J. Cryptology (2005) 18: 1–35
DOI: 10.1007/s00145-004-0102-6

Journal of
CRYPTOLOGY

Computationally Secure Oblivious Transfer*

Moni Naor
Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science,
Rehovot 76100, Israel
naor@wisdom.weizmann.ac.il

Benny Pinkas
HP Labs, 5 Vaughn Drive,
Princeton, NJ 08540, U.S.A.
benny.pinkas@hp.com, benny@pinkas.net

Communicated by Joan Feigenbaum

Received 16 January 2001 and revised 15 April 2004
Online publication 21 October 2004

Abstract. We describe new computationally secure protocols of 1-out-of-N oblivious transfer, k-out-of-N oblivious transfer, and oblivious transfer with adaptive queries. The protocols are very efficient compared with solutions based on generic two-party computation or on information-theoretic security. The 1-out-of-N oblivious transfer protocol requires only $\log N$ executions of a 1-out-of-2 oblivious transfer protocol. The k-out-of-N protocol is considerably more efficient than k repetitions of 1-out-of-N oblivious transfer, as is the construction for oblivious transfer with adaptive queries. The efficiency of the new oblivious transfer protocols makes them useful for many applications. A direct corollary of the 1-out-of-N oblivious transfer protocol is an efficient transformation of any Private Information Retrieval protocol to a Symmetric PIR protocol.

Key words. Cryptography, Privacy preserving computation, Secure function evaluation, Oblivious transfer.

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 compute jointly the value of $f(x, y)$ in a way that

* This paper is the full version of the sections discussing oblivious transfer in "Oblivious Transfer and Polynomial Evaluation" (31st STOC, 1999) and of the extended abstract "Adaptive Oblivious Transfer" (Crypto '99). The research of the first author was supported by Grant No. 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences. Most of the work by the second author 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.

does not reveal to each side more information than can be deduced from $f(x, y)$. (This definition can be generalized in a natural way to capture the case where each party learns a different output, or where only one party learns an output.) 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 efficiency of the resulting protocols is often not satisfactory since the number of cryptographic operations performed is proportional to the *size of the circuit computing* $f(x, y)$ [51], [31]. Even for relatively simple functions this may be prohibitively expensive. Therefore it is interesting to look for functions for which it is possible to design 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 (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 value, and he remains oblivious to the value she chooses. This is a well-known problem. We discuss 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 (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 oblivious transfers. Section 3 presents protocols for oblivious transfer with adaptive queries. Section 4 describes various applications for the protocols we present.

1.1. *Oblivious Transfer*

Oblivious Transfer (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 $1 \leq I \leq N$ 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 et al. [23], as a generalization of Rabin's "oblivious transfer" [49] (the notion was also developed independently by Wiesner in the 1970s, but was only published in [50]). 1-out-of-N OT was introduced by Brassard et al. [7] under the name ANDOS (all or nothing disclosure of secrets). For an up-to-date definition of OT and oblivious function evaluation see [29].

Since its proposal OT has enjoyed a large number of applications and in particular Kilian [35] and Goldreich and Vainish [32] 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 applications for 1-out-of-N OT in case N is relatively large, and some of them are described in Section 4. Another application is oblivious polynomial evaluation, which is described in [45].

Reductions between various types of OT protocols have been investigated extensively and it was concluded that the various types of OT are information-theoretically equivalent (See [6], [8], [18], [17], and [10]). 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. Furthermore, it was shown that an information-theoretic reduction from 1-out-of-N OT to 1-out-of-w OT (where both OTs operate on strings of the same length) must use at least $(N-1)/(w-1)$ invocations of 1-out-of-w OT in order to preserve the information-theoretic security [20]. 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], and [4]). Essentially every known suggestion of public-key cryptography allows one also to implement OT (although in general public-key cryptography and OT are incomparable under black-box reductions [27]), and the complexity of 1-out-of-2 OT is typical of public-key operations [6], [4]. Oblivious transfer based on the paradigm of [23] can be easily constructed using public-key systems if it is possible to generate two computationally indistinguishable strings, one of them being a public key and the other being random. Consequently, OT can be based on the existence of trapdoor permutations, the hardness of factoring, the Diffie–Hellman assumption, and the hardness of finding short vectors in a lattice (the Ajtai–Dwork cryptosystem). On the other hand, given an OT protocol it is a simple matter to implement secret-key exchange using it. Therefore from the work of Impagliazzo and Rudich [33] it follows that there is no black-box reduction to OT from one-way functions.

Complexity. Our working assumption is that 1-out-of-2 OT is an expensive operation compared with 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 achieve efficient constructions of 1-out-of-N OT protocols from 1-out-of-2 OT protocols, where the number of invocations of the 1-out-of-2 OT protocol is small. For instance, the 1-out-of-N OT constructions of [6] and [8] need N calls to the 1-out-of-2 OT protocol, and almost match the lower bound of [20]. In contrast our protocols need only $\log N$ calls to the 1-out-of-2 OT protocol plus $O(N)$ evaluations of a pseudo-random function (note also that there is an efficient construction in [8] of 1-out-of-2 OT of string inputs from 1-out-of-2 OT of bit inputs). We note that in a subsequent work [44] we described how to implement 1-out-of-N OT with an *amortized* overhead of a single 1-out-of-2 OT.

Another measure of complexity is communication complexity. While OT protocols might have communication complexity which is linear in N, Private Information Retrieval (PIR) protocols have sublinear communication complexity while guaranteeing privacy for the client but not necessarily for the server. Symmetric PIR (SPIR) protocols have sublinear communication complexity while providing privacy for both parties (therefore they are equivalent to OT with sublinear communication complexity). Section 4.1.1 presents a generic reduction from SPIR to PIR using 1-out-of-N OT. In conjunction with PIR constructions we can obtain a protocol for 1-out-of-N OT with $O(N^{\varepsilon}m)$ communication overhead under the Quadratic Residuosity Assumption, based on the PIR construction of [36] (where m is the security parameter, i.e., the length of the modulus), or a protocol with a communication overhead that is poly-logarithmic in N, based on the PIR protocol of [11].

1.2. *Correctness and Security Definitions*

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

We first define the input and output for 1-out-of-N OT. 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) Achieving the precise definition of general two-party or multi-party protocols is by no means simple and there is no consensus yet on the definition (see [3], [12], [29], and [40], 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 those due to Goldreich [29]. As the main purpose of our paper is to provide efficient constructions of OT rather than to concentrate on elaborate definitions, 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 issue of protecting the sender. Since our constructions offer computational security, they define privacy in the sense of computational indistinguishability (see, e.g., [28]).

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-of-N OT protocol is rather simple: it is required that *for any* $X_1, X_2, \ldots, X_N$, *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.*

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 with 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. This is the minimal information that the receiver learns in the any implementation of the protocol. We require that in the real implementation of the protocol, without a trusted party, the receiver does not learn more than in the ideal implementation.

More formally, the requirement is that *for every probabilistic polynomial-time machine* $\mathcal{A}'$ *substituting the receiver in the real implementation of the protocol, there exists a probabilistic polynomial-time machine* $\mathcal{A}''$ *that plays the receiver's role in the ideal implementation, such that for every input* $X_1, \ldots, X_N$ *of the sender, the outputs of* $\mathcal{A}'$ *and* $\mathcal{A}''$ *are computationally indistinguishable.* This requirement 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 $\mathcal{A}'$, which might behave arbitrarily, "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* (see, e.g., the proof of Lemma 2.3).

2. Protocols for 1-out-of-N Oblivious Transfer

In this section we describe efficient constructions of a 1-out-of-N OT 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 [45].

The 1-out-of-N OT protocol uses, in addition to 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. Consider, for example, a function F_K specified by a short key K, and assume that the function can only be accessed by the observer by adaptively specifying inputs and obtaining the value of the function on these inputs. (See [30], [28], [37], [46], and [47] for precise definitions and various constructions.) Our working assumption is that block ciphers (such as DES or AES) 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 more efficient than a typical public-key operation.

