MIT Lab of Computer Science

Cambridge, MA 02139

MIT-LCS-TR-714

April, 1997

Abstract

In this paper, we present a new application for $\binom{n}{1}$ oblivious transfer, which is an interactive protocol between two parties Alice and Bob, where Alice has n secrets and Bob has a query i. At the end of the protocol Bob has the ith secret and no other information about Alice's other secrets, while Alice does not get any information about i. This new application is the Secure Database Access problem. Motivated by this application, we propose an OT scheme which achieves low communication complexity and information theoretic security.

We use a distributed model for $\binom{n}{1}$ oblivious transfer, where Bob interacts with multiple "Alices". In this model, we base our scheme on *any* PIR scheme, which is a scheme where only the privacy of Bob is considered, and use it to construct an OT scheme, private for both parties, without paying too much in communication complexity. This results in the first sublinear information theoretic scheme for $\binom{n}{1}$ OT.

Further motivated by the application of $\binom{n}{1}$ OT for polynomial n, we raise the issue of repetition in $\binom{n}{1}$ OT, where both security and efficiency are important. We show that previous protocols for $\binom{n}{1}$ oblivious transfer fail in this setting.

[*]yael@theory.lcs.mit.edu

[†]tal@theory.lcs.mit.edu

1 Introduction

$\binom{n}{1}$ *oblivious transfer* (OT) is an interactive protocol between two parties Alice and Bob, where Alice has n secret bits and Bob has a query i. At the end of the protocol Bob has the ith secret and no other information about Alice's other secrets, while Alice does not get any information about i. Previously, Oblivious Transfer has been proven to be a very important cryptographic primitive for applications such as secret exchange, contract signing, and non-interactive zero knowledge proofs for NP, to name a few [17, 12, 15, 8]. Given the nature of these applications, the $\binom{n}{1}$ oblivious transfer protocol has always been used as a primitive within a larger two party or multi party computation, where n of a constant size was sufficient.

In this paper, we offer a new *direct* application of $\binom{n}{1}$ oblivious transfer in which the number of secrets n is *polynomial*. This application is the *Secure Database Access problem*, which involves a user who queries a database for the value in some location i, such that the database does not learn anything about the user's query i, and the user does not learn anything about the database except for the value in a single location i. This problem is equivalent to the $\binom{n}{1}$ OT problem.

The problem of secure database access, where security of both the user and database is considered, is a very natural problem, that arises in practice. 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 would like to keep the stock information private to itself, except for the particular stock that the user has paid for. Clearly, security of both should be maintained.

Having this application in mind, we are faced with a new problem in $\binom{n}{1}$ OT that non of the existing implementations address: reducing the cost of communication complexity, or the total amount of bits transferred between Alice and Bob, as n grows. Moreover, we want to achieve *information theoretical* security for both sides. In order to achieve these goals, we use a distributed model for $\binom{n}{1}$ OT, where the user interacts with multiple secret holders who do not communicate with each other.

Previous Work

A naive solution to the Secure Database Access problem would be to use an already existing $\binom{n}{1}$ OT protocol, such as [9, 14]. However, these protocols rely on cryptographic assumptions and their communication complexity is at least $\Omega(n, k)$ where k is a security parameter. In contrast, our goals are to achieves information theoretic results and a communication complexity which is sublinear in n.

Schemes that reduce the communication complexity were introduced for the Private Information Retrieval problem [11, 2], in which the user's query is protected by information theoretic security. This work achieved sublinear communication complexity by using a multi database model in which a constant number of databases rather than a single database are used. However, it does not achieve privacy for the database's information, since the user can get additional information about the database, other than the value in a single location. Here, we show how data privacy can be added to any PIR protocol without paying too much in communication complexity.

Another protocol called Instance Hiding [3, 4] allows for information theoretic security for both the user and the database, in a model where the database size n is exponential, and the number of databases needed is logarithmic. In contrast, here we consider n to be feasible (following the PIR model [11]), which allows us to achieve those results for a constant number of databases.

[7, 16] show that any two party protocol can be achieved in the two-prover IP model without cryptographic assumptions, which implies a distributed model for $\binom{2}{1}$ OT achieving information theoretic security. Although known reductions between $\binom{2}{1}$ OT and $\binom{n}{1}$ OT exist ([8, 12]), they cost a high price in communication complexity, and thus cannot be used to convert the [7] $\binom{2}{1}$ OT protocol into a sublinear $\binom{n}{1}$ OT protocol.

Our Contribution

• We show a direct application for $\binom{n}{1}$ OT – the Secure Database Access problem. This is the first application where n is polynomial. This motivates a new range of problems in $\binom{n}{1}$ OT, such as sublinear communication complexity, and repetitive OT (see below).

• We suggest a new model for OT – distributed $\binom{n}{1}$ OT, where n is polynomial, and there are multiple secret holders. This allows us to achieve the following properties:

- We show an efficient $\binom{n}{1}$ OT, using sublinear communication complexity. Specifically, starting from *any* private information retrieval protocol for constant k databases, we show a secure database access (equivalently, $\binom{n}{1}$ oblivious transfer) protocol for $k+2$ databases ("secret holders"), paying at most a logarithmic factor in communication complexity.

  For example, for $k = 2$, since currently the best known PIR protocol [2, 11] uses $O(n^{\frac{1}{3}})$ communication, our scheme uses $O(n^{1/3} \log n)$ communication. The same scheme can be used to achieve polylogarithmic communication complexity for a logarithmic number of databases. If we allow computational assumptions, our scheme can achieve $O(n^{\epsilon})$ communication complexity for any $\epsilon > 0$, using [10].

- Our scheme achieves an information theoretic OT which is not based on any cryptographic assumptions (which is impossible in the traditional non-distributed model).

