Oblivious Transfer and Polynomial Evaluation

Extended Abstract

Moni Naor* Benny Pinkas†

Abstract

We describe efficient constructions for two oblivious two-party computation problems: 1-out-of-N Oblivious Transfer and Oblivious Polynomial Evaluation. The oblivious polynomial evaluation protocol is based on a new intractability assumption which is closely related to noisy polynomial reconstruction. A direct corollary of the 1-out-of-N OT protocol is an efficient transformation of any Private Information Retrieval (PIR) protocol to a Symmetric PIR (SPIR) protocol without increasing the number of databases.

The new construction for 1-out-of-N OT is highly efficient – it requires only $\log N$ executions of a 1-out-of-2 OT protocol. We also present a construction for k-out-of-N OT which is more efficient than k repetitions of 1-out-of-N OT. The efficiency of the new OT protocols makes them useful for a variety of applications. These include *oblivious sampling* which can be used to securely compare the sizes of web search engines, protocols for privately solving the *list intersection problem* and for mutually authenticated key exchange based on (possibly weak) passwords, and protocols for anonymity preserving web usage metering.

1 Introduction

An oblivious evaluation protocol for a function $f(\cdot,\cdot)$ allows two parties, Alice who knows x and Bob who knows y, to jointly compute the value of $f(x,y)$ in a way that does not reveal to each side more than the information that can be deduced from $f(x,y)$. The fact that for every polynomially computable function $f(\cdot,\cdot)$ there exists such a (polynomially computable) protocol is one of the most remarkable achievements of research in foundations of cryptography. However, the resulting protocols are often not as efficient since the number of cryptographic operations performed is proportional to the *size of the circuit computing* $f(x,y)$ [43]. Even for relatively simple functions this may be prohibitively expensive. Therefore it is interesting to investigate for which functions it is possible to come up with a protocol that does not emulate the circuit for the function.

In this paper we present efficient protocols for two functions/problems. One is 1-out-of-N Oblivious Transfer where Bob knows N values and would like to let Alice choose any one of them in such a way that she does not learn more than one, and he remains oblivious to the value she chooses. This is a well known problem and we discuss 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 second problem is Oblivious Evaluation of Polynomials, where a polynomial P is known to Bob and he would like to let Alice compute the value $P(x)$ for an input x known to her in such a way that Bob does not learn x and Alice does not gain any additional information about P (except $P(x)$). This problem has not been investigated so far; we find it to be a useful primitive as it can act as a cheap replacement for pseudo-random functions, in case only k-wise independence is needed. We present three applications for it. One is the list intersection problem: a method for two agencies each having a list of names to find the common names on the lists without revealing other information. This has obvious significance in keeping privacy. Another application is for mutually authenticated key exchange based on (possibly weak) passwords, The third application is for an anonymity preserving initialization protocol for a method that allows metering on the web [35].

The protocol for 1-out-of-N Oblivious Transfer is based on standard cryptographic assumptions whereas the protocol for Oblivious Polynomial Evaluation (in addition to relying on the former) needs a new intractability assumption which is related to the hardness of the noisy polynomial reconstruction problem.

Organization: In the rest of this section we discuss in more detail oblivious transfer and oblivious function evaluation, and present appropriate security definitions. Section 2 presents protocols for 1-out-of-N and k-out-of-N OT. Section 3 discusses the security definition that is used for oblivious polynomial evaluation, and presents a protocol for this problem. Section 4 describes various applications for the protocols we present.

*Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by grant no. 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

†Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by an Eshkol Fellowship. E-mail: bennyp@wisdom.weizmann.ac.il.

STOC '99 Atlanta GA USA

1.1 Oblivious Transfer

Oblivious Transfer (abbrev. OT) refers to several types of two-party protocols where at the beginning of the protocol one party, the Sender (or sometimes Bob or B), has an input and at the end of the protocol the other party, the receiver (or sometime Alice or A), learns some information about this input in a way that does not allow the sender Bob 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 Bob learning anything about I.

1-out-2 OT was suggested by Even, Goldreich and Lempel [18], as a generalization of Rabin's "oblivious transfer" [40] (the notion was also developed independently by Wiesner in the 1970's, but not published till [42]). 1-out-of-N Oblivious Transfer was introduced by Brassard, Crépeau and Robert [6] under the name ANDOS (all or nothing disclosure of secrets). For an up-to-date definition of OT and oblivious function evaluation see Goldreich [22].

Since its proposal OT has enjoyed a large number of applications and in particular Kilian [29] and Goldreich and Vainish [25] have shown how to use OT in order to implement general oblivious function evaluation, i.e. how Alice and Bob can evaluate any function of their inputs without revealing more information than necessary. There are many application for 1-out-of-N OT in case N is relatively large. We will see two direct applications in Section 4 and will also use 1-out-of-N OT in our Oblivious Polynomial Evaluation protocol (Section 3).

Reduction between various types of oblivious transfer protocols has been investigated extensively and they all turn out to be information theoretically equivalent (See [5, 7, 14, 13, 9]). This is of interest, given the possibility of implementing OT using "physical means", e.g. via a noisy channel or quantum cryptography. However, some of these reductions are not particularly efficient and in this paper we use non-information theoretic reductions, i.e. we employ additional cryptographic primitives, to obtain very efficient reductions. In particular we apply pseudo-random functions which can be based on one-way functions[1].

Staying in the complexity based world, i.e. 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. [5, 18, 2]). Essentially every known suggestion of public-key cryptography allows also to implement OT (although there is no general theorem that implies this state of affairs), and the complexity of 1-out-of-2 OT is typical of public-key operations [5, 2]. OT can be based on the existence of trap-door permutations, factoring, Diffie-Hellman and the hardness of finding short vectors in a lattice (the Ajtai-Dwork cryptosystem). On the other hand, given an OT protocol it is a simple matter to implement secret-key exchange using it. Therefore from the work of Impagliazzo and Rudich [28] it follows that there is no black-box reduction of OT from one-way functions.

