SIAM J. COMPUT.
Vol. 35, No. 5, pp. 1254–1281

OBLIVIOUS POLYNOMIAL EVALUATION[*]

MONI NAOR[†] AND BENNY PINKAS[‡]

Abstract. Oblivious polynomial evaluation is a protocol involving two parties, a sender whose input is a polynomial P, and a receiver whose input is a value α. At the end of the protocol the receiver learns $P(\alpha)$ and the sender learns nothing. We describe efficient constructions for this protocol, which are based on new intractability assumptions that are closely related to noisy polynomial reconstruction. Oblivious polynomial evaluation can be used as a primitive in many applications. We describe several such applications, including protocols for private comparison of data, for mutually authenticated key exchange based on (possibly weak) passwords, and for anonymous coupons.

Key words. cryptography, secure computation, noisy polynomial reconstruction

AMS subject classifications. 94A60, 11T71

DOI. 10.1137/S0097539704383633

1. Introduction. A secure computation protocol for a function $f(\cdot,\cdot)$ allows two parties, a receiver who knows x and a sender who knows y, to jointly compute the value of $f(x, y)$ in a way that does not reveal to each side more information than it can learn 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 as one would desire, since the number of cryptographic operations that should be performed is proportional to the *size of the circuit computing* $f(x, y)$ [60]. Even for relatively simple functions this may be prohibitively expensive. It is therefore interesting to investigate for which functions it is possible to come up with a protocol that does not emulate a circuit computing the function.

1.1. Oblivious polynomial evaluation. In the oblivious polynomial evaluation (OPE) problem the input of the sender is a polynomial P of degree k over some field $\mathcal{F}$. The receiver can learn the value $P(x)$ for any element $x \in \mathcal{F}$ without learning anything else about the polynomial P and without revealing to the sender any information about x (for the precise definition of "learning" and "information" see section 1.2). We find this problem to be a useful primitive. For example, as it can act as a cheap replacement for pseudorandom functions in case only k-wise independence is needed.

Strongly polynomial overhead The overhead of an algorithm is strongly polynomial if it is bounded by a polynomial function of the number of data items in the input,

[*]Received by the editors January 15, 2004; accepted for publication (in revised form) September 28, 2005; published electronically March 17, 2006. This paper is the full version of the sections that describe oblivious polynomial evaluation in "Oblivious Transfer and Polynomial Evaluation," in Proceedings of the 31st ACM Symposium on the Theory of Computing, 1999.

http://www.siam.org/journals/sicomp/35-5/38363.html

[†]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel (moni.naor@weizmann.ac.il). Research supported by grant 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences.

[‡]Department of Computer Science, University of Haifa, Haifa 31905, Israel (benny@cs.haifa.ac.il, benny@pinkas.net). Most of this work was done at the Weizmann Institute of Science and was supported by an Eshkol grant of the Israel Ministry of Science. Part of this work was done at HP Labs, Technion City, Haifa 32000, Israel.

rather than a function of the *size* of the input values (e.g., the number of bits of numerical input values). In combinatorial optimization it is common to look for strongly polynomial algorithms for different problems, for example, for linear programming.

The computational overhead of cryptographic protocols is usually measured as the number of *public key operations*. We use this term to denote cryptographic operations which we do not know how to implement based on the existence of one-way functions, and for which the best known constructions are based on trapdoor functions, or on similar primitives. (In the context of this paper we usually only measure the number of invocations of a 1-out-of-2 oblivious transfer protocol.) Counting only public key operations is justified since the overhead of public key operations depends on the length of their inputs, and is greater by orders of magnitude than the overhead of symmetric key operations (i.e., operations based on one-way functions). Furthermore, the separation result of Impagliazzo and Rudich [38], and subsequent work, hint that it is unlikely that these operations can be based on the existence of one-way functions.

We say that a cryptographic protocol is *strongly polynomial* if the following two properties hold: (1) the number of public key operations performed by the protocol is bounded by a polynomial function of a security parameter and of the number of inputs (but not their size), and (2) the length of the inputs to the public key operations is linear in the security parameter. Note that the number of *symmetric* key operations that the protocol performs can be polynomial in the size of its inputs.

Known methods of implementing oblivious polynomial evaluation include generic protocols for secure two-party computation (e.g., [60]), or using homomorphic encryption (e.g., using Paillier's encryption system [56]). However, none of these methods is strongly polynomial in the sense defined above: the number of oblivious transfer operations in the generic construction, as well as the size of the input to the homomorphic encryption function, is linear in the size of the receiver's input in the OPE protocol.

In contrast, the number of oblivious transfers (OT) used by the OPE protocols presented in this paper does not depend on the size of the underlying field: the length of the items transferred in the oblivious transfer protocols is of size $\log|\mathcal{F}|$, but they require only $O(1)$ public key operations per transfer. Namely, if $\log|\mathcal{F}|$ is longer than the length of the input of the OT protocol, then the items to be transferred in the OT protocol are encrypted using random keys, and the corresponding keys are transferred in the actual OT protocol.

An alternative formulation of the "strongly polynomial" requirement for cryptographic protocols is to allow only "black box" access to field operations, while hiding the underlining field from the parties. In addition, the number of invocations of the black box must be independent of the size of the inputs. Note that our construction satisfies this requirement, whereas generic constructions, or constructions based on homomorphic encryption, do not.

It is interesting to note that the work on randomizing polynomials [36] implies that any function which has an efficient branching program representation can be efficiently reduced to computing polynomials. We explain in section 4.3 how to reduce two-party secure computation of such functions to OPE. In particular, this implies strongly polynomial protocols for computing any function which has an efficient branching program representation.

Our protocols can also be readily applied to oblivious computation of $g^{P(x)}$, where P is a polynomial, with no increase in their overhead, as described in section 4.1. In comparison, binary circuits that evaluate this function must compute exponentiations and are very large, and the overhead of the resulting secure protocols is high.

(Protocols based on homomorphic encryption, however, can compute $g^{P(x)}$ without increasing their overhead.) Oblivious evaluation of $g^{P(x)}$ yields k-wise independent outputs that might be useful for distributed implementations of public key operations.

