# Computationally Secure Oblivious Transfer*

Moni Naor† Benny Pinkas‡

**Abstract**

We describe a new construction for 1-out-of-$N$ oblivious transfer which is highly efficient – it requires only $\log N$ executions of a 1-out-of-2 oblivious transfer protocol. We also present a construction for $k$-out-of-$N$ oblivious transfer which is more efficient than $k$ repetitions of 1-out-of-$N$ oblivious transfer, and a construction for oblivious transfer with adaptive queries which is based on a new primitive – sum consistent synthesizers. The efficiency of the new oblivious transfer protocols makes them useful for a many applications. A direct corollary of the 1-out-of-$N$ oblivious transfer protocol is an efficient transformation of any Private Information Retrieval (PIR) protocol to a Symmetric PIR (SPIR) protocol without increasing the number of databases.

## 1 Introduction

An oblivious evaluation protocol for a function $f(\cdot,\cdot)$ allows two parties, Alice who knows $x$ and Bob who knows $y$, to jointly compute the value of $f(x,y)$ in a way that does not reveal to each side more information than can be deduced from $f(x,y)$. The fact that for every polynomially computable function $f(\cdot,\cdot)$ there exists such a (polynomially computable) protocol is one of the most remarkable achievements of research in foundations of cryptography. However, the resulting protocols are often not as efficient since the number of cryptographic operations performed is proportional to the *size of the circuit computing* $f(x,y)$ [48]. Even for relatively simple functions this may be prohibitively expensive. Therefore it is interesting to investigate for which functions it is possible to come up with a protocol that does not emulate the circuit for the function.

This paper presents efficient protocols for the basic two party problem of 1-out-of-$N$ oblivious transfer ($\mathrm{OT}_1^N$). In this problem Bob knows $N$ values and would like to let Alice choose any one of them in such a way that she does not learn more than one, and he remains oblivious to the value she chooses. This is a well known problem and we discuss

*This paper is the full version of the sections that describe oblivious transfer in "Oblivious Transfer and Polynomial Evaluation" (31st STOC, 1999) and of the extended abstract "Adaptive Oblivious Transfer" (Crypto '99).

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

‡STAR Lab, Intertrust Technologies, 4750 Patrick Henry Drive, Santa Clara CA 95054. Email: bpinkas@intertrust.com, benny@pinkas.net. Most of this work was done at the Weizmann Institute of Science and the Hebrew University of Jerusalem, and was supported by an Eshkol grant of the Israel Ministry of Science.

two new applications of it for Symmetric Private Information Retrieval (SPIR) and oblivious sampling (which is useful for example for checking the size of the index of a search engine). The paper also describes protocols for $k$-out-of-$N$ oblivious transfer ($\mathrm{OT}_k^N$), and for running oblivious transfer with $k$ *adaptive* queries ($OT_{k\times 1}^N$). All protocols are based on standard cryptographic assumptions.

**Organization:** In the rest of this section we discuss in more detail oblivious transfer and oblivious function evaluation, and present appropriate security definitions. Section 2 presents protocols for 1-out-of-$N$ and $k$-out-of-$N$ OT. Section 3 presents protocols for oblivious transfer with apadtive queries. Section 4 describes various applications for the protocols we present.

## 1.1 Oblivious Transfer

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 to figure out what she has learned. In this paper we are concerned with 1-out-of-2 OT protocols where the sender's input consists of two strings $(X_1, X_2)$ and the receiver can choose to get either $X_1$ or $X_2$ and learns nothing about the other string. Similarly, in 1-out-of-$N$ OT the sender has as input $N$ strings $X_1, X_2, \ldots X_N$ and the receiver can choose to get $X_I$ for some $1 \le I \le N$ of her choice, without learning anything about the other inputs and without the sender learning anything about $I$.

1-out-2 OT was suggested by Even, Goldreich and Lempel [23], as a generalization of Rabin's "oblivious transfer" [45] (the notion was also developed independently by Wiesner in the 1970's, but not published till [47]). 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 [28].

Since its proposal OT has enjoyed a large number of applications and in particular Kilian [34] and Goldreich and Vainish [31] have shown how to use OT in order to implement general oblivious function evaluation, i.e., to enable Alice and Bob to evaluate any function of their inputs without revealing more information than necessary. There are many application for 1-out-of-$N$ OT in case $N$ is relatively large, and two direct applications are described in Section 4. Another application is oblivious polynomial evaluation, which is described in [41].

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, 18, 17, 11]). This is of interest, given the possibility of implementing OT using "physical means", e.g. via a noisy channel or quantum cryptography. However, some of these reductions are not particularly efficient and in this paper we use non-information theoretic reductions, i.e., employ additional cryptographic primitives, to obtain very efficient reductions. In particular we apply pseudo-random functions which can be based on one-way functions.

Staying in the complexity based world, without physical realizations of OT channels but assuming that the adversary's power is limited to probabilistic polynomial time, OT can be implemented under a variety of assumptions (see e.g. [6, 23, 4]). Essentially every known suggestion of public-key cryptography allows also to implement OT (although in general public key cryptography and oblivious transfer are incomparable under black blox reductions [26]), and the complexity of 1-out-of-2 OT is typical of public-key operations [6, 4]. OT can be based on the existence of trapdoor permutations, factoring, Diffie-Hellman 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 [33] it follows that there is no black-box reduction of OT from one-way functions.

**Complexity:** Our working assumption is that 1-out-of-2 OT is an expensive operation compared to the evaluation of a pseudo-random function or a pseudo-random generator. This is justified both theoretically, by the separation of [33] mentioned above, and in practice, where one can model a pseudo-random function by very efficient block ciphers or keyed one-way hash functions which are several orders of magnitude more efficient than operations in public-key cryptography. Our goal is therefore to come-up with efficient constructions of 1-out-$N$ OT protocols from 1-out-2 OT protocols, where the number of invocations of the 1-out-2 OT protocol is small. For instance, the 1-out-$N$ OT constructions of [6, 8] need $N$ calls to the 1-out-2 OT protocol. In contrast our protocols need only $\log N$ calls to the 1-out-2 OT protocol plus $O(N)$ evaluations of a pseudo-random function (note that the construction of [8] of 1-out-2 string OT from 1-out-2 bit OT is efficient).

Another measure of complexity is communication complexity. In conjunction with recent work on private information retrieval (abbrev. PIR) we can obtain a protocol for 1-out-$N$ OT with $O(N^{\epsilon}m)$ communication overhead under the Quadratic Residuosity Assumption, based on the PIR construction of [35] (where $m$ is the security parameter, i.e. the length of the modulos), or an $O(m)$ protocol based on the PIR protocol of [12] (see Section 4.1.1).

### 1.2 Correctness and Security Definitions

When defining security for 1-out-of-$N$ oblivious transfer, there is no real difference between 1-out-of-2 OT and 1-out-of-$N$ OT and we will treat the former as 1-out-of-$N$ OT with $N = 2$.

We first define the input and output for 1-out-of-$N$ oblivious transfer. This is a two party protocol run between a receiver (sometimes called Alice, or $A$) and a sender (called Bob, or $B$).

- **Input**
  - Receiver: an index $1 \leq I \leq N$.
  - Sender: $N$ data elements $X_1, X_2, \ldots, X_N$.
- **Output**
  - Receiver: $X_I$.
  - Sender: nothing.

The definition of correctness is simple: At the end of a successful execution where all parties follow the protocol the receiver should obtain $X_I$ and be able to output it.

Oblivious transfer is a two party protocol and as such its definition of security can be derived from the security definition of such protocols. However, there are several obstacles: (i) Coming up with the precise definition of general multi-party protocols is a messy business and there is no consensus yet on the definition (see [3, 13, 28, 39], though the two-party case is less controversial). (ii) These definitions are rather complex, whereas the OT case is much simpler and does not require the full power of the general definitions. (iii) We feel that the constructions presented in this work are rather robust and should work with several definitions. However, the existing definitions are of course a good guideline and we follow most closely Goldreich's [28]. We keep the formalities at a bare minimum and ignore such important issues as uniformity.

The definition of security is separated into the issue of protecting the receiver and the issues of 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 1-out-$N$ OT protocol is rather simple: it is required that *for any* $1 \leq I, I' \leq N$ *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* $X_I$ *and in case the receiver tries to obtain* $X_{I'}$ *are computationally indistinguishable given* $X_1, X_2, \ldots X_N$.

**The Sender's Security - Comparison with the Ideal Model:** Here the issue is a bit trickier, since the receiver (or whatever machine which is substituted for her part) obtains 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.* The ideal implementation contains a trusted third party Charlie that gets the sender's input $X_1, X_2, \ldots X_N$ and the receiver's request $I$ and gives the receiver $X_I$. The 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 the single $X_I$ that the receiver has learned the rest of $X_1, X_2, \ldots X_N$ are semantically secure.