Let $\{F_K\colon \{0,1\}^m \mapsto \{0,1\}^m \mid K \in \{0,1\}^t\}$ 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 use a small set of $O(\log N)$ keys and mask each input with a combination 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): 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 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 OT. The first protocol solves the problem using $\log N$ applications of 1-out-of-2 OT. The second protocol is recursive and reduces 1-out-of-N OT to two invocations of 1-out-of-$\sqrt{N}$ OT. The 1-out-of-$\sqrt{N}$ OT protocols can be computed using the first protocol, or alternatively the recursion can be applied until a 1-out-of-2 OT protocol is needed (in any case, the total number of 1-out-of-2 OTs that should be executed is the same as in the first protocol). The latter protocol has a better initialization overhead, $O(N)$ invocations of a pseudo-random function, instead of $O(N \log N)$ invocations in the first protocol, but in both cases the main computational overhead is incurred during the transfer stage. The main reason for presenting both protocols is that later we present k-out-of-N OT protocols and adaptive OT protocols that are based on both types of 1-out-of-N OT protocols.

2.1. *A Protocol for* 1*-out-of-N Oblivious Transfer*

Protocol 2.1 (1-out-of-N OT). *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 ℓ 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* $K_j^{i_j}$.
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 OT protocol.*

Proof. It is straightforward to see that the protocol lets the receiver obtain any value she desires. As for the security analysis, it has to be argued that both the sender's security

[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 $I - 1$.

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* OT_1^2 *protocol.*

Proof. 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 $m + 1 \leq \ell$ indices 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, to show that the receiver's privacy is not preserved in the OT_1^2 protocol we assume that we are given a receiver A' in an OT_1^2 protocol, and our task is to compromise her privacy. We then simulate the part of the receiver in an OT_1^N protocol that is run with B. In the OT_1^2 protocols that correspond to the bits in which J_I agrees with J_{I+1} we ask to learn the input that corresponds to the appropriate bit of J_I (which is equal to the bit of J_{I+1}). We run the receiver A' in the 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* OT_1^2 *protocol does not provide the sender's privacy or the function* F *is not pseudo-random.*

Proof. The definition of the sender's privacy is based on comparison with the ideal model: we must show 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.

Assume that the OT_1^2 protocol preserves the sender's privacy, i.e., that the receiver learns only one of the two inputs of the sender (otherwise the lemma follows trivially). We next show how, given black-box access to the receiver $\mathcal{A}'$, it is possible to extract the indices of the keys that were learned by her in Step 2 of Protocol 2.1. Knowledge of these indices enable the identification of the item from the server's input that is learned by $\mathcal{A}'$ in the protocol, and construct an $\mathcal{A}''$ whose output is indistinguishable from that of $\mathcal{A}'$.

The extraction can be done by the following procedure: Run the protocol up to the end of Step 2, fix the state of the receiver $\mathcal{A}'$ 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 2ℓ experiments $\{C_{i,j} \mid 0 \leq i \leq 1,\ 1 \leq j \leq \ell\}$. 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 receiver learns K_j^i in Step 2 then her view in experiment $C_{i,j}$ (where r is used instead of K_j^i as an encryption key) is different by a non-negligible difference from her view in the runs of the protocol that use the original keys. Note that for every j this phenomenon occurs for at most one of the experiments $C_{0,j}$ and $C_{1,j}$ (otherwise it is easy to show that the OT_1^2 protocol does not preserve the sender's privacy). Therefore in at most one of the experiments $C_{0,j}$ and $C_{1,j}$ the distribution of the output of the receiver is different (by a non-negligible difference) than her output in the original protocol. Let $I_0^j \in \{0, 1\}$ be equal to the index $i \in \{0, 1\}$ for which this phenomenon occurs. (We can disregard the case where the receiver's output in both $C_{0,j}$ and $C_{1,j}$ is distributed as her output in the original protocol, since in that case we can deduce that the output of $\mathcal{A}'$ is independent of the server's inputs and therefore $\mathcal{A}''$ can be run in the ideal model. This is because in $C_{0,j}$ the receiver cannot learn the server's inputs for which the jth bit of the index is 0, and in $C_{1,j}$ she cannot learn the inputs in which this bit is 1, and therefore if none of these experiments changes her output, her output is not one of the server's inputs.) At the end of the set of experiments define $1 \leq I_0 \leq N$ as the concatenation of the bits I_0^j, for $1 \leq j \leq \ell$.

Now, given the procedure and a receiver algorithm $\mathcal{A}'$, we construct an algorithm $\mathcal{A}''$ that runs in the ideal model, and operates in the following way:

1. $\mathcal{A}''$ generates a set of keys in the same way that the sender does in Step 1 of Protocol 2.1, and engages in OT interactions with $\mathcal{A}'$ as in Step 2 of the protocol.
2. $\mathcal{A}''$ extracts, using the procedure outlined above, the index I_0 corresponding to the keys learned by $\mathcal{A}'$ in Step 2.
3. $\mathcal{A}''$ asks the third party Charlie for the value of X_{I_0}.
4. $\mathcal{A}''$ sends to $\mathcal{A}'$ a set of values $Y_1, \ldots, Y_N$, where Y_{I_0} is the encryption of X_{I_0} using the keys corresponding to the index I_0, and every other Y_J is an encryption of a random value with the keys corresponding to index J.
5. $\mathcal{A}''$ outputs the same output that $\mathcal{A}'$ outputs.

We claim that the outputs of A' and A'' are computationally indistinguishable: Assume that the output of $\mathcal{A}''$ is computationally *distinguishable* from the output of $\mathcal{A}'$, then we can construct a distinguisher between the output of the pseudo-random function F and random values.

Denote the ℓ keys whose values are *not* learned by $\mathcal{A}'$ in Step 2, as $K_1^{u_1}, \ldots, K_\ell^{u_\ell}$. We define $\ell + 1$ hybrids $H_1, \ldots, H_\ell + 1$. Hybrid H_i corresponds to running the receiver and sending it, in Step 3, a set of values $Y_1, \ldots, Y_N$ that is generated as in the protocol, except for the following difference: for every $J \neq I_0$, Y_J is generated using *random* values instead of the outputs of F that are keyed by $K_i^{u_i}, \ldots, K_\ell^{u_\ell}$.

Hybrid $H_{\ell+1}$ therefore corresponds to running the receiver in the original protocol. We claim that if F is pseudo-random then hybrid H_1 corresponds to running $\mathcal{A}'$ using the algorithm $\mathcal{A}''$ that runs in the ideal model, as we defined above. This follows from the fact that the difference between the two instances is that in H_1 each Y_J, for $J \neq I_0$, is generated by xoring the real value of X_J with a random value (instead of the output

of F), and in the invocation of $\mathcal{A}'$ these values are generated by xoring the real outputs of F with random values instead of the X_J's.

We thus get that the output of hybrids H_1 and $H_{\ell+1}$ are computationally distinguishable. Therefore there are two hybrids, H_i and H_{i+1}, $1 \leq i \leq \ell$, that are computationally distinguishable. This means that there is a distinguisher between random values and the output of F when it is keyed by $K_i^{u_i}$. □

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.

Improving the Preprocessing Overhead. Yuval Ishai [34] 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 (this construction is similar to the construction of pseudo-random functions in [30]). The advantage of this proposal is that the total number of evaluations of the pseudo-random function is linear in N. Protocol 2.2 in Section 2.2 introduces a different method for lowering the overhead of the preprocessing stage.

2.2. *A Recursive Protocol for* 1*-out-of-N Oblivious Transfer*

We present here a different protocol for 1-out-of-N OT, which 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 better than that of the previous protocol, and is $O(N)$ invocations of a pseudo-random function, instead of $O(N \log N)$ invocations. The protocol is used in the k-out-of-N protocol of Section 2.3.

Protocol 2.2 (1-out-of-N OT). *The sender's* (B) *input 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 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 she desires 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. Implementing 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 for Protocol 2.1.