The noisy polynomial reconstruction problem. The protocols we describe are based on intractability assumptions related to the noisy polynomial reconstruction problem. While being related to other computational problems, this problem has not been used before as an intractability assumption in a cryptographic protocol. Section 2 describes in detail the assumptions and the background. Interestingly, following the publication of the conference version of our work [51] Kiayias and Yung designed different cryptographic primitives which are based on variants of the noisy polynomial reconstruction problem [40, 41].

Overhead independent of the degree *of the polynomial.* We describe a protocol (Protocol 3.4) whose computational overhead (as measured by the number of 1-out-of-2 oblivious transfers that are computed) is independent of the degree of the polynomial. That is, for any degree d, the number of oblivious transfers that are required for an oblivious evaluation of a polynomial of degree d is the same as for a linear polynomial. (Of course, the total computation overhead of the evaluation of the polynomial depends on the degree of the polynomial, but the number of public key operations is independent of the degree.)

Applications. We envision two types of applications for oblivious polynomial evaluation. One is whenever k-wise independence can replace full independence or pseudorandomness (i.e., oblivious evaluation of a pseudorandom function, as in [54]). Such property is required, for example, for the application of constructing anonymous coupons that enable anonymous usage of limited resources (e.g., for constructing an anonymous complaint box). The other type of application uses OPE for preserving anonymity when the receiver must compute the value of a polynomial at a certain point. Applications of this nature include a protocol that allows reliable and privacy preserving metering (described in section 4.4), a method for distributed generation of RSA keys, designed by Gilboa [26], and a protocol for private computation of the ID3 data mining algorithms [45] (in that case the polynomial is used for a Taylor approximation of the logarithm function).

1.2. Correctness and security definitions. We now get to the delicate business of defining the security of oblivious polynomial evaluation. The OPE functionality requires privacy for both receiver and sender. In an OPE protocol neither party learns anything more than is defined by the OPE functionality. The strongest way of formalizing this notion and ensuring simple composability of the protocols is through the definition of secure two-party computation (see, e.g., Goldreich [29] and papers on composability, e.g., [10, 11]). However, this definition is rather complex, while there are many applications that do not require the full power of the general definition and could use "non-ideal" protocols. We therefore prefer to use a relaxed definition for OPE, which ensures privacy for both parties but does not require the sender to commit to its input (i.e., to commit to the polynomial P). We call this definition *private computation.* The definition of private computation is relevant also to the case of malicious parties (and is therefore stronger than a definition for the semi-honest case only). It preserves the privacy of the clients, but does not require simulation of the joint distribution of the view of a malicious sender and the output of an honest receiver, as is required by the general definition of secure computation.

We claim that this relaxation is justified by efficiency considerations, in particular when constructing specific OPE protocols rather than black-box reductions of OPE to other primitives. Furthermore, the definition of private computation is standard for related primitives such as oblivious transfer [58, 21, 53] or private information retrieval (PIR) [13, 43] (note that we present a reduction of OPE to oblivious transfer). Note also that the definition of private computation is equivalent to the definition of secure computation in the case of semi-honest parties. Furthermore, we deal with a receiver-sender (aka. client-server) scenario, where only one party, the receiver, has an output in the protocol. Therefore, the two definitions are equivalent with respect to a malicious *client*, as there is no issue of simulating the joint distribution of the client's view and the server's output.

The requirements of a private OPE protocol can be divided into *correctness, receiver privacy,* and *server privacy.* Let us first define these properties independently and then define a private OPE protocol as a protocol satisfying these three properties. In the definitions, the running time of polynomial time algorithms is polynomial in the size of their inputs, as well as in $\log |\mathcal{F}|$, where $\mathcal{F}$ is the field in which the polynomial P is defined, and in a security parameter k. (Note that the length of representations of elements in $\mathcal{F}$ must be polynomial in the security parameter since otherwise the cryptographic operations might be insecure given adversaries with poly-$\log |\mathcal{F}|$ running time.) We don't require in the definitions themselves that the number of public key operations is independent of $\mathcal{F}$. To simplify the notation we also omit any reference to auxiliary inputs.

We first define the input and output for the OPE functionality as a two party protocol run between a receiver and a sender over a field $\mathcal{F}$.

- Input:
  - Receiver: an input $x \in \mathcal{F}$.
  - Sender: a polynomial P defined over $\mathcal{F}$.
- Output:
  - Receiver: $P(x)$.
  - Sender: nothing.

Definition 1.1 (correctness, or functionality). *At the end of the protocol the receiver obtains the output of the OPE functionality, namely, $P(x)$.*

The definition of the receiver's privacy is simplified by the fact that the sender gets no output. It is as follows.

Definition 1.2 (receiver's privacy—indistinguishability). *For any probabilistic polynomial time $\mathcal{B}'$ executing the sender's part, for any x and x' in $\mathcal{F}$, the views that $\mathcal{B}'$ sees in case the receiver's input is x and in case the receiver's input is x' are computationally indistinguishable.*

The definition of sender's privacy is a bit trickier, since the receiver obtains some information, and we want to say that the receiver does not get more or different information than $P(x)$. We compare the protocol to the *ideal implementation.* In the ideal implementation there is a trusted third party Charlie, which gets the sender's polynomial P and the receiver's request x and gives $P(x)$ to the receiver. The privacy requirement is that the protocol does not leak to the receiver more information than in the ideal implementation.

Definition 1.3 (sender's security—comparison with the ideal model). *For any 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 view of $\mathcal{A}'$ and the output of $\mathcal{A}''$ are computationally indistinguishable.*

Definition 1.4 (private OPE protocol). *A two-party protocol satisfying Definitions* 1.1, 1.2 *and* 1.3.

Note that the definition of receiver privacy does *not* preclude the sender from cheating by using a polynomial of degree higher than the degree of P (and therefore it might not be possible to extract from the sender a degree k polynomial). We do not require that the sender be committed to a single polynomial, or that the receiver could verify that the value she receives corresponds to such a polynomial. Our construction allows such cheating; however, in many applications (including the ones described in this paper) this is immaterial.

