Distributed Oblivious Transfer

Moni Naor* Benny Pinkas **

Abstract. This work describes distributed protocols for oblivious transfer, in which the role of the sender is divided between several servers, and a chooser (receiver) must contact a threshold of these servers in order to run the oblivious transfer protocol. These distributed oblivious transfer protocols provide information theoretic security, and do not require the parties to compute exponentiations or any other kind of public key operations. Consequently, the protocols are very efficient computationally.

1 Introduction

Oblivious Transfer (abbrev. OT) refers to several types of two-party protocols where at the beginning of the protocol one party, the *sender*, has an input, and at the end of the protocol the other party, the *chooser* (sometimes called the receiver), learns some information about this input in a way that does not allow the sender to figure out what the chooser has learned. In this paper we are concerned with 1-out-of-2 OT protocols where the sender's input consists of two strings (m_0, m_1) and the chooser can choose to get either one of these inputs and learn nothing about the other string.

Distributed oblivious transfer protocols distribute the task of the sender between several servers. Security is ensured as long as a limited number of these servers collude. The constructions we describe have three major advantages compared to single server based oblivious transfer: (1) They are more efficient since they only involve the evaluation of polynomials over relatively small fields (and no exponentiations). (2) They provide information theoretic security, thus making the task of composing such a protocol with other protocols easier. (3) They also provide better security guarantee when applied to the multi party protocols based on the auction architecture of of [21] (see below).

The setting of distributed oblivious transfer involves, as in the basic 1-out-of-2 protocol, a sender with two inputs m_0, m_1, and a chooser with an input $\sigma \in \{0, 1\}$. There are also n servers $S_1, \ldots, S_n$. The sender generates for every server S_i a transfer function F_i, which is sent to the server. Apart from this message there is no interaction between the servers and the sender, or between

* Dept. of Computer Science and Applied Math, Weizmann Inst. of Science, Rehovot 76100, Israel. E-mail: naor@wisdom.weizmann.ac.il. Part of this work was done while visiting Stanford University and IBM Almaden Research Center. Partly supported by DOD Muri grant administered by ONR and DARPA contract F30602-99-1-0530.

** STAR Lab, Intertrust Tech., 4750 Patrick Henry Drive, Santa Clara, CA 95054. Most of this work was done while the author was at the Weizmann Institute of Science and the Hebrew University of Jerusalem, and was supported by an Eshkol grant of the Israeli Ministry of Science. E-mail: bpinkas@intertrust.com, benny@pinkas.net.

the servers themselves. Server S_i then uses the function F_i to answer a query of the chooser. The sender never interacts with the chooser and can be offline when the chooser sends his queries.

Related work. The notion of 1-out-2 oblivious transfer was suggested by Even, Goldreich and Lempel [13], as a generalization of Rabin's "oblivious transfer" [23]. Further generalization to 1-out-of-N oblivious transfer was introduced by Brassard, Crépeau and Robert [7] under the name ANDOS (all or nothing disclosure of secrets). For an up-to-date definition of OT and oblivious function evaluation see Goldreich [16].

Reductions between various types of oblivious transfer protocols have been investigated extensively and they all turn out to be information theoretically equivalent (See [6, 8, 12, 11, 9]). These reductions emphasize the importance of distributed oblivious transfer, since they enable other types of OT protocols to be based on the efficient constructions of distributed OT presented in this paper. In particular, a protocol for distributed 1-out-of-N OT can be constructed using the (non-information theoretic) reduction of Naor and Pinkas [20] to OT_1^2. The protocol uses $\log N$ invocations of distributed OT_1^2, and N invocations of a pseudo-random function. The resulting OT_1^N protocol is very efficient and does not require any public key operations.

Oblivious transfer protocols are the foundation of secure distributed computation. Since its proposal by Rabin [23] OT has enjoyed a large number of applications and in particular Kilian [19] and Goldreich and Vainish [17] have shown how to use OT in order to implement general oblivious function evaluation, i.e., to enable parties to evaluate any function of their inputs without revealing more information than necessary. Oblivious transfer can be implemented under a variety of assumptions (see e.g. [6, 13, 5]). Essentially every known suggestion of public-key cryptography allows also to implement OT (although there is no general theorem that implies this state of affairs), and the complexity of 1-out-of-2 OT is typical of public-key operations [6, 5]. OT can be based on the existence of trapdoor permutations, factoring, the Diffie-Hellman assumption and the hardness of finding short vectors in a lattice (the Ajtai-Dwork cryptosystem). On the other hand, given an OT protocol it is a simple matter to implement secret-key exchange using it. Therefore from the work of Impagliazzo and Rudich [18] it follows that there is no black-box reduction of OT from one-way functions. This result is quite discouraging if one attempts to improve the efficiency of OT protocols, since one-way functions are typically more efficient than public key operations by a few orders of magnitude.