2.3. *k-out-of-N Oblivious Transfer*

Some applications require a k-out-of-N OT protocol, i.e., a protocol which enables the receiver to choose *any* k out of N input strings. It is possible to implement this task by repeating Protocol 2.1 k times independently, but the overhead would be $kN\log N$ (or kN) applications of a pseudo-random function for preprocessing, and $k\log N$ OTs for the k transfers. Since N might be very large, the preprocessing overhead might be prohibitively expensive. An additional problem with repeating Protocol 2.1 k times is that the sender might not be *consistent* from round to round and he can thus induce a distribution on the receiver's output that is impossible in the ideal implementation. A simple solution for this issue is that the sender *commits* to the X_I's (once), and protects the keys that open the commitments using the masks $\bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$ (that are used in the 1-out-of-N protocol for simple encryption). The use of commitments introduces the selective decommitment problem, which is discussed in Section 3.3.3. Given the additional overhead and subtle issues of repeating Protocol 2.1, it is interesting to investigate whether it is possible 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. Next 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 OT. 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-of-$\sqrt{N}$ protocol k times *independently*. Then A is 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 works, 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 $k^4/\sqrt{N}$. 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 OT). *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 σ_j on $1 \cdots N$. 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 σ_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*

$$Y_I = X_I \bigoplus_{j=1}^{W} \left(F_{R^j_{\sigma_{j,R}(I)}}(I) \oplus F_{C^j_{\sigma_{j,C}(I)}}(I) \right),$$

  and sends them to A.
- *To reconstruct the desired inputs A computes for each $I_1, I_2, \ldots, I_k$ the value*

$$X_{I_i} = Y_{I_i} \oplus \bigoplus_{j=1}^{W} \left(F_{R^j_{\sigma_{j,R}(I_i)}}(I_i) \oplus F_{C^j_{\sigma_{j,C}(I_i)}}(I_i) \right).$$

The protocol preserves the privacy of the receiver since the OT 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 OT protocol which provides sender security with probability $1 - \delta$.*

Proof. Let S_1 be the set of k^2 input elements that are mapped by the first permutation σ_1, to the rectangle of size $k \times k$ whose keys are learned by the receiver in the first round (these keys can be extracted using similar methods to those used in the proof of Lemma 2.3). The input elements in S_1 are the only ones about which the receiver learns information in the first round. Note that in order to learn an input element the receiver must learn the keys associated with it in *every* round. Furthermore, in every round the receiver learns the keys of only k rows. In order for the receiver to be able to learn, in a subsequent round, the row keys of more than k of the elements of S_1, at least two of them must be mapped to the same row. The probability that there are two elements of S_1 that 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 δ. Setting $w > \log(1/\delta)/(4\varepsilon \log N)$ satisfies this requirement. □

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

Proof. The proof is composed of two steps. The first step 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 step 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. Let S_1 be the set of k^2 input elements which were mapped, in the first permutation σ_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 specific rectangle of size $k \times k$. Assume that the permutation maps the elements one by one into the rectangle. The probability that the ith element is mapped to the rectangle, given that j, $0 \leq j \leq i-1$, of the previously mapped elements were mapped to the rectangle, is $(k^2 - j)/(N - (i-1))$. This probability is at most $p = k^2/(N - k^2)$, regardless of the values of i and j. It therefore holds that for any set of t input items, $1 \leq t \leq N$, the probability that all t items are mapped by the permutation to the rectangle is at most p^t. Thus it is sufficient to examine the probability distribution where the probability of each element falling in the rectangle is p, independently of the other elements, and bound the probability that $2ek$ or more input elements are mapped to the rectangle. Let x_i be a random variable which is set to 1 if the ith element in S_1 is mapped to the rectangle by this probability distribution, 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 $X \geq 2ek$. Consider the following version of the Chernoff bound (see Theorem A.12 of [1]): *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 $\beta \geq 1$,

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

Returning to our proof, examine the following probability:

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

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

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

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 $(2ek)^2/\sqrt{N} \approx 30N^{-2\varepsilon}$. The probability that this happens in w rounds is $(30N^{-2\varepsilon})^w$ and should be bounded by δ. 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 $W = O(\log(1/\delta)/(\varepsilon \log N))$ communication rounds.

The preprocessing stage requires $2WN$ invocations of a pseudo-random function. The transfer stages require a total of $2W$ executions of k-out-of-$\sqrt{N}$ OT protocols. Each of these transfers can be implemented by running k invocations of 1-out-of-$\sqrt{N}$ OT, and using commitments to ensure that the elements transferred by the sender in each of them are consistent.

The computation overhead consists of the $2WN$ applications of a pseudo-random function for the initialization, additional $2Wk\sqrt{N}$ applications of a pseudo-random function for initializing the 1-out-of-$\sqrt{N}$ OT protocols, and $Wk \log N$ invocations of OT_1^2 for the actual transfers. Alternatively, each k-out-of-$\sqrt{N}$ OT protocol can be implemented recursively, using two k-out-of-$\sqrt[4]{N}$ OT protocols (this means that k must be $o((\sqrt{N})^{1/4}) = o(\sqrt[8]{N})$). The recursion could continue, but it results in even smaller bounds on k (e.g., the next step being $k = o(\sqrt[8]{N})$, which for $N = 10^6$ means a k smaller than 6). Furthermore, continuing the recursion does not improve the overhead. If the recursion stops in a k-out-of-$\sqrt[4]{N}$ protocol then the initialization overhead is reduced to $2WN + 4W^2k\sqrt[4]{N}$ applications of a pseudo-random function, which is still dominated by $2WN$, while the transfer overhead increases to $W^2k \log N$ invocations of OT_1^2.

3. Oblivious Transfer with Adaptive Queries

3.1. *Introduction*

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

This section presents several protocols for k successive (possibly adaptive) OTs, an operation which we denote as $OT^N_{k\times 1}$. The sender performs a single initialization of his input, which requires $O(N)$ work. Each transfer requires only $\log N$ OT^2_1's. In some of the protocols the parameter k does not affect the complexity, and the protocol can even be used for $k = N$ successive transfers.

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

- A large database should be queried in an adaptive fashion.
- The privacy of the 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.

3.1.1. *Protocol Structure*

Protocols for adaptive OT ($OT^N_{k\times 1}$) are composed of two phases, for initialization and for transfer.

The *initialization phase* is run by the sender (Bob) who knows 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 adaptive OT protocol uses commitments, or alternatively committing encryptions, 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 OTs 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.

3.1.2. *Correctness and Security Definitions*

The correctness and security definitions are slight modifications of the definitions for the OT^N_1 case, taking 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 $1 \le I \le N$, 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 adjusting the definition of OT 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 an $OT^N_{k\times 1}$ protocol is rather simple: *for any given* $X_1, X_2, \ldots, X_N$, *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 and in case the receiver tries to obtain $X_{I'}$ are computationally indistinguishable.

The Sender's Security—Comparison with the Ideal Model. We make again a comparison with 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, *operation respecting synthesizers* which are introduced in Section 3.2.1, 1-*out-of*-2 *OT* (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 us 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 scheme can be implemented using the construction of Naor [42], which is based only on the existence of one-way functions. In Section 3.3 we need to use a *trapdoor* commitment scheme (as explained in Section 3.3.3), which can be implemented, for example, using the construction of Chaum et al. [14].

3.2.1. *Operation-Respecting 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 *"operation respectfulness."* 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. Operation-respecting synthesizers can be constructed based on the decisional Diffie–Hellman assumption or based on a function modeled as a random oracle. Section 3.3 describes a construction of an $OT^N_{k\times 1}$ protocol using Diffie–Hellman-based synthesizers, and Section 3.4 describes a construction of an $OT^N_{k\times 1}$ protocol based on any operation-respecting synthesizer.

Pseudo-random synthesizers

Pseudo-random synthesizers were introduced by Naor and Reingold in [46]. 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 $\{S(x_i, y_j) \mid 1 \leq i, j \leq m\}$ of m^2 values is pseudo-random, i.e., indistinguishable from a truly random set (this is a special property which does not hold for any pseudo-random generator G, since it is concerned with inputs which are not independent).

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 any set of combined keys do not leak information about the values of other combined keys.

Operation-respecting synthesizers

We require an additional property from the pseudo-random synthesizers that we use. We define a specific arithmetic operation, such as addition or multiplication, and require *that the synthesizers have the same output for any two input vectors for which the result of applying the operation to the input variables is the same*. For example, for the addition operation and a two-dimensional synthesizer S this implies that for every x_1, y_1, x_2, y_2 that satisfies $x_1 + y_1 = x_2 + y_2$ it holds that $S(x_1, y_1) = S(x_2, y_2)$. More formally, the requirement is as follows:

Definition 3.1 (Operation-Respecting Synthesizer). Given an operation $\otimes$, a function S (defined over m inputs in a commutative group where $\otimes$ is defined) is an operation-respecting synthesizer if the following two conditions hold:

- S is a pseudo-random synthesizer.
- For every $x_1, \ldots, x_m$, and every $y_1, \ldots, y_m$ satisfying $\bigotimes_{i=1}^{m} x_i = \bigotimes_{i=1}^{m} y_i$, it holds that

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

An alternative, and equal, definition could define a function S to be an operation-respecting synthesizer if (1) S is a pseudo-random synthesizer, and (2) there exists an S' such that for all inputs $(x_1, \ldots, x_m)$ it holds that $S(x_1, \ldots, x_m) = S'(\bigotimes_{i=1}^{m} x_i)$.