• We also raise the question of repetition in $\binom{n}{1}$ oblivious transfer, namely we consider a scenario where multiple executions of $\binom{n}{1}$ oblivious transfer are necessary, using the same n secrets. We examine whether existing implementations for $\binom{n}{1}$ oblivious transfer allow for repetitive use maintaining security and efficiency. Surprisingly, the answer is negative.

Organization

In section 2 we give preliminaries and definitions of the already existing and our new model of $\binom{n}{1}$ OT. Then, in section 3, we present our implementation – the Random Pointer scheme – which guarantees information theoretic security for both parties while maintaining the communication complexity low. These properties are proven in section 4. In section 5, we outline possible generalizations. In section 6 we present the new open problem that deals with repetitive executions of $\binom{n}{1}$ OT, and show how existing $\binom{n}{1}$ OT implementations are not adequate for this purpose.

2 Preliminaries and Definitions

2.1 Oblivious Transfer

Oblivious transfer comes in various forms, including "standard" OT, $\binom{2}{1}$ OT, and $\binom{n}{1}$ OT. These variants are all equivalent, in the sense that reductions among them exist ([8, 12]). In this paper, we are interested in $\binom{n}{1}$ oblivious transfer, which is defined as follows.

$\binom{n}{1}$ **Oblivious Transfer:** This is an interactive protocol between two parties Alice and Bob. 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.

2.2 Distributed $\binom{n}{1}$ Oblivious Transfer

As described above, the model used in the traditional oblivious transfer consists of two parties, Alice and Bob. Alice has n secret bits and Bob has a query which is an index to one of those secrets. Implementations of traditional OT were shown using cryptographic assumptions (such as the existence of one way functions [14]), noisy channels [15, 16], or quantum computation [6]. The need to make some computational assumption is inherent in this model, because Alice has access to the complete transcript of the communication between her and Bob, and thus she can, information theoretically, determine exactly what Bob can infer about her data. Thus, this model does not allow us to implement an information theoretic protocol. We overcome this inherent problem by moving from the traditional model to a distributed one, as follows.

In the *distributed* OT model, the secret holder Alice is distributed into multiple holders who do not communicate with each other. More formally:

$\binom{n}{1}$ **Distributed Oblivious Transfer:** This is a protocol between k secret holders ("Alices") $A_1, \ldots, A_k$, holding n secret bits $S_1, \ldots, S_n$, and one user Bob, holding a selection index $i \in \{1, \ldots, n\}$. The protocol is run in two stages: the setup stage, and the online stage. After the initial setup stage, no two A_j, A_l, $j \neq l$ are allowed to communicate with each other.
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. $\forall j$, A_j learns nothing about the value of i.

This distributed model allows us to obtain the following properties:

Information theoretic OT Each Alice on her own, without communicating with the other Alices, receives a view which is completely independent of Bob's query i. Thus, no individual Alice can gain any information about i.
Note that this is impossible to achieve in the traditional OT model with a single Alice, since in this case Alice's view is the same as Bob's.

Sublinear communication complexity The total amount of bits exchanged between all the Alices and Bob for one query is sublinear in n.
All existing protocols in the traditional model fail to achieve this, because, since Bob's query is to remain secret, they are based on Alice sending to Bob information regarding *all* her secrets (in a way that will allow Bob to recover only one of them), and thus existing protocols require communication complexity which is at least linear in the number of secrets.

Note that by information theoretic OT we do not mean that the parties should be computationally unlimited in order to correctly execute the protocol, but rather that the security of the protocol is information theoretical. That is, polynomial computation power suffices to use the protocol, but even if the other side has unlimited computational power, she will not be able to extract more information than she was supposed to.

Within the distributed OT model, we implement a protocol which solves our goals mentioned above. This protocol uses private information retrieval (PIR) as a subprotocol. Since in this paper we present the problem in the context of oblivious transfer, where we have a secret holder (Alice) instead of a database as in the PIR scheme, for clarity we rename the PIR scheme to semi-oblivious transfer. In semi-oblivious transfer, we let go of the second condition in the definition of OT, allowing Bob to possibly get more information about the secrets other than S_i:

$\binom{n}{1}$ **Distributed Semi-Oblivious Transfer:** This is a protocol between k secret holders ("Alices") $A_1, \ldots, A_k$, holding n secret bits $S_1, \ldots, S_n$, and one user Bob, holding a selection index $i \in \{1, \ldots, n\}$. The protocol is run in two stages: the setup stage, and the online stage. After the initial setup stage, no two A_j, A_l, $j \neq l$ are allowed to communicate with each other.
At the end of the protocol, the following two conditions hold.

1. Bob learns the i'th secret S_i.
2. $\forall j$, A_j learns nothing about the value of i.

These definitions may be extended in the natural way to deal with longer secrets, consisting of l bits each.

Notation: The communication complexity required for a $\binom{n}{1}$ semi-oblivious transfer protocol for l-bit secrets using k secret holders, is denoted by $SOT_k(l, n)$.

2.3 Application: The Secure Database Access Problem

We describe here an application of $\binom{n}{1}$ oblivious transfer – the Secure Database Access problem, which is a direct application of $\binom{n}{1}$ OT for a polynomial n. In this problem, there is a user who wants to retrieve some information from a database. We assume the data is a string of n bits, and the user is interested in the i'th bit. The user wants to keep his interest i secret from the database, and the database does not want to give any additional information except one bit in a single location. At the end of the protocol, the user will have the i'th bit, but no other information about any other bit, and the database will have no information about i.

Clearly, secure database access (where security of both the user and the database is considered) is equivalent to $\binom{n}{1}$ oblivious transfer, and thus any solution for the latter will automatically translate into a solution for the former.