An issue that this definition does not handle is whether $\mathcal{A}'$ "knows" which input it has chosen, i.e. whether $I$ (for which $A$ learns $X_I$) is extractable. It turns out that our 1-out-of-$N$ construction enjoys this property *even if the original 1-out-of-2 OT protocol it is built from does not.*

## 2 1-out-of-$N$ Oblivious Transfer Protocol

In this section we describe efficient constructions of a 1-out-of-N Oblivious Transfer protocol, and a $k$-out-of-$N$ OT protocol. Section 4 describes two applications for these new protocols, and in particular a transformation of any Private Information Retrieval (PIR) protocol to a Symmetric PIR (SPIR) protocol, without using additional databases. A more involved application is that of oblivious polynomial evaluation which is described in [41].

The 1-out-of-$N$ OT protocol uses, in addition to the 1-out-of-2 OT, an additional cryptographic primitive, pseudo-random functions. A pseudo-random function is a function that cannot be distinguished from a truly random one by an observer granted access to the function in a black-box manner. Assume, for example, a function $F_K$, specified by a short key $K$ which can only be accessed by the observer by adaptively specifying inputs and obtaining the value of the function on these inputs. (See [29, 27, 36, 42, 43] for precise definitions and various constructions). Our working assumption is that block ciphers (such as DES, or triple DES) or *keyed* one-way hash functions (such as HMAC), can be modeled as a pseudo-random function. Therefore, the function $F_K(x)$ can be implemented by keying a block cipher with the key $K$ and encrypting $x$, or keying a hash function with $K$ and applying it to $x$. The evaluation of a pseudo-random function is therefore considerably cheaper than a typical public-key operation.

Let $\{F_K : \{0,1\}^m \mapsto \{0,1\}^m \mid K \in \{0,1\}^t\}$ be a family of pseudo-random functions. The 1-out-of-2 OT will be performed on strings of length $t$, since the transmitted strings are used as keys of $F$.

The main idea of the protocol is to have a rather small set of keys and to mask each input with a product of a *different* subset of the keys. The keys are not applied directly (which would leak information, for example if the keys were simply xored to the inputs and the receiver knows some of the $X_I$'s), but when a key $K$ is to be used to mask input $X_I$, the value $F_K(I)$ is used for masking. The complexity is measured in terms of the number of invocations of the 1-out-of-2 OT and the number of times the pseudo-random function $F_K$ is evaluated.

We present two protocols for 1-out-of-$N$ oblivious transfer. First a protocol which solves the problem using $\log N$ applications of 1-out-of-2 oblivious transfer, and then a recursive protocol which reduces 1-out-of-$N$ oblivious transfer to 1-out-of-$\sqrt{N}$ oblivious transfer.

## 2.1 A protocol for 1-out-of-$N$ oblivious transfer

**Protocol 2.1 (1-out-of-$N$ oblivious transfer)** *The input of the sender ($B$) is $X_1, X_2, \ldots, X_N$, where each $X_I \in \{0,1\}^m$ and $N = 2^\ell$. The receiver ($A$) would like to learn $X_I$;*

1. *$B$ prepares $\ell$ random pairs of keys*
$$(K_1^0, K_1^1), (K_2^0, K_2^1), \ldots (K_\ell^0, K_\ell^1)$$
*where for all $1 \le j \le \ell$ and $b \in \{0,1\}$, $K_j^b$ is a $t$-bit key to the pseudo-random function $F_K$.*
*For all $1 \le I \le N$ let $\langle i_1, i_2, \ldots i_\ell \rangle$ be the bits[1] of $I$. $B$ prepares $Y_I = X_I \oplus \bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$.*

2. *For $1 \le j \le \ell$, $A$ and $B$ engage in a 1-out-of-2 OT on the strings $\langle K_j^0, K_j^1 \rangle$. If $A$ would like to learn $X_I$ she should pick $K_j^{i_j}$.*

[1] To simplify the exposition we assume that the index $I$, which is in the range $[1, N]$, is represented by $\log N$ bits. The representation can be, for example, the binary representation of $I - 1$.

*3. $B$ sends to $A$ the strings $Y_1, Y_2, \ldots Y_N$.*

*4. $A$ reconstructs $X_I = Y_I \oplus \bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$.*

**Theorem 2.1** *Protocol 2.1 is a 1-out-of-N Oblivious Transfer protocol.*

**Proof:** It is straightforward to see that the receiver can get any value it wishes in the above protocol. As for the security analysis, it has to be argued that both the sender's security and the receiver's security are satisfied. Given that the 1-out-of-2 OT protocol maintains the computational indistinguishability of $A$'s choice, performing it $\log N$ times preserves the indistinguishability of *all* of $A$'s choices, i.e. for any $1 \leq I_1, I_2 \leq N$ the distributions that the sender $B$ sees when the receiver $A$ is retrieving $X_{I_1}$ or $X_{I_2}$ are computationally indistinguishable. This is proved in the following lemma. The sender's privacy is proved in Lemma 2.3.

**Lemma 2.2** *If the receiver's privacy is not preserved in protocol 2.1, then it is also not preserved in the* $\mathrm{OT}_1^2$ *protocol.*

**Proof of lemma:** Assume that the receiver's privacy is not preserved in protocol 2.1. Namely, there are two receiver's inputs $I_0, I_1$ for which the sender $B$ can distinguish the distributions that he sees when the receiver tries to retrieve $X_{I_0}$ or $X_{I_1}$. In this case, $B$ can be used to compromise the receiver's privacy in the 1-out-of-2 OT protocol: Let $m \geq 1$ be the Hamming distance between $I_0$ and $I_1$ and let $\langle J_0 = I_0, J_1, \ldots, J_m = I_1 \rangle$ be a sequence of $m \leq \ell$ indexes with Hamming distance 1 between each other. A hybrid argument shows that there must be a pair $J_I, J_{I+1}$ for which $B$ has a non negligible success probability in distinguishing between the case that $A$ is trying to learn $X_{J_i}$ and the case that she is trying to learn $X_{J_{I+1}}$. Now, in order to break the privacy of a receiver $A'$ in a given $\mathrm{OT}_1^2$ protocol, simulate the part of the receiver in a $\mathrm{OT}_1^N$ protocol with $B$. In the $\mathrm{OT}_1^2$ protocols that correspond to the bits in which $J_I$ agrees with $J_{I+1}$ ask to learn the input that corresponds to the bit of $J_I$ (which is equal to the bit of $J_{I+1}$). Run the receiver $A'$ in the $\mathrm{OT}_1^2$ protocol that corresponds to the bit in which $J_I$ differs from $J_{I+1}$. The output of $B$ distinguishes with non negligible probability between the two possible inputs of $A'$. This concludes the proof of the lemma. □

**Lemma 2.3** *If the sender's privacy is not preserved in protocol 2.1, then either the* $\mathrm{OT}_1^2$ *protocol does not provide sender's privacy, or the function $F$ is not pseudo-random.*

**Proof:** Assume that the $\mathrm{OT}_1^2$ protocol preserves the sender's privacy, i.e., that the receiver learns only one of the two inputs of the sender. We show that this implies that $F$ is not pseudo-random. Namely, that the receiver can be used as a black-box to decide whether a given a value is $F_K(I)$ or is random, for an unknown key $K$.

The assumption that the $\mathrm{OT}_1^2$ protocol preserves the sender's privacy implies that at the end of step 2 of protocol 2.1 the receiver learns only a single element of every pair $(K_j^0, K_j^1)$ of keys. We first extract which keys are learned by the receiver. Fix the state of the receiver at the end of this step, and run different experiments starting with this state. In order to find out which keys the receiver learned, run $2\ell$ experiments $\{C_{i,j} | 1 \leq j \leq \ell,\ 0 \leq i \leq 1\}$. In each experiment $C_{i,j}$ choose a random key $r$ and replace the values $F_{K_j^i}(I)$, in the generation of the encrypted inputs (the $Y_I$'s), with $F_r(I)$. If the distribution of the output of the receiver for experiment $C_{i,j}$ is different by non-negligible difference, from the distribution of her output in the runs of the protocol that use the original keys, then the receiver must have learned $K_j^i$ in step 2 (note that this phenomenon does not depend on the hardness of the pseudo-random function). At the end of this set of experiments extract, for each $1 \leq j \leq \ell$, the key, $K_j^0$ or $K_j^1$, that the receiver learned in step 2.

We now use the receiver to break the pseudo-randomness of the function $F$. Run $N$ new experiments, denoted $E_I$ for $1 \leq I \leq N$: In experiment $E_I$ replace the string $Y_I$ with a random value, and run the protocol to its end. For simplicity assume that the output of the receiver is either 0 or 1. Let $p_0$ be the probability that the output of the receiver in the original protocol is 1, and let $p_I$ be the probability that the output of experiment $E_I$ is 1. If the receiver can distinguish item $I$ from random, then $\varepsilon_I = |p_I - p_0|$ is non-negligible.

Let $I_0$ be the index that corresponds to the keys that the receiver learned in step 2 (that is to say that the receiver learned all the keys that encrypt $Y_{I_0}$). It holds that $\varepsilon_{I_0}$ is non-negligible. If the sender's privacy is not preserved in protocol 2.1, then there is an additional index $J$, such that $\varepsilon_J$ is also non-negligible.