Comment. In particular, we would use *addition-respecting synthesizers*, where the operation $\otimes$ is addition, and *multiplication-respecting synthesizers*, where the operation $\otimes$ is multiplication.

Construction 1 (Random-Oracle-Based Operation-Respecting Synthesizer). Let *RO* be a function which is modeled as a random oracle. An operation-respecting synthesizer can be realized as

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

In particular, we would use the addition operation and define an addition-respecting synthesizer as

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

Given that the inputs are taken from a domain that is exponentially large (as is the case in the constructions that we use, where each input element is a cryptographic key) this simple construction satisfies Definition 3.1: operation-respectfulness is satisfied trivially. The function S is a pseudo-random synthesizer as long as all the inputs to the random oracle RO are distinct, i.e., if there is a negligible probability that there are two combinations of input variables for which the sum of the input variables is equal given polynomially many uniformly distributed assignments to each of its input variables (the parameters should ensure that this happens with negligible probability).

This construction implies that it is plausible to assume that such functions exists, and also suggests a heuristic approach for constructing them using a "complex" function (e.g., SHA1). We should stress however that security in the random oracle model does not imply provable security (as was demonstrated by [13]). We prefer the number-theoretic construction that is presented next, but on the downside it requires exponentiations which are more complicated to compute than common realizations of "complex" functions.

The following construction introduces multiplication-respecting synthesizers based on the synthesizers of [47] whose security relies on the decisional Diffie–Hellman assumption (the DDH assumption is introduced and discussed in Section 3.3.1 below). This construction is used in Protocols 3.1 and 3.2 described below.

Construction 2 (DDH-Based Multiplication-Respecting Synthesizer). Let $\langle G_g, g\rangle$ be a group and a generator for which the DDH assumption holds. Let the input values $x_1, \ldots, x_m$ be elements in $\{1, \ldots, |G_g|\}$. A multiplication-respecting synthesizer can be realized as

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

This construction trivially satisfies Definition 3.1. The function is the Naor–Reingold DDH-based pseudo-random synthesizer [47]. Its definition ensures that two input vectors, for which the multiplication of the coordinates is equal, have the same output, and the pseudo-randomness is shown in [47].

The basic idea (which is insufficient, as we see below) of using an operation-respecting synthesizer S to construct an adaptive OT protocol is the following. 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). This basic protocol is insufficient, however, since it should also be ensured that Alice is unable to combine the values she obtains in different transfer phases.

3.2.2. *The Protocols*

We present two types of $OT^N_{k\times 1}$ protocols, protocols that are based on the Naor–Reingold synthesizer, and whose security depends on the DDH assumption, and protocols that

can be based on any operation-respecting synthesizer. We start with two DDH-based protocols. These protocols are somewhat simpler than the general construction, since the hardness of the discrete logarithm 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 an $OT^N_{k\times 1}$ protocol based on any operation-respecting synthesizer. This protocol is secure for at most k transfers, where k is a parameter that must satisfy $k = o(\sqrt[4]{N})$ and affects the complexity of the protocol.

3.3. *Protocols Based on the Decisional Diffie–Hellman Assumption*

This section presents two protocols that are based on the Naor–Reingold synthesizer, and consequently on the DDH assumption. The overhead of both protocols is of the same order.

The protocols presented here enable the sender to commit to n values, and ensure that in k transfer rounds the receiver cannot learn the commitment *keys* of more than k of these values. The proof showing that the receiver cannot use these k keys to learn more than k of the committed values is a little intricate, as it must handle the issue of *selective decommitment* and requires the use of trapdoor commitments and a slight change to the protocol. We defer the discussion of this proof to Section 3.3.3.

3.3.1. *The Decisional Diffie–Hellman Assumption*

The DDH assumption is used as the underlying security assumption of many cryptographic protocols (e.g., the Diffie–Hellman key agreement Protocol [19], the ElGamal encryption scheme [22], the Naor–Reingold pseudo-random functions [47], and the Cramer–Shoup construction of a cryptosystem secure against chosen ciphertext attacks [16]).

The DDH assumption is thoroughly discussed in [9]. 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 Naor and Reingold [47].

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 individual invocations of an $OT^{\sqrt{N}}_1$ protocol. In Section 3.3.4 we present a protocol which arranges the elements in a structure with $\log N$ dimensions and uses OT^2_1 as its basic primitive.

Let G be a group, and let G_g be a subgroup of G generated by g in which the DDH assumption holds. The protocol uses the operation-respecting, DDH-based, synthesizer of Construction 2. The range of that synthesizer is G_g, which is a non-standard range for the keys of commitment schemes. We therefore use, as suggested in [46] (based on an analysis using the leftover hash lemma), a method that uses pairwise independent hash

functions to generate an output which is indistinguishable from uniformly distributed bit strings.

Protocol 3.1. *B's input is* $X_1, X_2, \ldots, X_N$, *where* $N = 2^\ell$. *Rename these inputs as* $\{X_{i,j} \mid 1 \le i, j \le \sqrt{N}\}$.

1. **Initialization:**
   (a) *Let* $\{H\}$ *be a family of pairwise independent hash functions from* G_g *to* $\{0,1\}^{|G_g|/2}$. *B chooses a random function* $h \in \{H\}$.
   (b) *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* $K_{i,j} = h(g^{R_i C_j})$, *and a commitment* $Y_{i,j}$ *of* $X_{i,j}$ *using this key,* $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.
   (c) *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* $r_{\mathrm{R}}, r_{\mathrm{C}}$ (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 an* $OT_1^{\sqrt{N}}$ *protocol on the values* $\langle R_1 \cdot r_{\mathrm{R}}, R_2 \cdot r_{\mathrm{R}}, \ldots, R_{\sqrt{N}} \cdot r_{\mathrm{R}} \rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $R_i \cdot r_{\mathrm{R}}$.
   (c) *A and B engage in an* $OT_1^{\sqrt{N}}$ *protocol on the values* $\langle C_1 r_{\mathrm{C}}, C_2 \cdot r_{\mathrm{C}}, \ldots, C_{\sqrt{N}} \cdot r_{\mathrm{C}} \rangle$. *If A wants to learn* $X_{i,j}$ *she should pick* $C_j \cdot r_{\mathrm{C}}$.
   (d) *B sends to A the value* $g^{1/(r_{\mathrm{R}} r_{\mathrm{C}})}$.
   (e) *A reconstructs* $K_{i,j}$ *as* $K_{i,j} = h((g^{1/(r_{\mathrm{R}} r_{\mathrm{C}})})^{(R_i r_{\mathrm{R}}) \cdot (C_j r_{\mathrm{C}})})$, *and uses it as a decommitment key to open the commitment* $Y_{i,j}$ *and reveal* $X_{i,j}$.

Theorem 3.1. *Protocol* 3.1 *is an adaptive OT protocol, and can be used for N adaptive transfers.*

Proof. The receiver can clearly use the protocol to obtain any value she wishes to receive from the sender. The key is reconstructed is Step 2(e) using the multiplication-respecting synthesizer of Construction 2. Namely, $K_{i,j} = h(S(1/(r_{\mathrm{R}} r_{\mathrm{C}}), R_i r_{\mathrm{R}}, C_j r_{\mathrm{C}})) = h(g^{(1/(r_{\mathrm{R}} r_{\mathrm{C}})) \cdot (R_i r_{\mathrm{R}}) \cdot (C_j r_{\mathrm{C}})})$. The operation-respectfullnes of the synthesizer ensures that the key is reconstructed correctly.