As a side note we observe that a possible approach for ensuring correctness could use the verifiable secret sharing (VSS) schemes of Feldman and of Pedersen [23, 57], which let the sender commit to a single polynomial P before engaging in the OPE protocol. Since two polynomials of degree d agree in at most d locations, an OPE invocation in which the sender lets the user evaluate a different polynomial P' of the same degree is revealed with probability $1 - d/|\mathcal{F}|$ by a receiver that evaluates the polynomial at a *random* point. This approach does not work, however, if the sender has some information about the distribution of points in which the user might compute P.

Shared output OPE. A variant of OPE which might be useful in many applications (e.g., [26, 45]) is one where the output of the polynomial is shared between the two parties. As in the basic OPE protocol the input of the sender is a polynomial $P()$ and the input of the receiver is x. However, unlike the basic OPE protocol both parties have random outputs which sum up to $P(x)$. In other words, the sender and receiver have output y_S and y_R, respectively, such that each of these values is uniformly distributed in $\mathcal{F}$ and it also holds that $y_S + y_R = P(x)$.

There is a simple reduction from shared output OPE to basic OPE. Namely, given an input polynomial $P()$, the sender chooses a random $y_S \in \mathcal{F}$ and defines $P'(x) = P(x) - y_S$. The parties then perform an OPE of $P'(x)$ and the receiver sets $y_R = P'(x)$.

1.3. Related work.

1.3.1. Oblivious transfer.

The basic cryptographic primitive that is used by the protocols for oblivious polynomial evaluation is *oblivious transfer*. The notion of 1-out-of-2 oblivious transfer was suggested by Even, Goldreich and Lempel [21] as a generalization of Rabin's oblivious transfer [58]. A protocol for 1-out-of-2 OT involves a sender, which has two inputs x_0 and x_1, and a receiver whose input is a single bit, $b \in \{0, 1\}$. At the end of the protocol the receiver learns x_b and nothing about x_{1-b}, while the sender learns nothing about b. More generally, a k-out-of-N OT functionality is defined in the following way.

Definition 1.5. (k-out-of-N oblivious transfer functionality).

- *Parameters: Integers* $N > k \geq 1$.
- *Input:*
  - *Sender:* N *inputs* $x_1, \ldots, x_N$.
  - *Receiver:* k *inputs* $i_1, \ldots, i_k$ *which are integer values in the range* $[1, N]$.
- *Output:*
  - *Sender: nothing.*
  - *Receiver:* $x_{i_1}, \ldots, x_{i_k}$.

For a discussion of OT see Goldreich [29]. 1-out-of-N oblivious transfer was introduced by Brassard, Crépeau and Robert [8, 9] under the name ANDOS (all or nothing disclosure of secrets). They used information theoretic reductions to construct 1-out-of-N protocols from $N - 1$ invocations of a 1-out-of-2 protocol (it was later shown that such reductions must use at least $\Omega(N)$ invocations of 1-out-of-2 OT in order to preserve the information theoretic security [17]). Goldreich and Vainish [33] and Kilian [42] showed that oblivious transfer enables general secure two-party computation, with no additional assumptions (with security against semi-honest and malicious adversaries, respectively).

This paper describes reductions of oblivious polynomial evaluation to an ideal implementation of the OT functionality of Definition 1.5. Based on composition theorems (see, e.g., [29, 10, 11]) the ideal functionality can be replaced with actual implementations of OT. Such constructions of OT can be based on physical assumptions, such as the use of a noisy channel (see, e.g., [15]), or on computational assumptions. Constructions of OT based on computational assumptions can be divided to different categories, as listed below, according to the security definitions that they satisfy. The OPE constructions can be based on OT protocols from any of these categories. (Of course, security depends on the security of the oblivious transfer protocol. In particular, all OT protocols we describe provide security against semi-honest adversaries. All but the first category provide *privacy* in the face of malicious parties.)

(i) OT protocols which provide security in the semi-honest model. These include basic protocols based on the EGL (Even–Goldreich–Lempel) paradigm [21], which are based on using a public key encryption system with the additional property that the distribution of ciphertexts is independent of the encryption key (the original EGL protocol uses trapdoor permutations with a special property, but its structure is similar). In these protocols the receiver sends two encryption keys PK_0 and PK_1, while knowing only a single decryption key, corresponding to PK_b. The sender encrypts x_0 using the key PK_0, and encrypts x_1 using the key PK_1. The receiver then decrypts x_b but cannot decrypt x_{1-b}. Protocols based on this paradigm include the construction suggested by Bellare and Micali [4], and generic constructions based on the existence of trapdoor permutations.

OT protocols of this type can be made secure with respect to malicious parties by applying the GMW (Goldreich–Micali–Wigderson) [31] paradigm, i.e., "compiling" a semi-honest protocol using generic techniques to obtain a protocol which ensures that the operation of the parties follows the protocol. This is usually done by adding zero-knowledge proofs in which the parties prove that they operate according to the protocol. If the zero-knowledge proofs enable extraction then the protocols are simulatable, but only for a single invocation at a time, rather than for parallel or concurrent invocations of the protocol.

(ii) OT protocols which provide information-theoretic security for the sender with respect to a corrupt receiver, and computational security for the receiver (e.g., the two round protocols of [52, 1, 39]). Although these protocols provide information-theoretic security, they do not enable extraction of the receiver's input. Therefore they do not enable easy simulation of the output that is obtained by the receiver. (The GMW paradigm can be applied to this type of protocol as well, but usually with considerable degradation in efficiency. In addition, the information-theoretic security might be lost.)

(iii) Fully simulatable OT protocols, in particular for parallel or concurrent invocations. These include the concurrent OT protocol of [24] (which assumes that the inputs are independent), the universally composable protocols of [11], and the universally composable committed OT of [25].

(iv) Protocols based on the random oracle model. In this model it is possible to design very efficient protocols based on the EGL or the Bellare–Micali paradigms. These protocols are secure against malicious parties and fully simulatable. Their security relies, however, on the random oracle model.