The element at position $J$ can be used to break the pseudo-randomness of $F$, in the following way. Denote the Hamming distance between $I_0$ and $J$ as $h$. This means that $Y_J$ is encrypted using the outputs of $h$ pseudo-random functions whose keys are unknown to the receiver. Assume first that $h = 1$. Denote by $j'$ the bit in which $I_0$ and $J$ differ, and assume, wlog, that this bit is set to 0 in $I_0$. Therefore, the receiver did not learn $K_{j'}^1$ in step 2 of the protocol.

It is now possible to break the pseudo-randomness of $F_K$, keyed by a key $K$ whose value is unknown to us. First, for every $I \neq J$ in which the bit $j'$ is 1, ask to receive the value $F_K(I)$ and use it to replace $F_{K_{j'}^1}(I)$ in the encryption $Y_I$. Then ask to receive a challenge which is either $F_K(J)$ or random, and use it to replace $F_{K_{j'}^1}(J)$ in the encryption $Y_J$. Run the receiver on the resulting input. If her output is 1 then announce that the challenge is random, otherwise announce that it is $F_K(J)$.

If the Hamming distance $h$ is greater than 1, then a hybrid argument shows that there are two indexes $J_0, J_1$ with Hamming distance 1, for which $|p_{J_1} - p_{J_0}|$ is non-negligible. These two indexes can be used instead of $I_0$ and $J$ (in exactly the same manner) in order to break the pseudo-randomness of $F$. This concludes the proof of the lemma and of the theorem. □

**Complexity:** The computational complexity of the protocol is $N \log N$ evaluations of the pseudo-random function $F_K$ in the preprocessing of step 1, and $\log N$ invocations of the 1-out-of-2 OT protocol in the transfer stage. The communication overhead involves the sender sending to the receiver $N$ encryptions, one of each of his input items.

Yuval Ishai (personal communication) has suggested an improvement to the preprocessing complexity of the above protocol – each $Y_I$ should be masked by $\bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(pref_j(I))$ where $pref_j(I)$ denotes the first $j$ bits of $I$. The advantage of this proposal is that the total number of evaluations of the pseudo-random function is linear $N$. We suggest a different method for lowering the complexity in Protocol 2.2 in Section 2.1.1.

### 2.1.1 A recursive protocol for 1-out-of-$N$ oblivious transfer

A different protocol for 1-out-of-$N$ oblivious transfer is recursive, and reduces the 1-out-of-$N$ problem to two 1-out-of-$\sqrt{N}$ protocols. The recursion can be continued, or alternatively, the 1-out-of-$\sqrt{N}$ protocols can be run with $O(\sqrt{N})$ overhead. The preprocessing overhead of this protocol is only $O(N)$. The protocol is used in the $k$-out-of-$N$ protocol of Section 2.2.

**Protocol 2.2 (1-out-of-$N$ oblivious transfer)** *The sender ($B$'s) input is $X_1, X_2, ...X_N$ where each $X_I \in \{0,1\}^m$ and $N = 2^\ell$. The receiver ($A$) would like to learn $X_I$;*

1. *$B$ prepares two sets of $\sqrt{N}$ randomly chosen keys*

   $$R_1, R_2, \ldots, R_{\sqrt{N}}$$

   *(for the rows) and*

   $$C_1, C_2, \ldots, C_{\sqrt{N}}$$

   *(for the columns), each $t$-bits long. $B$ arranges the $N$ inputs in a $\sqrt{N} \times \sqrt{N}$ matrix, i.e. each input is indexed now as $X_{i,j}$. Bob sets $Y_{i,j} = X_{i,j} \oplus F_{R_i}(j) \oplus F_{C_j}(i)$.*

2. *$A$ and $B$ engage in a 1-out-of-$\sqrt{N}$ $OT$ protocol on $R_1, R_2, \ldots R_{\sqrt{N}}$ and on $C_1, C_2, \ldots C_{\sqrt{N}}$ (e.g. by invoking Protocol 2.1 twice). If $A$ would like to learn $X_{i,j}$ she should pick $R_i$ and $C_j$.*

3. *$B$ sends to $A$ all the $Y_{i,j}$'s.*

4. *$A$ reconstructs $X_{i,j} = Y_{i,j} \oplus F_{R_i}(j) \oplus F_{C_j}(i)$.*

It is clear that the receiver can get any value it wishes in the above protocol. The complexity of the protocol is $2N$ evaluations of $F_K$ for preprocessing, and two invocations of the 1-out-of-$\sqrt{N}$ protocol for the transfer. Implementation of the 1-out-of-$\sqrt{N}$ protocols using Protocol 2.1 involves $\sqrt{N}\log N$ evaluations of the function $F_K$ and $\log N$ calls to the 1-out-of-2 OT. Protocol 2.2 can be described as being two-dimensional whereas Protocol 2.1 is $(\log n)$-dimensional. The proofs of security are straightforward modifications of the proofs of Protocol 2.1.

## 2.2 $k$-out-of-$N$ Oblivious Transfer

Some applications can use a $k$-out-of-$N$ oblivious transfer protocol, i.e., a protocol which enables the receiver to choose *any* $k$ out of $N$ input strings. It is possible to achieve this task by repeating Protocol 2.1 $k$ times independently[2], but the overhead would be $kN \log N$ (or $kN$) applications of a pseudo-random function for preprocessing, and $k \log N$ oblivious transfers for the $k$ transfers. Since $N$ might be very large, the preprocessing overhead might be prohibitively expensive. It is, therefore, interesting to investigate whether it is possible

[2] A problem with repeating Protocol 2.1 $k$ times is that the sender might not be *consistent* from round to round and thus induce a distribution on the receiver's output that is impossible in the ideal implementation. A simple solution is that the sender commits to the $X_I$'s (once), and protects the keys that open the commitments using the masks $\bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$.

to keep the price low in terms of pseudo-random function evaluations while keeping the number of 1-out-of-2 OTs proportional to $k$. Following we describe a $k$-out-of-$N$ scheme which achieves this property.

The scheme as described works for $k \ll N$. Consider first running Protocol 2.2 to perform a $k$-out-of-$N$ oblivious transfer. Suppose that in step 2 we let $A$ obtain $k$ of the keys $R_1, R_2, \ldots R_{\sqrt{N}}$ and $k$ of the keys $C_1, C_2, \ldots C_{\sqrt{N}}$, by repeating the 1-out-$\sqrt{N}$ protocol $k$ times *independently*. Then $A$ would be able to obtain any $k$ values she wishes. However, she gets more information than that: if she is interested in $X_{i,j}$ and $X_{i',j'}$ then, by learning the keys $(R_i, R_{i'}, C_j, C_{j'})$ she can actually also learn $X_{i,j'}$ and $X_{i',j}$. The total number of items she can learn is, therefore, $k^2$, but all other values remain hidden. Furthermore, after the execution of the protocol these $k^2$ values are well defined. The main idea we use is to use this protocol to learn *shares* of the inputs, and repeat the protocol again after randomly permuting the locations in the matrix. It is important that the permutation be revealed *only after* the first protocol is executed, so that the receiver has effectively committed to a set $S \subset \{1, \ldots N\}$ of $k^2$ values.

To get some basic intuition why this protocol should work, note that a good permutation might be one where no two elements of $S$ are mapped to the same column or to the same row. The probability that there are two different values in $S$ which are mapped to the same row is at most $\frac{k^4}{\sqrt{N}}$. If this procedure is repeated several times, each time with independent keys and without revealing the new permutation before the previous stage is over, the probability can be made arbitrarily small.

**Protocol 2.3 ($k$-out-of-$N$ oblivious transfer)**
*The input to $B$ is $X_1, X_2, \ldots X_N$, where $N = 2^\ell$, and $A$ would like to learn $X_{I_1}, \ldots X_{I_k}$*

- *Repeat for $j = 1$ to $W$*
  1. *$B$ chooses two random sets of $\sqrt{N}$ keys, $R_1^j, R_2^j, \ldots R_{\sqrt{N}}^j$ and $C_1^j, C_2^j, \ldots C_{\sqrt{N}}^j$.*
  2. *$B$ chooses a random permutation $\sigma_j$ on $1 \ldots N$. For any $I$ we let $\sigma_{j,R}(I)$ be the first $\ell/2$ bits of $\sigma_j(I)$ and $\sigma_{j,C}(I)$ be the second $\ell/2$ bits of $\sigma_j(I)$. Bob arranges the $N$ inputs in a $\sqrt{N} \times \sqrt{N}$ matrix, i.e. input $X_I$ is indexed now as $X_{\sigma_{j,R}(I), \sigma_{j,C}(I)}$*
  3. *$B$ sends $\sigma_j$ to $A$.*
  4. *$A$ and $B$ engage in two $k$-out-of-$\sqrt{N}$ OT protocols, one for the keys $R_1^j, R_2^j, \ldots R_{\sqrt{N}}^j$, and the other for the keys $C_1^j, C_2^j, \ldots C_{\sqrt{N}}^j$. For all $1 \le I \le k$ $A$ picks $R_{\sigma_{j,R}(I)}^j$ and $C_{\sigma_{j,C}(I)}^j$.*
- *$B$ computes $Y_I = X_I \bigoplus_{j=1}^{W} (F_{R_{\sigma_{j,R}(I)}^j}(I) \oplus F_{C_{\sigma_{j,C}(I)}^j}(I))$, and sends them to $A$.*
- *To reconstruct the desired inputs $A$ computes for each $I_1, I_2, \ldots I_k$ the value*

$$X_{I_i} = Y_{I_i} \oplus \bigoplus_{j=1}^{W} (F_{R_{\sigma_{j,R}(I_i)}^j}(I_i) \oplus F_{C_{\sigma_{j,C}(I_i)}^j}(I_i))$$

The protocol preserves the privacy of the receiver since the oblivious transfer protocols that are run have this property (the proof is similar to that of Protocol 2.1). As for the privacy of the sender, we present two different proofs: First a simple proof for the case $k < N^{1/8}$, and then a more intricate proof for the case $k < N^{1/4}$.