There are many works which solve problems which are related (at least syntactically) to ours. The work of Beaver et. al. [4] on locally random reductions enables to distribute a function between many servers, such that a user can compute the function by contacting these servers. The construction guarantees that the servers cannot learn which values the users compute, but on the other hand it does not provide security against a user who attempts to compute the function in many locations. This is also the case with PIR (private information retrieval) protocols [10]. SPIR protocols [15] address the security of the sender as well, but the emphasis of both these types of protocols is different than ours: they consider communication overhead as the major resource that must be minimized (at the cost of increasing the computation overhead). In the PIR context Gertner et. al. [14] proposed a system where the database owner solicits the help of several servers which are not fully trusted. A related line of work is that of "commodity based cryptography" [3], where OT is treated as a resource, but our work puts a much more stronger emphasis on simplicity and efficiency.

Very recently Rivest has considered a model with a "trusted initializers" who (similarly to the sender in our scenario) participates only in and initial setup [24]. The difference with our setting (i) The trusted party should provide secret information to the receiver/chooser as well; this is unacceptable in application such as the privacy preserving architecture discussed below. (ii) the online sender knows the the values m_0 and m_1, whereas the servers in our scenario do not gain information about them.

Application to the privacy preserving architecture An architecture for executing auctions, economic mechanism design and negotiations was proposed in [21]. The goal is to preserve the privacy of the inputs of the participants (so that no nonessential information about them is divulged, even a posteriori) while maintaining communication and computational efficiency. This goal is achieved by adding another party, the auction *issuer*, in addition to the bidders and the auctioneer. This party's role is to generate the programs ("garbled circuits") for computing the auctions prior to the auction and to run a variant of OT called proxy OT after the the bids have been submitted. Other than that it does not take an active part in the protocol. The auction issuer is not a trusted party, but is assumed not to collude with the auctioneer. In the original protocol of [21] the privacy of bidders is preserved as long as the auction issuer and the auctioneer do not collude.

Employing the distributed oblivious transfer protocols proposed in this paper allows splitting the role of the auction issuer into two parts (this was the motivation for our work). One of them needs a central server that acts only offline. It prepares the garbled circuits and acts as the sender preparing the inputs for the n servers in the distributed OT protocol. During the execution of the auction these n servers, called the online auction servers, operate after the bids are submitted. The central auction issuer can be better safeguarded than the online servers, since it operates offline. Privacy is guaranteed as long as the auctioneer does not collude with a coalition of *several* (more than the given threshold) of the online auction servers.

2 Definitions

A distributed k-out-of-n OT_1^2 protocol involves three types of parties:

- A **sender** which has two inputs m_0, m_1. It is convenient to assume that both these inputs are elements in a field $\mathcal{F}$.
- A **chooser** that has an input bit $\sigma \in \{0, 1\}$.
- Additional n **servers**, $S_1, \ldots, S_n$.

The protocol is composed of the following functional steps:

- The sender generates for each server S_i a function F_i, which depends on (m_0, m_1) and on random coin tosses of the sender.
- The chooser contacts k different servers. She sends to server S_i a query q_i which is a function of σ and of i, and of private random coin tosses. The server answers the query with $F_i(q_i)$.

A distributed k-out-of-n OT_1^2 protocol must guarantee the following properties:

- **Reconstruction:** If the chooser receives information from k servers she can compute m_σ. That is, there is an efficient algorithm for computing m_σ from any set $\{i_j, F_{i_j}(q_{i_j})\}_{j=1}^k$.
- **Sender's privacy:** Given any k values $\{i_j, F_{i_j}(q_{i_j})\}_{j=1}^k$ the chooser must gain information about a single input m_σ, and no information about the other input of the sender. (A weaker requirement is that she can compute at most a single linear combination of m_0 and m_1.)
- **Chooser's privacy:** No coalition of less than t servers gains any information about σ, where t is a parameter in the range $1 \leq t \leq k$. The parameter t should ideally be as close as possible to k.
- **Chooser-servers collusion:** A coalition of the chooser with ℓ corrupt servers cannot learn about m_0, m_1 more than be learned by the chooser herself (where ℓ is a parameter).

An additional requirement is that if the chooser receives information from less than k servers she gains no information about m_0 or m_1. There might be applications in which this requirement is not important, since the emphasis might be on the chooser having to contact *at most* k servers. This requirement is not supported in all of the protocols that we present. Namely, in the protocol of Section 3.2 the receiver can obtain information about a single input after receiving information from less than k servers. However, in this case she compromises her own privacy and risks that a coalition of fewer than k servers can learn σ.

Note that the privacy of both the sender and the receiver is based on information theory and does not depend on any computational assumption. Furthermore, the protocol is very simple, the chooser simply asks server S_i for a value of $F_i(\cdot)$ and receives an answer, and this process is considerably more efficient than a OT_1^2 protocol (since in all protocols F_i is simply a polynomial).

The privacy of the sender depends on the chooser getting shares from at most k servers. We discuss in Section 5 how to ensure that this is indeed the case.

The protocols use bivariate polynomials in a way which is similar to that used by the oblivious polynomial evaluation protocols of [20]: The sender defines a bivariate polynomial $Q(x, y)$ which hides his input, and the chooser defines a secret univariate polynomial $S(x)$ and interpolates $Q(x, S(x))$ which reveals to her one value of the sender's input. However, in [20] a single sender knows the polynomial Q and the chooser uses OT_1^N in order to learn the values of this polynomial at different locations, without revealing them to the sender. In the current work each server knows part of the polynomial, and the chooser simply asks servers to reveal to her values of the polynomial at different points. The chooser does not have to use OT in order to hide these points from the servers, since as long as not too many of them collude they cannot learn her input.