Most constructions of OT protocols that are described in the literature are of 1-out-of-2 OT. Efficient implementations of 1-out-of-N and k-out-of-N oblivious transfer protocols are described in [53]. These protocols are based on efficient computationally secure reductions to 1-out-of-2 oblivious transfer. A 1-out-of-N oblivious transfer is reduced to $\log N$ invocations of 1-out-of-2 OT, and the k-out-of-N protocol is considerably more efficient than k repetitions of 1-out-of-N oblivious transfer. Furthermore, more recent constructions [52] reduce the amortized overhead of oblivious transfer, if multiple invocations of this protocol should be run (as is the case in the OPE constructions). A direct implementation of 1-out-of-N OT, e.g., by the protocols which provide information-theoretic security for the sender [52, 1, 39], is quite efficient for the receiver which has to do only $O(1)$ work, while the sender has to perform $O(N)$ exponentiations.

It was recently shown how to extend oblivious transfer in the sense that two parties can execute a large number of OTs (a number polynomial in k, where k is a security parameter of a pseudorandom function), at the cost of running only k OTs and executing additional invocations of symmetric cryptographic functions for every OT [35] (this is an efficient realization of a generic construction of Beaver [2]). The security of this construction is based on a nonstandard assumption (or alternatively on the random oracle assumption), and security against malicious parties is obtained using a cut-and-choose method that involves running multiple invocations of the system.

In our work we measure the computational overhead by the number of OTs that are executed by the parties. The use of this criterion makes sense since all other operations are either arithmetic or are symmetric crypto operations, which are considerably more efficient. It is preferable to minimize the number of OT operations even given the recent work on extending oblivious transfer, since that construction depends on a new assumption, involves some additional constants, and requires a cut-and-choose solution against malicious parties.

We present a protocol that uses a minimal number of OTs in the sense that this number is independent of the size of the field and of the degree of the polynomial. We assume that each 1-out-of-2 OT operation is atomic and can accommodate an input of arbitrary size (this is justified since the OT can be used to transfer one of two keys whose length is equal to the security parameter, and these keys can be used to encrypt each of the two inputs, which can be of arbitrary size).

1.3.2. Secure two-party computation. The idea of secure two-party computation was introduced by Yao [60]. His construction enables one party, the sender, to define a function F and enable another party, the receiver, to compute the value of F at a single point x without learning anything else about F, and without disclosing to the sender any information about x. The construction is based on describing F as a binary circuit and evaluating the circuit. Its computational overhead is composed of running an oblivious transfer protocol for every input wire of the circuit, and computing a pseudorandom function for every gate. The bulk of the communication overhead is incurred by sending a table, of size linear in the security parameter of the pseudorandom function, for every gate. The overhead, therefore, strongly depends on the size of the representation of F as a binary circuit. In the case of a polynomial of degree d this circuit should compute x^d and its size is $O(|x|^2 \cdot \log d)$. In particular, the size of the circuit depends on the size of the field over which the polynomial is defined. (It is also possible to construct a circuit that uses an FFT approach for computing x^d. The size of this circuit, too, depends on the size of the field over which the polynomial is defined.)

In another generic construction, Naor and Nissim [49] show that any two-party protocol can be transformed into a secure protocol with the effect that a protocol with communication complexity of c bits is transformed to a secure protocol which performs c invocations of oblivious transfer (or SPIR) from a database of length 2^c. A simple (insecure) protocol for oblivious polynomial evaluation has a single round and a communication overhead of $\log|\mathcal{F}|$ bits (the receiver simply sends x to the sender). Applying the Naor–Nissim transformation to this protocol results in a secure protocol that executes an OT/SPIR out of a table of $|\mathcal{F}|$ elements. Namely, the server constructs a table of $|\mathcal{F}|$ items, containing the value of $P(x)$ for every $x \in \mathcal{F}$. The receiver then reads a single entry of this table using OT or SPIR. This protocol is definitely not strongly polynomial as its overhead is linear in $|\mathcal{F}|$.

2. Intractability assumptions.

This section contains definitions of two new pseudorandomness assumptions. They are later used for constructing protocols for oblivious polynomial evaluation. The assumptions are closely related to the noisy polynomial reconstruction problem, or the list decoding problem of Reed–Solomon codes. We first describe this well-known problem, and then introduce the pseudorandomness assumptions.

2.1. The noisy polynomial reconstruction problem.

The noisy polynomial reconstruction problem is described by the following definition.

DEFINITION 2.1 (polynomial reconstruction). *A polynomial reconstruction algorithm has the following functionality:*

- *Input: Integers k and t, and n points $\{(x_i, y_i)\}_{i=1}^n$, where $x_i, y_i \in \mathcal{F}$ ($\mathcal{F}$ is a field).*
- *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 related to the list decoding problem, which is motivated by coding theory and was first defined by Elias [20] (and sometimes also 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 that 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 2.2 (polynomial list reconstruction). *A polynomial list reconstruction algorithm has the following functionality.*

- *Input: Integers k and t, and n points $\{(x_i, y_i)\}_{i=1}^n$, where $x_i, y_i \in \mathcal{F}$ ($\mathcal{F}$ is a field).*
- *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 classic algorithm of Berlekamp and Massey (see, e.g., [48]) solves the polynomial reconstruction problem in polynomial time for $t \geq \frac{n+k}{2}$ (in this range there is a unique solution). Sudan [59] presented a polynomial algorithm that works if $t \geq \sqrt{2kn}$, and later Guruswami and Sudan [34] 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. (Later, Coppersmith and Sudan [14] showed an improved noisy interpolation algorithm which removes random errors applied to curves of the form $\langle x, p_1(x), p_2(x), \ldots, p_c(x)\rangle$, where $p_1, \ldots, p_c$ are polynomials. We are interested in the case $c = 1$, for which this algorithm is not better than the Guruswami–Sudan algorithm.)