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. 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, without loss of generality, 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 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 OT 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 privacy of the sender B is guaranteed by the privacy of the sender in the OT protocols, and by the pseudo-randomness of the Naor–Reingold synthesizer (that is based on the DDH assumption). To prove that the privacy of the sender is preserved we compare A with 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 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 that can adaptively ask and obtain the commitment keys of k elements.*

Proof. We show that for every algorithm A' run by the receiver, there is an algorithm A'' that she can run in the ideal model, such that the outputs of A' and A'' are computationally indistinguishable. We consider the following scenarios in which the receiver could operate:

- E_1—the two-party interaction between the receiver and the sender, as defined by Protocol 3.1.
- E_2—this scenario is the same as scenario E_1, except for the following change: instead of the receiver running OT protocols in Steps 2(b) and 2(c), she interacts with an ideal party, sends it indexes i and j of her choice, and receives the values $R_i \cdot r_{\mathrm{R}}$ and $C_j \cdot r_{\mathrm{C}}$.
- E_3—in this scenario the receiver interacts with a trusted party, and the input elements of the sender are committed to using random keys. The receiver sends to the trusted party indexes of input elements X_I she wants to learn, and receives the keys corresponding to these elements.

Suppose that there is a receiver algorithm A' that is executed in scenario E_1, for which there is no algorithm A'', which is executed in scenario E_3, such that the outputs of A' and A'' are computationally indistinguishable. In other words, given A' then for every A'' there is a distinguisher between the outputs of A' and A''.

Consider now algorithms A^* that operate in scenario E_2. Then either it holds that for every A^* there is a distinguisher between the outputs of A' and A^* (we denote this as case 1), or otherwise there is an A^* whose output is computationally indistinguishable from that of A', and therefore it holds that for every A'' operating in E_3 there is a distinguisher between the outputs of A'' and this A^* (we denote this as case 2).

Consider case 1. The only difference between scenarios E_1 and E_2 is that in E_2 the receiver obtains values $R_i \cdot r_{\mathrm{R}}$ and $C_j \cdot r_{\mathrm{C}}$ of her choice, whereas in E_1 she uses 1-out-of-$\sqrt{N}$ OT protocols to obtain these values. The existence of the distinguisher therefore contradicts (using a hybrid argument) the sender's privacy in the 1-out-of-$\sqrt{N}$ OT protocol.

Consider now case 2. In scenario E_2 the receiver can only learn tuples of the 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 her learning tuples $V_2 = (g^{R_i C_j / r_1 r_2}, r_1, r_2)$, with r_1, r_2 that are uniformly chosen, which have the same distribution. The receiver can easily compute from this information tuples $V_3 = (g^{R_i C_j}, g^{R_i C_j / r_1 r_2}, r_1, r_2)$, which of course do not contain more information than the keys $g^{R_i C_j}$ alone (that enable the receiver to generate tuples in the same distribution as that of V_3).

In other words, the receiver A^* can only obtain in scenario E_2 values $g^{R_i C_j}$ of her choice, and no other information. These keys were generated using the operation-respecting synthesizer. The only difference from the operation of A'' in scenario E_3 is that in that scenario the keys are random. Therefore, a distinguisher between the outputs of A'' and A^* contradicts the pseudo-randomness of the synthesizer. □

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 implement these exponentiations efficiently 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 an $OT_{k\times 1}^{\sqrt{N}}$ protocol. A slight complication is the fact that in each transfer round the sender uses different keys $r_{\mathrm{R}}, r_{\mathrm{C}}$, and the transfer should be for values of the form $R_i \cdot r_{\mathrm{R}}$ or $C_i \cdot r_{\mathrm{C}}$, which are different in each round. Therefore, the $OT_{k\times 1}^{\sqrt{N}}$ protocol should not use commitments, but rather the sender should send, in each round, encrypted values of $\{R_i \cdot r_{\mathrm{R}}, C_i \cdot r_{\mathrm{C}}\}_{i=1}^{\sqrt{N}}$. This does not enable the sender to change from round to round the values that are transferred in the $OT_{k\times 1}^{N}$ protocol, since the $OT_{k\times 1}^{N}$ 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 proven 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 that 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*. In the scenario of $OT^N_{k\times 1}$ there is a way to enable this property by using *trapdoor commitments*. These are commitment schemes that have a trapdoor that enables opening them to arbitrary values (see [24] for a detailed discussion of trapdoor commitments). The main idea of amending the OT protocol using trapdoor permutations is to use a trapdoor commitment scheme and let the *receiver* choose its public parameters. In general, trapdoor commitment schemes can be based on the existence of one-way functions (see Section 4.9.2.3 of [28]). We describe the construction using a specific trapdoor commitment scheme of Chaum et al. [14]. (For the sake of clarity we do not describe the use of the trapdoor commitment scheme in the body of the protocols that are detailed in the paper, but rather in this separate section.) Next we describe in more detail the changes to the protocols:

- In the beginning of the protocol the receiver A sends to B the trapdoor to a commitment scheme and proves that the trapdoor is correct. (When using the commitment scheme of [14] the receiver sends two values $g_1, g_2 \in G$ and proves in zero-knowledge that she knows the discrete logarithm 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 uses the trapdoor commitments in his part of the protocol.
  In particular, the commitments of [14] are of the form $g_1^a g_2^b$. (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 R_I and computes $C_I = g_1^{X_I} g_2^{R_I}$; (ii) takes the output of the synthesizer and uses it as a key to encrypt (X_I, R_I) by xoring it; and (iii) sends the two results (C_I and the encrypted (X_I, R_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 an R_I that is consistent with it, and give an output of the synthesizer that "decrypts" these values.

More formally, suppose that there is a receiver A' that receives all the commitments, asks for the keys of k of them, 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 the opening of any commitment to any value. Then the N commitments of the form $\langle (X_I, R_I) \oplus K_I, C_I \rangle$ (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 distinguish from random one of the values of the original set of committed values.

3.3.4. *A Protocol Using* OT_1^2

The following protocol constructs $OT_{k\times 1}^N$ using direct invocations of a simple OT_1^2 protocol.

Protocol 3.2. *B's input is* $X_1, X_2, \ldots, X_N$, *where* $N = 2^\ell$.

1. **Initialization:**
   (a) *Let* $\{H\}$ *be a family of pairwise independent hash functions from* G_g *to* $\{0,1\}^{|G_g|/2}$. *B chooses a random function* $h \in \{H\}$.
   (b) *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*[2] *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* $K_I = h(g^{\prod_{j=1}^{\ell} a_j^{i_j}})$, *and a commitment* Y_I *of* X_I *using this key,* $Y_I = commit_{K_I}(X_I)$.
   (c) *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 an* 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* $K_I = h((g^{1/(r_1 r_2 \cdots r_\ell)})^{(a_1^{i_1} r_1)\cdots(a_\ell^{i_\ell} r_\ell)})$, *and uses it to open the commitment* Y_I *and reveal* X_I.

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

Proof. The protocol uses a multiplication-respecting synthesizer with $\ell+1$ inputs. The key is initialized as $S(1, a_1^{i_1}, \ldots, a_\ell^{i_\ell})$, and is reconstructed in Step 2(d) by computing $S(1/(r_1 r_2 \cdots r_\ell), a_1^{i_1} r_1, \ldots, a_\ell^{i_\ell} r_\ell)$. The receiver can therefore obtain any value it wishes to receive in the protocol. The privacy of A is guaranteed by the privacy of the OT_1^2 protocols, and the proof is similar to that of Lemma 3.2 . The receiver A is 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 DDH assumption, and is proven identically to the security of the sender in Protocol 3.1. □

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

[2] 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.

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 [39] 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.

3.4. *Protocols Based on Any Operation-Respecting Synthesizer*

This section presents an $OT_{k\times 1}^N$ protocol which can be based on any operation-respecting synthesizer. In order to emphasize the issues underlying the protocol, we first describe a protocol that is *insecure*, examine it, and construct a secure protocol.

3.4.1. *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 $\{x_{i,j} \mid 1 \le i, j \le \sqrt{N}\}$, where $N = 2^\ell$. Let $S(x, y)$ be an operation-respecting 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 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 an $OT_1^{\sqrt{N}}$ 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 an $OT_1^{\sqrt{N}}$ 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 protocol, 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 problem does not exist for DDH-based solutions since the DDH assumption implies that it is hard to compute relations similar to the linear relations outlined above.)