[1] A pseudo-random function is one that cannot be distinguished from a truly random one by a polynomial-time observer who is given access to the function in a black-box manner. (See [23, 21, 31, 37, 38] for a precise definition and various constructions).

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 an application of a pseudo-random generator. This is justified both theoretically, by the separation of [28] mentioned above, and in practice, where one can model a pseudo-random function by very efficient block ciphers which are several orders of magnitude more efficient than operations in public-key cryptography. Our goal is therefore to come-up with efficient constructions of 1-out-N OT protocols from 1-out-2 OT protocols, where the number of invocations of the 1-out-2 OT protocol are small. For instance, the constructions of [5, 7] need N calls to the 1-out-2 OT protocol. In contrast our protocols need only $\log N$ calls to the 1-out-2 OT protocol plus $O(N)$ evaluations of a pseudo-random function. (Note that the construction of [7] of 1-out-2 string OT from 1-out-2 bit OT is efficient.)

Another measure of complexity is communication complexity. In conjunction with recent work on private information retrieval (abbrev. PIR) we can obtain a protocol for 1-out-N OT with $O(N^{\epsilon}m)$ communication overhead under the Quadratic Residuosity Assumption, based on the PIR construction of [30] (m is the security parameter, i.e. the length of the modulos), or an $O(m)$ protocol based on the PIR protocol of [10]. (see Section 4.1.1).

1.2 Oblivious Polynomial Evaluation

In the Oblivious Polynomial Evaluation problem the sender knows a polynomial P of degree k over some field $\mathcal{F}$. The receiver can choose to get the value $P(x)$ for any element $x \in \mathcal{F}$ without learning anything else about the polynomial P and without the sender getting any information about x (for the precise definition of learning and information see Section 1.3). Oblivious polynomial evaluation can be performed using general protocols for oblivious function evaluation. However, these protocols operate on the *binary* circuit which computes the function and therefore depend on the *size* of the field $\mathcal{F}$ over which the polynomial is defined. In contrast, the complexity of the protocol we present in Section 3 does not depend on the size of the underlying field, except that it uses 1-out-of-2 OT protocols over that field. The issue of finding protocols for oblivious polynomial evaluation that do not depend on the field size might be viewed as being the equivalent of searching for strongly polynomial algorithms in combinatorial optimization (e.g. for linear programming). Moreover, our protocol can be readily applied to obliviously compute $g^{P(x)}$, where P is a polynomial, with no increase in the protocol's complexity. The binary circuits that evaluate this function should compute exponentiations and are therefore very large. Oblivious evaluation of $g^{P(x)}$ yields k-wise independent outputs which might be useful for distributed implementations of some Public Key operations.

The protocol we describe is based on an intractability assumption relate to the *noisy polynomial reconstruction problem*. While very related to other computational problems it has not been used as such in a cryptographic protocol. Section 3.1 describes in detail the assumption and the background.

We envision two types of applications for Oblivious Polynomial Evaluation. One is whenever k-wise independence can replace full independence or pseudo-randomness (i.e. oblivious evaluation of a pseudo-random function, as in [39]). Such an example is given is Section 4.2.2. Another is for preserving anonymity where the value of P at a given point acts as a token which Alice receives from Bob. An Application of this nature is described in Section 4.2.3 for a protocol that

allows reliable metering.

1.3 Correctness and Security Definitions

We now get to the delicate business of defining the security of both variants of OT we deal with in this paper: 1-out-of-N OT and Oblivious Polynomial Evaluation. There is no real difference between 1-out-of-2 OT and 1-out-of-N and we will treat the former as 1-out-of-N OT with $N = 2$. Providing a definition of security for OT is an involved business, since one must take into account all possible behaviors of the sender and the receiver.

The definition of correctness is simple: the sender's input is $X_1, X_2, \ldots X_N$ the receiver's input is $1 \leq I \leq N$. At the end of a successful execution where all parties follow the protocol the receiver should obtain X_I and be able to output it.

Oblivious Transfer is a two party protocol and as such its definition of security can be derived from the security definition of such protocols. However, there are several obstacles: (i) coming up with the precise definition of general multi-party protocols is a messy business and there is no consensus yet on the definition (see [1, 11, 22, 34], 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 of our paper are rather robust and should work with several definitions. However, the existing definitions are of course a good guideline and we follow most closely Goldreich's [22]. We kept the formalities at a bare minimum and ignored such important issues as uniformity.

We separate the definition of security into protecting the sender and protecting the receiver.

The Receiver's Security - Indistinguishability: Given that under normal operation the sender gets no output from the protocol the definition of the receiver's security in a 1-out-N OT protocol is rather simple: for any $1 \leq I, I' \leq N$ and for any probabilistic polynomial time $\mathcal{B}'$ executing the sender's part, the views that $\mathcal{B}'$ sees in case the receiver tries to obtain X_I and in case the receiver tries to obtain $X_{I'}$ are computationally indistinguishable given $X_1, X_2, \ldots X_N$.

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

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

Security of Oblivious Polynomial Evaluation: As in the case 1-out-of-N OT, the sender gets no output, which simplifies the definition. The receiver's security is that for any x and x' in $\mathcal{F}$ 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 $P(x)$ and in case the receiver tries to obtain $P(x')$ are computationally indistinguishable given P[2].

As for the sender's security, we compare to the *ideal implementation*, using a trusted third party Charlie that gets the sender's polynomial P and the receiver's request x and gives the receiver $P(x)$. 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 implementation such that the outputs of $\mathcal{A}'$ and $\mathcal{A}''$ are computationally indistinguishable.

Note that this definition does *not* preclude the sender from cheating by using a polynomial of degree higher than k, (it might not be possible to extract from the sender a degree k polynomial.) Our construction enables such cheating; however, in many applications (including the ones described in this paper) this is immaterial.