3 The Random Pointer Scheme

In this section we present a scheme that achieves sublinear communication complexity and information theoretic security in the distributed $\binom{n}{1}$ oblivious transfer model. Our scheme uses any semi-oblivious scheme of k secret holders to obtain a distributed OT scheme with $k+2$ secret holders, paying only a logarithmic factor in communication complexity.

3.1 Overview

We start by recalling that sublinear information-theoretical schemes for *semi-oblivious* transfer exist in a distributed model (using any private information retrieval scheme, such as [2, 11]). However, those schemes are only concerned with Bob's privacy, and not Alice's. That is, Bob can get more information about the secrets, in addition to just S_i. Thus, in order to achieve privacy for both parties, we must prevent Bob from getting this extra information.

We achieve this using the following idea: There are $k+2$ secret holders: $A_1, \ldots, A_k, R_1, R_2$. Those secret holders are not allowed to communicate amongst themselves. R_1 and R_2 each consist of a random string with an equal number of zeros and ones. $A_1, \ldots, A_k$ contain the original data and a copy of R_1 and R_2. During the final stage of the protocol Bob 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 Bob obtained by communicating with the A_is). Using the values of those pointers Bob can compute the value of his query

$$R_1(j) \oplus R_2(l) = S_i \tag{1}$$

The values of these pointers are chosen by A in such a way that a pair of pointers only gives information about at most one secret bit.

The rest of the interaction between Bob and $A_1, \ldots, A_k$ serves the purpose of allowing Bob to obtain an appropriate pair of indices (j, l) that satisfy (1), without revealing any information about his selection index i. This is done by running a distributed semi-oblivious transfer subprotocol in which $A_1, \ldots, A_k$ use n pairs of the form $(j_1, l_1), (j_2, l_2), \ldots, (j_n, l_n)$ for the n secrets, and i as the selection index of Bob.

Using this general paradigm, we need to carefully adjust the details of the protocol so that it indeed implements sublinear, information theoretical, distributed $\binom{n}{1}$ oblivious transfer.

In order for Bob to receive the correct secret S_i in (1), the pairs used as secrets in the subprotocol must satisfy

$$R_1(j_r) \oplus R_2(l_r) = S_r \quad \forall r \in \{1, \ldots, n\} \tag{2}$$

These secrets cannot be chosen deterministically, because (j_i, l_i) will be sent to R_1 and R_2 respectively in the clear by Bob, so it should not reveal any information about his interest i. Thus, $A_1, \ldots, A_k$ need to share some randomness (in our case, they share a few random permutations on n bits).

Before turning to describing the details of the protocol, let us summarize the intuition behind this idea. Since the subprotocol that we want to use (semi-oblivious transfer) leaks excess information about A's secrets, we run the subprotocol with secrets that will not contain any useful information for Bob. In our case, these are the pairs of locations of the form (j_r, l_r), which can be viewed as "pointers" to more useful information. These locations without the actual content of R_1, R_2 in these locations, give no information about the original secrets $S_1, \ldots, S_n$. However, these locations *together* with the content of R_1, R_2 in these locations, give the original secrets, as implied by (2). Since Bob is allowed to get only one value from each of R_1 and R_2, we can prove that he does not get any information about the secrets, except for a single secret S_i. In addition, privacy of Bob is still maintained, because he talks to $A_1, ..., A_k$ using SOT, and R_1 and R_2 each get a uniformly distributed location.

3.2 The Scheme

This protocol is an interaction between Bob, holding a selection index i, and distributed holders $A_1, \ldots, A_k$, R_1, R_2, where $A_1, \ldots, A_k$ hold n secret bits $S_1, \ldots, S_n$ and R_1, R_2 hold random bits (see below). It uses as a subprotocol, call it P, a semi-oblivious transfer scheme (equivalently, private information retrieval), for k distributed holders. P should actually be a semi-oblivious scheme that transfers secrets which are *strings*, rather than single bits. This can always be achieved by simply repeating a (single-bit) semi-oblivious scheme for every bit in the string, or by using a more efficient scheme, such as the one described in [11]. For efficiency reasons, we require the subprotocol to run in time sublinear in n.

Initial Setup for the Secret Holders At this stage we describe what contents each party gets.

- 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.
- $A_1, \ldots, A_k$ each have the secrets $S_1, \ldots, S_n$, the contents of R_1, R_2, and three random permutations $\pi_1, \pi_2^0, \pi_2^1 : \{1, \ldots, n\} \to \{1, \ldots, n\}$. (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 stress that R_1 and R_2 only need to contain a random string, and are not required to know the secrets of the protocol, or the random permutations which are shared by $A_1, \ldots, A_k$.

After the initial setup stage, once Bob steps into the picture, the secret holders are not allowed to communicate with each other.

$\binom{n}{1}$ **Oblivious Transfer Protocol (on-line stage)**

- Bob chooses three random shifts $s_1, s_2^0, s_2^1 \in_U \{1, \ldots, n\}$ and sends them to each of $A_1, \ldots, A_k$.
- $A_1, \ldots, A_k$ each compute three new permutations $\sigma_1, \sigma_2^0, \sigma_2^1$, which are π_1, π_2^0, π_2^1 shifted by s_1, s_2^0, s_2^1 respectively. That is, $\forall r \in \{1, \ldots, n\}$, $\sigma_1(r) = \pi_1(r) + s_1 \pmod n$, and similarly for σ_2^0 and σ_2^1.
- $A_1, \ldots, A_k$ each compute n pairs $(j_1, l_1), (j_2, l_2), \ldots, (j_n, l_n)$ from $\sigma_1, \sigma_2^0, \sigma_2^1, \{S_1 \ldots, S_n\}$, and the content of R_1, R_2, as follows:
  - $j_r = \sigma_1(r)$ for $r = 1, \ldots, n$, hence all the j's are chosen completely randomly.
  - l_r's $(r = 1, \ldots, n)$ are chosen randomly so that the contents in the j locations and the l locations will xor to the secret bits. To do that, start by letting $b = R_1(j_r) \oplus S_r$ and $m = \sigma_2^b(r)$. Note that in order to satisfy (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).
- $A_1, \ldots, A_k$ and Bob run the subprotocol P with $(j_1, l_1), (j_2, l_2), \ldots, (j_n, l_n)$ as the secrets, and i as the selection index of Bob. At the end of the subprotocol, Bob has the pair $(j, l) = (j_i, l_i)$.
- Bob sends j to R_1, and l to R_2.