**Theorem 2.4** *For $k \le N^{1/8-\varepsilon}$, Protocol 2.3 with $W = w+1$ rounds, where $w > \log(1/\delta)/(4\varepsilon \log N)$, is a $k$-out-of-$N$ oblivious transfer protocol which provides sender security with probability $1-\delta$.*

**Proof:** Let $S_1$ be the set of $k^2$ input elements which are mapped by the first permutation $\sigma_1$, to the rectangle of size $k \times k$ whose keys are learned by the receiver. In order for the receiver to be able to learn, in a subsequent round, the keys of more than $k$ of these elements, at least two of the elements must be mapped to the same row. The probability that there are two elements of $S_1$ which are mapped by a random permutation to the same row, is at most $k^4/\sqrt{N}$. The probability that this happens in each of the $w$ subsequent rounds of the protocol, is at most $(k^4/\sqrt{N})^w \le N^{-4\varepsilon w}$ and should be smaller than $\delta$. Setting $w > \log(1/\delta)/(4\varepsilon \log N)$ satisfies this requirement. $\square$

**Theorem 2.5** *For $k \le N^{1/4-\varepsilon}$, Protocol 2.3 with $W = w+2$ rounds, where $w > \log(1/\delta)/(2\varepsilon \log N)$, is a $k$-out-of-$N$ oblivious transfer protocol which provides sender security with probability $1-\delta$.*

**Proof:** The proof is composed of two stages. The first stage shows that, with high probability, after two rounds of the protocol the receiver knows the keys of at most $O(k)$ input elements. The second stage shows that after sufficiently more rounds, the receiver knows the keys of only $k$ elements.

**Lemma 2.6** *After two rounds of protocol 2.3 it holds with overwhelming probability that the receiver knows the keys of at most $(2e+1)k$ input elements.*

**Proof of lemma:** Let $S_1$ be the set of $k^2$ input elements which were mapped, in the first permutation $\sigma_1$, to the rectangle of size $k \times k$ whose keys are learned by the receiver. Examine first the probability that the second permutation maps more than $2ek$ of these elements to a rectangle of size $k \times k$. The probability that a single input element is mapped to the rectangle is $p = (k/\sqrt{N})^2 = N^{-(1/2+2\varepsilon)}$. Let $x_i$ be a random variable which is set to 1 if the $i$'th element in $S_1$ is mapped to the rectangle, and is 0 otherwise. Therefore, $\Pr(x_i = 1) = p$. Define the random variable $X$ as the number of elements of $S_1$ which are mapped to the rectangle, i.e., $X = \sum_{i=1}^{k^2} x_i$.

We use the Chernoff bound to estimate the probability that $X \ge 2ek$. Consider the following version of the Chernoff bound (see [1], Theorem A.12): *Let $x_1, \ldots, x_\ell$ be mutually independent random variables, with*

$$\begin{aligned} Pr[x_i = 1] &= p \\ Pr[x_i = 0] &= 1-p \end{aligned}$$

*Then, for all $\beta \ge 1$*

$$Pr\left[\frac{1}{\ell}\sum_{i=1}^{\ell} X_i \ge \beta p\right] < \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^{p\ell}.$$

Returning to our proof, examine the following probability

$$\Pr(X \geq 2ek) = \Pr(\frac{1}{k^2}\sum_{i=1}^{k^2} x_i \geq 2e/k) = \Pr(\frac{1}{k^2}\sum_{i=1}^{k^2} x_i \geq p\beta)$$

where $\beta = 2e/(kp) = 2eN^{1/4+3\varepsilon}$. Therefore,

$$\Pr(X \geq 2ek) < (\frac{e^{\beta-1}}{\beta^\beta})^{pk^2} < (\frac{e}{\beta})^{\beta pk^2} = (\frac{e}{\beta})^{2ek} < N^{-(e/2)\cdot(N^{1/4}-\varepsilon)}$$

Now, the number of possible rectangles is $[\binom{\sqrt{N}}{k}]^2 \approx N^{N^{1/4}}$. Therefore, the probability that there is a rectangle which contains more than $2ek$ elements is about $N^{N^{1/4}}N^{-(e/2)N^{1/4}} = N^{-0.7N^{1/4}}$, and is negligible. This concludes the proof of the lemma.

The remainder of the proof follows the lines of the proof of Theorem 2.4. It bounds the probability that the receiver can learn the keys of more than $k$ of the $2ek$ elements whose keys she knows after the first two rounds. The probability that this happens in a single round is smaller than the probability that two of the elements are mapped to the same row, i.e., smaller than $\frac{(2ek)^2}{\sqrt{N}} \approx 30N^{-2\varepsilon}$. The probability that this happens in $w$ rounds is $(30N^{-2\varepsilon})^w$ and should be bounded by $\delta$. Therefore, if $N^{2\varepsilon} \leq 30$ then in order for a protocol to be secure with probability $1-\delta$, it should set $w \geq \log(1/\delta)/(2\varepsilon \log N)$ (or more accurately, $w \geq \log(1/\delta)/(2\varepsilon \log N + \log 30)$). □

**Complexity:** The protocol consists of $W$ rounds of communication. The preprocessing stage requires $2WN$ invocations of a pseudo-random function. The transfer stages require a total of $2W$ executions of $k$-out-of-$\sqrt{N}$ oblivious transfer protocols. Each of these could be performed recursively, or by running $k$ invocations of 1-out-of-$\sqrt{N}$ oblivious transfer. The latter option has a total overhead of $2Wk\sqrt{N}$ applications of a pseudo-random function, and $Wk \log N$ invocations of $\mathrm{OT}_1^2$.

# 3 Oblivious Transfer with Adaptive Quqeries

## 3.1 Introduction

The $k$-out-of-$N$ oblivious transfer protocol (Protocol 2.3) enables the receiver to *simultaneously* obtain any $k$ out of the $N$ values. However it is not secure to use this protocol to perform $k$ adaptive transfers of a single value, since the privacy of the sender is based on hiding the permutations from the receiver.

This section presents several efficient protocols for $k$ successive (possibly adaptive) oblivious transfers, an operation which we denote as $OT_{k\times 1}^N$. The sender has to perform a single initialization of his input, which requires $O(N)$ work. Each transfer requires only $\log N$ $\mathrm{OT}_1^2$'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.

### 3.1.1 Motivation

Adaptive oblivious transfer ($OT^N_{k\times 1}$) protocols are useful whenever the following three requirements hold:

- 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.

Three different applications of this type are described in Section 4.2.

### 3.1.2 Protocol structure

Adaptive oblivious transfer ($OT^N_{k\times 1}$) protocols are composed of 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 protocol uses commitments rather than simple encryptions in order to prevent Bob from changing the data elements between different invocations of the transfer stage.)

The transfer phase is used to transfer 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 $X_I$. An $OT^N_{k\times 1}$ protocol supports up to $k$ successive transfer phases.

### 3.1.3 Correctness and Security Definitions

The correctness and security definitions are slight modifications of the definitions for the $\mathrm{OT}^N_1$ case, which take into account the fact that the receiver's operation can be adaptive.

The definition of correctness is simple: The sender's input is $X_1, X_2, \ldots X_N$. At each transfer phase the receiver's input is $1 \leq I \leq N$, and at the end of this transfer the receiver should obtain $X_I$ and be able to output it. 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.

The definition of security is separated into the issue of protecting the receiver and the issue of protecting the sender, and is based on adapting the definition of oblivious transfer to the adaptive case.

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

**The Sender's Security - Comparison with Ideal Model:** We make again a 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. The 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 the $X_{I_1}, \ldots, X_{I_k}$ that the receiver has learned the rest of $X_1, X_2, \ldots X_N$ are semantically secure.

## 3.2 The $OT^N_{k\times 1}$ Protocols

The protocols use three cryptographic primitives, *sum consistent synthesizers* which are introduced in Section 3.2.1, *1-out-of-2 Oblivious Transfer* (which were described in Section 1.1), and *commitments.*

*Commitment schemes* are used to ensure that the sender does not change his inputs 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 infeasible to generate open a commitment yielding two different $x$'s. The commitment protocols we use are those of Chaum, van Heijst and Pfitzmann [14] in Section 3.3 and of Naor [40] in Section 3.4.