2 1-out-of-N Oblivious Transfer Protocol

In this section we describe rather efficient constructions of a 1-out-of-N Oblivious Transfer protocol, and a k-out-of-N OT protocol. Section 4 describes two applications for these new protocols, and in particular a transformation of any Private Information Retrieval (PIR) protocol to a Symmetric PIR (SPIR) protocol, without using additional databases. A more involved application is that of oblivious polynomial evaluation which is described in Section 3.

In order to design the 1-out-of-N OT protocol we need another cryptographic primitive in addition to the 1-out-of-2 OT – a pseudo-random function. Let $\{F_K : \{0,1\}^m \mapsto \{0,1\}^m \mid K \in \{0,1\}^t\}$ be a family of pseudo-random functions. The 1-out-of-2 OT will be performed on strings of length t, since the transmitted strings will be used as keys of F.

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

Protocol 2.1 *B's 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 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 $1 \leq j \leq \ell$ and $b \in \{0,1\}$ each K_j^b is a t-bit key to the pseudo-random function F_K.

For all $1 \leq I \leq N$ let $\langle i_1, i_2, \ldots i_\ell \rangle$ be the bits of I. B prepares $Y_I = X_I \oplus \bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$.

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

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

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

[2] The only difference between this case and the OT case is that the quantification for all x and x' may be different in the uniform case.

It is clear that the receiver can get any value it wishes in the above protocol. The complexity of the protocol is $N \log N$ evaluations of F_K plus $\log N$ invocations of the 1-out-of-2 OT protocol.

Security analysis: It has to be argued that both the sender's security and the receiver's security are satisfied. Given that the 1-out-of-2 OT 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 B sees when A is retrieving X_{I_1} or X_{I_2} are computational indistinguishable. As for the security of the sender, one property of the protocol that simplifies the proof is that the Sender gives $Y_1, Y_2, \ldots Y_N$ only *after* the 1-out-2 OT protocols are executed (this enables to extract from the simulator of the receiver the index I of the item it has learned, even if the 1-out-of-2 OT protocols do not enjoy this extraction property). This allows us to make all of them or part of them random and examine $\mathcal{A}'$ reaction. Therefore if the claim is that $\mathcal{A}'$ can learn more than in the ideal implementation we can test and see on which I's $\mathcal{A}'$ responds when the real X_I is given and when a random one is substituted. If there is more than one such I then it means that $\mathcal{A}'$ has broken either the 1-out-of-2 OT protocol or the pseudo-random function. This method allows us to essentially extract the inputs A has requested. We therefore get the following lemma:

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

Yuval Ishai (personal communication) has suggested an improvement to the above protocol - each Y_I is masked by $\bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(pref_j(I))$ where $pref_j(I)$ denotes the first j bits of I. The advantage of this proposal is that the total number of evaluations of the pseudo-random function is linear N. We will see a different method for lowering the complexity in Protocol 2.2 in Section 2.1 as part of the k-out-N protocol.

2.1 k-out-of-N Oblivious Transfer

For some applications we actually need a k-out-of-N Oblivious Transfer protocol, i.e. that the receiver should be able to choose *any* k out of the N strings. It is possible to use Protocol 2.1 by repeating it k times independently[3]. Since N might be very large it is very interesting to investigate whether it is possible to keep the price low in terms of pseudo-random function evaluations while the number of 1-out-of-2 OTs will be proportional to k. Following we describe a k-out-of-N scheme which achieves this property.

The scheme as described works for $k \ll N$. Consider first the following protocol for 1-out-of-N OT which is based on a recursive call to Protocol 2.1.

Protocol 2.2 *B's input is* $X_1, X_2, \ldots X_N$

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* $C_1, C_2, \ldots C_{\sqrt{N}}$ *by invoking Protocol 2.1 twice. If A would like to learn* $X_{i,j}$ *she should pick* R_i *and* C_j.

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

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

It is clear that the receiver can get any value it wishes in the above protocol. The complexity of the protocol is $2N$ evaluations of F_K plus two invocations of the 1-out-of-$\sqrt{N}$ protocol which involve $\sqrt{N} \log N$ evaluations of the function F_K and $\log N$ calls to the 1-out-of-2 OT. This protocol can be described as being two-dimensional whereas Protocol 2.1 was $\log n$-dimensional.

We wish to use this protocol to perform a k-out-of-N OT protocol. Suppose that in step 2 we let A obtain k of the keys $R_1, R_2, \ldots R_{\sqrt{N}}$ and k of the keys $C_1, C_2, \ldots C_{\sqrt{N}}$, by repeating the 1-out-$\sqrt{N}$ protocol k times *independently*. Then A would be able to obtain any k values she wishes. However, she gets more information than that: if she is interested in $X_{i,j}$ and $X_{i',j'}$ then she can actually learn also $X_{i,j'}$ and $X_{i',j}$. The total number of values she receives is k^2 but all other values remain hidden. Furthermore, after the execution of the protocol these k^2 values are well defined. The idea is to repeat the protocol a second time 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 A has affectively committed to the a set $S \subset \{1, \ldots N\}$ of k^2 values. A good permutation is one where no two elements of S are mapped to the same column or to the same row. The probability that there are two different values in S which are mapped to the same row is at most $\frac{k^4}{\sqrt{N}}$. If this 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.

The protocol we get is therefore as follows:

Protocol 2.3 *The input to B is* $X_1, X_2, \ldots X_N$ *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* $1 \ldots N$. *For any I we let* $\sigma_{j,R}(I)$ *be the first* $\ell/2$ *bits of* $\sigma_j(I)$ *and* $\sigma_{j,C}(I)$ *be the second* $\ell/2$ *bits of* $\sigma_j(I)$. *Bob arranges the N inputs in a* $\sqrt{N} \times \sqrt{N}$ *matrix, i.e. input* X_I *is indexed now as* $X_{\sigma_{j,R}(I),\sigma_{j,C}(I)}$

  3. *B sends* σ_j *to A.*

  4. *A and B engage in a k-out-of-*$\sqrt{N}$ *OT protocol on keys* $R_1^j, R_2^j, \ldots R_{\sqrt{N}}^j$ *and on keys* $C_1^j, C_2^j, \ldots C_{\sqrt{N}}^j$ *by invoking Protocol 2.1 k times. For all* $1 \leq I \leq k$ *A picks* $R^j_{\sigma_{j,R}(I)}$ *and* $C^j_{\sigma_{j,C}(I)}$.