Why secret sharing isn't enough: The first naive approach for designing a distributed OT_1^2 scheme is probably to suggest using simple k-out-of-n secret sharing for sharing m_0 and m_1 between the servers. Namely, each input should be divided into n shares, and each of the n servers is given a share. The chooser should obtain k shares of one of the schemes to reconstruct one of the inputs. The problem with this method is, of course, that the chooser must hide from the servers the identity of the input whose shares it requires. This essentially requires the chooser to run a OT_1^2 protocol with each of the servers.

3 Protocols for Distributed Oblivious Transfer

This section describes several protocols for distributed OT_1^2. The protocols follow the generic structure described in Table 1.

1. **Input:** The sender's input is a pair $m_0, m_1 \in \mathcal{F}$. The chooser's input is $\sigma \in \{0, 1\}$.
2. The sender generates a bivariate polynomial $Q(x, y)$, s.t. $Q(0, 0) = m_0$, $Q(0, 1) = m_1$.
3. The sender sends the univariate polynomial $Q(i, \cdot)$ to server S_i.
4. The chooser chooses a random polynomial S s.t. $S(0) = \sigma$, and defines a univariate polynomial R to be $R(x) = Q(x, S(x))$. The degree of R is $k - 1$.
5. The chooser asks server S_i for the value $R(i) = Q(i, S(i))$.
6. After receiving k values of R the chooser interpolates R and computes $R(0)$.

Fig. 1. The basic steps of the distributed OT_1^2 protocol.

The main difference between the different protocols is the type of the polynomial $Q(x, y)$ that is generated by the sender. This choice affects all other parameters of the protocol. In particular, the first type of protocols uses a polynomial $Q(x, y)$ which is defined as the sum of a polynomial in x and a linear polynomial in y, and has no monomials which include both x and y. We denote such polynomials as *sparse*. Since the sender is only required to compute sparse polynomials, his task is greatly reduced (compared to the computation of full polynomials). This type of protocols is secure as long as there is no collaboration between the chooser and a corrupt server. It is also possible to make it immune against a collusion between the chooser and a single (or a few) servers.

We describe a different type of protocols which can protect the sender's privacy against a collusion between the chooser and a large set of servers. This type of protocols uses *full* bivariate polynomials in which the coefficients of all the monomials are non-zero (with high probability).

3.1 Using a sparse polynomial

The most basic and straightforward protocol employs a bivariate polynomial, where the degree of y is 1 and there are no monomials which contain both x and y. The protocol is described in Figure 2. It has the following properties.

Initialization: The sender generates a linear polynomial $P_y(y) = b_1 \cdot y + b_0$, s.t.

$$P_y(0) = m_0, P_y(1) = m_1. \qquad (I.e., m_0 = b_0,\ m_1 = b_1 + b_0.)$$

The sender generates a random masking polynomial $P_x(x)$ of degree $k-1$, s.t. $P_x(0) = 0$. Namely, $P_x(x) = \sum_{j=1}^{k-1} a_j x^j$. It also defines a bivariate polynomial

$$Q(x,y) = P_x(x) + P_y(y) = \sum_{j=1}^{k-1} a_j x^j + b_1 y + b_0$$

The sender provides server S_i with the function $F_i(y)$ which is the result of substituting $x = i$ in the polynomial Q. Namely,

$$F_i(y) = Q(i,y) = \sum_{j=1}^{k-1} a_j i^j + b_1 y + b_0 = b_1 y + (\sum_{j=1}^{k-1} a_j i^j + b_0)$$

Transfer: The chooser generates a random polynomial $S(x)$ of degree $k-1$, subject to the constraint $S(0) = \sigma$. I.e, $S(x) = \sum_{j=0}^{k-1} s_j x^j$ where $s_0 = \sigma$.
Consider the polynomial $R(x)$ which is generated by substituting $S(x)$ instead of y in Q,

$$R(x) = Q(x, S(x)) = \sum_{j=1}^{k-1} a_j x^j + b_1 \sum_{j=0}^{k-1} s_j x^j + b_0 = \sum_{j=1}^{k-1} (a_j + b_1 s_j) x^j + b_1 s_0 + b_0$$

The chooser's goal is to interpolate R and compute $R(0) = Q(0, S(0)) = Q(0,\sigma) = m_\sigma$. The degree of R is $k-1$, and therefore the chooser should obtain k values of R in order to interpolate it. She approaches k different servers and asks server S_i for the value $F_i(S(i)) = Q(i, S(i)) = R(i)$. After receiving k answers she can interpolate R and compute $R(0) = m_\sigma$.

Fig. 2. A distributed OT_1^2 protocol using a sparse linear polynomial.