The "polynomial list reconstruction" problem is defined as a worst case problem regarding the *number* of polynomials that might appear in the solution list. In other words, the definition requires that all polynomials within a given distance be listed. Goldreich et al. [32] 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 that 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 noisy polynomial reconstruction problem has a single solution.

We note that given an input to a noisy polynomial reconstruction problem it is possible to randomize it to obtain an input that 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 it only randomizes 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.

2.2. The intractability assumptions—pseudorandomness. We present two new intractability assumptions. The first assumption is equivalent to a conjecture that given a randomly chosen input to the polynomial list reconstruction problem the value of the polynomial at $x = 0$ is pseudorandom. The second assumption states that the value $P(0)$ is pseudorandom even given some additional hints about the location of the values of P. Section 3 describes OPE protocols that are based on the assumptions.

The intractability assumptions depend on the following parameters:

(i) $\mathcal{F}$, the field over which the polynomial is defined.

(ii) k, the degree of the hidden polynomial.

(iii) n, the number of correct values of the polynomial, which is also the number of queries made in the oblivious evaluation protocol. (This parameter corresponds to t in the definition of the polynomial list reconstruction problem, Definition 2.2. We change the notation to agree with the notation used later in this paper.)

(iv) m, the expansion ratio (namely, the ratio between the total number of points and n). This parameter corresponds to n/t in Definition 2.2.

Setting precise parameter sizes to satisfy the intractability assumptions is beyond the scope of this paper, and we therefore do not make any precise recommendations with regard to parameter sizes.

The first intractability assumption. This intractability assumption simply assumes that given an input to the polynomial list reconstruction problem, with all x_i being distinct, the value of the polynomial at $x = 0$ is pseudorandom. That is, it is infeasible to distinguish between two such inputs corresponding to polynomials with different values at $x = 0$. To define this more formally, we use the following notation:

Let $A_{n,m}^{k,\alpha}$ denote the probability distribution of sets generated in the following way:

1. Pick a random polynomial P over $\mathcal{F}$, of degree at most k, for which it holds that $P(0) = \alpha$.
2. Generate nm random values $x_1, \ldots, x_{nm}$ in $\mathcal{F}$ subject to the constraint that all x_i values are distinct and different from 0.
3. Choose a random subset S of n different indices in $[1, nm]$, and set $y_i = P(x_i)$ for all $i \in S$. For every $i \notin S$ set y_i to be a random value in $\mathcal{F}$.
4. Output the set $\{(x_i, y_i)\}_{i=1}^{nm}$.

The pseudorandomness assumption is based on the notion of computationally indistinguishability, as defined by Goldreich in [28, page 104]. It sets the size of the parameters to be polynomial in a security parameter ℓ, and requires that the resulting probability ensembles are computationally indistinguishable.

Assumption 1 (*first pseudorandomness assumption*). Let ℓ be a security parameter, and let $n(\ell), m(\ell), k(\ell), F(\ell)$ be polynomially bounded functions that define the parameters n, m, k and the description of the field $\mathcal{F}$. Let $\alpha(\ell)$ and $\alpha'(\ell)$ be any polynomially bounded functions defining elements of the field $\mathcal{F}$, and define $\alpha = \alpha(\ell)$ and $\alpha' = \alpha'(\ell)$. Let $\mathcal{A}^{k,\alpha}_{n,m}$ and $\mathcal{A}^{k,\alpha'}_{n,m}$ be random variables that are chosen according to the distributions $A^{k,\alpha}_{n,m}$ and $A^{k,\alpha'}_{n,m}$, respectively.

The assumption is that the probability ensembles $\{\mathcal{A}^{k,\alpha}_{n,m}\}$ and $\{\mathcal{A}^{k,\alpha'}_{n,m}\}$ are computationally indistinguishable for adversaries whose running time is polynomial in the security parameter ℓ.

Pseudorandomness Assumption 1 is related to the assumption that polynomial list reconstruction is hard. Assumption 1 is stronger, since in addition to assuming that reconstructing the polynomial is hard, it assumes that all x_i are distinct and that it is even hard to learn information about the value of the polynomial at 0.

Assumption 1 is weaker than an assumption that states that it is hard to distinguish between any probability ensemble $\{\mathcal{A}^{k,\alpha}_{n,m}\}$ and a probability ensemble that generates sets with the same number of *random* (x, y) values. If the latter assumption is true, then so is Assumption 1. (Assume that Assumption 1 does not hold. Then there is a distinguisher such that the probability of its output being 1 given an input from $\{\mathcal{A}^{k,\alpha}_{n,m}\}$ has a nonnegligible difference from the probability of it having a 1 output given an input from $\{\mathcal{A}^{k,\alpha'}_{n,m}\}$. The probability of a 1 output given an input from the "random" ensemble must have a nonnegligible difference from at least one of these two probabilities.) The converse might not be true. It might be that it is easy to distinguish between an input from the "random" ensemble and an input from $\{\mathcal{A}^{k,\alpha}_{n,m}\}$, yet for all α values, the inputs from the different $\{\mathcal{A}^{k,\alpha}_{n,m}\}$ ensembles are indistinguishable.

The pseudorandomness assumption is of course false for any choice of parameters for which the polynomial list reconstruction problem is easy, e.g., when $m < n/k$. (This corresponds to the number of correct points, n, being more than the square root of the total number of points, nm, times the degree of polynomial, k. This equation therefore agrees with the threshold for which the noisy polynomial problem is easy.) In the other direction, it is an open problem to find a reduction from the polynomial list reconstruction to the assumption.

The second intractability assumption. The second assumption states that given n sets with m points in every set, such that a polynomial P agrees with at least one point in each set, the value of $P(0)$ is pseudorandom. Namely, the total number of points, and the number of correct points, are as in pseudorandomness Assumption 1, but in addition there is a partition into sets and it is promised that the polynomial P agrees with at least one point in each set. It is hard to come up with a reduction to this assumption from pseudorandomness Assumption 1 since the reduction must map each of the n points of P into a different set. The new assumption seems stronger than pseudorandomness Assumption 1 since the problem is easier. Namely, the adversary is given an additional hint—the partition into sets containing at least one correct value of the polynomial.