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

- *Bob computes* $Y_I = X_I \bigoplus_{j=1}^{w}(F_{R^j_{\sigma_{j,R}(I)}}(I) \oplus F_{C^j_{\sigma_{j,C}(I)}}(I))$, *and sends them to A.*
- *To reconstruct the desired inputs A computes for each* $I_1, I_2, \ldots I_k$ *the value*

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

As for the complexity of the protocol, it requires $2wN$ invocations of the pseudo-random function as well as $2wk \log N$ executions of the 1-out-of-2 OT protocol. In order for the probability of A learning no more than k values be smaller than δ, we need w to be such that $(\frac{k^4}{\sqrt{N}})^w < \delta$, i.e $w > \log\delta/\log(k^4/\sqrt{N})$. We therefore get that when $k \leq N^{1/8}$ this process works (this analysis only requires that the mapping of the data elements is pair-wise independent, and therefore the representation of the mapping can be short. A different analysis shows that the protocol can be used for up to $k \leq N^{1/4}$ and a similar protocol can work up to $k \leq N^{1/3}$). Note that the description of the permutations σ_j can be short, since it is enough that it be pair-wise independent.

Protocol 2.3 enables A to *simultaneously* get k out of the N values. However it is not secure to use this protocol to perform k adaptive transfers of a single value to A, e.g. when A should learn the first value, use it to decide what is the second value she wants to get and learn it, and so on. Such operations might be useful for performing oblivious searches in data structures. This work was recently extended [36] to construct a protocol for k adaptive oblivious transfers based on a new cryptographic primitive – *sum consistent synthesizers.*

3 Protocols for Oblivious Polynomial Evaluation

In this section we describe a protocol for oblivious evaluation of a polynomial, and the intractability assumption on which it is based. Section 4 contains applications for this oblivious evaluation protocol.

3.1 Intractability Assumption

The intractability assumption that we use is closely related to the noisy polynomial reconstruction problem, or the list decoding problem of Reed-Solomon codes. Informally our intractability assumption is as follows: Let P be a k degree polynomial over a field F. Then given $n > k$ sets, each containing $m-1$ random elements and one element which is the value of P at a known point different from 0, the value $P(0)$ is pseudorandom.

3.1.1 The noisy polynomial reconstruction problem

The noisy polynomial reconstruction problem is the following:

Definition 3.1 (Polynomial reconstruction)
INPUT: *Integers* k *and* t, *and* n *points* $\{(x_i, y_i)\}_{i=1}^n$, *where* $x_i, y_i \in \mathcal{F}$.
OUTPUT: *Any univariate polynomial* P *of degree at most* k *such that* $P(x_i) = y_i$ *for at least* t *values* $i \in [1, n]$.

The noisy polynomial reconstruction problem is closely connected to the list decoding problem, which is motivated by coding theory and was first defined by Elias [17] (and sometimes termed as the bounded-distance decoding problem). The input to this problem is a received word, and the output is a list of all code words which are within some distance from the received word. For the case of Reed-Solomon codes the list decoding problem can be formulated as follows:

Definition 3.2 (Polynomial list reconstruction)
INPUT: *Integers* k *and* t, *and* n *points* $\{(x_i, y_i)\}_{i=1}^n$, *where* $x_i, y_i \in \mathcal{F}$.
OUTPUT: *All univariate polynomials* P *of degree at most* k *such that* $P(x_i) = y_i$ *for at least* t *values* $i \in [1, n]$.

For given values of k and n, and in particular for a given message rate k/n, it is preferable to obtain solutions for minimal values of t. The classical algorithm of Berlekamp and Massey (see e.g. [33]) solves the polynomial list reconstruction problem in polynomial time for $t \geq \frac{n+k}{2}$. Sudan [41] presented a polynomial algorithm which works if $t \geq \sqrt{2kn}$, and very recently Guruswami and Sudan [26] have presented a polynomial algorithm that solves the problem for $t \geq \sqrt{kn}$, and thus improves upon the Berlekamp-Massey algorithm for every value of the message rate k/n.

The polynomial list reconstruction problem is defined as a worst case problem regarding the number of polynomials that might appear in the solution list, i.e. all polynomials within a given distance should be listed. Goldreich et al [24] have shown that for $t > \sqrt{kn}$ the number of possible polynomials in the solution to the problem is bounded by a polynomial in n. We are more interested in the problem of finding just a single polynomial which fits t or more of the given n points, since our constructions use random instances for which it holds with high probability that the corresponding instances of the noisy polynomial reconstruction problem have just a single solution.

We note that given an input to the noisy polynomial reconstruction problem it is possible to randomize it to obtain an input which corresponds to a random polynomial. Specifically, for parameters k and t and given n points $\{(x_i, y_i)\}_{i=1}^n$ the randomization is achieved by choosing a random polynomial R of degree k and constructing a new instance of the problem with input $\{(x_i, y_i + R(x_i))\}_{i=1}^n$. While this is by no means a reduction from the worst case problem to the average case (since we only randomize the polynomial but not the noise), it might hint that solving the problem in the average case might not be much simpler than solving it in the worst case.

3.1.2 The intractability assumption – pseudorandomness

The intractability assumption depends on the following parameters: $\mathcal{F}$, the field over which operations are performed; k, the degree of the hidden polynomial; n, the number of sets; and m, the number of points in a set.