- **Reconstruction:** After receiving information from k servers, the chooser can learn m_σ, by interpolating the polynomial R.
- **Sender's privacy:** After receiving information from k servers, the chooser cannot learn more than a single linear equation of m_0 and m_1 (this is proved in theorem 1). We later show in Section 4 how to ensure that the chooser learns exactly m_0 or m_1 and not any other combination of these values.
- Information from less than k servers does not reveal to the chooser any information about m_0 and m_1 (since the degree of x in Q is $k-1$).
- **Chooser's privacy:** No coalition of at most $t = k-1$ servers can learn any information about σ (this is proved in Theorem 2 and is based on the degree of S being $k-1$).
- **No security against chooser-server collusion:** A coalition of the chooser with one corrupt server reveals to the chooser both m_0 and m_1 (after running the protocol). At the end of this Section we describe a method to address this problem if the chooser colludes with a single corrupt server (or a small

number of corrupt servers). Section 3.2 describes a scheme which is secure against a collusion between the receiver and a large number of servers.

- **Overhead:** The sender has to choose $O(K)$ elements and has to send to each server $O(1)$ elements. Each server has to compute a linear polynomial a single time. The chooser should contact k servers, and her total communication overhead is $O(k)$. The computation of m_σ involves interpolation of a $k-1$ degree polynomial in order to find its free coefficient. This can be done in $O(k^2)$ multiplications using Lagrange's interpolation formula, or $O(k \log^2 k)$ multiplications using FFT (see e.g. [1] p. 299). The operations are done over the field $\mathcal{F}$ which can be rather small[1] and are therefore efficient by a few orders of magnitude compared to the public key operations required (following [18]) for non-distributed oblivious transfer.

Proofs of privacy

Theorem 1 (Sender's privacy). *After receiving information from k servers, the chooser cannot learn more than a single linear combination of m_0 and m_1.*

Proof: When the chooser sends to server i the query y_i, she receives the answer $F_i(y_i) = Q(i, y_i) = \sum_{j=1}^{k-1} a_j i^j + b_1 y_i + b_0$. The receiver therefore obtains the following set of k equations:

$$\underbrace{\begin{pmatrix} i_1^{k-1} & i_1^{k-2} & \cdots & i_1 & y_{i_1} & 1 \\ i_2^{k-1} & i_2^{k-2} & \cdots & i_2 & y_{i_2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ i_k^{k-1} & i_k^{k-2} & \cdots & i_k & y_{i_k} & 1 \end{pmatrix}}_{A} \cdot \begin{pmatrix} a_{k-1} \\ \vdots \\ a_1 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} F_{i_1}(y_{i_1}) \\ F_{i_2}(y_{i_2}) \\ \vdots \\ F_{i_k}(y_{i_k}) \end{pmatrix}$$

It should be shown that no matter what values the chooser assigns to the y_i's, she does not learn more than a single linear combination of b_0, b_1. In other words, that the rows of the matrix A do not span both the vector $e_k = (0, \ldots, 0, 1, 0)$ and the vector $e_{k+1} = (0, \ldots, 0, 0, 1)$. The matrix A has $k+1$ columns and k rows. Consider the matrix A' with $k+1$ rows which is formed by taking the first $k-1$ rows of A and appending to them the vectors e_k, e_{k+1}. The determinant of A' is different than 0 (since the sub-matrix of size $(k-1) \times (k-1)$ in the upper-left corner is Van Der Monde). Therefore, the first $k-1$ rows of A do not span any of e_k, e_{k+1}, and the matrix A which has just a single additional row cannot span both vectors. $\square$

Theorem 2 (Chooser's privacy). *A coalition of $k-1$ servers does not learn any information about σ.*

[1] Typically the field should contain m_0, m_1. However, if these elements are large the sender can choose two random keys k_0, k_1 (say, 128 bits long) and use them to encrypt m_0, m_1, respectively. The OT protocol should be run for the inputs k_0, k_1, and therefore the field $\mathcal{F}$ should only be large enough to contain them.

Proof: The coalition receives $k - 1$ values of $S(i)$ for $i \neq 0$. The polynomial S is of degree k and is random except for $S(0) = \sigma$. The information that the coalition learns could have been equally likely derived from a polynomial S with $S(0) = 0$ as from a polynomial with $S(0) = 1$. □

How to protect against a collusion between the chooser and a single server: The main drawback of the protocol is that a collusion between the chooser and one of the servers reveals both m_0 and m_1. This happens since each server S_i knows a polynomial $F_i(y)$ which reveals $b_1 = m_1 - m_0$. We describe below a simple solution against a collusion between a chooser and a *single* server. This solution is general and is good for *any* distributed OT scheme. The aim of the rest of the paper is to deal with larger collusions.

In order to protect against a coalition of the chooser with a *single* server, the sender divides the n servers into all possible n subsets of $n - 1$ servers. It defines n random shares $\{m_{0,i}\}_{i=1}^{n}$ that satisfy $m_0 = \oplus_{i=1}^{n} m_{0,i}$, and similarly shares $\{m_{1,i}\}_{i=1}^{n}$ that satisfy $m_1 = \oplus_{i=1}^{n} m_{1,i}$. Next, it defines n schemes for $(k-1)$-out-of-$(n-1)$ distributed OT_1^2. The ith scheme enables to transfer either one of $(m_{i,0}, m_{i,1})$, and is assigned to the members of the ith subset of servers.

The chooser should contact k servers, and run the n distributed OT_1^2 protocols, learning $\{m_{\sigma,i}\}_{i=1}^{n}$. She should then combine the results to compute m_σ.

