Oblivious Transfer with Adaptive Queries

Moni Naor * Benny Pinkas **

Dept. of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

Abstract. We provide protocols for the following two-party problem: One party, the sender, has N values and the other party, the receiver, would like to learn k of them, deciding which ones in an adaptive manner (i.e. the ith value may depend on the first $i-1$ values). The sender does not want the receiver to obtain more than k values. This is a variant of the well known Oblivious Transfer (OT) problem and has applications in protecting privacy in various settings.
We present efficient protocols for the problem that require an $O(N)$ computation in the preprocessing stage and fixed computation (independent of k) for each new value the receiver obtains. The on-line computation involves roughly $\log N$ invocations of a 1-out-2 OT protocol. The protocols are based on a new primitive, *sum consistent synthesizers*.

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 (or sometimes Bob or B), has an input and at the end of the protocol the other party, the receiver (or sometime Alice or A), learns some information about this input in a way that does not allow the sender Bob to figure out what she has learned. In particular, in 1-out-of-N OT (OT_1^N) protocols the sender's input consists of N strings $X_1, X_2, \ldots, X_N$; the receiver can choose to get one element X_I and does not learn any information about any other string, and the sender does not learn anything about I.

Recently Naor and Pinkas [23] suggested efficient constructions of OT_1^N based on protocols for OT_1^2. In particular the overhead of the initialization work of the sender is $O(N)$ invocations of a pseudo-random function (which can be realized by a block cipher) and the transfer requires only $\log N$ invocations of OT_1^2. This low overhead makes the OT_1^N protocol rather efficient even if N is very large (say, even if there are a billion elements for Alice to choose from).

Many applications might require the receiver to obliviously obtain several elements held by the sender. It is very inefficient to invoke independent runs

* Research supported by grant no. 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

** Research supported by an Eshkol Fellowship of the Israeli Ministry of Science. E-mail: bennyp@wisdom.weizmann.ac.il.

of a OT_1^N protocol because of the $O(N)$ overhead of the initialization phase (remember, N might be very large). An efficient protocol for k-out-of-N OT (OT_k^N) was described in [23], with complexity which is $O(N)$ for the initialization phase and $O(k)$ OT's for the transfer phase (we omit small logarithmic factors). However, that protocol required the receiver to get all k elements *simultaneously*. That is to say that she had to decide in advance which k elements to get, and was unable to adaptively pick the elements to be transfered to her[1].

We present several efficient protocols for k successive (possibly adaptive) oblivious transfers, an operation which we denote as $\mathrm{OT}_{k\times 1}^N$. The sender has to perform a single initialization of his input, which requires $O(N)$ work. Each transfer requires only about $\log N$ OT_1^2's. In some of the protocols the parameter k does not affect the complexity, and the protocol can even be used for N successive transfers.

1.1 Motivation

$\mathrm{OT}_{k\times 1}^N$ protocols are useful whenever the following three properties are required:

- A large database should be queried in an adaptive fashion.
- The privacy of the the party which performs the queries should be preserved.
- The owner of the database does not want to reveal to the other party more than a minimal amount of information.

We describe below three applications of this type:

Oblivious search: Bob owns a database which Alice wants to search. Suppose first that the database is sorted and Alice is using *binary search*. The two parties can invoke a $\mathrm{OT}_{\log N\times 1}^N$ protocol to perform this search without revealing to Bob the element that Alice is searching for, and while limiting Alice's knowledge to $\log N$ elements. Alternatively, the data elements can be ordered by a *two-level hash*, using two hash functions. The first function maps data elements into bins and the second function further maps the elements that were mapped to the same bin (this is how *perfect hash* functions [15] are constructed). Our protocols can be used to let Alice obliviously determine whether an element is is in the table. It first computes by herself the bin to which the element should have been mapped, then performs an oblivious transfer to get the (second) hash function that is used in that bin, and then another oblivious transfer to check the final location into which the element should have been mapped.

[1] The protocol uses several random mappings of the data elements to cryptographic keys. It consists of several stages, and the receiver learns the ith mapping only after stage $i-1$ is complete. For every k data elements that the receiver wishes to learn the protocol guarantees that the probability (taken over the mappings) that she is able to learn another element, is small. However once she knows all the mappings she is able to ask for k data elements that enable her to learn more elements. If this protocol were used for adaptive OT the receiver would have been able to learn all the mappings after the first transfer phase.

Patent or legal databases: Suppose that Bob holds a patent database. He does not want to give the whole database to other parties, but is willing to let other people search the database using a World-Wide-Web interface. Alice has a bright idea which she wants to patent and as a first step she wants to conduct a search for related patents. She is afraid that if she conducts the search on Bob's database he might learn what she is interested in and might reveal her idea. Alice and Bob can use $\mathrm{OT}^N_{k\times 1}$ to enable Alice to search Bob's database without revealing her queries to him. This solution also applies to searches in legal databases such as Lexis-Nexis or Westlaw.

Medical data: Suppose that Bob holds a database of medical information. For proprietary or privacy reasons Bob does not want to reveal the whole database to other parties but he is willing to let them use it for their research. Alice wants to conduct a research about a certain disease and has a list of patients that have this disease. She wants to search Bob's database for records related to these patients, but cannot reveal their identities to Bob. Alice and Bob can use $\mathrm{OT}^N_{k\times 1}$ to enable Alice to gather the relevant information from Bob's database.

1.2 Protocol structure

$\mathrm{OT}^N_{k\times 1}$ protocols contain two phases, for initialization and for transfer.

The initialization phase is run by the sender (Bob) who owns the N data elements. Bob typically computes a commitment to each of the N data elements, with a total overhead of $O(N)$. He then sends the commitments to the receiver (Alice).

The transfer phase is used to transmit a single data element to Alice. At the beginning of each transfer Alice has an input I, and her output at the end of the phase should be data element X_I. The transfer phase typically involves the invocation of several OT^m_1 protocols, where m is small (either constant or $\sqrt{N}$). In these OT's Alice obtains keys which enable her to open the commitment to X_I (the protocol uses commitments rather than simple encryptions in order to prevent Bob from changing the data elements between invocations of transfers). An $\mathrm{OT}^N_{k\times 1}$ protocol supports up to k successive transfer phases.