The definition of the assumption uses the following notation.

Let $C^{k,\alpha}_{n,m}$ denote the probability distribution of sets generated in the following way:

1. Pick a random polynomial P over $\mathcal{F}$, of degree at most k, for which it holds that $P(0) = \alpha$.
2. Generate nm random values $x_1, \ldots, x_{nm}$ in $\mathcal{F}$ subject to the constraint that all x_i values are distinct and different from 0.
3. Choose a random subset S of n different indices in $[1, nm]$, and set $y_i = P(x_i)$ for all $i \in S$. For every $i \notin S$ set y_i to be a random value in $\mathcal{F}$.
4. Partition the nm (x_i, y_i) pairs to n random subsets subject to the following constraints:
   - (i) The subsets are disjoint.
   - (ii) Each subset contains exactly one pair whose index is in S (and therefore for this pair it holds that $y_i = P(x_i)$).
   - (iii) Each subset contains exactly $m-1$ pairs whose indices are not in S (and therefore have a random y_i value).

   Output the resulting subsets.

The intractability assumption says that for any α, α' the two probability ensembles $\{C^{k,\alpha}_{n,m}\}, \{C^{k,\alpha'}_{n,m}\}$ are computationally indistinguishable. It depends on the parameters F, k, m, and n.

Assumption 2 (*second pseudo-randomness assumption*). Let ℓ be a security parameter, and let $n(\ell), m(\ell), k(\ell), F(\ell)$ be polynomially bounded functions that define the parameters n, m, k and the description of the field $\mathcal{F}$. Let $\alpha(\ell)$ and $\alpha'(\ell)$ be any polynomially bounded functions defining elements of the field $\mathcal{F}$, and define $\alpha = \alpha(\ell)$ and $\alpha' = \alpha'(\ell)$. Let $\mathcal{C}^{k,\alpha}_{n,m}$ and $\mathcal{C}^{k,\alpha'}_{n,m}$ be random variables that are chosen according to the distributions $C^{k,\alpha}_{n,m}$ and $C^{k,\alpha'}_{n,m}$, respectively.

The assumption is that the probability ensembles $\{\mathcal{C}^{k,\alpha}_{n,m}\}$ and $\{\mathcal{C}^{k,\alpha'}_{n,m}\}$ are computationally indistinguishable.

As with Assumption 1 there is an easy reduction from the problem of breaking this pseudorandomness assumption to the noisy polynomial reconstruction problem. In addition, if pseudorandomness Assumption 1 does not hold in the worst case, then pseudorandomness Assumption 2 does not hold in the worst case (since any input for the latter can be transformed to an input for the first assumption by simply ignoring the partition into subsets). This reduction is also true in the average case: consider all sets generated by the distributions $A^{k,\alpha}_{n,m}$ and $C^{k,\alpha}_{n,m}$ subject to the constraint that no more than n points in a set agree with the polynomial P (the probability that this property does not hold for a specific set is at most $n(m-1)/|\mathcal{F}|$, which we consider to be negligible). Denote the resulting collections of sets as $\hat{A}^{k,\alpha}_{n,m}$ and $\hat{C}^{k,\alpha}_{n,m}$. A set in $\hat{A}^{k,\alpha}_{n,m}$ is matched to a set in $\hat{C}^{k,\alpha}_{n,m}$ if they contain exactly the same points. Each of the sets in $\hat{A}^{k,\alpha}_{n,m}$ is therefore matched with the exactly same number of sets in $\hat{C}^{k,\alpha}_{n,m}$, and no two sets in $\hat{A}^{k,\alpha}_{n,m}$ are matched with the same set in $\hat{C}^{k,\alpha}_{n,m}$. Now, picking a random set in $\hat{C}^{k,\alpha}_{n,m}$ and removing the partition into subsets results in the matching set in $\hat{A}^{k,\alpha}_{n,m}$. Therefore, this procedure has the same probability of hitting any set in $\hat{A}^{k,\alpha}_{n,m}$.

It is an interesting open problem to provide a reduction to pseudorandomness Assumption 2 from pseudorandomness Assumption 1, or from the polynomial reconstruction problem. The problem with designing such a reduction is that its output should comply with the input distribution of pseudorandomness Assumption 2, namely, in each subset there should be (with high probability) a single value of the polynomial P.

Remark. The pseudorandomness assumption that was used in an earlier version of this work [51] was broken by Bleinchenbacher and Nguyen [6] and by Boneh [7]. The assumption was similar to pseudorandomness Assumption 2, but with the additional requirement that in each subset of the sets generated by $C_{n,m}^{k,\alpha}$ all the points have the *same x coordinate*. This property enables reduction of this problem to an instance of the lattice shortest vector problem, and solving it using the LLL algorithm [44]. We remark that it is unknown how to employ this attack against pseudorandomness Assumption 2. Furthermore, the overhead of the oblivious evaluation scheme that is based on pseudorandomness Assumption 2 is not greater than that of the scheme that is based on the broken assumption.

3. Protocols for oblivious polynomial evaluation.

This section describes protocols for oblivious evaluation of polynomials. The protocols reduce computing the OPE functionality to computing an OT functionality. We first describe a generic OPE protocol (Protocol 3.1), and then describe two instantiations of the generic protocol using each of the two pseudorandomness assumptions (Protocols 3.2 and 3.3, respectively). These protocols provide privacy against semi-honest or malicious senders, and against semi-honest receivers.

The protocols ensure that a malicious receiver learns at most a single linear equation of the coefficients of the polynomial, but this equation might not correspond to a legitimate value of the polynomial. We show that if the polynomial P that is evaluated is linear, then the protocols are secure even against malicious receivers, since the linear equation always corresponds to a value of the polynomial.

Security against malicious receivers, and improved efficiency. We then describe Protocol 3.4 which is secure against malicious receivers. Furthermore, Protocol 3.4, although a little more complicated conceptually, has computational overhead which is better than that of the previous protocols: the number of oblivious transfers is independent of the degree of the polynomial P and is equal to the number of oblivious transfers that are required in the case of a linear polynomial.