### 3.2.1 Sum Consistent Synthesizers

The $OT^N_{k\times 1}$ protocols are based on encrypting 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 decrypt just a single 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 function modeled as a random oracle. Section 3.3 presents an $OT^N_{k\times 1}$ protocol which uses synthesizers based on decisional Diffie-Hellman assumption, and section 3.4 describes a construction of an $OT^N_{k\times 1}$ protocol based on any sum consistent synthesizer.

#### Pseudo-random synthesizers

Pseudo-random synthesizers were introduced by Naor and Reingold in [42]. A pseudo-random synthesizer $S$ is an efficiently computable function of $\ell$ 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 $\{S(x_i, y_j) \mid 1 \leq i, j \leq m\}$ of $m^2$ values is pseudo-random, i.e. indistinguishable from truly random[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 can 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 encrypted using the *combined* key $S(R_i, C_j)$. It is ensured that the values of some combined keys do not leak information about the values of other combined keys.

#### Sum consistent synthesizers

We require an additional property from the pseudo-random synthesizer functions that we use: *that they 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 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 3.1 (sum consistent synthesizer)** *A function $S$ (defined over $m$ inputs in a commutative group) is a* sum consistent synthesizer *if the following two conditions hold:*

- *$S$ is a pseudo-random synthesizer.*
- *For every $x_1, \ldots, x_m$, and every $y_1, \ldots, y_m$ that satisfy $\sum_1^m x_i = \sum_1^m y_i$, it holds that*

$$S(x_1, x_2, \ldots, x_m) = S(y_1, y_2, \ldots, y_m)$$

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

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

This simple construction means that (1) it is plausible to assume that such functions exists, and (2) suggests a heuristic approach for constructing them using a "complex" function (e.g. SHA1). We prefer the number-theoretic construction that we present next, but on the downside it requires exponentiations which are more complicated to compute than common realizations of "complex" functions.

The following construction introduces sum consistent synthesizers based on the synthesizers of [44] whose security relies on the decisional Diffie-Hellman assumption (the DDH assumption is introduced and discussed in Section 3.3.1 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 $x_1, \ldots, x_m$ be elements in $\{1, \ldots, |G_g|\}$. A sum consistent synthesizer can be realized as*

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

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

[4]In fact, in the usual representation 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 out purposes.

A sum consistent synthesizer $S$ is used in adaptive oblivious transfer in the following way. Suppose that the elements are arranged and encrypted as entries in a square, as described above. Then for each transfer protocol 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 $(i, j)$. It should be endured that Alice is unable to combine the values she obtains in different transfer phases, and this requirement complicates the protocols a little.

### 3.2.2 The protocols

We present two types of $OT^N_{k\times 1}$ protocols, protocols whose security depends 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 $OT^N_{k\times 1}$ with any $k < N$, and their efficiency does not depend on $k$. We then present a $OT^N_{k\times 1}$ protocol based on any sum consistent synthesizer. This protocol is secure for at most $k$ transfers, where $k$ is a parameter which affects the complexity of the protocol.

## 3.3 Protocols based on the Decisional Diffie-Hellman Assumption

This section presents two protocols which are based on the decisional Diffie-Hellman assumption. The protocols are very efficient, except for the fact that the basic operation they use is an exponentiation in a group in which the DDH assumption holds.

### 3.3.1 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 [20], the ElGamal encryption scheme [22], the Naor-Reingold pseudo-random functions [44], and the Cramer-Shoup construction of a cryptosystem secure against chosen ciphertext attacks [16]).

The DDH assumption is thoroughly discussed in [10]. The assumption is about a cyclic group $G$ 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 $[1, |G|]$.

Our protocols essentially encrypt the data elements using a key which is the output of the DDH based pseudo-random synthesizer of [44].

### 3.3.2 A two-dimensional protocol

The following protocol arranges the elements in a two-dimensional structure of size $\sqrt{N} \times \sqrt{N}$. It uses $OT^{\sqrt{N}}_{k\times 1}$ as a primitive. This primitive can either be realized recursively, or by $k$

repetitions of a $OT_1^{\sqrt{N}}$ protocol. In Section 3.3.4 we present a protocol which arranges the elements in a $\log N$ dimensional structure and uses $\mathrm{OT}_1^2$ as its basic primitive.

Let $G$ be a group, and let $G_g$ be a subgroup of $G$ generated by $g$ in which the decisional Diffie-Hellman assumption holds.

**Protocol 3.1** *$B$'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 $A$ would like to 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 $1 \le i, j \le \sqrt{N}$, $B$ 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 an input element). 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 $OT_1^{\sqrt{N}}$ protocol on 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 $OT_1^{\sqrt{N}}$ protocol on 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 $X_{i,j}$.*

**Theorem 3.1** *Protocol 3.1 is an adaptive oblivious transfer protocol, which can be used for $N$ adaptive transfers.*

**Proof:** The receiver can clearly use the protocol to obtain any value it wishes to receive from the sender. The use of commitments ensures that the sender cannot change the input elements from round to round.

The privacy of the receiver $A$ is guaranteed by the security of the $OT_1^{\sqrt{N}}$, as is shown by lemma 3.2. The privacy of the sender $B$ is guaranteed by lemma 3.3.

**Lemma 3.2** *If the receiver's privacy is not preserved in protocol 3.1, then it is also not preserved in the $OT_{k \times 1}^{\sqrt{N}}$ protocol.*

**Proof of lemma:** Assume that the receiver's privacy is not preserved in protocol 3.1. Namely, there are two receiver's inputs, $X_{i_1,j_1}, X_{i_2,j_2}$, for which the sender $B$ can distinguish the distributions that he sees when the receiver tries to retrieve $X_{i_1,j_1}$ or $X_{i_2,j_2}$. In this case, $B$ can be used to compromise the receiver's privacy in the 1-out-of-$\sqrt{N}$ OT protocol.

Assume, wlog, that the indexes of the two inputs are equal in one coordinate, say $i_1 = i_2$. (Otherwise, it is possible to use a hybrid argument to show that the sender can distinguish between two inputs which are equal in one of their coordinates. Namely, that the sender can distinguish the case that the receiver's $A$ input is $X_{i_1,j_2}$, either from the case that $A$'s input is $X_{i_1,j_1}$, or from the case that her input is $X_{i_2,j_2}$.)

Now, in order to break the privacy of a receiver $A'$ in a given 1-out-of-$\sqrt{N}$ OT protocol, run the part of the receiver in protocol 3.1 with $B$. In the oblivious transfer protocol that corresponds to coordinate $i$ ask to learn the input that corresponds to $i_1 = i_2$. Then run the receiver $A'$ in the 1-out-of-$\sqrt{N}$ OT protocol that corresponds to the $j$ coordinate. The output of the sender $B$ distinguishes with non negligible probability between the two possible inputs of $A'$. This concludes the proof of the lemma.

The security of the sender $B$ 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, and 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 given the $k$ keys she obtained she does not learn more than $k$ committed values. This statement seems trivial, but the formal proof of this property turns out to be rather subtle (since it involves the the problem of *selective decommitment*) and is discussed in Section 3.3.3. We provide here a proof that the receiver does not learn more than $k$ keys, and defer to Section 3.3.3 the discussion on learning more than $k$ input items.

**Lemma 3.3** *In $k \leq N$ invocations of the above protocol, the receiver $A$ does not learn more information than a party $A'$ which can adaptively ask and obtain the commitment keys of $k$ elements.*

**Proof :** The security of the $OT_1^{\sqrt{N}}$ 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 $r_1, r_2$ were chosen at random by $B$. This is equivalent to $A$ learning a triplet $V_2 = (g^{R_i C_j / r_1 r_2}, r_1, r_2)$, which has the same distribution. $A$ can easily compute from this information the 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 (which enables $A$ to generate tuples in the same distribution as that of $V_4$). The pseudo-randomness of the output of the DDH based synthesizer ensures that $A$ does not gain from $k$ such keys more information than is available to $A'$ which simply asks and receives $k$ keys. This concludes the proof of the lemma and of the theorem. $\square$

**Complexity:** The initialization phase requires $B$ to compute all $N$ commitment keys, i.e. to compute $N$ exponentiations (see in Protocol 3.2 a discussion on how to efficiently implement these exponentiations by utilizing the structure of the exponents). Each transfer phase requires two invocations of an $OT_1^{\sqrt{N}}$ protocol. These can be realized by independent $OT_1^{\sqrt{N}}$ protocols (which each require $O(\sqrt{N})$ initialization work by $B$). The $k$ calls to $OT_1^{\sqrt{N}}$ can also be realized by $k$ calls to a $OT_{k\times 1}^{\sqrt{N}}$ protocol. A slight complication is the fact that in each transfer round the sender uses different keys $r_R, r_C$, and the transfer should be for values of the form $R_i \cdot r_R$ or $C_i \cdot r_C$, which are different in each round. Therefore, the $OT^{\sqrt{N}}_{k\times 1}$ protocol should not use commitments, but rather the sender should send, in each round, encrypted values of $\{R_i \cdot r_R, C_i \cdot r_C\}_{i=1}^{\sqrt{N}}$. This does not enable the sender to change from round to round the values that are transferred in the $OT^N_{k\times 1}$ protocol, since the $OT^N_{k\times 1}$ protocol employs commitments.