We use the following notation:

- Let P be a polynomial over $\mathcal{F}$, and let $A_x^{P,m}$ be a random set with m distinct elements of F subject to the constraint that one of the elements equals $P(x)$.
- Define $A^{P,m}_{x_1,\ldots,x_n}$ as the sequence $\langle x_1, A_{x_1}^{P,m}\rangle, \ldots, \langle x_n, A_{x_n}^{P,m}\rangle$, and let $A_n^{P,m}$ be the family of all such sequences where the x_i-s are distinct and different from 0.
- Let $A_n^{k,\alpha,m}$ be the set of all $A_n^{P,m}$ for which P is of degree at most k and $P(0) = \alpha$.

- Let $A_n^{R,m}$ be the family of all sets of $n \times (m+1)$ elements of $\mathcal{F}$, $\{x_i, z_{i,1}, \ldots, z_{i,m}\}_{i=1}^n$, such that for all i the m elements $\{z_{i,1}, \ldots, z_{i,m}\}$ are distinct.

The intractability assumption is that given a random element of $\{A_n^{k,\alpha,m}\}_{\alpha \in \mathcal{F}}$ the value $P(0)$ is pseudorandom. It depends on the parameters $\mathcal{F}, k, m$, and n. More accurately,

Definition 3.3 (Pseudorandomness assumption) *For every algorithm D with running time polynomial in k, m, n, and in $|\mathcal{F}|$, and for every $\alpha \in \mathcal{F}$, the difference between the probability that D outputs 1 when its output is randomly chosen in $A_n^{k,\alpha,m}$, and the probability that D output 1 when its output is randomly chosen in $A_n^{R,m}$, is smaller than any polynomial in k, m, n, and $|\mathcal{F}|$.*

There is an easy reduction from the problem of breaking the pseudorandomness assumption to the noisy polynomial reconstruction problem with parameters k, n and $t = n/m$. The algorithm D should choose an element at random from every set $A_{x_i}^{P,m}$, and with high probability it obtains an instance of the polynomial list reconstruction problem with $t \geq n/m$ correct values of the polynomial. In particular, setting the parameter m in the pseudorandomness assumption to be smaller than $\frac{n}{\sqrt{nk}} = \sqrt{n/k}$ yields problem instances which are easily solvable by the polynomial list reconstruction algorithm of [26]. It is an interesting open problem to provide a reduction from the polynomial reconstruction problem to the pseudorandomness assumption, suggesting that solving the latter problem for values of m substantially larger than $\sqrt{n/k}$ is hard.

Conjecture 3.1 *If the noisy polynomial reconstruction problem is hard then the pseudorandomness assumption holds.*

3.2 Oblivious Polynomial Evaluation

In this section we describe a protocol for oblivious evaluation of a polynomial. The protocol involves a receiver A and a sender B and has the following specifications:

- B's input: A polynomial $P(x)$ of degree d_P in the field $\mathcal{F}$.
- A's input: A value $\alpha \in \mathcal{F}$.
  A's output: $P(\alpha)$.
- Protocol security parameters: $m, d_{Q,x}$ (s.t. $d_{Q,x}$ is an integer multiple of d_P), which would be discussed below.

At the end of the protocol the parties should not learn anything but their specified outputs. The protocol is based on the sender hiding P in an otherwise random bivariate polynomial and then running 1-out-of-m OT protocols (where m is rather small) with the receiver to reveal to her just enough information to enable the computation of $P(\alpha)$. The protocol is as follows:

B hides P in a bivariate polynomial: (see Figure 1) B chooses a random bivariate polynomial $Q(x,y)$ in $\mathcal{F}$ subject to the following three constraints: (i) The degree of y in Q is the same as the degree of P, that is $d_{Q,y} = d_P$. (ii) The degree of x in Q equals a parameter $d_{Q,x}$. (iii) $Q(0,\cdot) = P(\cdot)$, i.e. $\forall y\ Q(0,y) = P(y)$.

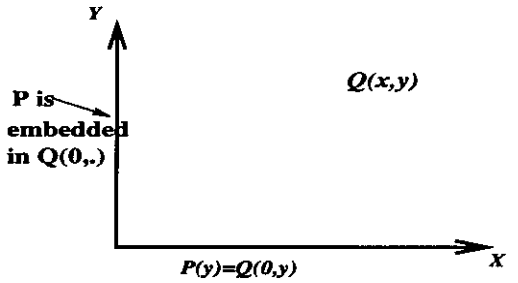

Figure 1: The polynomial P embedded in the bivariate polynomial Q.

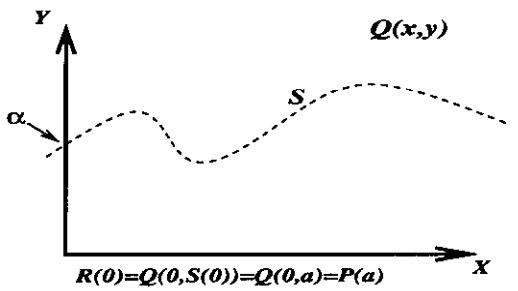

Figure 2: The polynomial S defines a polynomial R s.t. $R(0) = P(\alpha)$.

A hides α in a univariate polynomial: (see Figure 2) A chooses a random polynomial S of degree d_S, such that $S(0) = \alpha$. The degree d_S is defined as $d_S = d_{Q,x}/d_P$.

A's plan is to use the univariate polynomial $R(x) = Q(x, S(x))$ in order to learn $P(\alpha)$: it holds that $R(0) = Q(0, S(0)) = P(S(0)) = P(\alpha)$ and then if A is able to interpolate R she can learn $R(0) = P(\alpha)$. The degree of R is $d_R = d_{Q,x} + d_P d_S = 2d_{Q,x}$.