1.3 Correctness and Security Definitions

The definition of correctness is simple: The sender's input is $X_1, X_2, \ldots X_N$. At each transfer phase j (where $1 \leq j \leq k$) the receiver's input is $1 \leq I_j \leq N$ which may depend on all the previous information she learned. At the end of this transfer phase the receiver should obtain X_{I_j}. Note that this implies that the sender essentially commits to his inputs at the beginning of the protocol and cannot change the X's between transfers[2].

[2] Our protocols enable the sender to prevent the receiver from obtaining certain data elements at certain times (of course, independently of the items which were previously transferred). It is possible to amend our protocols to ensure an "all or nothing"

The definition of security is separated into two issues: protecting the receiver and protecting the sender.

The Receiver's Security - Indistinguishability: Given that under normal operation the sender gets no output from the protocol the definition of the receiver's security in a $\mathrm{OT}^N_{k\times 1}$ protocol is rather simple: for any step $1 \le t \le k$, for any previous items $I_1, \ldots, I_{t-1}$ that the receiver has obtained in the first $t-1$ transfers, for any $1 \le I_t, I'_t \le N$ and for any probabilistic polynomial time $\mathcal{B}'$ executing the sender's part, the views that $\mathcal{B}'$ sees in case the receiver tries to obtain X_{I_t} and in case the receiver tries to obtain $X_{I'_t}$ are computationally indistinguishable given $X_1, X_2, \ldots X_N$.

The Sender's Security - Comparison with Ideal Model: Here the issue is a bit trickier, since the receiver (or whatever machine is substituted for her part) gets some information and we want to say that the receiver does not get more or different information than she should. We make the comparison to the *ideal implementation*, using a trusted third party Charlie that gets the sender's input $X_1, X_2, \ldots X_N$ and the receiver's requests and gives the receiver the data elements she has requested. Our requirement is that for every probabilistic polynomial-time machine $\mathcal{A}'$ substituting the receiver there exists a probabilistic polynomial-time machine $\mathcal{A}''$ that plays the receiver's role in the ideal model such that the outputs of $\mathcal{A}'$ and $\mathcal{A}''$ are computationally indistinguishable. This implies that except for $X_{I_1}, \ldots, X_{I_k}$ that the receiver has learned the rest of $X_1, X_2, \ldots X_N$ are semantically secure.

1.4 Previous work

The notion of 1-out-2 OT (OT^2_1) was suggested by Even, Goldreich and Lempel [16] as a generalization of Rabin's "oblivious transfer" [26] (oblivious transfer was also developed independently by Wiesner in the 1970's, but not published till [28]). For an up-to-date and erudite discussion of OT see Goldreich [18]. 1-out-of-N Oblivious Transfer was introduced by Brassard, Crépeau and Robert [3, 4] under the name ANDOS (all or nothing disclosure of secrets). They used information theoretic reductions to construct OT^N_1 protocols from N applications of a OT^2_1 protocol (lately it was shown that such reductions must use at least N OT^2_1's in order to preserve the information theoretic security [12]). Our work builds upon the simple OT^N_1 and OT^N_k protocols of [23], which are based on efficient computationally secure reductions to OT^2_1.

1.5 Comparison to Private Information Retrieval (PIR)

Private Information Retrieval (PIR) schemes [8] allow a user to access a database consisting of N elements $X_1, X_2, \ldots X_N$ and read any element she wishes without the owner learning which element was accessed. Compared to Oblivious Transfer

behavior or the sender, i.e. that in each transfer phase he either lets the receiver learn any data element she chooses, or quits the protocol. Note that in any two-party protocol it is impossible to prevent a party from quitting the protocol.

protocols, the emphasis in PIR is on the communication complexity which must be $o(N)$. On the other hand, PIR schemes do not protect the owner of the database and do not prevent the user from learning more than a single element.

The first constructions of PIR schemes were based on using separate databases which do not communicate, but more recent constructions [20, 5] use only a single database. More recently attention was given to the question of protecting the database as well, i.e. that the user will not learn more than a single data element. A PIR scheme that enjoys this property is called SPIR (for Symmetric PIR). In [17] an information theoretic transformation of any PIR scheme into a SPIR scheme was proposed at the cost of increasing the number of servers (and introducing the assumption of a separation between the servers). The OT_1^N protocols of [23] enable a straightforward and efficient transformation of any PIR scheme into a SPIR scheme without increasing the number of database servers. In [7] PIR schemes with more involved queries (such as keyword retrieval) are discussed.

The $\mathrm{OT}_{k\times 1}^N$ protocols that we introduce would enable even more efficient future transformations from PIR to SPIR. They can be used to transform a protocol for k adaptive invocations of PIR to a protocol for k adaptive invocations of SPIR (currently there are no adaptive PIR protocols, but when such a protocol is introduced it would be possible to immediately transform it to a SPIR protocol).

On a more practical level, we believe that it is preferable to use the computation efficient OT_1^N and $\mathrm{OT}_{k\times 1}^N$ protocols rather than the communication efficient PIR protocols. The $\mathrm{OT}_{k\times 1}^N$ protocols require $O(N)$ communication at the end of the initialization phase and before the transfer phases begin, and the communication overhead of the transfer phases is negligible. For many applications the communication in the initialization phase is not an issue, and can be done using devices such as DVD's, jaz drives, or a fast communication network. In contrast, single server PIR protocols [20, 5] are very costly to implement since each transfer requires $O(N)$ exponentiations (which must be done after the receiver sends her query).

2 Cryptographic Tools

The protocols use three cryptographic primitives, *sum consistent synthesizers* which are introduced in Section 2.1, *1-out-of-2 Oblivious Transfer*, and *commitments*.

Protocols for **1-out-of-2 Oblivious Transfer** (OT_1^2) can be constructed under a variety of assumptions (see e.g. [3, 16, 1]). Essentially every known suggestion of public-key cryptography allows also to implement OT, (*but there is no general theorem that implies this state of affairs*). OT can be based on the existence of trapdoor permutations, factoring, the Diffie-Hellman assumption (both the search and the decision problems, the latter yields more efficient constructions) 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 [19] it follows that there is no black-box reduction of OT from one-way functions.