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

The proofs for correctness, security, and efficiency properties of our protocol are presented in the next section.

4 Analysis of the Random Pointer Scheme

In this section we analyze the complexity and security of our protocol. In particular, we show that it achieves sublinear communication complexity, and that it satisfies the definition of distributed $\binom{n}{1}$ oblivious transfer, including correctness and information theoretic security for both parties.

Assumption: In the analysis, we consider any user Bob which may be malicious and deviate from the protocol. As for the secret holders, we first make the usual assumption that they *want* to send the secret to Bob, so that they won't send junk instead of the real secrets[1]. However, if we limit ourselves to this assumption only, then if all of $A_1, \ldots, A_k$ and one of the R_j collaborate during setup time, and deviate from the protocol during the online stage, then R_j can get information about Bob's query[2].

Thus, we need to make one of the following assumptions, in order to protect the privacy of Bob against the random holders R_1, R_2. Either assume that the secret holders are honest but curious, namely they follow the protocol, but may try to extract as much information as possible about the identity of Bob's query. Alternatively, we may make the assumption that the random holders (R_1, R_2) do not know the random permutations shared by $A_1, \ldots, A_k$. This assumption is satisfied if we require that R_1, R_2 do not get any communication from $A_1, \ldots, A_k$ during the setup stage. Under this assumption, all the secret holders may be malicious, and deviate from the protocol. This assumption is reasonable, since we can think of R_1, R_2 as auxiliary databases (consisting of a random string), provided by an independent source, such as a special server for this purpose (and they may be determined in advance, independent of the secrets, or chosen later, after the secret holders chose their permutations). Note that these random holders do not need to know anything about the secrets, and no communication from the secret holders to the random holders is required at any stage of the protocol.

4.1 Correctness and Obliviousness

Notation: Denote our scheme by *RP* (random pointer scheme). RP_P will denote our random pointer scheme when used with the underlying semi oblivious transfer protocol P.

The following three theorems establish the required properties to prove that our random pointer scheme satisfies the definitions of distributed oblivious transfer. Recall that the definition consists of three properties that must hold at the end of the execution: (1) Bob learns S_i for his selection index i *(correctness)*; (2) Bob gains no further information about the other secrets S_j for $j \neq i$ *(privacy of secret holders)*; and (3) $\forall j$, A_j learns nothing about the value of i *(privacy of recipient)*.

Correctness

Theorem 1 *If P is a semi-oblivious transfer scheme, then RP_P is correct, i.e if Bob follows the protocol with selection index i, the value he obtains at the last step is the secret S_i.*

Proof: By reduction from the correctness of P, after running P with $A_1, \ldots, A_k$, Bob receives the pair $(j, l) = (j_i, l_i)$ corresponding to his selection index i. From the way l_i was constructed, it is a location in which R_2 has the bit $b = R_1(j) \oplus S_i$. Thus, $R_1(j) \oplus R_2(l) = S_i$ and Bob receives the correct secret S_i. $\square$

[1] This is a common assumption in the OT model, and is quite natural, for example if we view the secret holders as a commercial database which sells data, and charges per query.

[2] For example, they could agree on a fixed permutations to use, ignoring Bob's shifts, and then when R_j receives the user's query he knows which location it corresponds to, according to the fixed permutation.

Privacy of secret holders

Theorem 2 (informal statement) *For any strategy Bob' (possibly cheating), if all holders follow the protocol, Bob' cannot get any information about more than one secret S_i of his choice.*

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

Let Bob' be any strategy for the recipient. Bob' runs a semi-oblivious subprotocol P with $A_1, ..., A_k$ and the secrets $(j_1, l_1), \ldots, (j_n, l_n)$, at the end of which he receives (j_i, l_i) and possibly additional information about these secrets which the subprotocol leaks. We assume a worst case in which Bob' receives the full information about all the secrets, 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, Bob' cannot obtain any information about the real secrets $S_1, \ldots, S_n$ other than a single secret S_i 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 Bob' who sends queries j, l to R_1, R_2 respectively. (This is the assumption mentioned above. In reality, the view of Bob' can be derived from $V(j, l)$, but is possibly much smaller). Note that an honest Bob should set $j = j_i, l = l_i$, but we allow a possibly cheating Bob', 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 D be the domain of all possible partial views V^-. Thus, $|D| = 2n!\binom{2n}{n}$. We will prove that the partial view V^- is uniformly distributed over D, and from this we will be able to prove that the distribution of the complete view V depends only on one secret.

In what follows, the notation $X \sim U[D]$ means that the random variable X is distributed uniformly over the domain D.

Theorem 2 *$\forall j, l$, the distribution of $V(j, l)$ may depend on at most one secret. More specifically, for any possible view $V(j_r, l_{r'}) \in D \times \{0, 1\}$,*

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

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 Bob case), then the view provides complete information about S_r (since $\epsilon = 1$, so $S_r = R_1(j_r) \oplus R_2(l_r)$), whereas if j, l do not correspond to such a pair, only partial information about $S_{r'}$ is provided (since there is a positive probability for both $S_{r'} = 0$ and $S_{r'} = 1$).