3.1. A generic protocol.

All the protocols involve a receiver A and a sender B and have the following specifications:

- Input:
    - Sender: a polynomial $P(y) = \sum_{i=0}^{d_P} b_i y^i$ of degree d_P in the field $\mathcal{F}$.
    - Receiver: a value $\alpha \in \mathcal{F}$.
- Output:
    - Sender: nothing.
    - Receiver: $P(\alpha)$.
- Protocol security parameters: m, k, which are discussed below.

At the end of the protocol the parties learn nothing but their specified outputs. The generic protocol (Protocol 3.1) is described in Figure 1. It is based on the sender hiding P in a *bivariate* polynomial, and running oblivious transfer protocols with the receiver to reveal to her just enough information to enable the computation of $P(\alpha)$. (In this respect, the protocol is somewhat similar to the instance hiding construction of [3, 46].)

Note that the protocol uses a polynomial $Q(x, y)$ which is defined by $d + d_P + 1$ coefficients only, whereas a random bivariate polynomial of the same degrees is defined by $(d + 1)(d_P + 1)$ coefficients.

The only step that has to be further specified is step 3, in which the receiver learns $d_R + 1$ values of R. This is the only step that involves interaction between the two parties, and as such determines the overhead of the protocol. Below are descriptions

PROTOCOL 3.1 (a generic protocol for oblivious polynomial evaluation).

1. **The sender hides P in a bivariate polynomial:** *(see Figure 2) The sender generates a random masking polynomial $P_x(x)$ of degree d, s.t. $P_x(0) = 0$. Namely, $P_x(x) = \sum_{i=1}^{d} a_i x^i$. The parameter d equals the degree of P multiplied by the security parameter k (i.e., $d = k \cdot d_P$).*

   The sender then defines a bivariate polynomial

$$Q(x,y) = P_x(x) + P(y) = \sum_{i=1}^{d} a_i x^i + \sum_{i=0}^{d_P} b_i y^i$$

   for which it holds that $\forall y \;\; Q(0,y) = P(y)$.
2. **The receiver hides α in a univariate polynomial:** *(see Figure 3) The receiver chooses a random polynomial S of degree k, such that $S(0) = \alpha$. The receiver'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, therefore, if the receiver is able to interpolate R she can learn $R(0) = P(\alpha)$. The degree of R is $d_R = d = k \cdot d_P$.*
3. **The receiver learns points of R:** *The receiver learns $d_R + 1$ values of the form $\langle x_i, R(x_i) \rangle$.*
4. **The receiver computes $P(\alpha)$:** *The receiver uses the values of R that it learned to interpolate $R(0) = P(\alpha)$.*

FIG. 1. *A generic protocol for oblivious polynomial evaluation.*

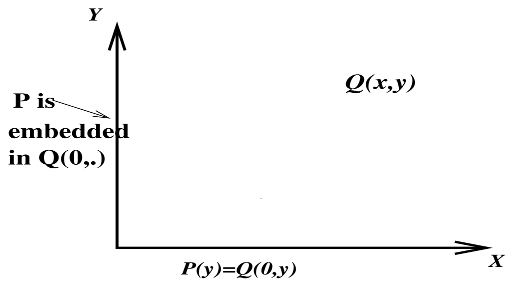

FIG. 2. *The polynomial P embedded in the bivariate polynomial Q s.t. $Q(0,y) = P(y)$.*

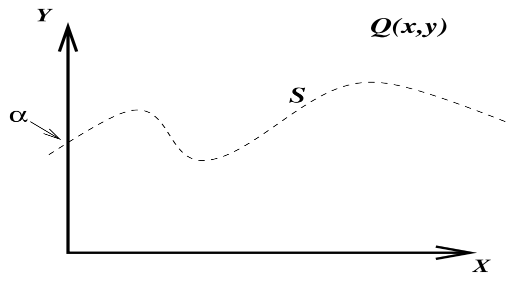

FIG. 3. *The polynomial S defines a polynomial R s.t. $R(0) = Q(0, S(0)) = Q(0,\alpha) = P(\alpha)$.*

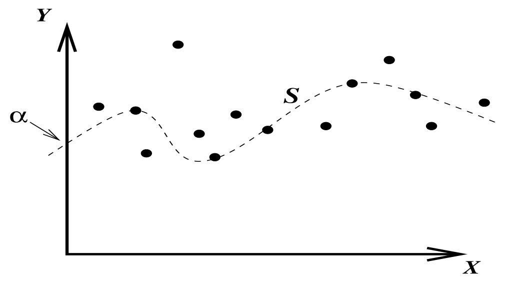

FIG. 4. *The receiver uses n-out-of-N OT to learn n values of R hidden between N values.*

of two protocols for oblivious polynomial computation in which step 3 is based on each of the two pseudorandomness assumptions.

3.2. Detailed protocols. This section describes instantiations of Protocol 3.1 based on the two pseudorandomness assumptions.

3.2.1. A protocol based on pseudorandomness Assumption 1. The first instantiation is a reduction to an n-out-of-N oblivious transfer protocol (see, e.g., [53]), in which the receiver learns n of the N inputs of the sender while hiding these values from the sender.

PROTOCOL 3.2 (oblivious polynomial evaluation based on Assumption 1). *The protocol is the generic protocol (Protocol* 3.1*), where the third step is run as follows:*

1. *The receiver sets* $n = d_R + 1 = d + 1 = kd_P + 1$ *and chooses* $N = nm$ *distinct random values* $x_1, \dots, x_N \in \mathcal{F}$*, all different from* 0.
2. *The receiver chooses a random set* T *of* n *distinct indices* $1 \le i_1, \dots, i_n \le N$*. She then defines* N *values* y_i*, for* $1 \le i \le N$*. The value* y_i *is defined as* $S(x_i)$ *if* i *is in* T*, and is a random value in* $\mathcal{F}$ *otherwise.*
3. *The receiver sends the* N *points* $\{(x_i, y_i)\}_{i=1}^N$ *to the sender.*
4. *The receiver and sender execute an* n*-out-of-*N *oblivious transfer protocol for the* N *values* $Q(x_1, y_1), \dots, Q(x_N, y_N)$ *(see Figure* 4*). The receiver chooses to learn* $\{Q(x_i, y_i)\}_{i \in T}$*.*

The correctness of the protocol is based on observing that the receiver learns $d_R + 1$ values of the polynomial R, and can, therefore, interpolate R and compute $R(0) = P(\alpha)$. We prove the security of the protocol in section 3.3 below.