Commitment schemes are used to make sure that the sender does not change values between rounds. In a commitment scheme there is a *commit phase* which we assume to map a random key k and a value x to a string $commit_k(x)$, and a *reveal phase* which in our case would simply be revealing the key k which enables to compute x. The commitment should have the properties that given $commit_k(x)$ the value x is indistinguishable from random, and that it is infeasible to generate a commitment yielding two different x's. The commitment protocols we use are those of Chaum et al [6] in Section 4, and of Naor [22] in Section 5.

2.1 Sum Consistent Synthesizers

Our constructions of $\mathrm{OT}^N_{k\times 1}$ protocols are based on committing to the data elements using pseudo-random synthesizers with a special property, which we call *sum consistency*. Each transfer phase reveals information which is sufficient to reveal just one data element, but cannot be used in conjunction with information from other transfer phases. Sum consistent synthesizers can be constructed based on the Decisional Diffie-Hellman assumption or based on a random oracle. We present in Section 4 a $\mathrm{OT}^N_{k\times 1}$ protocol which uses synthesizers based on the Decisional Diffie-Hellman assumption. In Section 5 we present a construction of a $\mathrm{OT}^N_{k\times 1}$ protocol based on any sum consistent synthesizer.

Pseudo-random synthesizers: Pseudo-random synthesizers were introduced by Naor and Reingold in [24]. A pseudo-random synthesizer S is an efficiently computable function of ℓ variables $x_1, \ldots, x_\ell$, that satisfies the following property: *given polynomially-many uniformly distributed assignments to each of its input variables, the output of S on all the combinations of these inputs is pseudo-random.* Consider for example a synthesizer $S(x, y)$ with two inputs. Then for random sets of inputs $\langle x_1, \ldots, x_m\rangle, \langle y_1, \ldots, y_m\rangle$, the set $\{S(x_i, y_j) \,|\, 1 \le i, j \le m\}$ (which contains m^2 elements) is pseudo-random. That is to say that this set is indistinguishable from a truly random set[3].

We use this property of synthesizers in order to encrypt the data elements. For example, the elements can be arranged in a square and a random key could be attached to every row and every column (say, key R_i to row i, and key C_j to column j). The element in position (i, j) can be committed to using the combined key $S(R_i, C_j)$. It is ensured that the values of any set of combined keys do not leak information about the values of other combined keys.

We require an additional property from the pseudo-random synthesizers that we use: *they should have the same output for any two input vectors for which the sum of the input variables is the same.* For example, for a two dimensional

[3] This is a special property which does not hold for any pseudo-random generator G, since it involves inputs which are not independent.

synthesizer S this implies that for every x_1, y_1, x_2, y_2 that satisfy $x_1+y_1 = x_2+y_2$ it holds that $S(x_1, y_1) = S(x_2, y_2)$. More formally, the requirement is as follows:

Definition 1 ((sum consistent synthesizer)). *A function S (defined over ℓ inputs in a commutative group) is a* sum consistent synthesizer *if the following two conditions hold:*

- *S is a pseudo-random synthesizer.*
- *For every $x_1, \ldots, x_\ell$, and every $y_1, \ldots, y_\ell$ that satisfy $\sum_1^\ell x_i = \sum_1^\ell y_i$, it holds that*
$$S(x_1, x_2, \ldots, x_\ell) = S(y_1, y_2, \ldots, y_\ell)$$

The sum consistency property does not contradict the pseudo-randomness of the synthesizer. Suppose that S is a two dimensional sum consistent synthesizer, and let $\langle x_1, \ldots, x_m \rangle$ and $\langle y_1, \ldots, y_m \rangle$ be two random sets of inputs, whose size is polynomial in the security parameter of the synthesizer. Then there is an exponentially small probability that there is a pair $(x_i, y_j), (x'_i, y'_j)$ for which $x_i + y_j = x'_i + y'_j$. If ℓ is large enough then an ℓ dimensional synthesizer might contain such pairs with non-negligible probability. However then the range of possible ℓ-tuple inputs is exponentially large, and the probability of sampling such a pair is exponentially small.

Construction 1 (Random oracle based sum consistent synthesizer) *Let RO be a random oracle. A sum consistent synthesizer can be realized as*

$$S(x_1, x_2, \ldots, x_\ell) = RO(x_1 + x_2 + \cdots + x_\ell)$$

This simple construction means that (1) it is plausible to assume that such functions exist, and (2) suggests a heuristic approach for constructing such functions using a "complex" function (e.g. MD5). We prefer the number-theoretic construction that we present next, but on the downside it requires modular exponentiations which are more complicated to compute than common realizations of "complex" functions.

Another construction of sum consistent synthesizers is based on the synthesizers of [25] whose security relies on the Decisional Diffie-Hellman assumption (the DDH assumption is introduced and discussed in Section 4.1 below)[4].

Construction 2 (DDH based sum consistent synthesizer) *Let $\langle G_g, g \rangle$ be a group and a generator for which the Decisional Diffie-Hellman assumption holds. Let the input values $x_1, \ldots, x_\ell$ be elements in $\{1, \ldots, |G_g|\}$. A sum consistent synthesizer can be realized as*

$$S(x_1, x_2, \ldots, x_\ell) = g^{x_1 x_2 \cdots x_\ell}$$

[4] In fact, in the usual representation it seems that these synthesizers have the same output for input vectors for which the *multiplication* of the input elements is equal. It is possible to look at a different representation of the inputs which results in the same outputs for *sum consistent* inputs. Both representations are sufficient for our purposes.

We use sum consistent synthesizers S in the following way which is depicted in Figure 1. Suppose that the elements are arranged as entries in a square and are committed to using $S(R_i, C_j)$, as described above. Then for each transfer phase Bob can choose a random value r, and let Alice obtain one of the values $\langle R_1 + r, R_2 + r, \ldots, R_{\sqrt{N}} + r\rangle$, and one of the values $\langle C_1 - r, C_2 - r, \ldots, C_{\sqrt{N}} - r\rangle$. Alice can compute $S(R_i + r, C_j - r) = S(R_i, C_j)$ and obtain the key that hides data element (i, j). We should also make sure that Alice is unable to combine the values she obtains in different transfer phases, and this requirement complicates the protocols a little.