### 3.3.3 Solving the selective decommitment problem

To show that the receiver in protocol 3.1 does not learn more than $k$ input items, it should be proved that party $A'$ which sees $N$ commitments and then asks for the keys of $k$ of them is not able to get information about more than $k$ committed values. $A'$ should, therefore, be simulated by a party who can adaptively ask and get $k$ of the committed values and sees nothing else (as in the ideal model). 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 trivial to prove that this is indeed the case. The problem is that it is hard to simulate the operation of $A'$ because it is unknown at the time of generating the commitments which of them she would like to open. See [21] for a discussion of this issue.

To enable the simulation it should be possible to open in the simulation *any commitment* to *any value*. Luckily, in the scenario of $OT^N_{k\times 1}$ there is a way to enable this property by amending the protocol in the following way. (For the sake of clarity we do not describe these modifications in the body of the protocols in the paper.)

- In the beginning of the protocol the receiver $A$ 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 $\log_{g_1} g_2$ for the values $g_1, g_2$ that would be used there).

- $B$ would use commitments of the form $g_1^a g_2^b$, which were suggested in [14]. (These commitments 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: (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. (3) sends the two results ($C_I$ and the encrypted $(X_I, Y_I)$) to $A$.

In the protocol, when the receiver computes the output of the synthesizer she can use it to compute $X_I$ and use the commitment 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.

More formally, suppose that there is a receiver $A'$ which receives all the commitments, asks for the keys of $k$ of the commitments, and is able to distinguish from random more than $k$ of the inputs. This $A'$ can be used to break the security of the commitment scheme in the following manner. The reduction first extracts from $A'$ the discrete log of $g_2$ to the base $g_1$. This information enables to open any commitment to any value. Then, the $N$ commitments, of the form $\langle (X_I, Y_I) \oplus K_I, C_I \rangle$ (where the $K_I$'s are random keys which are unknown to us), are sent to $A'$. When $A'$ asks to open commitment $I$, we can choose what value $X_I$ to reveal for this commitment, compute the appropriate values of $Y_I$ and $K_I$, and send $K_I$ to $A'$. If, after receiving $k$ such values, $A'$ is able to distinguish from random more than $k$ committed values, we can use her output to distniguish from random one of the committed values.

#### 3.3.4 A protocol using $\mathrm{OT}_1^2$

The following protocol constructs $OT_{k\times 1}^N$ using direct invocations of a simple $\mathrm{OT}_1^2$ protocol.

**Protocol 3.2** *$B$'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 $\ell$ 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 $\mathrm{OT}_1^2$ protocol on the strings $\langle a_j^0 r_j, a_j^1 r_j \rangle$. 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 $X_I$.*

**Theorem 3.4** *Protocol 3.2 is an adaptive oblivious transfer protocol, which can be used for $N$ adaptive transfers.*

**Proof:** It is clear that the receiver can obtain any value it wishes to receive in the Protocol. The privacy of $A$ is guaranteed by the privacy of the $\mathrm{OT}_1^2$ protocols, and the proof is similar to that of lemma 3.2 . The receiver $A$ is also ensured by the commitments that the sender $B$ 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 identically to the security of the sender in protocol 3.1.

[5]Note also that $B$ can set every $a_j^0$ to be equal to 1 without affecting the security of the system. The gain from this is a reduction in the size of the keys that $B$ needs to keep.

**Complexity:** 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 commitment 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 elements to the same exponent (see [38] for a survey of such techniques). It is an interesting problem to find a way to utilize the special structure of the exponents (being all the multiplications of all the subsets of $\ell$ elements) to compute the $N = 2^\ell$ commitment keys more efficiently.

The transfer part of the protocol requires $\ell = \log N$ invocations of an $\mathrm{OT}_1^2$ protocol. In addition $A$ and $B$ should each compute a single exponentiation.

## 3.4 Protocols Based on Any Sum Consistent Synthesizer

This section presents a $OT^N_{k\times 1}$ protocol which can be based on any sum consistent synthesizer. We first describe a protocol which is *insecure*, examine it, and construct a secure protocol.

### An insecure protocol

The following protocol is **insecure**. The protocol is based on organizing the input elements in a matrix, encrypting each element with the corresponding row and column keys, and letting the receiver learn the keys of $k$ locations.

**Protocol 3.3** *$B$'s input is $\{x_{i,j} | 1 \leq i, j \leq \sqrt{N}\}$, where $N = 2^\ell$. 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 $1 \leq i, j \leq \sqrt{N}$, $B$ prepares a commitment key $K_{i,j} = S(R_i, C_j)$, 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:** *for each $X_{i,j}$ that $A$ wants to learn the parties invoke the following protocol:*
   - *(a) $B$ chooses random elements $r_R, r_C$, 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 $OT_1^{\sqrt{N}}$ protocol on 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 $OT_1^{\sqrt{N}}$ protocol on 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 seems to be correct and secure. It 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 invocations of the protocols, and use linear relations between the keys to learn more keys than she is entitled to. In particular, 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 of $K_{i,j}, K_{i',j}, K_{i,j'}$ and use them to compute the key $K_{i',j'}$.

**The secure protocol**

In order to transform the above protocol to be secure, we use a mapping which ensures that no linear relations exist between the keys of different entries. In particular, we use the following construction of a set of matrices:

**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 the elements $1, \ldots, N$.*
- *Define a $2t\sqrt{N}$ dimensional vector space $V$, whose coordinates correspond 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, or column, index).*
- *For each element $x$ denote its row and column in matrix $i$ as $R_i(x), C_i(x)$. Construct a vector $v_x \in V$ in which the coordinates that correspond to the locations of $x$ in the matrices are set to 1. I.e., coordinates $(i, 1, R_i(x))$ and $(i, 2, C_i(x))$ are 1 for $1 \le i \le t$, 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.*

The motivation for this construction is to allocate keys to inputs according to a mapping defined by a relation free set of matrices. In this mapping there are no linear relation between keys and the resulting protocol is good for oblivious transfer with adaptive queries. The security of the protocol is based the following lemma.

**Lemma 3.5** *A randomized mapping of $N$ elements to $t$ matrices, where $t = \frac{\log(N/k)}{\log(\sqrt{N}/k)} + 1 = \frac{3\log N - 4\log k}{\log N - 2\log k}$, is with high probability $k$-out-$n$ 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 non-zero values of all $k+1$ vectors are concentrated in at most $(k+1)/2$ coordinates.

Since the mapping to locations in matrices 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 $(\frac{k+1}{2\sqrt{N}})^{(k+1)t}$. 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 $t \geq (\log(eN/(k+1))/\log(2\sqrt{N}/(k+1)) \approx \frac{\log(N/k)}{\log(\sqrt{N}/k)}$. Therefore, setting $t = \frac{\log(N/k)}{\log(\sqrt{N}/k)} + 1$ satisfies the requirement of the theorem. □

It should be interesting to come up with an explicit construction of $k$-out-$n$ relation free matrices.

### An overview of the protocol

On a high level, the transformation of protocol 3.3 is as follows. In the *initialization* phase $B$ takes a $k$-out-of-$N$ relation free construction of $t$ matrices and maps the $N$ elements to the $t$ matrices according to the construction (we use the random construction of lemma 3.5). He 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 $2t$ $OT_1^{\sqrt{N}}$ protocols, which let $A$ learn each of 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 inputs that generated the key to the commitment that hides $X_I$, and so $A$ is able to open this commitment,

### The protocol

**Protocol 3.4 (Adaptive oblivious transfer based on any sum-consistent synthesizer)** *$B$'s input is $\{x_{i,j} | 1 \leq i,j \leq \sqrt{N}\}$, where $N = 2^\ell$. $B$ maps the inputs into $t$ square matrices independently at random. Let $x_R^m$ denote the row into $x$ is mapped in matrix $m$, and let $x_C^m$ denote the column into $x$ is mapped in matrix $m$.*

*Let $S(x,y)$ be a sum consistent synthesizer with two inputs.*

1. **Initialization:**

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

   $$(R_1^1, R_2^1, \ldots, R_{\sqrt{N}}^1) \;\; (C_1^1, C_2^1, \ldots, C_{\sqrt{N}}^1), \ldots, (R_1^t, R_2^t, \ldots, R_{\sqrt{N}}^t) \;\; (C_1^t, C_2^t, \ldots, C_{\sqrt{N}}^t)$$

   *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 on 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* $r_R^1, r_C^1, \ldots, r_R^t, r_C^t$, *such that their sum is* $0$. *(*$r_R^m$ *is used to randomize the row keys of matrix* $m$, *and* $r_C^m$ *is used to randomize the column keys of matrix* $m$*).*

(b) *For every matrix* $1 \le m \le t$ *A and B engage in the following protocols:*
   - *A* $OT_1^{\sqrt{N}}$ *protocol on the values* $\langle R_1^m + r_R^m, R_2^m + r_R^m, \ldots, R_{\sqrt{N}}^m + r_R^m \rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $R_{(x_{i,j})_r^m}^m + r_R^m$.
   - *A* $OT_1^{\sqrt{N}}$ *protocol on the values* $\langle C_1^m + r_C^m, C_2^m + r_C^m, \ldots, C_{\sqrt{N}}^m + r_C^m \rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $C^m(x_{i,j})_c^m + r_C^m$.