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

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 secrets except $S_{r'}$. 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: Since π_1 is a uniformly distributed permutation, so is $\sigma_1 = \pi_1 + s_1$, namely $(j_1, \ldots, j_n) = (\sigma_1(1), \ldots, \sigma_1(n)) = (\pi_1(1) + s_1, \ldots, \pi_1(n) + s_1)$ is uniformly distributed over all permutations on $\{1, \ldots, n\}$ (recall that addition here is modulo n).

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 *$\forall j$, $[l_1, \ldots, l_n \,|\, R_1(j), j_1, \ldots, j_n] \sim U$ 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 $\{b_r = R_1(r) \oplus S_r\}_{r=1}^n$. Then, for $r \in \{1, \ldots, n\}$, l_r is chosen to be the index of the m_r'th bit with value b_r in R_2, where $m_r = \sigma_2^{b_r}(r)$. Thus, for any particular sequence $l_1, \ldots, l_n$, $Prob[l_1, \ldots, l_n \,|\, R_1, R_1(j), j_1, \ldots, j_n] = Prob[\forall r : R_2(r) = b_r \wedge \sigma_2^{b_r}(r) = m_r$ if l_r is the m_r'th bit with value b_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 σ_2^0 and σ_2^1 are both uniformly distributed permutations (because $\sigma_2^b = \pi_2^b + s_2^b$). We have some number k of restrictions on the values of σ_2^0 and $n-k$ restrictions on the values of σ_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[3]. Thus for each sequence we have the same probability, and thus $[l_1, \ldots, l_n \,|\, R_1, R_1(j), j_1, \ldots, j_n] \sim U$ 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. $\Box$

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

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

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

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, $R_2(l_{r'}) = R_1(j_{r'}) \oplus S_{r'}$, and thus $R_2(l_{r'}) = 0 \iff R_1(j_{r'}) = S_{r'}$. Therefore,

$$Prob[R_2(l_{r'}) = 0 \,|\, R_1(j_r), j_1, \ldots, j_n, l_1, \ldots, l_n] =$$
$$= Prob[R_1(j_{r'}) = S_{r'} \,|\, R_1(j_r), j_1, \ldots, j_n, l_1, \ldots, l_n] = Prob[R_1(j_{r'}) = S_{r'} \,|\, R_1(j_r)]$$
$$= \begin{cases} \frac{\binom{n-2}{\frac{n}{2}}}{\binom{n-1}{\frac{n}{2}}} & \text{if } r \neq r' \text{and } S_{r'} = R_1[j_r] \\ \frac{\binom{n-2}{\frac{n}{2}-1}}{\binom{n-1}{\frac{n}{2}}} & \text{if } r \neq r' \text{and } S_{r'} = \overline{R_1[j_r]} \\ 1 & \text{if } r = r' \text{and } S_{r'} = R_1[j_r] \\ 0 & \text{if } r = r' \text{and } S_{r'} = \overline{R_1[j_r]} \end{cases}$$

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 $R_1[j_{r'}] = S_{r'}$ is $\binom{n-2}{\frac{n}{2}}$, if $S_{r'} = R_1[j_r]$, and $\binom{n-2}{\frac{n}{2}-1}$ otherwise.

[3] 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.

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

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

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 D. Now, from lemma 4 we have that

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

Combining these equations, we get

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

which completes the proof of the theorem. □

Privacy of recipient

We prove that the RP scheme is private for Bob, provided that the secret holders follow the protocol (honest but curious). As explained in the beginning of the section, this assumption can be removed if R_1, R_2 do not know the random permutations, and a similar proof will work for that case.

Theorem 3 *If P is a semi-oblivious transfer scheme, then RP_P is recipient-private, i.e. for any honest-but-curious strategies $A'_1, \ldots, A'_k, R'_1, R'_2$, if Bob follows the protocol for a selection index i, no secret holder can get any information about i.*

Proof: For A'_r, $1 \le r \le k$, Bob's communication with A'_r is identical to the communication in the underlying P, and therefore for these holders the theorem follows by reduction from the privacy of recipient for P.
For R'_1, the only communication R'_1 gets is the index $j = \sigma_1(i) = \pi_1(i) + s_1 (\text{mod } n)$. Since s_1 is a random shift uniformly distributed in $\{1, \ldots, n\}$, j is a uniformly distributed index in $\{1, \ldots, n\}$, independent of i. Thus, R'_1 cannot get any information about i.
For R'_2, the only communication R'_2 gets is the index l, which is the location of the m'th b-bit in R_2, where $b = R_1(j) \oplus S_i$, and $m = \sigma_2^b(i) = \pi_2^b(i) + s_2^b (\text{mod } n)$. 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 i. m is uniformly distributed in $\{1, \ldots, n\}$ by randomness of the shift s_2^b, as above. We showed that b and m are both distributed independent of i, in fact uniformly, and thus l is also uniformly distributed (in $\{1, \ldots, 2n\}$), independent of i. □

4.2 Complexity

Space Complexity: R_1, R_2 require $O(n)$ space, and $A_1, \ldots, A_k$ require $O(n \log n)$ space. Specifically, R_1 is a n-bit string[4], R_2 is a $2n$-bit string, and $A_1, \ldots, A_k$ each hold n secret bits, the same $n + 2n$ bits as in R_1 and R_2, and $3 \log n! < 3n \log n$ bits for the three permutations, for a total of $O(n \log n)$ bits.

Communication Complexity: Recall that $SOT_k(l, n)$ denotes the communication complexity required for a $\binom{n}{1}$ semi-oblivious transfer protocol for l-bit secrets using k secret holders.

[4] Even a slightly shorter $\log \binom{n}{\frac{n}{2}}$-bit string suffices, since we need to specify an n-bit string with equal number of 0's and 1's. A similar observation holds for R_2.