This protocol ensures that a coalition of $t = k - 2$ servers cannot learn which element the receiver learned, and that any k servers enable the receiver to learn only a single share. A coalition of the chooser with a single server cannot learn any additional information, since this server has no information about one of the OT_1^2 schemes. This method can be generalized to handle a collusion of the chooser with t servers, but this would require running $\binom{n}{t}$ distributed OT_1^2 protocols.

3.2 Using a full polynomial

In order to protect against large chooser-servers collusions, the sender should use a bivariate polynomial which includes all possible monomials, and in which the degree of y is high. This approach yields a tradeoff between the number of servers that can compromise the chooser's privacy, and the size of a chooser-servers collusion that can compromise the sender's privacy. The protocol is described in Figure 3.

The protocol has the following properties:

- **Reconstruction:** As in the previous protocol, after receiving information from k servers the chooser can learn m_σ, since the degree of R is k.
- **Sender's privacy:** After receiving information from k servers, the chooser cannot learn more than a single linear equation of m_0 and m_1. This is proved in Theorem 3 in the Appendix. We show in Section 4 how to ensure that she learns exactly m_0 or m_1.
- **Chooser's privacy:** No coalition of at most $t = d_s = (k-1)/(2d_y)$ servers can learn any information about σ (if the chooser acts according to the protocol). This follows from the degree of S.

Initialization: The sender generates a random bivariate polynomial $Q(x,y)$ of degree d_x in x and degree d_y in y, subject to the constraints

$$Q(0,0) = m_0, \; Q(0,1) = m_1.$$

Namely, $Q(x,y) = \sum_{j=0}^{d_x} \sum_{l=0}^{d_y} a_{j,l} x^j y^l$, where $a_{0,0} = m_0$ and $\sum_{l=0}^{d_y} a_{0,l} = m_1$. It should also hold that $d_x = (k-1)/2$ (the parameter k must be even).
The sender sends to server S_i the function $F_i(y)$ which is the result of substituting $x = i$ in the polynomial Q. Namely,

$$F_i(y) = \sum_{l=0}^{d_y} (\sum_{j=0}^{d_x} a_{j,l} \cdot i^j) \cdot y^l.$$

Transfer: The chooser generates a random polynomial $S(x)$ of degree d_s, where the degree satisfies[a] $d_y d_s = d_x = (k-1)/2$. The polynomial S is random subject to the constraint $S(0) = \sigma$.
Consider the polynomial $R(x)$ which is generated by substituting $S(x)$ instead of y in Q,

$$R(x) = Q(x, S(x))$$

The chooser should interpolate R and compute $R(0) = Q(0, S(0)) = Q(0,\sigma) = m_\sigma$. The degree of R is $k - 1 = d_x + d_y d_s$, and therefore the chooser should obtain k values of R in order to interpolate it. She approaches k different servers and asks server S_i for the value $F_i(S(i)) = Q(i, S(i)) = R(i)$. After receiving k answers she can interpolate R and compute $R(0) = m_\sigma$.

[a] We assume that the degrees are chosen such that this equality holds. Otherwise it must hold that $d_y d_s < d_x$.

Fig. 3. A distributed OT_1^2 protocol using a full polynomial.

- Information from less than k servers might reveal to the chooser information about m_0 or m_1 (e.g., if she sets $S(x)$ to be of degree smaller than d_x, the degree of $R = Q(x, S(x))$ would be smaller than k). However, this affects the chooser's privacy, namely reveals σ to a coalition of less than $(k-1)/(2d_y)$ servers. If the chooser receives information from less than d_x servers she learns no information about either m_0 or m_1.
- **Security against chooser-servers collusion:** A coalition of the chooser with $d_x - \frac{2d_x}{d_y+1}$ corrupt servers, does not reveal to the chooser more than a single linear equation of m_0 and m_1. This is proved in Theorem 4 in the Appendix.
- **Overhead:** The sender in preparing the polynomial has to choose $O(kd_y)$ elements and send d_y elements per server. Each server has to compute a polynomial of degree d_y a single time. The overhead of the chooser is as in the sparse polynomial scheme.

This construction, therefore, gives a tradeoff between chooser privacy against a coalition of corrupt servers, and sender's privacy against a coalition between the chooser and corrupt servers. Once n and k are fixed, The tradeoff depends on a parameter d_y. The size of a coalition of corrupt servers against which the chooser is secure is $(k-1)/(2d_y) = d_x/d_y$, whereas the size of a coalition of corrupt servers that can help the chooser learn more than a single input is $d_x - \frac{2d_x}{d_y+1}$.

4 Preventing the chooser from learning linear combinations

Suppose that the chooser must be forced to learn either m_0 or m_1, and it is required to prevent her from learning linear combinations of the two inputs[2].

The following method can be used to ensure that the chooser learns either m_0 or m_1, but not any other linear combination of the two inputs. We describe it for the protocol of Section 3.1 which uses a sparse bivariate polynomial.

The protocol is run simultaneously with two polynomials $P_y^1 = (a \cdot m_1 - b \cdot m_0)y + m_0 \cdot b$, and $P_y^2 = (a-b)y + b$, and corresponding polynomials Q^1 and Q^2. (The first polynomial hides m_1 multiplied by a, and m_0 multiplied by b, whereas the second polynomial hides a and b). The chooser sends a single value $S(i)$ to server i and receives the values $Q^1(i, S(i))$ and $Q^2(i, S(i))$.