3.4.2. *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. The mapping is composed of N permutations that map the N input elements to locations in each of a set of matrices (each such location corresponds to a key in the protocol). The desired property is that for every k inputs it holds there is no linear combination of the keys associated with them, which covers the keys of a different input. The following construction of a set of permutation matrices is used to construct a protocol for $k = o(\sqrt{N})$:

Definition 3.2 (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 $\{(i, j, k) \mid 1 \leq i \leq t,\ 1 \leq j \leq 2,\ 1 \leq k \leq \sqrt{N}\}$ (i.e., i represents the matrix, j indicates a choice of 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 $1 \leq i \leq t$, 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 relations between keys and the resulting protocol is good for OT with adaptive queries. The security of the protocol is based on the following lemma.

Lemma 3.5. *Consider a set of t matrices of size $\sqrt{N} \times \sqrt{N}$, and a set of t random permutations of $\{1, \ldots, N\}$. These permutations map the values $1, \ldots, N$ to locations in each of the matrices. Then with high probability this set is k-out-of-n relation-free for*

$$t = \frac{2\log(N/k)}{\log(\sqrt{N}/k)} + 1 = \frac{5\log N - 6\log k}{\log N - 2\log k}.$$

Proof. The vectors corresponding to the matrices contain $2t\sqrt{N}$ coordinates. Call the coordinates of 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 the number of these vectors is at least 2. We examine

the probability that this property holds for a single region. This probability is the same as that of throwing $s = k + 1$ balls into $n = \sqrt{N}$ bins independently at random, and witnessing the event that there are at least two balls in every non-empty bin (there are at most $s/2$ such bins). When this event happens, it is possible to remove one ball from every bin that has an odd number of balls and partition the remaining balls (of which there are at least $s/2$) into pairs, such that the balls in each pair fall into the same bin. (If each bin contains either none or two remaining balls, then there is only one such partition into pairs. If there is a bin with four or more balls then there are several such partitions.) The probability of such a partition happening is

$$\frac{(s/2)!}{(s/4)! \cdot 2^{s/4}} \cdot \frac{1}{n^{s/4}} < \left(\frac{s}{4n}\right)^{s/4}.$$

In other words, it is at most $((k+1)/4\sqrt{N})^{(k+1)/4}$.

The probability that this property holds for all regions is at most $((k+1)/4\sqrt{N})^{(k+1)t/2}$. We apply the probabilistic method and require that this probability be smaller than the inverse of the number of subsets of $k+1$ elements, $1/\binom{N}{k+1} \approx ((k+1)/eN)^{k+1}$. This holds for

$$t \geq 2 \cdot \frac{\log(eN/(k+1))}{\log(4\sqrt{N}/(k+1))} \approx \frac{2\log(N/k)}{\log(\sqrt{N}/k)}.$$

Therefore, setting $t = 2\log(N/k)/\log(\sqrt{N}/k) + 1$ satisfies the requirement of the theorem. □

It should be interesting to design an explicit construction of k-out-of-n relation-free matrices.

An overview of the protocol

On a high level Protocol 3.3 is transformed in the following way: 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 an operation-respecting 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 zero. A and B run $2t$ $OT_1^{\sqrt{N}}$ protocols, which let A learn each of the relevant inputs of 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 OT Based on Any Operation-Respecting Synthesizer). *B's input is $\{x_{i,j} \mid 1 \leq i, j \leq \sqrt{N}\}$, where $N = 2^\ell$. B maps the inputs into t square matrices*

independently at random. Let x_{R}^m denote the row into which x is mapped in matrix m, and let x_{C}^m denote the column into which x is mapped in matrix m.

Let $S(x_1, \ldots, x_{2t})$ be an operation-respecting 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 $1 \le i, j \le \sqrt{N}$, B prepares a commitment key

$$K_{i,j} = S\left(R^1_{(x_{i,j})^1_{\mathrm{R}}}, C^1_{(x_{i,j})^1_{\mathrm{C}}}, \ldots, R^t_{(x_{i,j})^t_{\mathrm{R}}}, C^t_{(x_{i,j})^t_{\mathrm{C}}}\right).$$

   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:** *The parties invoke the following protocol for each $X_{i,j}$ that A wants to learn*:
   (a) *B chooses random elements $r_{\mathrm{R}}^1, r_{\mathrm{C}}^1, \ldots, r_{\mathrm{R}}^t, r_{\mathrm{C}}^t$, such that their sum is zero. (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*:
   - *An $OT_1^{\sqrt{N}}$ protocol on the values $\langle R_1^m + r_{\mathrm{R}}^m, R_2^m + r_{\mathrm{R}}^m, \ldots, R_{\sqrt{N}}^m + r_{\mathrm{R}}^m\rangle$. If A wants to learn $X_{i,j}$ she should pick $R^m_{(x_{i,j})_r^m} + r_{\mathrm{R}}^m$.*
   - *An $OT_1^{\sqrt{N}}$ protocol on the values $\langle C_1^m + r_{\mathrm{C}}^m, C_2^m + r_{\mathrm{C}}^m, \ldots, C_{\sqrt{N}}^m + r_{\mathrm{C}}^m\rangle$. If A wants to learn $X_{i,j}$ she should pick $C^m_{(x_{i,j})_{\mathrm{c}}^m} + r_{\mathrm{C}}^m$.*
   (c) *A reconstructs $K_{i,j}$ as*

$$K_{i,j} = S\left(R^1_{(x_{i,j})^1_{\mathrm{R}}} + r_{\mathrm{R}}^1, C^1_{(x_{i,j})^1_{\mathrm{C}}} + r_{\mathrm{C}}^1, \ldots, R^t_{(x_{i,j})^t_{\mathrm{R}}} + r_{\mathrm{R}}^t, C^t_{(x_{i,j})^t_{\mathrm{C}}} + r_{\mathrm{C}}^t\right)$$

   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 $\sqrt{N}$ only a few matrices are needed).

Theorem 3.6. *The above $OT_{k\times 1}^N$ protocol with*

$$t = \frac{2\log(N/k)}{\log(\sqrt{N}/k)} + 1 = \frac{5\log N - 6\log k}{\log N - 2\log k}$$

matrices is secure.

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

Proof. The privacy of the receiver is preserved since each $OT_1^{\sqrt{N}}$ protocol preserves her privacy. As for the sender's privacy, the same arguments that were used to prove Lemma 3.3 show that if the output of the receiver is computationally distinguishable from that of a receiver in the ideal model, then either (1) the sender's privacy is not preserved in the $OT_1^{\sqrt{N}}$ protocol, or (2) given the information the receiver is prescribed to learn in k invocations of the OT protocols she can learn the commitment keys of $k+1$ or more inputs. Case (1) should not happen since we assume the $OT_1^{\sqrt{N}}$ to be secure. As for case (2), 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 learned 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.) Now, after k invocations of the protocol the receiver learns k linear equations of the keys. If these k equations span the commitment keys of $k+1$ elements then this is a contradiction to the k-out-of-N relation-freeness property of the matrices. If they do not, then the receiver can learn information about at most k keys. The methods described in Section 3.3.3 (discussing the selective decommitment problem) can be used in order to show that these k keys do not enable the receiver to learn more than k committed input elements. □

4. Applications

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

4.1. *Applications of* 1*-out-of-N Oblivious Transfer*

Two applications of 1-out-of-N OT are described here. Another application is oblivious polynomial evaluation [45].

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-communicating replicated databases) was shown to be unnecessary by Kushilevitz and Ostrovsky [36], who proposed a PIR scheme with a single server—where the user's security depends on the Quadratic Residuosity assumption modulo a Blum integer and the communication complexity is $O(N^{\varepsilon}m)$ (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. [11] where the communication complexity is poly-logarithmic in N.