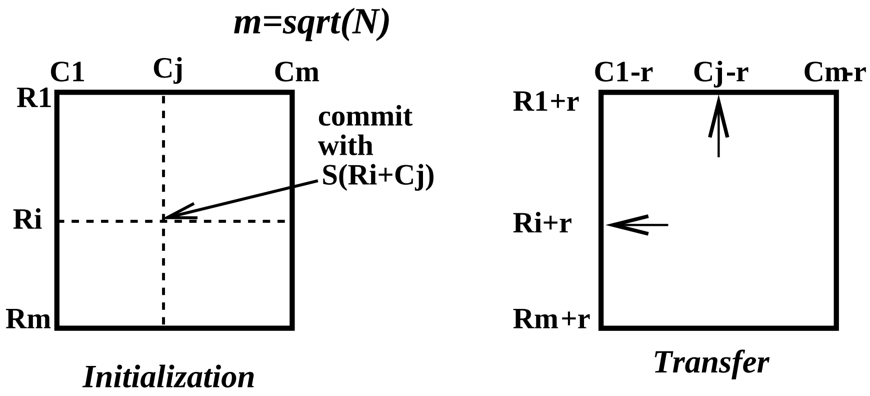

Fig. 1. The stages of the protocol.

3 The New $\mathrm{OT}^N_{k\times 1}$ Protocols

We present two types of $\mathrm{OT}^N_{k\times 1}$ protocols of the above flavor, protocols whose security depend on the Decisional Diffie-Hellman assumption, and protocols which can be based on any sum consistent synthesizer. We start with two DDH based protocols. These protocols are somewhat simpler than the general construction, since the hardness of the discrete log problem prevents some attacks which are possible in the general case. The DDH based protocols can be used to transfer any number of elements. That is, they are good for $\mathrm{OT}^N_{k\times 1}$ with any $k < N$, and their efficiency does not depend on k. We then present a $\mathrm{OT}^N_{k\times 1}$ protocol based on *any* sum consistent synthesizer. This protocol is secure for at most k transfers, where k is a parameter which affects (logarithmically) the complexity of the protocol.

4 Protocols based on the Decisional Diffie-Hellman Assumption

Following we present two protocols which are based on the Decisional Diffie-Hellman assumption. The protocols are very efficient, except that the basic operation they use is an exponentiation in a group in which the DDH assumption holds.

4.1 The Decisional Diffie-Hellman assumption

The Decisional Diffie-Hellman assumption (DDH assumption) is used as the underlying security assumption of many cryptographic protocols (e.g. the Diffie-Hellman key agreement protocol [11], the ElGamal encryption scheme [14], the Naor-Reingold pseudo-random functions [25], and the Cramer-Shoup construction of a cryptosystem secure against chosen ciphertext attacks [10]). This assumption is very appealing since there is a worst-case average-case reduction which shows that the underlying decision problem is either very hard on the average case or very easy in the worst case [25, 27].

The DDH assumption is thoroughly discussed in [2]. The assumption is about a cyclic group G_g and a generator g. Loosely speaking, it states that no efficient algorithm can distinguish between the two distributions $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^c\rangle$, where a, b, c are randomly chosen in $[1, |G_g|]$.

Our protocols essentially commit to the data elements using a key which is the output of the DDH based pseudo-random function or synthesizer of [25]. They are defined for a group G_g which is generated by a generator g, and for which the Decisional Diffie-Hellman assumption holds. It is very attractive to use in these protocols the non-interactive OT_1^2 protocols of Bellare and Micali [1]. The combination of these protocols with the proof techniques of [9] yields a very efficient OT_1^2 protocol which is based on the Decisional Diffie-Hellman assumption.

4.2 A two-dimensional protocol

The following protocol arranges the elements in a two-dimensional structure of size $\sqrt{N} \times \sqrt{N}$. It uses $\mathrm{OT}_1^{\sqrt{N}}$ as a basic primitive (which can be efficiently implemented using [23]). In Section 4.3 we present a protocol which arranges the elements in a $\log N$ dimensional structure and uses OT_1^2 as its basic primitive.

Protocol 1 *B's input is $X_1, X_2, ...X_N$, where $N = 2^\ell$. Rename these inputs as $\{x_{i,j} | 1 \le i, j \le \sqrt{N}\}$. In each invocation of the protocol the receiver A should learn a different element $X_{i,j}$.*

1. **Initialization:**
   (a) *B prepares $2\sqrt{N}$ random keys*

$$(R_1, R_2, \ldots, R_{\sqrt{N}}) \;\; (C_1, C_2, \ldots, C_{\sqrt{N}})$$

which are random integers in the range $1, \ldots, |G_g|$.

For every pair $1 \leq i, j \leq \sqrt{N}$, *B prepares a commitment key* $K_{i,j} = g^{R_i C_j}$, *and a commitment* $Y_{i,j}$ *of* $X_{i,j}$ *using this key,* $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.

(b) *B sends to A the commitments* $Y_{1,1}, \ldots, Y_{\sqrt{N},\sqrt{N}}$.

2. **Transfer** *(this part takes place when A wants to learn* $X_{i,j}$*).*
For each $X_{i,j}$ *that A wants to learn the parties invoke the following protocol:*
   (a) *B chooses random elements* r_R, r_C *(*r_R *is used to randomize the row keys, and* r_C *is used to randomize the column keys).*
   (b) *A and B engage in a* $\mathrm{OT}_1^{\sqrt{N}}$ *protocol for the values* $\langle R_1 \cdot r_R, R_2 \cdot r_R, \ldots, R_{\sqrt{N}} \cdot r_R \rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $R_i \cdot r_R$.
   (c) *A and B engage in a* $\mathrm{OT}_1^{\sqrt{N}}$ *protocol for the values* $\langle C_1 \cdot r_C, C_2 \cdot r_C, \ldots, C_{\sqrt{N}} \cdot r_C \rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $C_j \cdot r_C$.
   (d) *B sends to A the value* $g^{1/(r_R r_C)}$.
   (e) *A reconstructs* $K_{i,j}$ *as* $K_{i,j} = (g^{1/(r_R r_C)})^{(R_i r_R) \cdot (C_j r_C)}$, *and uses it to open the commitment* $Y_{i,j}$ *and reveal* $X_{i,j}$.