Theorem 4 *The random pointer scheme for $k+2$ holders uses communication complexity of $O(k \log n) + SOT_k(2\log n + 1, n)$.*

Proof: The communication in this scheme consists of $3\log n$ bits sent by Bob in the first step to each of $A_1, \ldots, A_k$ (indicating the three shifts s_1, s_2^0, s_2^1), $\log n + \log 2n$ bits sent by Bob in the last step to R_1 and R_2 (indicating the locations j, l respectively), their two answer bits, and the communication required by the underlying semi-oblivious protocol P for n secrets of length $\log n + \log 2n = 2\log n + 1$ each (recall that the secrets for the underlying subprotocol are of the form $(j_1, l_1), \ldots, (j_n, l_n)$ where j_r, l_r are location into n-bit and $2n$-bit strings). Altogether, this gives $(3k+4)\log n + 4 + SOT_k(2\log n + 1, n) = O(k\log n) + SOT_k(2\log n + 1, n)$. □

Corollary 1 *Starting from any semi-oblivious protocol, protecting Bob only, an $\binom{n}{1}$ oblivious transfer protocol protecting both sides can be constructed, paying a logarithmic factor in communication complexity.*

Proof: Clearly, an $SOT_k(l, n)$ protocol can be implemented by executing an $SOT_k(1, n)$ protocol l times, considering each bit in the secret separately, one at a time[5]. Since in theorem 1 the dominating communication complexity in the random pointer scheme is $SOT_k(2\log n + 1, n)$, the corollary follows. □

We note that when a more efficient approach for $SOT_k(l, n)$ rather than the one bit at a time is possible, it should be used to obtain further savings in communication. For example, [11] shows that when a semi-oblivious protocol satisfies a certain additivity condition on the reconstruction function for Bob, then $SOT_k(l, n)$ can be solved within l times the complexity of $SOT_k(1, \frac{n}{l} + 1)$ (see [11] for details).

Using the best upper bounds known to date for semi-oblivious protocols, yields the following.

Corollary 2 *The random pointer scheme can be used with known subprotocols to achieve the following results:*

- *$k+2$-holders scheme with communication complexity $O(n^{\frac{1}{2k-1}} \log n)$, for every constant k. (For $k = 2$ this is $O(n^{\frac{1}{3}} \log n)$).*
- *A scheme for a logarithmic number of holders, with polylogarithmic communication complexity.*
- *A computational version (relying on the existence of one way functions) for a constant number of holders, with communication complexity of n^{ϵ}, for every $\epsilon > 0$.*

Proof: These results follow directly from combining the previous corollary with the known protocols of [11] for 2 databases, [2] for any constant number k of databases, [11] for $\log n + 1$ databases, and the [10] computational protocol for 2 databases. □

5 Generalizations

The random pointer scheme can be generalized to support more general variants of OT, such as *privacy with respect to coalitions* of secret holders, or oblivious transfer of secrets consisting of *blocks* of bits. In the following we show how an underlying SOT scheme P supporting the generalized variant, can be extended to a generalized OT scheme.

5.1 Privacy With Respect to Coalitions

So far we were concerned with the privacy of Bob with respect to each single holder (either an A or an R), assuming there is no communication between different holders. This protocol can be extended to allow privacy with respect to coalitions of up to t holders who may communicate with each other. We say that a distributed $\binom{n}{1}$ oblivious transfer scheme is *t-private* if no t holders together may obtain from their joint view any information about Bob's selection index i. Note that 1-private OT means the regular distributed $\binom{n}{1}$ oblivious transfer as defined before.

[5]Note that here we need not worry about a cheating Bob who may ask for different bits from different secrets at each execution, since this is a *semi-oblivious* protocol, meaning we only care about Bob's privacy, and not Alice's.

The random pointer scheme as described above achieves only 1-privacy. There are three types of coalitions that could potentially violate the privacy of Bob, a coalition between the two R's, a coalition between the Alices, and a coalition between a combination of Alices and R's.

In order to allow for coalitions of size t between the R's we increase the number of R's to be $t+1$: $R_1, \ldots, R_t$ corresponding to R_1 in the original scheme, and R_{t+1} corresponding to R_2 in the original scheme. This way any coalition of up to t random holders from $R_1, \ldots, R_{t+1}$ is missing at least one random holder, and thus their view is uniformly distributed.

In order to allow for coalitions of size t between the Alices we use an underlying SOT scheme which is t-private, such as the one suggested in [11].

In order to allow for coalitions of size t between A's and R's, we can extend the RP scheme to achieve t-privacy, by using the following idea: instead of having just one level of pointers from Alice to R, we propose to use t levels of pointers between Alice and R, such that at least $t+1$ holders will have to form a coalition in order to gain some information about the Bob's query. Thus, if the 1-privacy scheme included k Alices and 2 R's, we now have a level of k Alices, a level of $t+1$ R's, and between them we use $t-1$ intermediate levels of secret holders called *AR*, where each level consists of k *AR*'s. Altogether we use $kt+t+1$ holders, denoted as follows (next to each level of holders we denote the secret string associated with that level).

Holders			Secrets
A^1_1,	$\ldots$,	A^1_k,	$[S_1, \ldots, S_n]$
AR^2_1,	$\ldots$,	AR^2_k,	$[\vec{r^2} = r^2_1, \ldots, r^2_{2n}]$
	$\vdots$		
AR^t_1,	$\ldots$,	AR^t_k,	$[\vec{r^t} = r^t_1, \ldots, r^t_{2n}]$
$R_1, \ldots, R_t,\ R_{t+1}$.			

$S_1, \ldots, S_n$ are the original secrets, and each $\vec{r^l}$ is an independent random string of length $2n$ consisting of half 0's and half 1's.

We denote the holders in the intermediate levels by *AR* because their role in the protocol is to play Alice's role (like in the RP scheme), but their content consists of random bits, and thus they can be viewed as random holders, and don't need to know the original secrets.