(c) *A reconstructs* $K_{i,j}$ *as*

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

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

The following theorem states that protocol 3.4 is secure if enough matrices are used (fortunately, if $k$ is not too close to $\sqrt{N}$ very few matrices are needed).

**Theorem 3.6** *The above* $OT_{k\times 1}^N$ *protocol with* $t = \frac{\log(N/k)}{\log(\sqrt{N}/k)} + 1 = \frac{3\log N - 4\log k}{\log N - 2\log k}$ *matrices is secure.*

Note that this yields a construction for any $k = o(\sqrt{N})$. In particular, if $k < N^{1/4}$ it is enough to use only 4 matrices, and $k < N^{1/3}$ requires only 5 matrices.

**Proof:** The privacy of the receiver is preserved since each $OT_1^{\sqrt{N}}$ protocol preserves her privacy. The properties of these protocols ensure also that in each invocation of a $OT_1^{\sqrt{N}}$ protocol the receiver learns only a single item of the sender's input. Consequently, in every transfer stage the receiver learns one row key and one column key of each matrix, where each of these keys is added to a random value and the sum of all the random values is zero. The receiver learns, therefore, a single linear combination of all the row and column keys. Note that we do not assume that in each run of the transfer protocol $A$ has learnt the sum of keys which correspond to one of the elements. In each transfer protocol she could have obtained the sum of keys of her choice, not necessarily corresponding to an element $X_I$, but the equations she learnt span $k$ equations which correspond to elements $X_I$.

Assume that the receiver has run the transfer protocol $k$ times, learned $k$ linear relations of the keys, and used them to learn the commitment keys of $k+1$ elements. This contradicts the $k$-out-of-$N$ relation freeness property of the matrices. $\square$

# 4 Applications

This section describes the basic details of several applications of oblivious transfer and of oblivious transfer with adaptive queries.

## 4.1 Applications of 1-out-of-$N$ OT

Two applications of 1-out-of-$N$ oblivious transfer are described here. Another application is oblivious polynomial evaluation [41]

### 4.1.1 PIR to SPIR transformation

The problem of allowing search in databases so that the database owner does not learn what is being searched has received a great deal of attention. A system that allows a user to access a database consisting of $N$ words $\langle W_1, W_2, \ldots W_N \rangle$, read any word it wishes without the owner learning which word was accessed, and use $o(N)$ communication, is called PIR (for Private Information Retrieval) [15]. There are various proposals for implementing such schemes, where the emphasis is on the communication complexity. Some of the proposals require that the user communicates with several servers maintained by the database owner where these servers are certified somehow not to communicate with each other. This assumption (of non-communication replicated data-bases) was shown to be unnecessary by Kushilevitz and Ostrovsky [35] where a PIR scheme with a single server was proposed – where the user's security depends on the Quadratic Residuocity assumption modulo a Blum integer and the communication complexity is $O(N^{\epsilon}m)$ (where $m$ is the security parameter, i.e., the length of a number that is hard to factor). Another proposal of such a scheme is by Cachin et al [12] where the communication complexity is just $m$.

More recently attention was given to the question of protecting the database as well, i.e., that the user will not learn more than one word of data (or as many words as he or she paid for). A PIR scheme that enjoys this property is called SPIR. In [25] a transformation of any PIR scheme into a SPIR scheme was proposed at the cost of increasing the number of servers (and introducing the separation of servers assumption). Kushilevitz and Ostrovsky [35] too, suggest an adaptation of their single server PIR protocol to enable the receiver to learn just a single element of the database, and transform it to a SPIR protocol by combining a zero-knowledge proof in which the receiver proves that it followed the protocol. We show here that the combination of 1-out-of-N oblivious transfer with any PIR protocol provides a SPIR protocol which does not require to add any new servers and and requires relatively little work on behalf of the parties involved. In particular, the overhead is just the sum of the overheads of the PIR and oblivious transfer protocols, and is reasonable for any application for which the overhead of PIR is reasonable.

It is not hard to see the connection between the PIR/SPIR setting and the 1-out-of-$N$ OT setting. As described in table 1, both PIR and OT protocols, provide privacy for the receiver, but PIR protocols emphasize communication complexity, whereas oblivious transfer protocols emphasize the server's privacy. One can regard a SPIR construction as a combination of 1-out-of-$N$ OT and PIR protocols, which provides low communication complexity, and privacy for both parties. SPIR is essentially oblivious transfer with $o(N)$ communication. The important feature of Protocol 2.1 for 1-out-of-$N$ OT is that for the receiver to obtain the value of the desired $X_I$ she does not need all of $Y_1, Y_2, \ldots Y_N$ but only $Y_I$. Therefore if instead of step 3 in Protocol 2.1 the sender and the receiver perform a PIR reading of $Y_1, Y_2, \ldots Y_N$, then the receiver can get sufficient information without giving the sender any information about the value she is interested in. The added communication complexity to the PIR protocol is the $\log N$ invocations of the 1-out-of-2 OT protocol. The evaluations of $F_K$ do not add to the communication complexity, but add to the work done by the database. Therefore one can use this protocol to transform any PIR protocol to a SPIR protocol without increasing the number of databases.

| | PIR | OT | | SPIR |
|---|---|---|---|---|
| Receiver Privacy | + | + | | + |
| Sender Privacy | | + | $\Rightarrow$ | + |
| Communication | $o(N)$ | $O(N)$ | | $o(N)$ |

Table 1: A comparison of PIR, oblivious transfer, and SPIR protocols.

#### PIR and oblivious transfer with adaptive queries

$\mathrm{OT}_1^N$ protocols enable an efficient transformation of any PIR protocol to a SPIR protocol. The adaptive oblivious transfer protocols we introduced would enable even more efficient future transformations from PIR to SPIR, by transforming a protocol for $k$ adaptive invocations of PIR to a protocol for $k$ adaptive invocations of SPIR (the problem is that currently there are no adaptive PIR protocols, but when such a protocol is introduced, $OT_{k\times 1}^N$ would enable to immediately transform it to a SPIR protocol).

#### PIR vs. oblivious transfer

On a more practical level, we believe that it is preferable to use the computation efficient $\mathrm{OT}_1^N$ and $OT_{k\times 1}^N$ protocols rather than the communication efficient PIR protocols. Oblivious transfer protocols, including the protocol that can handle adaptive queries, require $O(N)$ communication only at the end of the initialization phase and before the transfer phases begin. For many applications this communication overhead is not an issue. Communication of Gigabytes of data is cheap and simple, using detachable storage devices (such as DATs or DVDs) or fast communication networks. In contrast, single server PIR protocols [35, 12] are very costly to implement since they require $O(N)$ exponentiations per transfer.

### 4.1.2 Oblivious sampling

In this section we briefly describe an application of 1-out-of-$N$ OT protocols to a problem suggested by Andrei Broder.

Consider the following scenario: Cold Fusion (Trademark) is a search engine claiming to be the largest on the west coast. Alice would like to check this claim and measure the number of URLs *indexed* by Cold Fusion, i.e., web pages which can be searched for using Cold Fusion's search interface. She might also like to check the overlap between the pages indexed by Cold Fusion and by other search engines, and this task also requires a random sample of the pages indexed by Cold Fusion (see [9]). One possibility is for Cold Fusion to give Alice the list of the URLs it has indexed; Alice then will make sure that all (or most of) these URLs are indeed indexed actively, i.e. that the corresponding page can be retrieved in a search (it is much easier to gather many URLs without indexing the content). This can be done by sampling a few of them, retrieving the corresponding page and searching for the page via the public interface of Cold Fusion. The problem is of course that the list of URLs is a trade secret and Cold Fusion will not reveal it even to a study that will declare it to be the largest search engine. Therefore we are looking for a sampling procedure which will allow Alice to select a few URLs from the list, and then search for them in Cold Fusion's web interface. The selection procedure must

- Keep most of the list (the part not sampled) secret from Alice.
- Prevent Cold Fusion from learning which URLs were selected - otherwise it can quickly add them to the active index. (There is a great difference between finding and storing web pages on one hand, and indexing them so that they can be searched for through a search engine's interface, on the other hand. Alice wants to check how many web pages are searchable, and therefore should keep the URLs she selected secret from Cold Fusion, and then quickly search for them through its web interface).