A learns points of R: (see Figure 3) Once A learns $d_R + 1$ values of the form $\langle x_i, R(x_i)\rangle$ she can compute $R(0)$. Therefore she does the following:

- A chooses $n = d_R + 1$ different points $x_1, \ldots, x_n$, all different from 0.
- For each $1 \leq j \leq n$ the following is performed:
  - A chooses a masking set of $m-1$ random field elements $y_{j,1}, \ldots, y_{j,m-1}$. She sends to B the value x_j and the list $\langle y_{j,1}, \ldots, y_{j,m-1}, S(x_j)\rangle$, permuted randomly. Denote the permuted list elements by $\langle z_{j,1}, \ldots, z_{j,m}\rangle$.
  - B and A perform a 1-out-of-m OT protocol with the m values $\langle Q(x_j, z_{j,1}), \ldots, Q(x_j, z_{j,m})\rangle$. A should choose to learn the value $Q(x_j, S(x_j)) = R(x_j)$.
- After learning the n values $\{R(x_j)\}_{j=1}^n$ the receiver A can interpolate $R(\cdot)$ and compute $R(0) = P(\alpha)$.

It is straightforward to verify that the above protocol enables A to obtain any value $P(\alpha)$ she desires. The complexity of the protocol is $n = d_R + 1 = 2d_{Q,x} + 1$ executions of the 1-out-of-m OT protocol.

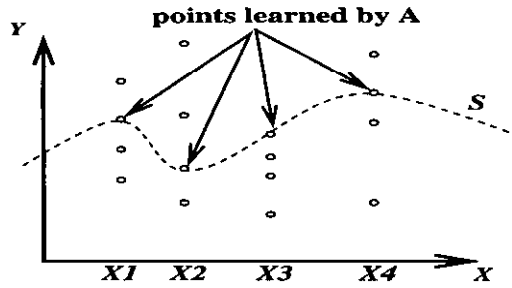

Figure 3: For every x_i the receiver uses OT to learn a value of S hidden between m values.

As for the security of the protocol it has to be shown that both A's and B's privacy are satisfied. These properties can be reduced to the security of the 1-out-of-m OT protocol and the pseudorandomness assumption (which is only needed to show A's privacy). Following we discuss the proofs of security based on a perfectly secure 1-out-of-m OT protocol.

Sketch of B's privacy: B hides P in a random bivariate polynomial Q, and A obtains a systems of $2d_{Q,x}+1$ equations of the coefficients of Q, using different x values. The following theorem (proved in the full version) states that A can learn only a single linear equation of the coefficients of P (and this in particular implies that A can learn only a single value of the polynomial P).

Theorem 3.1 *Let $Q(x,y)$ be a bivariate polynomial in which x is of degree k and y of degree ℓ. Denote by $P(y) = \sum_{j=0}^{\ell} a_{0,j}y^j = P(0,y)$ the polynomial which is equal to Q constrained to the line $x=0$ (i.e. to the y axis). Then given any $2k+1$ values $Q(x_i,y_i)$ where all the x_i-s are distinct and different from 0,*

- *It is impossible to obtain more than a single linear relation between the coefficients $\{a_{0,i} | 0 \le i \le \ell\}$.*
- *It is possible to construct a polynomial $Q^*(x,y)$ of the same degrees as Q which is random except that it satisfies the equations defined for Q by the $2k+1$ values.*

Sketch of A's privacy: A is trying to hide a polynomial S of degree $k = d_S = d_{Q,x}/d_P$. It sends to B $n = 2d_{Q,x}+1$ points of S, each hidden among $m-1$ additional random values. The pseudorandomness assumption assures that the value of $S(0)$ is pseudorandom if the right security parameters are used. In particular, A should use $m \gg \frac{n}{\sqrt{nk}} = \sqrt{n/k} = \sqrt{2d_P}$.

A note on parallel executions: Note that the protocol has a rather handy property, a *single* invocation of the protocol can be used to let A obliviously compute the values of *several* polynomials at the same point x. The parties can therefore run with no additional overhead (in terms of the number of OT's) a protocol in which B's input is n polynomials $\langle P_1,\ldots,P_n\rangle$, A's input is x, and A's output is $\langle P_1(x),\ldots,P_n(x)\rangle$. To achieve this B should define appropriate polynomials $\langle Q_1,\ldots,Q_n\rangle$ and in every step in which B transfers a single value $Q(i,j)$ in the original protocol, he should transfer $\langle Q_1(i,j),\ldots,Q_n(i,j)\rangle$ in the multi-polynomial protocol. Note that the protocol has the additional property that it is ensured that A computes the values of all the polynomials at the *same point* x. We use this protocol in Section 4.2.2 to construct an efficient protocol for the intersection problem.

4 Applications

4.1 Applications for 1-out-of-N OT

In this section we describe two direct applications to the 1-out-of-N OT protocol.

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 recently. A system that allows a user to access a database consisting of N words $W_1, W_2, \ldots W_N$ and read any word it wishes without the owner learning which word was accessed is called PIR (for Private Information Retrieval) [12]. There are various proposals for implementing such schemes, where the emphasis is on the communication complexity. Some of the proposals require that the user communicate with several servers maintained by the database owner where these servers are certified somehow not to communicate with each other. This assumption (of non-communication replicated data-bases) was shown to be unnecessary by Kushilevitz and Ostrovsky [30] where a PIR scheme with a single server was proposed - the user's security is under the Quadratic Residuocity assumption modulo a Blum integer and the communication complexity is $O(N^\epsilon m)$ (where m is the security parameter, i.e. the length of a number that is hard to factor). Another proposal of such a scheme is by Cachin et al [10] where the communication complexity is just m.

More recently attention was given to the question of protecting the database as well, i.e. that the user will not learn more than one word of data (or as many words as he paid for). A PIR scheme that enjoys this property is called SPIR. In [20] 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). We now show that adding servers is not necessary and that using our protocols it is possible to transform any PIR scheme into a SPIR scheme with relatively little work on behalf of the parties involved.

It is not hard to see the connection between the PIR/SPIR setting and the 1-out-of-N OT setting. One can regard a SPIR construction as 1-out-of-N OT protocol construction with the emphasis on the communication complexity. The important feature of Protocol 2.1 for 1-out-of-N OT is that for A to obtain the value of the desired X_I she does not need all of $Y_1, Y_2, \ldots Y_N$ but only Y_I. Therefore if instead of step 3 in Protocol 2.1 Alice and Bob perform a PIR reading of $Y_1, Y_2, \ldots Y_N$, then Alice can get sufficient information without giving Bob 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.