The protocol follows the same idea as the basic RP scheme. Bob starts by running the SOT protocol P with the first level of holders, where the secrets used by the holders are pointers (indices) into the second level's secret string corresponding to the original secrets. Thus, Bob semi-obliviously receives a pointer i_2 into $\vec{r^2}$ that he is now interested in. He runs P again with the second level holders, using i_2 as his selection index, and obtains a new index i_3 into $\vec{r^3}$ and so on. After t steps Bob has obtained a private index i_t into $\vec{r^t}$. He now runs P with the holders at level t, to receive $t+1$ pointers $j_1, \ldots, j_t, j_{t+1}$ into $R_1, \ldots, R_t, R_{t+1}$ respectively. Now he can ask each random holder for the value in the corresponding location, and xor the values to obtain his answer.

The pointers used by the holders as secrets are obtained in the same manner as in the original scheme, namely via random permutations that are shared between the holders in each level, and are used to calculate pointers with a suitable bit value into the next level's string.

In order to make the above idea work, we need to make one additional modification. To see why, note that in the scheme described above, if one of the holders in level t communicates with one of the random holders, they can find out which bit in $\vec{r^t}$ Bob was interested in. This gives away the value of the bit of Bob's interest (although not its index, since they don't know the mappings from $\{S_1, \ldots, S_n\}$ to the string $\vec{r^t}$).

To solve this problem, before running P with a certain level l, Bob first sends a random bit $b_l \in \{0,1\}$ to all holders of that level. The holders xor all the secret bits with b_l, and proceed as above. At the end of the protocol, Bob xors $b_1 \oplus \ldots \oplus b_t$ and all the bits he received from $R_1, \ldots, R_{t+1}$ to obtain his desired bit.

Now, any coalition of up to t holders from different levels cannot contain one holder from each level and a random holder (since this would consist of $t+1$ holders), and thus cannot have *all* the permutations connecting the original secrets with the last level secrets *and* the location Bob has asked from the random holder, and thus cannot have any information about Bob's original interest.

5.2 OT of Blocks of Secrets

Another possible generalization, is the transfer of *block secrets* consisting of $l > 1$ bits each. Note, that unlike semi-oblivious transfer, this cannot be done simply by running the original scheme for each bit[6]. A simple way to extend the RP scheme to block secrets, is to change R_1 and R_2 to consist of blocks of length l, so that Bob receives two indices of blocks in R_1 and R_2 such that the xor of the two is the secret.

In order for the original protocol and proof to follow through in this setting, R_1 should contain n l-bit blocks, chosen randomly so that each possible block appears the same number of times. R_2 should contain n copies of each possible block, in random order, so it needs to have $n2^l$ l-bit blocks.

6 Repetitive $\binom{n}{1}$ Oblivious Transfer

In the context of the Secure Database Access problem, repetitive executions of the Oblivious Transfer protocol are very desirable. By repetitive executions we mean that the protocol is used multiple times with the same secrets. In order to allow repetitions, two issues must be examined:

Security: Executing the scheme k times will give Bob information about only k secrets of his choice, but no more than that, and will give Alice no information at all regarding Bob's choices of secrets. This extends the traditional definition of $\binom{n}{1}$ oblivious transfer which guarantees this for a single execution ($k = 1$).

Efficiency: Executing the scheme k times can be done in a reasonable complexity. In particular, we want to achieve secure repetitive $\binom{n}{1}$ oblivious transfer without recomputing the whole protocol from scratch with each execution of the protocol.

For simplicity, we present repetitive oblivious transfer with respect to the standard model, with a single secret holder. It is easy to extend the notion to distributed repetitive oblivious transfer, similarly to our approach in the previous sections.

Repetitive Oblivious Transfer: This is a protocol between Alice, who has n secret bits $S_1, \ldots, S_n$, and Bob who has a selection index $i \in \{1, \ldots, n\}$. We break the protocol into a setup stage, and an on-line stage. We say that the protocol is *secure for k repetitions*, if after performing the setup stage once and the on-line stage k times, with selection indices $i_1, \ldots, i_k$ respectively, the following three conditions hold.

1. Bob learns the secrets at his selection indices, namely $S_{i_1}, \ldots, S_{i_k}$.
2. Bob gains no further information about the other secrets S_j , $j \notin \{i_1, \ldots, i_k\}$.
3. Alice learns nothing about the values of $i_1, \ldots, i_k$.

Clearly, for the repetitive scenario it is desirable to have as much of the work load as possible done during the setup stage, so that the on-line stage (which is the one being repeated) is as efficient as possible.

We inspect existing protocols, and show that they are not secure even for 2 executions, unless all the setup (such as choosing a one way function, etc) is computed from scratch every time, which makes it too inefficient for repetitive applications.

6.1 Open Question

In our scheme, it is clear that if all random strings of the secret holders are chosen independently every time, then the scheme can be repeated without losing privacy. This may be a reasonable solution for constant number of repetitions (since several random strings can be generated in advance), or for computational security (using short pseudo random seeds). However, it is too expensive if we require a large number of repetitions and insist on information theoretic security.

Other existing schemes also fail to solve this problem, as we show below. Thus, the problem of designing a repetitive (efficient) $\binom{n}{1}$ oblivious transfer protocol remains as an important and useful open problem, and we are currently working towards solutions in this directions.

[6] If we did this, Bob could ask for l different bits from different blocks, thus obtaining information dependent on more than one secret.

6.2 Problems with Repetition for Existing Protocols

It is interesting to find out, that existing implementations of $\binom{n}{1}$ oblivious transfer are generally not satisfactory for applications that require repetitions. In this section we look at some of the existing schemes and their repetition security.