1-out-of-$N$ OT (and in particular $k$-out-of-$N$ OT protocols) can be used for *oblivious sampling*, i.e. to let the receiver sample a random element from a large set of elements known to the sender, without giving the sender any information about the item that the receiver chooses. In the case of the search engines application this enables Alice and Cold Fusion to solve their problem at a small cost, even if Cold Fusion's databases consists of hundreds of millions of pages. If Cold Fusion claims to have indexed $N$ URLs it should feed them as the input $X_1, \dots X_N$ to the 1-out-of-$N$ OT protocol, whose main computational overhead is logarithmic in $N$.

The only problem remaining is the duplication problem – what if Cold Fusion duplicates some of the URLs in order to claim a larger set (namely, taking a database of $N'$ URLs, and creating a database of $N = cN'$ URLs by duplicating each URL $c$ times). This can be solved using a hash tree structure, but there is also a very simple solution using the following procedure – for some of the sampled URLs Alice does the following: she sends the URL to Cold Fusion and asks it what was the index $I$ that Alice chose when she sampled the URL. If Cold Fusion does not answer correctly – then Alice can conclude that Cold Fusion was cheating by duplicating URLs.

## 4.2 Applications of Oblivious Transfer with Adaptive Queries

Protocols for oblivious transfer with adaptive queries are useful when a large database should be queried adaptively while hiding the queries from the database owner, and hiding the contents of the database (exept for the answer for the queries) from the party making the queries. Following are some examples for such applications.

**Oblivious search:** Bob owns a database which Alice wants to search. Suppose first that the database is sorted and Alice would be using *binary search*. The two parties can invoke a $OT^N_{\log N \times 1}$ 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 [24] are constructed).

Note that the hash function that Bob uses in the first stage can be made public, but the hash functions of the second stage should be kept secret from Alice since they disclose information about Bob's inputs. Protocols for oblivious transfer with adaptive queries can be used to let Alice obliviously determine whether an element is in the table. She first uses the hash function of the first stage to compute by herself the bin to which the element should have been mapped. She 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.

**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. Furthermore, whenever she finds certain information on one of her patients, she wants to examine records of related patients (e.g., family members or patients which were treated similarly). Alice and Bob can use $OT^N_{k\times 1}$ to enable Alice to gather the relevant information from Bob's database.

**Patent database:** 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. This search is adaptive: after she retreives a certain patent Alice might ask to read patents which are referenced by this patent. She is afraid that if she conducts the search on Bob's database he might learn what she is interested in and maybe reveal her idea. Alice and Bob can use $OT^N_{k\times 1}$ to enable Alice to search Bob's database without revealing her queries to him.

## Acknowledgments

We would like to thank Sanjeev Arora, Andrei Broder, Oded Goldreich, and Yuval Ishai.

## References

[1] N. Alon and J. Spencer, **The Probabilistic Method**, Wiley, 1992.

[2] R. Anderson and M. Kuhn, *Tamper Resistance – A Cautionary Note*, Usenix Electronic Commerce Workshop, Oakland (1996), 1–11.

[3] D. Beaver, *Foundation of Secure Interactive Computation*, Advances in Cryptology - Crypto '91, pp. 377–391, 1991.

[4] M. Bellare and S. Micali, *Non-interactive oblivious transfer and applications*, Advances in Cryptology - Crypto '89, pp. 547-557, 1990.

[5] G. Brassard, **Modern cryptology**, LNCS, vol. 325, Springer, 1988.

[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 Verlag, 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] K. Bharat and A. Broder. *A technique for measuring the relative size and overlap of public web search engines,* In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pp. 379–388. Elsevier Science, April 1998.

[10] D. Boneh, *The Decision Diffie-Hellman Problem*, Proc. of the Third Algorithmic Number Theory Symposium, Springer-Verlag LNCS 1423 (1998) 48–63.

[11] C. Cachin, *On the foundations of oblivious transfer*, Advances in Cryptology - Eurocrypt '98, LNCS 1403, pp. 361-374. Springer-Verlag, 1998.

[12] C. Cachin, S. Micali and M. Stadler, *Computationally Private Information Retrieval With Polylogarithmic Communication*, Advances in Cryptology – Eurocrypt '99, LNCS, Springer-Verlag, 1999.

[13] R. Canetti, *Security and Composition of Multiparty Cryptographic Protocols*, manuscript, 1998.

[14] D. Chaum, E. van Heijst, and B. Pfitzmann, *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, Proc. Advances in Cryptology – Crypto '91.

[15] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan, *Private Information Retrieval,* 36th FOCS, pp. 41–50, 1995.

[16] 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.

[17] C. Crépeau, *Equivalence between two flavors of oblivious transfers*, Advances in Cryptology – Crypto '87 , LNCS 293, pp. 350–354, 1988.

[18] C. Crépeau and J. Kilian, *Achieving oblivious transfer using weakened security assumptions*, FOCS '88, pp. 42–52, 1988.

[19] A. De Santis, Y. Desmedt, Y. Frankel and M. Yung, *How to share a function securely*, Proc. 26th STOC, pp. 522-533, 1994.

[20] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, vol. 22(6), 1976, pp. 644-654.

[21] C. Dwork, M. Naor, O. Reingold and L. Stockmeyer, *Magic Functions*, 40th FOCS, pp. 523-534, 1999.

[22] 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.

[23] S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM **28**, pp. 637–647, 1985.

[24] M. L. Fredman, J. Komlos and R. Szemeredi, *Storing a sparse table with $O(1)$ worst case access time*, JACM 31 (1984), 538–544.

[25] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, *Protecting Data Privacy in Private Information Retrieval Schemes*, Proc. 30th STOC, 1998.

[26] Y. Gertner, S. Kannan, T. Malkin, O. Reingold and M. Viswanathan *The relationship between public key encryption and oblivious transfer*, manuscript, 2000.

[27] O. Goldreich, **Foundations of Cryptography (Fragments of a Book)**, 1995. Electronic publication: `http://www.eccc.uni-trier.de/eccc/info/ECCC-Books` (Electronic Colloquium on Computational Complexity).

[28] O. Goldreich, *Secure Multi-Party Computation* (working draft) Version 1.1, 1998.

[29] O. Goldreich, S. Goldwasser and S. Micali, *How to construct random functions*, J. of the ACM., vol. 33, pp. 792-807, 1986.

[30] O. Goldreich, M. Sudan and R. Rubinfeld, *Learning Polynomials with Queries: The Highly Noisy Case*, Proc. 36th FOCS, pp. 294-303, 1995.

[31] O. Goldreich and R. Vainish, *How to Solve any Protocol Problem - An Efficiency Improvement*, Advances in Cryptology - Crypto '87, LNCS 293, 73–86. Springer-Verlag, 1988.

[32] R. Impagliazzo and M. Naor, *Efficient Cryptographic schemes provably secure as subset sum*, Journal of Cryptology, vol 9, pp. 199-216, 1996.

[33] R. Impagliazzo and S. Rudich, *Limits on the Provable Consequences of One-Way Permutations,* STOC '89, pp. 44–61, 1989.

[34] J. Kilian, **Use of Randomness in Algorithms and Protocols**, MIT Press, Cambridge, Massachusetts, 1990.

[35] E. Kushilevitz and R. Ostrovsky, *Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval*, 38th FOCS, pp. 364-373, 1997.

[36] M. Luby, **Pseudo-randomness and applications**, Princeton University Press, 1996.

[37] S. Lucks, *Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys*, Proc. of Security Protocol Workshop '97, `http://www.dmi.ens.fr/~vaudenay/spw97/spw97_Luc3.ps.gz`.

[38] A. J. Menezes, P. C. van Oorschot and Scott A. Vanstone, **Handbook of Applied Cryptography**, CRC Press, 1996.

[39] S. Micali and P. Rogaway, *Secure Computation*, Advances in Cryptology – Crypto '91, pp. 392–404. LNCS 576, Springer-Verlag, 1992.

[40] M. Naor, *Bit Commitment Using Pseudo-Randomness*, Journal of Cryptology, vol 4, 1991, pp. 151-158.

[41] M. Naor and B. Pinkas, *Oblivious Polynomial Evaluation*, manuscript, 2000.

[42] M. Naor and O. Reingold, *Synthesizers and their application to the parallel construction of pseudo-random functions*, 36th FOCS, pp. 170-181, 1995.

[43] M. Naor and O. Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited*, J. of Cryptology, vol. 12, pp. 29–66, 1999. Preliminary version appeared in 29th STOC, pp. 189-199, 1997.

[44] M. Naor and O. Reingold, *Number-Theoretic constructions of efficient pseudo-random functions*, 38th FOCS, pp. 458-467, 1997.

[45] M. O. Rabin, *How to exchange secrets by oblivious transfer*, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.

[46] M. Sudan, *Decoding of Reed Solomon codes beyond the error-correction diameter*, Journal of Complexity 13(1), pp. 180-193, 1997.

[47] S. Wiesner, *Conjugate coding*, SIGACT News **15**, pp. 78–88, 1983.

[48] A.C. Yao, *How to Generate and Exchange Secrets*, 27th FOCS, pp. 162–167, 1986.