If the chooser operates according to the protocol, she learns the values $m_0 \cdot b$ and b if $S(0) = 0$, and can then compute m_0. Similarly, she can compute m_1 if she sets $S(0) = 1$.

The chooser cannot learn any other linear combination of m_0 and m_1. The important property of the protocol is that the chooser learns the same linear combination of the coefficients of both P_y^1 and P_y^2. Suppose that in this combination the coefficient of y is multiplied by α and the free coefficient is multiplied by β. The chooser therefore learns the following equations:

$$\begin{pmatrix} m_1\alpha & m_0(\beta-\alpha) \\ \alpha & \beta-\alpha \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

If this matrix is non singular then any value of m_0, m_1 corresponds to a different pair a, b, and no information is divulged about m_0 or m_1. The matrix is singular only if $m_0 = m_1$ (but we can ensure that this does not happen if we append a different prefix to each input), or if $\alpha = 0$ or $\alpha = \beta$. These last two cases reveal to the chooser the value of m_0 or m_1, respectively, and are therefore legitimate.

5 Ensuring that a chooser does not obtain more than k shares

Distributed oblivious transfer prevents the chooser from learning more than a single input as long as she does not obtain information from more than k

[2] A heuristic approach for achieving this property might encrypt the inputs m_0 and m_1 using two random keys k_0 and k_1, respectively, and run the distributed OT protocol to let the chooser learn either k_0 or k_1. If the chooser chooses to learn a linear combination of both keys then presumably she would not be able to decrypt any of the encryptions. This approach can be proved to be secure in the random oracle world, i.e. if a function H which is modeled as a random oracle is used to encrypt each m_i using k_i.

servers. This property raises the following question: how should we ensure that the chooser receives information from at most k servers? (note that this problem does not exist if the system implements an n-out-of-n access structure). This issue might be regarded as orthogonal to the schemes themselves. Alternatively, there might be some centralized mechanism for limiting the number of servers that send information to the chooser. However, it might be difficult to operate such a mechanism in a distributed setting.

We now describe two solutions that are applicable for the case $k > n/2$ (or any other quorum system). The solutions can be combined with any protocol for distributed OT. Therefore there is no need to postulate any external mechanism enforcing the limit on the number of servers accessed in this case.

A solution for $k > n/2$ (or any other quorum system): The servers share a key K for a pseudo-random function F (pseudo-random functions are commonly modeled by block ciphers). The key K is known to each of the servers. Denote the subset of k servers that the user approaches as $\mathcal{S}$, $|\mathcal{S}| = k$. The user sends the names of all servers in $\mathcal{S}$ to each of the servers she contacts.

Each such server, S_i, operates as follows:

- It verifies that $\mathcal{S}$ contains the names of k servers including S_i, and that it did not previously send an answer to the chooser for a different set $\mathcal{S}'$ which contains S_i (for the same OT).
- It computes $\alpha_S = \oplus_{S_i \in S} F_K(S, S_i)$, where F_K is a pseudo-random function F keyed by K.
- It sends to the chooser its answer, as defined in the distributed OT protocol, encrypted by α_S. In addition it sends her $F_K(S, S_i)$.

After receiving answers from all servers in S the chooser can compute α_S and decrypt the answers. Since $k > n/2$, every two different subsets of k servers, $\mathcal{S}$ and $\mathcal{S}'$, intersect, and therefore the chooser cannot compute both α_S and $\alpha_{S'}$.

The above solution can be generalized to any access structure which is based on a quorum system[3]. Assume, for simplicity, that each quorum contains the same number of servers, k. The system should use a k-out-of-n threshold access structure. In addition each server S_i should verify that $\mathcal{S}$ is a legitimate quorum which contains S_i, and encrypt its answer with α_S as described above. Since each two quorums intersect, the chooser can only decrypt k answers of a single quorum.

A solution for $k > n/2$ (and any other quorum system) secure against chooser-servers coalition: The drawback of the previous solution is that even a single server cooperating with the chooser can reveal K and enable the chooser to decrypt messages from more than k servers. The following solution solves the problem chooser-server coalition, provided the size of the coalition is less than $2k - n$.

[3] A quorum system is a collection of subsets of some ground set such that any two subsets intersect. They have been considered for cryptographic purposes previously, e.g. [22].

The sender defines in advance $n(n-1)$ strings $\{\alpha_{i,j}\}_{1\leq i,j\leq n, i\neq j}$ for every *ordered* pair of servers, and gives server S_i the $2(n-1)$ strings $\{\alpha_{i,j}, \alpha_{j,i} \mid i \neq j\}$. The chooser sends to server S_i the set $\mathcal{S}$ of k servers which she is querying. The server first verifies that $S_i \in \mathcal{S}$ and that it was not asked to answer the chooser using a different set $\mathcal{S}'$ of servers. It then sends its answer encrypted by $\oplus_{S_j\in\mathcal{S},\, j\neq i}\alpha_{i,j}$. It also sends to the chooser the values $\{\alpha_{j,i} \mid S_j \in \mathcal{S},\ j \neq i\}$. The chooser must receive answers from all the servers in $\mathcal{S}$ before she can decrypt them. This method can be applied to any access structure which is based on a quorum system, provided a coalition does not cover any intersection of quorums.