More recently attention was given to the question of protecting the database as well, i.e., ensuring that the user does 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 Symmetric PIR, or SPIR. In [26] 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 [36] suggest an adaptation of their single server PIR protocol to enable the receiver to learn only 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 OT with any PIR protocol provides a SPIR protocol which does not require adding any new servers and requires relatively little work on behalf of the parties involved. In particular, the overhead is only the sum of the overheads of the PIR and OT 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 OT protocols emphasize the server's privacy. One can regard a SPIR construction as a combination of 1-out-of-N OT and PIR which provides low communication complexity, and privacy, for both parties. SPIR is essentially OT 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 the encrypted values $Y_1, Y_2, \ldots, Y_N$ 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 and Oblivious Transfer with Adaptive Queries. OT_1^N protocols enable an efficient transformation of any PIR protocol to a SPIR protocol. Adaptive OT protocols could 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

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

	PIR	**OT**		**SPIR**
Receiver privacy	+	+		+
Sender privacy		+	$\Rightarrow$	+
Communication	$o(N)$	$O(N)$		$o(N)$

a protocol is introduced, $OT^N_{k\times 1}$ would enable us to transform it immediately to an SPIR protocol).

PIR versus Oblivious Transfer. On a more practical level, we believe that it is preferable to use the computation efficient OT^N_1 and $OT^N_{k\times 1}$ 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 [36], [11] are very costly to implement since they require $O(N)$ exponentiations or modular multiplications 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: a search engine claims to have the largest searchable database of all search engines. Alice would like to check this claim and measure the number of URLs *indexed* by this search engine, i.e., web pages that can be searched for using its search interface. She might also like to check the overlap between the pages indexed by this search engine and by other search engines, and this task also requires a random sample of the pages indexed by the search engine (see [5]). One possibility is for the search engine to give Alice the list of the URLs it has indexed; Alice will then 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 the search engine. The problem is of course that the list of URLs is a trade secret and the search engine 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 the search engine's web interface. The selection procedure must:

- Keep most of the list (the part not sampled) secret from Alice.
- Prevent the search engine 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 the search engine, 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

the search engine to solve their problem at a small cost, even if the engine's databases consists of hundreds of millions of pages. If the search engine claims to have indexed N URLs it should feed them as the input $X_1, \ldots, X_N$ 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 the search engine 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 the search engine and asks it what was the index I that Alice chose when she sampled the URL. If the search engine does not answer correctly—then Alice can conclude that the search engine was cheating by duplicating URLs.

4.2. *Applications for Searching*

Oblivious transfer is useful for making private search queries in a large private database. Namely, for ensuring that the party doing the queries learns nothing more than the result of each query, and that the queries themselves are kept hidden from the database owner.

If the items in the database are indexed from 1 to N and N is of moderate size, then the party requesting the query (the *receiver*) should simply run a 1-out-of-N OT in which she retrieves the item she is interested in. This type of search is useful for example for searching *patent databases*, i.e., in a scenario where Bob holds a patent database, does not want to give the whole database to other parties, but is willing to let other people search it. Alice wants to search the database while hiding her queries (which might reveal the great new invention she is working on). Note that in this scenario Alice's search might be adaptive: after she retrieves a certain patent Alice might ask to read patents which are referenced by this patent. Alice and Bob can use $OT^N_{k\times 1}$ to enable Alice to search Bob's database adaptively without revealing her queries to him.

4.2.1. *Keyword Search*

Assume again that Bob owns a database which Alice wants to search, but now each record has a keyword (e.g., a user name) which identifies it. Namely, the database is composed of items $X_{k_1}, \ldots, X_{k_N}$, where $k_1, \ldots, k_N$ are the keywords of the records, and Alice's input is a keyword s. The situation is complicated since the keywords might come from a large domain which contains much more than N items.

Suppose first that the database is sorted according to the keywords and Alice would be using *binary search*. The two parties can invoke an $OT^N_{\log N\times 1}$ protocol to perform this search without revealing to Bob the element that Alice is searching for. This search limits Alice's knowledge to $\log N$ elements of the database.

Ideally, we would like to limit Alice's output to a single value, namely to let her learn X_s if $s \in \{k_1, \ldots, k_N\}$, and nothing otherwise. We design such a solution using *perfect hash* functions [25]. A hash function is perfect for an input set $k_1, \ldots, k_N$ if it maps it to its range without collisions, and we are interested in perfect hash functions whose range is of size $O(N)$. The common structure for constructions of perfect hash functions uses *two-level hash*, using two levels of hash functions. The first function maps data elements

into bins. A different hash function is associated with each bin, and is used to map further the elements that were mapped to the bin.

Keyword search can be implemented using perfect hash functions in the following way. Bob uses a perfect hash function H to map the keywords of his input to a domain $\{1, \ldots, N'\}$, where $N' = O(N)$. He chooses a random pairwise independent function R, constructs a table T of size N', and stores the values $R(k_i)$ in location $H(k_i)$, namely in the location to which k_i is mapped by the perfect hash function. In the retrieval stage Alice computes $H(s)$ and uses OT to learn entry $H(s)$ of the table T. She compares this value with $R(s)$. If these two values are equal she concludes that the entry keyed by s is in the database. (She can then also use a key derived from $R(s)$ to decrypt an encryption of X_s, if such an encryption is stored in this entry.)

It is obvious that Bob learns nothing about Alice's query, and that if the hash function is perfect then the protocol is correct. It is more complicated to design the protocol such that Bob's privacy is also preserved. First, Alice should not be able to compute the hash function $R()$ more than a single time. Otherwise, after she learns the value of the entry of T she could try several keywords and check whether they are mapped by the function R to this value. Luckily, $R()$ can be implemented as a linear polynomial, and there are efficient protocols [45] that enable Bob to let Alice obliviously compute a single output of the polynomial. It is slightly harder to let Alice compute the output $H(s)$ of the hash function H, while hiding from her any information about H except for $H(s)$. This is essential since constructions of perfect hash functions depend on the input set $k_1, \ldots, k_N$ in order to prevent collisions. Therefore if Alice learns information about H she might learn about Bob's input (for example, if she learns that $H(s) = H(t)$ she knows that it cannot be the case that both s and t are in Bob's input).

4.2.2. *An Oblivious Perfect Hash Function*

We describe a construction of a perfect hash function with a range of size $O(N)$, similar to the class of perfect hash functions suggested in [48]. Unlike the construction of [25], it can be evaluated (at least once) without revealing information about the input set $k_1, \ldots, k_N$. The first step of the construction uses a hash function f to map the inputs to N bins such that the following property holds: Let b_i be the number of items which are mapped to the ith bin, then $\sum_{i=1}^{N} \binom{b_i}{2} < \sum_{i=1}^{N} (b_i)^2 < cN = O(N)$. This property is satisfied by at least half of the choices of hash functions of the form $h_a(x) = (ax \bmod p) \bmod N$, where p is prime and we assume that $k_i \in \{0, p-1\}$ (see Corollary 8.20 in [41]).

The next step maps the items into a table T of size cN. First choose a random pairwise independent hash function g, whose range is $[1, cN]$, and which can be made public. Then for every bin assign a secret offset d_i. Item x in bin i is mapped to a location in T by the function $g(x) + d_i \bmod cN$. For the analysis, first note that the expected number of collisions of two items taken from the *same* bin is $(1/cN) \sum_{i=1}^{N} \binom{b_i}{2} < 1$ and therefore the constant c can be set to ensure that a pairwise independent hash function generates such a collision with probability smaller than $1/2$. It is also required to choose offsets d_i such that no collisions occur between items from different bins. This is indeed possible if offsets are assigned starting from the most populated bin to the least populated one. Assume, without loss of generality, that $b_1 \geq b_2 \geq \cdots \geq b_N$. The offset assigned to bin i should prevent collisions with items of bins $b_1, \ldots, b_{i-1}$. The number of possible

collisions is at most $b_i \cdot (b_1 + b_2 + \cdots + b_{i-1}) \leq b_1^2 + b_2^2 + \cdots + b_{i-1}^2 < cN$, and therefore there is always an offset which does not cause any collision.

The last step chooses a random rotation $0 \leq r \leq N-1$, assigns the value $d_i + r \mod N$ to bin i, and defines the hash function as $H(x) = g(x) + d_{f(x)} + r \mod cN$.

Performing the search

The most straightforward method for performing the search requires Alice to perform one *oblivious* evaluation of the hash function f of the first level. The search is composed of the following steps: (1) Alice obliviously computing $f(s)$. (2) Alice using 1-out-of-N OT to learn the offset $r + d_{f(s)} \mod cN$. (3) Alice using 1-out-of-cN OT to learn the value in entry $g(s) + d_{f(s)} + r \mod cN$ in table T. (4) Alice computing an oblivious evaluation of $R(s)$, and comparing the result to the entry of T that she learned. Note that the only information learned by Alice is one output of the function f, two outputs of the function R, and the value of an offset. Bob's privacy is preserved since the functions f and R are pairwise independent, and each offset is distributed uniformly at random.