6.2.1 Oblivious Transfer Based on any One Way Trapdoor Function

A general $\binom{n}{1}$ oblivious transfer protocol based on any one way trapdoor function is described in [14]. In what follows we provide a brief sketch of the protocol, and show that it is not secure for repetitions.

Sketch of Protocol: Alice and Bob agree on a one way trapdoor function f, and let B be a hard core predicate for f. Bob sends to Alice n numbers $y_1, \ldots, y_n$, where the one at his selection index i is of the form $y_i = f(x)$ for a randomly chosen x, and the other $n-1$ numbers are chosen randomly. Alice gets the list $y_1, \ldots, y_n$ and sends back $S_1 \oplus B(f^{-1}(y_1)), \ldots, S_n \oplus B(f^{-1}(y_n))$. Bob is able to find S_i, which is the exclusive-or of two values he knows: $B(f^{-1}(y_i)) = B(x)$ and $B(f^{-1}(y_i)) \oplus S_i$. Bob cannot find any other secret S_j for $j \neq i$, since it is masked by $B(f^{-1}(y_j)$ which Bob has no information about.

Indeed, one application of the scheme gives Bob only one secret. However, applying the same scheme twice, gives Bob much more information than just 2 secrets (unless the one way function and hard core predicate are chosen afresh every single time). In fact, in two applications Bob can recover *all* the secrets, as follows.

- **First iteration of the scheme:** Bob sends to Alice

  $$f(x),\ y_2,\ y_3, \ldots, y_n$$

  for random $x, y_2, \ldots, y_n$

- Bob receives from Alice

  $$z_1 = S_1 \oplus B(x), \quad z_2 = S_2 \oplus B(f^{-1}(y_2)), \quad \ldots, \quad z_n = S_n \oplus B(f^{-1}(y_n))$$

- Bob calculates $\mathbf{S_1} = z_1 \oplus B(x)$

- **Second iteration of the scheme:** Bob sends to Alice

  $$y_2, \ldots, y_n,\ y'$$

  for a random y'

- Bob receives from Alice

  $$w_1 = S_1 \oplus B(f^{-1}(y_2)), \ldots, w_{n-1} = S_{n-1} \oplus B(f^{-1}(y_n)),\ w_n = S_n \oplus B(f^{-1}(y'))$$

- Bob calculates

  $$\mathbf{S_2} = z_2 \oplus w_1 \oplus S_1, \quad \mathbf{s_1} = z_3 \oplus w_2 \oplus S_2, \quad \ldots, \quad \mathbf{S_n} = z_n \oplus w_{n-1} \oplus S_{n-1}$$

Bob has all n secrets now.

6.2.2 Oblivious Transfer Based on Quadratic Residuosity

Brassard Crepeau and Robert [9] suggest a $\binom{n}{1}$ oblivious transfer protocol based on the quadratic residuosity assumption (QR). This protocol has weaknesses in terms of repetition security, but it is much better than the previous one in this respect.

Sketch of Protocol: Alice sends to Bob all secrets, encrypted (using QR). Bob selects the encrypted secret of his choice, and encrypts it again using his own key, and sends to Alice. Alice decrypts the value she had received, thus removing her encryption from it. The resulting value is sent to Bob, who can now remove his encryption from it as well, and get his desired secret. This idea as described above does not quite work yet, since Bob may get some other value (e.g. the xor of two secrets) rather than one of the secrets. To get around this problem, the protocol is modified such that Bob sends to Alice a value called a $\sigma-$packet P_σ, together with a proof of validity for P_σ. (see [9] for details).

The same P_σ may be used for repetitive applications of the scheme, achieving only partial security (e.g. Alice can tell whether Bob's questions (selection indices) are all different or not). Using a new P_σ for every repetition avoids this leakage of information, but considerably increases the on line stage complexity.

6.2.3 Non Interactive Oblivious Transfer

A different flavor of oblivious transfer was introduced by Bellare and Micali [5], where the goal is to eliminate interaction from the oblivious transfer protocol. Their non interactive protocol violates the definition of repetition security, since Bob can hold only one selection index i, and if the protocol is to be repeated, he will get the secret at the same location every single time.

Sketch of Protocol: This protocol is based on the Diffie-Helman assumption. The idea is that Bob publishes a set of n public keys, such that he knows the discrete logarithm of exactly one of them (the protocol provides a way for Alice to make sure Bob cannot know the discrete logarithm of more than one of his public keys, relying on the Diffie-Hellman assumption). Now, to perform an oblivious transfer Alice sends the n secrets encrypted with the n public keys, and Bob can decrypt only the one corresponding to the public key whose discrete log he knows.

Since the selection index of Bob is predetermined from the moment he publishes his public keys, it is clear that repeating the scheme k times, Bob will always have the same selection index (i.e. $i_1 = \ldots = i_k$ in our definition of repetitive oblivious transfer), and the scheme is not secure for repetitions.

Acknowledgment

We wish to thank Shafi Goldwasser for her invaluable input throughout this work, and for her helpful suggestions, comments, and pointers.

References

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

[2] A. Ambainis. Upper bound on the communication complexity of private information retrieval. ICALP 97.

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

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

[5] M. Bellare, S. Micali. Non-Interactive Oblivious Transfer and Applications. CRYPTO 1989.

[6] C. Bennett, G. Brassard, C. Crepeau, M. H. Skubiszewska. Practical Quantum Oblivious Transfer. CRYPTO 1991

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

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

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

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

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

[12] C. Crepeau, J. Kilian. Weakening Security Assumptions and Oblivious Transfer. CRYPTO 1988.

[13] C. Crepeau, J. Kilian. Achieving Oblivious Transfer Using Weakened Security Assumptions. FOCS 1988.

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

[15] J. Kilian. Founding Cryptography on Oblivious Transfer. STOC 1988.

[16] J. Kilian. PhD Thesis, MIT.

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

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