The receiver can clearly obtain any value she wishes to receive in the above protocol (the sum consistency ensures that she reconstructs the same key that was used by B for the commitment).

As for the complexity of the protocol, the initialization phase requires B to compute all N commitment keys, i.e. to compute N exponentiations (see in protocol 2 a discussion on how to efficiently implement these exponentiations by utilizing the structure of the exponents). Each transfer phase requires two invocations of an $\mathrm{OT}_1^{\sqrt{N}}$ protocol (which each requires $O(\sqrt{N})$ initialization work by B).

The privacy of A is guaranteed by the security of the $\mathrm{OT}_1^{\sqrt{N}}$ protocols which do not disclose to B information about A's choices. The use of commitments ensures that B cannot change the X_i's between transfers.

The security of B is guaranteed by the Decisional Diffie-Hellman assumption. To show this we compare A to a party A' who instead of running the transfer phases simply asks and receives the keys for k commitments. We prove that A does not gain more information than A'. To complete the proof of security it is required to simulate A' and show that she does not learn more than k committed values. We present a theorem which states the security property and a sketch of its proof. The formal proof of this property turns out to be rather subtle since it involves the problem of *selective decommitment* which is described below.

Theorem 1 *In k invocations of the above protocol A does not learn more information than a party A′ which can adaptively ask and obtain the commitment keys of k elements.*

Proof sketch: The security of the $\mathrm{OT}_1^{\sqrt{N}}$ protocol ensures that in each invocation of the transfer phase A can only learn a triplet of the following form $V_1 = (g^{1/r_1 r_2}, R_i r_1, C_j r_2)$, where $r1, r_2$ were chosen at random by B. This is equivalent to A learning a triplet $V_2 = (g^{R_i C_j / r_1 r_2}, r_1, r_2)$. A can easily compute from this information the following tuple $V_3 = (g^{R_i C_j}, g^{R_i C_j / r_1 r_2}, r_1, r_2)$ which of course does not contain more information than the key $g^{R_i C_j}$ alone (the key enables A to generate tuples in the same distribution as that of V_3). The pseudo-randomness of the output of the synthesizer ensures that A does not gain from k such keys more information about other keys than is available to A' which simply asks and receives k keys. It is left to prove that A does not learn anything from the commitments that she is given. This proof involves the selective decommitment problem, which we describe below.

The selective decommitment problem and its resolution: Consider the following seemingly benign problem. A party A receives N commitments to N values. She then chooses k of the commitments, receives their keys, and opens them. It seems obvious that A does not gain information on the unopened values. A should therefore be simulated in the ideal model by a party who can ask and get k of the committed values and sees nothing else. Although there does not seem to be any obvious way for A to take advantage of the fact that she sees the commitments before asking for the keys, it is not known how to prove that this is indeed the case. The problem is that it is hard to simulate the operation of A since it is unknown at the time of generating the commitments which of them she would ask to open. The number of the possible subsets of k commitments whose keys A might request is exponential. Note that it is possible to prove that A does not learn information about any other *single* element (or about a constant number of elements) since the simulation can with high probability "guess" the identity of this element. The selective decommitment problem is discussed in [13].

To enable the simulation it should be possible to open *in the simulation* any commitment to any value. Luckily, in the scenario of the $\mathrm{OT}^N_{k\times 1}$ there is an easy way to enable this. We describe below the solution for the DDH based protocol, which uses the trapdoor commitments of Chaum et al [6]. The solution for the protocols which use any sum consistent synthesizer is more complex and uses the commitments of Naor [22]. We do not describe it here.

In the case of DDH based $\mathrm{OT}^N_{k\times 1}$ the protocol should be amended as follows (but for the sake of clarity we do not describe these modifications in the body of the protocols in the paper): (1) In the beginning of the protocol A should send to B two values $g_1, g_2 \in G$ and prove (in zero-knowledge) that she knows the discrete log of g_2 to the base g_1. (In the simulation we would extract $\log_{g_1} g_2$ of the g_1, g_2 that would be used there). (2) B would use *trapdoor commitments* of the form $g_1^a g_2^b$ (which can be opened in an arbitrary way in the simulation, where $\log_{g_1} g_2$ is known). B commits to the value X_I in the following way, suggested in [6]: (i) chooses a random Y_I and computes $C_I = g_1^{X_I} g_2^{Y_I}$; (ii) takes the output of the synthesizer and uses it as a key to encrypt (X_I, Y_I) by xoring it, and sends these two results to A.

When the output of the synthesizer is computed it can be used to compute X_I and the commitment can be used to verify the result. In the simulation it is possible given X_I to find a Y_I which would be consistent with it, and give an output of the synthesizer which "decrypts" these values.

4.3 A protocol using OT_1^2:

The following protocol uses simple OT_1^2 in a straightforward manner.

Protocol 2 *B's input is $X_1, X_2, ...X_N$, where $N = 2^\ell$. In each invocation of the protocol the receiver A would like to learn a different element X_I.*

1. **Initialization:**
   (a) *B prepares ℓ random pairs of keys*
   $$(a_1^0, a_1^1), (a_2^0, a_2^1), \ldots (a_\ell^0, a_\ell^1)$$
   where for all $1 \le j \le \ell$ and $b \in \{0, 1\}$ each a_j^b is a random integer[5] *in the range $1, \ldots, |G_g|$.*
   For all $1 \le I \le N$ let $\langle i_1, i_2, \ldots i_\ell \rangle$ be the bits of I. B prepares a commitment key $K_I = g^{\Pi_{j=1}^{\ell} a_j^{i_j}}$, and a commitment Y_I of X_I using this key, $Y_I = commit_{K_I}(X_I)$.
   (b) *B sends to A the commitments $Y_1, Y_2, \ldots Y_N$.*
2. **Transfer:** *For each X_I that A wants to learn the parties invoke the following protocol:*
   (a) *B chooses random elements $r_1, \ldots, r_\ell$. Element r_i will be used to randomize the keys of the i'th coordinate.*
   (b) *For each $1 \le j \le \ell$, A and B engage in a OT_1^2 protocol for the values $\langle a_j^0 r_j, a_j^1 r_j \rangle$. If A wants to learn X_I she should pick $a_j^{i_j} r_j$.*
   (c) *B sends to A the value $g^{1/r_1 r_2 \cdots r_\ell}$.*
   (d) *A reconstructs K_I as $K_I = (g^{1/(r_1 r_2 \cdots r_\ell)})^{(a_1^{i_1} r_1) \cdots (a_\ell^{i_\ell} r_\ell)}$, and uses it to open the commitment Y_I and reveal X_I.*

The receiver can clearly obtain any value she wishes to receive in the above transfer protocol.