References

1. Aho A., Hopcroft J. and Ullman J., *The design and analysis of computer algorithms*, Addison-Wesley, 1974.
2. D. Beaver, *Foundation of Secure Interactive Computation*, Advances in Cryptology - Crypto '91, pp. 377–391, 1991.
3. D. Beaver, "Commodity-Based Cryptography", STOC 1997, pp. 446-455.
4. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. "Locally Random Reductions: Improvements and Applications", Journal of Cryptology 10(1): 17-36 (1997).
5. M. Bellare and S. Micali, *Non-interactive oblivious transfer and applications*, Advances in Cryptology - Crypto '89, pp. 547-557, 1990.
6. G. Brassard, C. Crépeau and J.-M. Robert *Information Theoretic Reduction Among Disclosure Problems*, 27th FOCS, pp. 168–173, 1986.
7. G. Brassard, C. Crépeau and J.-M. Robert, *All-or-Nothing Disclosure of Secrets*, Advances in Cryptology - Crypto '86, LNCS 263, Springer, pp. 234–238, 1987.
8. G. Brassard, C. Crépeau and M. Santha, *Oblivious Transfer and Intersecting Codes*, IEEE Trans. on Inform. Theory, Vol. 42(6), pp. 1769–1780, 1996.
9. C. Cachin, *On the foundations of oblivious transfer*, Advances in Cryptology - Eurocrypt '98, LNCS 1403, pp. 361-374. Springer, 1998.
10. B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan, *Private Information Retrieval*, J. of ACM 45(6), 1998, pp. 965–981.
11. C. Crépeau, *Equivalence between two flavors of oblivious transfers*, Advances in Cryptology – Crypto '87 , LNCS 293, pp. 350–354, 1988.
12. C. Crépeau and J. Kilian, *Achieving oblivious transfer using weakened security assumptions*, FOCS '88, pp. 42–52, 1988.
13. S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM **28**, pp. 637–647, 1985.
14. Y. Gertner, S. Goldwasser, T. Malkin, *A Random Server Model for Private Information Retrieval or How to Achieve Information Theoretic PIR Avoiding Database Replication*,. RANDOM 1998, LNCS 1518, Springer, pp. 200–217.
15. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, *Protecting Data Privacy in Private Information Retrieval Schemes*, Proc. 30th STOC 1998, pp. 151–160.
16. O. Goldreich, *Secure Multi-Party Computation* (working draft) Version 1.1, 1998.
17. O. Goldreich and R. Vainish, *How to Solve any Protocol Problem - An Efficiency Improvement*, Advances in Cryptology - Crypto '87, LNCS 293, 1988, pp. 73–86.
18. R. Impagliazzo and S. Rudich, *Limits on the Provable Consequences of One-Way Permutations,* STOC '89, pp. 44–61, 1989.
19. J. Kilian, **Use of Randomness in Algorithms and Protocols**, MIT Press, Cambridge, Massachusetts, 1990.

20. M. Naor and B. Pinkas, *Oblivious Transfer and Polynomial Evaluation*, Proc. of the 31st ACM Symp. on Theory of Computer Science, 1999, pp. 245–254.
21. M. Naor, B. Pinkas and R. Sumner, *Privacy Preserving Auctions and Mechanism Design*, Proc. of the 1st ACM conf. on Electronic Commerce, November 1999, pp. 129–139 .
22. M. Naor and A. Wool, *Access Control and Signatures via Quorum Secret Sharing*, IEEE Transactions on Parallel and Distributed Systems 9(9), 1998, pp. 909–922.
23. M. O. Rabin, *How to exchange secrets by oblivious transfer*, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.
24. R. Rivest, *Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer*, manuscript. Available: `http://theory.lcs.mit.edu/~rivest/publications.html`

A Privacy for the Protocol which uses Full Polynomials

A.1 Sender's privacy

We first prove that if $d_y = 1$ then the chooser can learn only a single linear equation of m_0 and m_1, and then prove this for any degree d_y.

Lemma 1. *Let $Q(x, y)$ be a bivariate polynomial in which x is of degree d_x and y is linear. Denote by $P(y) = ay + b = Q(0, y)$ the polynomial which is equal to Q constrained to the line $x = 0$ (i.e. to the y axis). Any $2d_x + 1$ values $Q(x_i, y_i)$ where all the x_i-s are distinct and different from 0 do not yield more than a single linear equation on the coefficients a and b.*

Proof: Denote the polynomial as $Q(x,y) = \sum_{i=0}^{d_x} \sum_{j=0}^{1} a_{i,j} x^i y^j$ (i.e. $a = a_{0,1}$ and $b = a_{0,0}$). The $2d_x + 1$ values of $Q(x, y)$ define $2d_x + 1$ linear relations for the $2d_x + 2$ coefficients $a_{i,j}$. Assume wlog that these equations are linearly independent (otherwise Alice has made redundant queries). Note that this implies that not all y_i values are the same (if all y_i were the same then for all $1 \le i \le d_x + 1$ columns i and $d_x + 1 + i$ would have been linearly dependent).