4.1.2 Oblivious sampling

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

Consider the following scenario: Cold Fusion (Trademark) is a search engine claiming to be the largest on the west coast. Alice would like to check this claim and measure the number of URLs *indexed* by Cold Fusion. She might also like to check the overlap between the pages indexed by Cold Fusion and by other search engines, and this task also requires a random sample of the pages indexed by Cold Fusion (see [8]). One possibility is for Cold Fusion to give Alice the list of the URLs it has indexed; Alice then will make sure that all (or most of) these URLs are indeed indexed actively, i.e. that the corresponding page can be retrieved in a search (it is much easier to gather many URLs without indexing the content). This can be done by sampling a few of them, retrieving the corresponding page and searching for the page via the public interface of Cold Fusion. The problem is of course that the list of URLs is a trade secret and Cold Fusion is not going to 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 in such a way that

- Will keep most of the list (the part not sampled) secret from Alice.
- Will not allow Cold Fusion to learn which URLs were selected - otherwise it can quickly add them to the active index.

Our 1-out-of-N OT (and in particular k-out-of-N OT protocols) can be used for *oblivious sampling*, i.e. to let Alice sample a random element from a large set of elements known to Bob, without giving Bob any information about the item that Alice chooses. In the case of the search engines application this enables Alice and Cold Fusion to solve their problem at a small cost, even if Cold Fusion's databases consists of hundreds of millions of pages. If Cold Fusion claims to have indexed N URLs it should feed them as the input $X_1, \ldots X_N$ to the receiver in the 1-out-of-N OT protocol. The only problem remaining is the duplication problem – what if Cold Fusion duplicates some of the URLs in order to claim a larger set. This can be solved using a hash tree structure, but there is also a very simple solution using the following procedure – for some of the sampled URLs Alice does the following: she sends the URL to Cold Fusion and asks it what was the index I that Alice chose when she sampled the URL. If Cold Fusion does not answer correctly – then Alice can conclude that it was cheating by duplicating URLs.

4.2 Applications for Oblivious Evaluation of Polynomials

There are two main types of applications in which oblivious polynomial evaluation is useful: First, when it is required to enable the receiver to obliviously obtain a value from a k-wise independent space. For many applications such values are as good as completely random values. The second application is when it is desired to preserve anonymity in cryptographic protocols which require a user to get a value of a polynomial held by the sender without revealing the choice to the sender. We give examples of both types of applications below.

4.2.1 Comparing information without leaking it

Imagine two parties, A and B, who each think about a name (e.g. of a "suspect") and would like to check whether they both think about the same person. This problem was thoroughly discussed by Fagin, Naor, and Winkler [19]. Oblivious evaluation of linear polynomials can be used to construct a very simple solution to this problem.

Assume that A thinks about α and B about β. A should generate a random linear polynomial $P_A(\cdot)$, and B should generate a random linear polynomial $P_B(\cdot)$. They run the oblivious polynomial evaluation twice, switching roles. In the first invocation A learns a value of B's polynomial (she should choose to learn $P_B(\alpha)$), and in the second invocation B learns a value of A's polynomial (he should choose to learn $P_A(\beta)$). They then compute the two values, $P_A(\alpha) + P_B(\alpha)$ (computed by A), and $P_A(\beta) + P_B(\beta)$ (computed by B). If $\alpha = \beta$ then the two values are the same, otherwise they are different with high probability.

Application to passwords: The above protocol can be used for a mutually authenticated key exchange based on (possibly weak) passwords. Consider a user who wishes to login to a remote server over an insecure network. She does not want to send her password in the clear, and is not even sure that the remote party is the required server. An additional problem is that the password might not have enough entropy and might therefore be susceptible to dictionary attacks. Assume that there is no Public Key Infrastructure, and that the user does not carry with her a public key of the remote server. There have been several solutions to this problem (see e.g. [3, 32]), but it seems that they cannot be proven to be secure without postulating the existence of a Random Oracle.

The most natural formulation of the scenario is as a "comparing information without leaking it" problem. If the right user contacts the right server they both should be thinking of the same password, and they can verify whether this is the case. The above protocol can thus be used for password authentication, and it can furthermore be used to generate a session key for the two parties, whose entropy does not depend on the entropy of the passwords. Each of the parties, the user and the remote server, should choose a random linear polynomial and (obliviously) compute the sum of the two polynomials at x =password. They use the first half of the output for authentication and the second half as a session key. If this protocol is used then the probability of cheating is bounded by the probability of the adversary guessing the password.

4.2.2 The list intersection problem

A generalization of the above problem is the list intersection problem, in which A and B each have a list of names and would like to know the intersection of these lists. This might be required in order to preserve the privacy of A and B, and in some cases the privacy of the parties listed in the lists. For instance, in some countries strict privacy laws prevent government agencies from disclosing their databases to other agencies, but it is still required sometimes to check the intersection of two databases, for example the intersection of the database of people receiving unemployment payments and the database of those who recently bought expensive houses.

The list intersection problem can be formulated as follows: A's input is a list of n elements $A_L = \{a_1, \ldots, a_n\}$,

and B's input is a list $B_L = \{b_1, \ldots, b_n\}$. The required output should be $\{x | x \in A_L \cap B_L\}$, and should become known to both parties. (We do not attempt to handle the *early stopping* problem in this scenario).

The list intersection problem can be solved using a simple protocol which uses polynomials of degree n. A and B should each choose a random polynomial (P_A and P_B) of degree n. A should obliviously obtain $\{P_B(a_i)\}_{i=1}^n$ and compute $\{P_A(a_i) + P_B(a_i)\}_{i=1}^n$. B should compute $\{P_A(b_i) + P_B(b_i)\}_{i=1}^n$ in a similar way. A pair $a_i = b_j$ results in equality between two items in the two lists (a property which otherwise happens with low probability). The protocol requires each party to perform n oblivious evaluations of a polynomial of degree n (it is possible to adapt the protocol to use polynomials of lower degree, by randomly hashing the elements into bins and using a different polynomial for every bin, but such protocols leak some information). A different protocol which uses oblivious evaluations of linear polynomials can be obtained from the solution to the following problem.