The initialization phase requires B to compute all N commitment keys. This can be done with exactly N exponentiations if the order in which the keys are computed follows a Gray code (i.e. the Hamming distance between each two consecutive words is 1). The computation can be further improved by using efficient techniques for raising the same number to many powers, or for raising many numbers to the same exponent (see [21] for a survey of such techniques). It is an interesting problem to find a way to utilize the special structure of the exponents (being the multiplications of all the subsets of ℓ elements) to compute the $N = 2^\ell$ commitment keys more efficiently.

The transfer part of the protocol requires $\ell = \log N$ invocations of an OT_1^2 protocol. In addition A and B should each compute a single exponentiation.

The privacy of A is guaranteed by the privacy of the OT_1^2 protocols. A is also ensured by the security properties of the commitments that B cannot change the values of the X_I's between transfers.

The security of B is guaranteed by the Decisional Diffie-Hellman assumption, and is proven in a similar way to the security of protocol 1.

[5] Note that B can set every a_j^0 to be equal to 1 without affecting the security of the system. This results in a reduction in the size of the keys that B needs to keep.

5 Protocols Based on Any Sum Consistent Synthesizer

We describe an *insecure* $\mathrm{OT}^N_{k\times 1}$ protocol which can be based on any sum consistent synthesizer, examine it, and transform it to a secure protocol.

5.1 An Insecure Protocol

The following protocol is **insecure**.

Protocol 3 (an insecure protocol)
B's input is $\{x_{i,j} | 1 \le i,j \le \sqrt{N}\}$. *Let* $S(x,y)$ *be a sum consistent synthesizer with two inputs.*

1. **Initialization:**
   (a) *B prepares* $2\sqrt{N}$ *random keys*
   $$(R_1, R_2, \ldots, R_{\sqrt{N}}) \ \ (C_1, C_2, \ldots, C_{\sqrt{N}})$$
   For every pair $1 \le i,j \le \sqrt{N}$, *B prepares a commitment key* $K_{i,j} = S(R_i, C_j)$ *and uses the key to generate a commitment* $Y_{i,j}$ *of* $X_{i,j}$, $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.
   (b) *B sends to A the commitments* $Y_{1,1}, \ldots, Y_{\sqrt{N},\sqrt{N}}$.
2. **Transfer:**
   The parties invoke the following protocol for each $X_{i,j}$ *that A wants to learn:*
   (a) *B chooses random elements* r_R, r_C, *such that* $r_R + r_C = 0$ *(*r_R *is used to randomize the row keys, and* r_C *is used to randomize the column keys).*
   (b) *A and B engage in a* $\mathrm{OT}^{\sqrt{N}}_1$ *protocol for the values* $\langle R_1 + r_R, R_2 + r_R, \ldots, R_{\sqrt{N}} + r_R\rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $R_i + r_R$.
   (c) *A and B engage in a* $\mathrm{OT}^{\sqrt{N}}_1$ *protocol for the values* $\langle C_1 + r_C, C_2 + r_C, \ldots, C_{\sqrt{N}} + r_C\rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $C_j + r_C$.
   (d) *A reconstructs* $K_{i,j}$ *as* $K_{i,j} = S(R_i + r_R, C_j + r_C))$, *and uses it to open the commitment* $Y_{i,j}$ *and reveal* $X_{i,j}$.

The security problem: The above protocol enables A to learn any value she wishes and protects her privacy. However the protocol is insecure for B because A can combine information she learns in different transfers, and use linear relations between the keys to learn more keys than she is entitled to. For example she can use the relation $(R_i + C_j) + (R_{i'} + C_{j'}) = (R_{i'} + C_j) + (R_i + C_{j'})$. She can thus ask to learn the keys that generate $K_{i,j}, K_{i',j}$, and $K_{i,j'}$ and use the information she receives to illegally compute the key $K_{i',j'}$.

5.2 Fixing the protocol

In order to transform the above protocol to be secure we use the following construction of a set of matrices. It is used to ensure the non-linearity of the information that A learns.

Construction 3 (k-out-of-N relation free matrices) .

- *Let $M_1, \ldots, M_t$ be t matrices of size $\sqrt{N} \times \sqrt{N}$, each containing a permutation of all the elements $1, \ldots, N$.*
- *Consider a collection of N vectors, which each have $2t\sqrt{N}$ coordinates corresponding to the rows and columns of each of the matrices. Denote the coordinates as $\{(i, j, k) \mid 1 \le i \le t,\ 1 \le j \le 2,\ 1 \le i \le \sqrt{N}\}$ (i.e. i represents the matrix, j is either a row or a column, and k is the row (column) index).*
- *For each element $1 \le x \le N$ define a vector v_x. Denote the row and column of x in matrix i as $R_i(x), C_i(x)$. The coordinates in v_x that correspond to the locations of x in the matrices are set to 1. I.e. the $2t$ coordinates $(i, 1, R_i(x))$ and $(i, 2, C_i(x))$ are 1 and all other coordinates are 0.*
- *The t matrices are* k-out-of-N relation free *if the vectors corresponding to any $k+1$ elements are linearly independent.*

For simplicity assume that the non-linearity is defined over the field $GF(2)$. The following lemma suggests a random construction of a k-out-N relation free set.

Lemma 1 *A random mapping of N elements to t matrices, where*

$$t \ge \frac{\log(N/(k+1))}{\log(\sqrt{N}/(k+1))},$$

is with high probability k-out-of-n relation free.

Proof: The vectors contain $2t\sqrt{N}$ coordinates. Call the coordinates that correspond to the row (or column) keys of a certain matrix a *region*. The vectors contain $2t$ regions, each with $\sqrt{N}$ coordinates. Each vector has in each region a single coordinate with a '1' value.