An alternative method relaxes the need for an oblivious evaluation of $f(s)$. The system is changed and includes $e = \log(1/\varepsilon)$ first level hash functions $f_1, \ldots, f_e$ (where ε is an error probability), which are public and are known to Alice. These hash functions are chosen by Bob independently of his input. It therefore holds that with probability at least $1 - \varepsilon$ there is a function f_i that maps the inputs such that the sum of the squares of the number of items mapped to the bins is less than cN. Assume, without loss of generality, that this property holds for f_1. Bob constructs e sets of second level hash functions, and e tables $T_1, \ldots, T_e$. He chooses random values $r_2, \ldots, r_e$ and sets *all* the entries of table T_i ($i \geq 2$) to be r_i. Then for each input value k_j he sets the value of the entry to which it is mapped in T_1 to be $R(k_j) \oplus r_2 \oplus \cdots \oplus r_e$.

During the search operation Alice finds the locations to which s is mapped in each of the e tables (note that she knows the functions $f_1, \ldots, f_e$ and does not need to compute them obliviously). She then uses invocations of 1-out-of-cN OT to learn the values in these entries. She computes the exclusive-or of these values and compares it with $R(s)$. Note that in this process all the f_i functions are treated symmetrically, and Alice does not learn which one of them supports perfect hashing of the input set.

5. Conclusions

Oblivious transfer is a fundamental tool of secure computation. We presented efficient constructions of a computationally secure 1-out-of-N OT protocol, variants that support multiple and adaptive queries, and several applications. Another application is communication preserving protocols for secure function evaluation [43]. Our approach uses a logarithmic number of 1-out-of-2 OTs compared with $\Omega(N)$ 1-out-of-2 OTs that must be used by any information-theoretic secure construction. It is also more efficient than a generic solution that applies Yao's two-party protocol to a circuit that selects one of N inputs (that solution might use $\log N$ 1-out-of-2 OTs and $O(N)$ communication, but the constants are larger than in our solution).

Acknowledgments

We thank Sanjeev Arora, Andrei Broder, Oded Goldreich, and Yuval Ishai, as well as the anonymous referees.

References

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

[2] R. Anderson and M. Kuhn, Tamper resistance – a cautionary note, *Proc. Usenix Electronic Commerce Workshop*, *Oakland*, pp. 1–11, 1996.

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

[6] G. Brassard, C. Crépeau, and J.-M. Robert, Information theoretic reduction among disclosure problems, *Proc. 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, Berlin, pp. 234–238, 1987.

[8] G. Brassard, C. Crépeau, and M. Santha, Oblivious transfer and intersecting codes, *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1769–1780, 1996.

[9] D. Boneh, The decision Diffie–Hellman problem, *Proc. Third Algorithmic Number Theory Symposium*, LNCS 1423, Springer-Verlag, Berlin, pp. 48–63, 1998.

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

[11] C. Cachin, S. Micali and M. Stadler, Computationally private information retrieval with polylogarithmic communication, *Advances in Cryptology – Eurocrypt* '99, LNCS 1592, Springer-Verlag, Berlin, pp. 402–414, 1999.

[12] R. Canetti, Security and composition of multiparty cryptographic protocols, *J. Cryptology*, vol. 13, pp. 143–202, 2000.

[13] R. Canetti, O. Goldreich, and S. Halevi, The random oracle methodology, revisited, *J. ACM*, vol. 51, no. 4, pp. 557–594, 2004.

[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, *Proc. 36th FOCS*, pp. 41–50, 1995.

[16] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attacks, *Proc. Advances in Cryptology – Crypto* '98, LNCS 1462, Springer-Verlag, Berlin, pp. 13–25, 1998.

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

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

[19] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[20] Y. Dodis and S. Micali, Lower bounds for oblivious transfer reductions, *Proc. Advances in Cryptology – Eurocrypt* '99, LNCS 1592, Springer-Verlag, Berlin, pp. 42–55, 1999.

[21] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer, Magic functions, *Proc. 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, LNCS 196, Springer-Verlag, Berlin, pp. 10–18, 1985.

[23] S. Even, O. Goldreich, and A. Lempel, A randomized protocol for signing contracts, *Comm. ACM*, vol. 28, pp. 637–647, 1985.
[24] M. Fischlin, Trapdoor commitment schemes and their applications, Ph.D. dissertation, University of Frankfurt am Main, 2002.
[25] M. L. Fredman, J. Komlos, and R. Szemeredi, Storing a sparse table with $O(1)$ worst case access time, *J. ACM*, vol. 31, pp. 538–544, 1984.
[26] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, Protecting data privacy in private information retrieval schemes, *Proc*. 30*th STOC*, pp. 151–160, 1998.
[27] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan, The relationship between public key encryption and oblivious transfer, *Proc*. 41*st FOCS*, pp. 325–335, 2000.
[28] O. Goldreich, *The Foundations of Cryptography - Volume* 1, *Basic Tools*, Cambridge University Press, Cambridge, 2001.
[29] O. Goldreich, *The Foundations of Cryptography - Volume* 2, *Basic Applications*, Cambridge University Press, Cambridge, 2004.
[30] O. Goldreich, S. Goldwasser, and S. Micali, How to construct random functions, *J. ACM*, vol. 33, pp. 792–807, 1986.
[31] O. Goldreich, S. Micali, and A. Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, *Proc*. 19*th STOC*, pp. 218–229, 1987.
[32] O. Goldreich and R. Vainish, How to solve any protocol problem - an efficiency improvement, *Advances in Cryptology - Crypto '*87, LNCS 293, Springer-Verlag, Berlin, pp. 73–86, 1988.
[33] R. Impagliazzo and S. Rudich, Limits on the provable consequences of one-way permutations, *Proc. STOC '*89, pp. 44–61, 1989.
[34] Y. Ishai, Personal communication, 1998.
[35] J. Kilian, *Use of Randomness in Algorithms and Protocols*, MIT Press, Cambridge, MA, 1990.
[36] E. Kushilevitz and R. Ostrovsky, Replication is not needed: single database, computationally-private information retrieval, *Proc*. 38*th FOCS*, pp. 364–373, 1997.
[37] M. Luby, *Pseudo-Randomness and Applications*, Princeton University Press, Princeton, NJ, 1996.
[38] S. Lucks, Open key exchange: how to defeat dictionary attacks without encrypting public keys, *Proc. Security Protocol Workshop '*97, pp. 79–90, 1997.
[39] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.
[40] S. Micali and P. Rogaway, Secure computation, *Advances in Cryptology – Crypto '*91, LNCS 576, Springer-Verlag, Berlin, pp. 392–404, 1992.
[41] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, 1995.
[42] M. Naor, Bit commitment using pseudo-randomness, *J. Cryptology*, vol. 4, pp. 151–158, 1991.
[43] M. Naor and K. Nissim, Communication preserving protocols for secure function evaluation, *Proc*. 33*rd STOC*, pp. 590–599, 2001.
[44] M. Naor and B. Pinkas, Efficient oblivious transfer protocols, *Proc. SODA* 2001 (*SIAM Symposium on Discrete Algorithms*), pp. 448–457, 2001.
[45] M. Naor and B. Pinkas, Oblivious polynomial evaluation, Manuscript, 2004.
[46] M. Naor and O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions, *Proc*. 36*th FOCS*, pp. 170–181, 1995.
[47] M. Naor and O. Reingold, Number-theoretic constructions of efficient pseudo-random functions, *Proc*. 38*th FOCS*, pp. 458–467, 1997.
[48] R. Pagh, Hash and displace: efficient evaluation of minimal perfect hash functions, *Proc. WADS '*99, LNCS 1663, Springer-Verlag, Berlin, pp. 49–54, 1999.
[49] M. O. Rabin, How to exchange secrets by oblivious transfer, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.
[50] S. Wiesner, Conjugate coding, *SIGACT News*, vol. 15, pp. 78–88, 1983.
[51] A. C. Yao, How to generate and exchange secrets, *Proc*. 27*th FOCS*, pp. 162–167, 1986.