The equations can be represented by a matrix A with $2d_x + 1$ rows and $2d_x + 2$ columns,

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{d_x} & y_1 & y_1 x_1 & \cdots & y_1 x_1^{d_x} \\ 1 & x_2 & \cdots & x_2^{d_x} & y_2 & y_2 x_2 & \cdots & y_2 x_2^{d_x} \\ \vdots & & & & \vdots & & & \\ 1 & x_{2d_x+1} & \ldots & x_{2d_x+1}^{d_x} & y_{2d_x+1} & y_{2d_x+1} x_{2d_x+1} & \ldots & y_{2d_x+1} x_{2d_x+1}^{d_x} \end{pmatrix} \begin{pmatrix} a_{0,0} \\ \vdots \\ a_{d_x,0} \\ a_{0,1} \\ \vdots \\ a_{d_x,1} \end{pmatrix} = \begin{pmatrix} P(x_1, y_1) \\ P(x_2, y_2) \\ \vdots \\ P(x_{2d_x+1}, \\ y_{2d_x+1}) \end{pmatrix}$$

We will prove that it cannot be the case that both $e_{0,0}$ and $e_{0,1}$ are defined by these equations. In other words, let $e_{i,j}$ be the $2d_x + 2$ entry vector in which all entries are 0 except for the $(i + 1 + j \cdot (d_x + 1))$'th entry which is 1 (i.e. only the coefficient of $a_{i,j}$ is 1). We will prove that the rows of the matrix A cannot span both $e_{0,0}$ and $e_{0,1}$.

The vector space is of dimension $2d_x + 2$, the vectors $e_{0,0}$ and $e_{0,1}$ are orthogonal and the rank of A is $2d_x + 1$ (all its rows are linearly independent). Therefore A spans a vector in the linear subspace generated by $e_{0,0}$ and $e_{0,1}$. Assume wlog that this vector is of the form $v = (\alpha, 0, \ldots, 0, 1, 0, \ldots, 0)$, i.e. that its first entry equals α and its $(d_x + 2)$'th entry equals 1. The vector v can be represented as a linear combination of the rows of A, and we can therefore replace one of the rows of A (say the last row) with v. Wlog we prove that this revised matrix (and therefore also A) cannot span $e_{0,0}$ in addition to v. Consider the matrix B' which is constructed by adding to the revised matrix the row $e_{0,0}$. It has $2d_x + 2$ rows and $2d_x + 2$ columns.

$$B' = \begin{pmatrix} 1 & 0 & \cdots 0 & 0 & 0 & \cdots 0 \\ \alpha & 0 & \cdots 0 & 1 & 0 & \cdots 0 \\ 1 & x_1 & \cdots x_1^{d_x} & y_1 & y_1 x_1 & \cdots y_1 x_1^{d_x} \\ \vdots & & & \vdots & & \\ 1 & x_{2d_x} & \ldots x_{2d_x}^{d_x} & y_{2d_x} & y_{2d_x} x_{2d_x} & \ldots y_{2d_x} x_{2d_x}^{d_x} \end{pmatrix}$$

The lemma is proven by the following claim, which shows that all the rows of B' are linearly independent. The proof appears in the full version of the paper.

Claim: The determinant of a matrix B' in which all the x_i-s are distinct and different from 0 and not all y_i values are equal, cannot be 0.

Following is a privacy theorem for polynomials in which the degree of y is greater than linear. The proof is similar to that of Lemma 1.

Theorem 3. *Let $Q(x, y)$ be a bivariate polynomial in which x is of degree d_x and y of degree d_y. Denote by $P(y) = \sum_{j=0}^{d_y} a_{0,j} y^j = P(0, y)$ the polynomial which is equal to Q constrained to the line $x = 0$ (i.e. to the y axis). Denote the coefficients of the elements free of x, i.e. $a_{0,0}, a_{0,1}, \ldots, a_{0,d_y}$, as the y coefficients. Then given any $2d_x + 1$ values $Q(x_i, y_i)$ where all the x_i-s are distinct and different from 0, at most a single linear relation is defined between the y coefficients.*

A.2 Chooser-servers collusion

The following theorem demonstrates that a collusion between the chooser and $d_x - \frac{2d_x}{d_y+1}$ servers (in addition to the k servers that were contacted by the chooser), cannot learn about m_0, m_1 more than can be learned by the chooser herself. The proof appears in the full version of the paper.

Theorem 4. *Let $Q(x, y)$ be a bivariate polynomial in which x is of degree d_x and y of degree d_y. Denote by $P(y) = \sum_{j=0}^{d_y} a_{0,j} y^j = P(0, y)$ the polynomial which is equal to Q constrained to the line $x = 0$ (i.e. to the y axis). Denote the coefficients of the elements free of x, i.e. $a_{0,0}, a_{0,1}, \ldots, a_{0,d_y}$, as the y coefficients. Then given any $2d_x + 1$ values $Q(x_i, y_i)$ where all the x_i-s are distinct and different from 0, and given the restrictions of $Q(x, y)$ to ℓ different x values, where $\ell \le d_x - \frac{2d_x}{d_y+1}$, at most a single linear relation is defined between the y coefficients.*