Consider a set of $k+1$ *linearly dependent* vectors. Then each coordinate either has no vectors in which it is set to 1, or it is set to 1 in at least two vectors. Therefore in each region the 1 values of all $k+1$ vectors are concentrated in at most $(k+1)/2$ coordinates.

Since the mapping to matrices locations is random the probability that this property holds for a single region is at most $(\frac{k+1}{2\sqrt{N}})^{(k+1)/2}$. The probability that it holds for all regions is therefore bounded by $(\frac{k+1}{2\sqrt{N}})^{(k+1)t}$. We apply the probabilistic method and require this probability to be smaller than the inverse of the number of subsets, $1/\binom{N}{k+1} \approx (\frac{k+1}{eN})^{k+1}$. This holds for $t \ge (\log(N/(k+1)))/\log(\sqrt{N}/(k+1))$. $\Box$

Note that this randomized construction is good for every $k = o(\sqrt{N})$. In particular, if $k < N^{1/4}$ then $t = 3$ satisfies the condition of the theorem, and for $k < N^{1/3}$ it suffices to use $t = 4$. It should be interesting to come up with an explicit construction of k-out-n relation free matrices (the problem seems to be related to the minimal weight of linear codes).

The transformation of protocol 3 is based on mapping the data elements to keys using a set of k-out-of-n relation free matrices. This ensures that the receiver can learn at most k linear equations which correspond to k relevant synthesizer inputs. The full description of the secure protocol appears in Section 5.3. We first describe only the differences from protocol 3.

In the *initialization* phase B uses a k-out-of-N relation free construction of t matrices and maps the N elements to entries in the t matrices. Namely, the element whose index is x is mapped in each matrix to the entry which contains x. The sender publishes the mapping and makes it available to A. B chooses random keys for every row and column from each matrix (a total of $2t\sqrt{n}$ keys). The commitment key for each element is the output of a sum consistent synthesizer with $2t$ inputs, which are the keys corresponding to the rows and columns to which the element is mapped in each of the matrices.

In each *transfer* phase B chooses $2t$ random hiding elements r_i whose sum is 0. A and B run $2t$ $\mathrm{OT}_1^{\sqrt{N}}$ protocols, which let A learn all the relevant inputs to the synthesizer, each summed with the corresponding random element r_i. The sum of these values equals the sum of the synthesizer inputs that generated the key to the commitment, and so A is able to open it.

The following theorem states that this protocol is secure if enough matrices are used (fortunately, if k is not too close to $\sqrt{N}$ very few matrices are needed).

Theorem 2 *The above* $\mathrm{OT}_{k\times 1}^{N}$ *protocol with a set of k-out-of-N relation free matrices is secure.*

Proof sketch: The properties of the $\mathrm{OT}_1^{\sqrt{N}}$ protocol ensure A's privacy. It is required to prove that A cannot use the information she obtained in k invocations of the transfer protocol to learn more than k elements.

The protocol is run with t matrices which are k-out-of-n relation free. B uses in each transfer phase a new set of $2t$ random hiding elements, r_i, which sum to 0. Therefore A can learn in each transfer phase at most a single linear equation which does not involve the hiding elements. This equation is the sum of one key from each row and from each column.

In order to avoid the selective decommitment problem we present the proof assuming that B generated N encryptions of the data items, and not commitments. The proof for a protocol which uses commitments is more involved and uses the commitments of [22].

First assume that in each transfer phase A follows the protocol and learns an equation which corresponds to a key which was used to encrypt a data element. The relation freeness ensures that the k equations that A learns do not span any key which was used for encrypting another data element.

Note that for every synthesizer output that A obtains she also learns the sum of its inputs. In order to handle this in the simulation the encryptions there should be done with a different set of keys for each data element. I.e. instead of using $2t\sqrt{N}$ keys there would be N sets of $2t$ keys, so that the Ith set is used for encrypting to X_I. When A asks in the simulation for element X_I she receives the sum of the keys in the Ith set. The pseudo-randomness of the synthesizer ensures that she cannot distinguish this view from the real distribution.

Consider now the case in which A does not play by the rules and in some transfer phases asks to learn linear equations which *do not correspond to any* *of the encryption keys.* Then in some later transfer phase she might learn an equation which, together with the previous equations she learned, reveals *several* keys. This might pose a problem in the simulation since it would be required to supply A with a single value which agrees with all the keys that she is supposed to learn in this phase. However observe that the j equations that A learns in the first j transfer phases span a subspace of dimension j of the N equations that were used for the encryptions. The value that A obtains in the jth phase of the simulation could be set to correspond to a new vector of this subspace which is learned by A in this phase . $\square$

5.3 A Protocol Based on Any Sum Consistent Synthesizer

The following protocol is a secure version of protocol 3.

Protocol 4 *B's input contains the N elements $\{x_{i,j} | 1 \leq i,j \leq \sqrt{N}\}$. B maps the inputs into a set of k-out-of-N relation free matrices, which contains t matrices. Let x^m_R and x^m_C denote the row and column into which x is mapped in matrix m.*

Let $S(x_1, \ldots, x_{2t})$ be a sum consistent synthesizer with $2t$ inputs.

1. **Initialization:**
   (a) *B prepares $2t\sqrt{N}$ random keys*

   $$(R^1_1, R^1_2, \ldots, R^1_{\sqrt{N}})\ (C^1_1, C^1_2, \ldots, C^1_{\sqrt{N}}), \ldots, (R^t_1, R^t_2, \ldots, R^t_{\sqrt{N}})\ (C^t_1, C^t_2, \ldots, C^t_{\sqrt{N}})$$

   For every pair $1 \leq i,j \leq \sqrt{N}$, B prepares a commitment key

   $$K_{i,j} = S(R^1_{(x_{i,j})^1_R}, C^1_{(x_{i,j})^1_C}, \ldots, R^t_{(x_{i,j})^t_R}, C^t_{(x_{i,j})^t_C})$$