One-to-many intersection: A special case of the list intersection problem is where A's list is just a single element. This problem is interesting in its own right (imagine for example that B holds a list of people with a bad record and A wants to check whether a certain person appears in the list). The problem can be solved using a protocol which requires A to obliviously compute n linear polynomials at the same point x. Luckily, the oblivious polynomial evaluation protocol of Section 3.2 can achieve this task using the same number of oblivious transfers required for evaluating a single polynomial. Therefore the overhead of the oblivious evaluations in the protocol for one-to-many intersection is just that of a single oblivious evaluation of a linear polynomial. We describe below how the one-to-many protocol can be used to compute the intersection between two lists of size n with complexity equal to that of n oblivious evaluations of a linear polynomial.

Following is the protocol for computing one-to-many-intersection. A's input is an element x, B's input is a list $b_1, \ldots, b_n$.

- B chooses n random linear polynomials $\langle P_1, \ldots, P_n \rangle$.
- For every $1 \leq i \leq n$, A obliviously computes $P_i(x)$.
- B publishes $(P_1(b_1), P_2(b_2), \ldots, P_n(b_n))$, permuted randomly.
- A checks whether one of the values she computed appears in the list that B published.

If there exists an i for which $x = b_i$ then A finds an equality, otherwise the probability that an equality occurs is low. The computation and communication overheads are linear in n, and the protocol for obliviously computing the polynomials has the overhead of obliviously computing just a single linear polynomial.

Using the one-to-many protocol for computing list intersection: If A has n elements the above protocol can be run n times to let A check whether each of her elements is in B's list. However if B should also learn the intersection the protocol is a little more complex, since if A just tells B which values she computed, B (who knows the polynomials) can check whether different elements (which are not is his list) are in A's list.

The solution is that B defines n^2 linear polynomials $\{P_{i,j}^B \mid 1 \leq i,j \leq n\}$, and A obliviously computes n^2 values $\{P_{i,j}^B(a_i) \mid 1 \leq i,j \leq n\}$. A defines n^2 linear polynomials $\{P_{i,j}^A \mid 1 \leq i,j \leq n\}$ and B obliviously computes $\{P_{i,j}^A(b_j) \mid 1 \leq i,j \leq n\}$ (note that A must simultaneously compute the values of a row of B's polynomials at a single point, while B must use A's polynomials column by column). If $a_i = b_j$ then $P_{i,j}^B(a_i) + P_{i,j}^A(a_i) = P_{i,j}^B(b_j) + P_{i,j}^A(b_j)$. The two parties should therefore each publish a matrix of n^2 values and look for equalities between the matrices' entries. The protocol requires $O(n^2)$ communication, but the complexity of the oblivious computations of each party is just that of n evaluations of linear polynomials.

4.2.3 Anonymous initialization for metering

A scheme for efficient and secure metering of clients' visits to servers was suggested in [35]. This scheme uses a trusted audit authority A. Very crude description of the scheme can be is as follows:

- The audit authority A generates a random bivariate polynomial $P(x,y)$, where the degree of x is $k-1$.
- Each client C receives from A an initialization data $P(u,\cdot)$.
- When C visits a server S it sends it the value $P(u,S)$.
- After k clients visit S it can interpolate the value $P(0,S)$ which serves as a proof for being visited by k clients.

A modification of this scheme described in [35] preserves the anonymity of clients towards servers. However, if the audit agency A cooperated with a server S they were able to check which clients visit A. If that scheme is used together with oblivious polynomial evaluation, then A would not able to know which initialization values where taken by each client and then even a coalition of A and a server S would not be able to connect different visits by the same client.

Acknowledgments

We would like to thank Sanjeev Arora, Andrei Broder, Oded Goldreich, Yuval Ishai, Amit Klein, Ronitt Rubinfeld, and Madhu Sudan.

References

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

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

[3] S.M. Bellovin and M. Merritt, *Encrypted key exchange: Password-based protocols secure against dictionary attacks*, Proc. of the 1992 IEEE Computer Society Conference on Research in Security and Privacy, pp. 72–84, 1992.

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

[5] G. Brassard, C. Crépeau and J.-M. Robert *Information Theoretic Reduction Among Disclosure Problems*, 27th FOCS, pp. 168–173, 1986.

[6] G. Brassard, C. Crépeau and J.-M. Robert, *All-or-Nothing Disclosure of Secrets*, Advances in Cryptology - Crypto '86, LNCS 263, Springer Verlag, pp. 234–238, 1987.

[7] G. Brassard, C. Crépeau and M. Santha, *Oblivious Transfer and Intersecting Codes*, IEEE Trans. on Inform. Theory, Vol. 42(6), pp. 1769–1780, 1996.

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

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

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

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

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

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

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

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

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

[17] P. Elias, *List decoding for noisy channels*, TR 335, Research Laboratory for Electronics, MIT, 1957.

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

[19] R. Fagin, M. Naor and P. Winkler, *Comparing Information Without Leaking It*, Communications of the ACM **39**, pp. 77-85, 1996.

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

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

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

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

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

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

[26] V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and algebraic-geometric codes*, 39th FOCS, 1998.

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

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

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

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

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

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

[33] F. J. MacWilliams, N. Sloane. **The Theory of Error Correcting Codes**, North Holland, Amsterdam, 1977.

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

[35] M. Naor and B. Pinkas, *Secure and Efficient Metering*, Advances in Cryptology – Eurocrypt '98, LNCS 1403, Springer-Verlag, 1998.

[36] M. Naor and B. Pinkas, *Adaptive Oblivious Transfer*, manuscript, 1999.

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

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

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

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

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

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

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