   That is, the output of the synthesizer for the row and column keys that correspond to the locations of the input in each of the matrices. B prepares a commitment $Y_{i,j}$ of $X_{i,j}$ using this key, $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.
   (b) *B sends to A the commitments $Y_{1,1}, \ldots, Y_{\sqrt{N},\sqrt{N}}$.*
2. **Transfer:** *for each $X_{i,j}$ that A wants to learn the parties invoke the following protocol:*
   (a) *B chooses random elements $r^1_R, r^1_C, \ldots, r^t_R, r^t_C$, such that their sum is 0. (r^m_R is used to randomize the row keys of matrix m, and r^m_C is used to randomize the column keys of matrix m).*
   (b) *For every matrix $1 \leq m \leq t$, A and B engage in the following protocols:*
      – *A $\mathrm{OT}^{\sqrt{N}}_1$ protocol for the values $\langle R^m_1 + r^m_R, R^m_2 + r^m_R, \ldots, R^m_{\sqrt{N}} + r^m_R \rangle$. If A wants to learn $X_{i,j}$ she should pick $R^m_{(x_{i,j})^m_r} + r^m_R$.*
      – *A $\mathrm{OT}^{\sqrt{N}}_1$ protocol for the values $\langle C^m_1 + r^m_C, C^m_2 + r^m_C, \ldots, C^m_{\sqrt{N}} + r^m_C \rangle$. If A wants to learn $X_{i,j}$ she should pick $C^m(x_{i,j})^m_c + r^m_C$.*
   (c) *A reconstructs $K_{i,j}$ as*

   $$K_{i,j} = S(R^1_{(x_{i,j})^1_R} + r^1_R, C^1_{(x_{i,j})^1_C} + r^1_C, \ldots, R^t_{(x_{i,j})^t_R} + r^t_R, C^t_{(x_{i,j})^t_C} + r^t_C)$$

   and uses it to open the commitment $Y_{i,j}$ and reveal $X_{i,j}$.

References

1. M. Bellare and S. Micali, *Non-interactive oblivious transfer and applications*, Proc. Advances in Cryptology - Crypto '89, Springer-Verlag LNCS 435 (1990), 547-557.
2. D. Boneh, *The Decision Diffie-Hellman Problem*, Proc. of the Third Algorithmic Number Theory Symposium, Springer-Verlag LNCS 1423 (1998) 48–63.
3. G. Brassard, C. Crépeau and J.-M. Robert *Information Theoretic Reduction Among Disclosure Problems*, 27th Annual Symposium on Foundations of Computer Science, 1986, 168–173.
4. G. Brassard, C. Crépeau and J.-M. Robert, *All-or-Nothing Disclosure of Secrets*, Proc. Advances in Cryptology - Crypto '86, Springr-Verlag LNCS 263 (1987), 234–238.
5. C. Cachin, S. Micali and M. Stadler, *Computationally Private Information Retrieval With Polylogarithmic Communication*, Proc. Advances in Cryptology – Eurocrypt '99, Springr-Verlag LNCS 1592 (1999), 402–414.
6. D. Chaum, E. van Heijst, and B. Pfitzmann, *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, Proc. Advances in Cryptology – Crypto '91.
7. B. Chor, N. Gilboa, and M. Naor, *Private information retrieval by keywords*, manuscript, 1998.
8. B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan, *Private Information Retrieval*, JACM 45 (1998), 965–981. Preliminary version appeared in Proc. 36th IEEE Symposium on Foundations of Computer Science, 1995.
9. R. Cramer, I. Damgrd, B. Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, Proc. Advances in Cryptology – Crypto '94, Springr-Verlag LNCS 839 (1994), 174–187.
10. R. Cramer and V. Shoup, *A practical public key cryptosystem provably secure against adaptove chosen ciphertext attacks*, Proc. Advances in Cryptology – Crypto '98, Springr-Verlag LNCS 1462 (1998), 13–25.
11. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, vol. 22(6), 1976, 644–654.
12. Y. Dodis and S. Micali, *Lower bounds for oblivious transfer reductions*, Proc. Advances in Cryptology – Eurocrypt '99, Springr-Verlag LNCS 1592 (1999), 42–54.
13. C. Dwork, M. Naor, O. Reingold and L. Stockmeyer, *Magic functions*, manuscript, 1999.
14. T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, Proc. Advances in Cryptology – Crypto '84, Springr-Verlag LNCS 196 (1985), 10–18.
15. M. L. Fredman, J. Komlos and R. Szemeredi, *Storing a sparse table with $O(1)$ worst case access time*, JACM 31 (1984), 538–544.
16. S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM **28**, 1985, 637–647.
17. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, *Protecting Data Privacy in Private Information Retrieval Schemes*, Proc. of the 30th ACM Symp. on the Theory of Computing, 1998.
18. O. Goldreich, *Secure Multi-Party Computation* (working draft) Version 1.1, 1998. Available at `http://philby.ucsd.edu/books.html`
19. R. Impagliazzo and S. Rudich, *Limits on the Provable Consequences of One-Way Permutations,* Proc. of the 20th ACM Symp. on the Theory of Computing, 1988.

20. E. Kushilevitz and R. Ostrovsky, *Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval*, Proc. 38th IEEE Symp. on Foundations of Computer Science, 1997
21. A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, **Handbook of Applied Cryptography**, CRC Press, 1996.
22. M. Naor, *Bit Commitment Using Pseudo-Randomness*, Journal of Cryptology, vol. 4, 1991, 151–158.
23. M. Naor and B. Pinkas, *Oblivious Transfer and Polynomial Evaluation*, Proc. 31th ACM Symp. on Theory of Computing, 1999, 245–254.
24. M. Naor and O. Reingold, *Synthesizers and their application to the parallel construction of pseudo-random functions*, Proc. 36th IEEE Symp. on Foundations of Computer Science, 1995, 170–181.
25. M. Naor and O. Reingold, , *Number-Theoretic constructions of efficient pseudo-random functions*, Proc. 38th IEEE Symp. on Foundations of Computer Science, 1997, 458–467.
26. M. O. Rabin, *How to exchange secrets by oblivious transfer*, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.
27. M. Stadler, *Publicly verifiable secret sharing*, Proc. Advances in Cryptology – Eurocrypt '96, Springr-Verlag LNCS 1070 (1996), 190–199.
28. S. Wiesner, *Conjugate coding*, SIGACT News **15**, 1983, 78–88.