3.2.2. A protocol based on pseudorandomness Assumption 2. The second protocol is based on using n sets of points, such that each set contains a point of the polynomial R. The receiver runs an independent oblivious transfer protocol for each set, in which it learns a value of R.

PROTOCOL 3.3 (oblivious polynomial evaluation based on Assumption 2). *The protocol is the generic protocol (Protocol* 3.1*), where the third step is run as follows:*

1. *The receiver sets* $n = d_R + 1 = kd_P + 1$*, defines* $N = nm$ *and chooses* N *distinct random values* $x_1, \dots, x_N \in \mathcal{F}$*, all different from* 0.
2. *The receiver chooses a set* T *of* n *randomly-chosen distinct indices* $1 \le i_1, i_2, \dots, i_n \le N$*, subject to the constraint that* $(j-1)m + 1 \le i_j \le jm$ *for* $1 \le j \le n$*.*
3. *The receiver defines* N *values* y_i*, for* $1 \le i \le N$*. The value* y_i *is defined as* $S(x_i)$ *if* i *is in* T*, and is random otherwise.*
4. *The receiver partitions the* N *pairs* $\{(x_i, y_i)\}_{i=1}^N$ *into* n *subsets* $B_1, \dots, B_n$*, where subset* B_j *contains the* m *pairs indexed* $(j-1)m+1$ *to* jm*. This means that each subset* B_j *contains exactly one pair from* T*.*

5. *The receiver sends the n subsets $B_1, \dots, B_n$ to the sender.*
6. *The receiver and sender execute n protocols of* 1*-out-of-m oblivious transfer, one for each subset. The protocol for subset B_j enables the receiver to learn one of the values $Q(x_i, y_i)$ for the m points (x_i, y_i) in B_j. (The receiver should choose to learn $Q(x_i, y_i)$ for which $y_i = P(x_i)$.)*

3.2.3. Properties of the protocols.

Correctness. It is straightforward to verify that both protocols enable the receiver to obtain any value $P(\alpha)$ she desires. She can do this by choosing a polynomial S for which $S(0) = \alpha$, learning $d_R + 1$ values of R, and interpolating $R(0) = P(\alpha)$.

Complexity. The main overhead of the protocols, in terms of both computation and communication, is the overhead of the oblivious transfer stage. The overhead depends on the degree of P and on the security parameters k and m. Namely, the overhead of each protocol is that of running the following primitives:

(i) Protocol 3.2 (based on Assumption 1): running a single invocation of $(kd_P + 1)$-out-of-$[(kd_P + 1) \cdot m]$ oblivious transfer.

(ii) Protocol 3.3 (based on Assumption 2): running $(kd_P + 1)$ invocations of 1-out-of-m oblivious transfer.

The actual overhead of the protocol depends on the value that is set to the parameter m, as a function of k and d_P, in order for the relevant security assumption to hold (this value might be different in each of the protocols). It might seem that Protocol 3.3, which uses $(kd_P + 1)$ invocations of 1-out-of-m oblivious transfer, is always inferior to Protocol 3.2, which uses a single invocation of $(kd_P + 1)$-out-of-$[(kd_P + 1) \cdot m]$ oblivious transfer. Also, the fact that in Protocol 3.3 each subset is known to contain a value of the polynomial S might make the task of attacking the receiver easier, and require the use of larger parameter m to ensure the security of the protocol. We describe both protocols since the choice of parameters k and m might be different in the two cases and result in Protocol 3.3 being the more efficient. Also, recent work on extending oblivious transfer [35] leads to more efficient implementation of multiple invocations of OT.

We note that the multiple invocations of the oblivious transfer protocol in Protocol 3.3 can be run in parallel. The protocol is secure if the specific oblivious transfer protocol which is used can be securely run in parallel with respect to the relevant adversary (i.e., with respect to either semi-honest or malicious adversaries).

3.2.4. A note on evaluating multiple polynomials with the same receiver input. The OPE protocols have a rather handy property: a *single* invocation of a protocol can be used to let the receiver compute the values of *several* polynomials at the *same* point x, with the same overhead (in terms of the number of oblivious transfers) as computing the value of a single polynomial. This can be done since the choices of the receiver in the OT protocols depend on its input x alone, and this input is the same in all invocations of the OPE protocol. The parallel invocation is done by the parties running a protocol in which the sender's input is n polynomials $\langle P_1, \dots, P_n \rangle$, the receiver's input is x, and the receiver's output is $\langle P_1(x), \dots, P_n(x) \rangle$. The sender defines appropriate polynomials $\langle Q_1, \dots, Q_n \rangle$ and in every step in which the sender transfers a single value $Q(i, j)$ in the original protocol, it transfers n values $\langle Q_1(i, j), \dots, Q_n(i, j) \rangle$ in the multipolynomial protocol. Note that this variant ensures that the receiver computes the values of all the polynomials at the *same point x*. This variant of the protocol is used in Protocol 3.4 in section 3.4, and is also used in [26] for distributed generation of RSA keys.

3.3. Security analysis for semi-honest behavior. In order to prove the security of the protocol one must show that the privacy of both parties is preserved. The privacy of the receiver is based on the pseudorandomness assumption which is used (namely, Assumption 1 or 2), while the privacy of the sender is independent of this assumption. The proof employs a hybrid model in which there is access to an ideal OT functionality, as is defined in Definition 1.5. Composition theorems (see, e.g., [29, 10, 11]) can then be used to replace the ideal OT functionality with a specific implementation of OT, as is described in section 1.3.1.

3.3.1. The receiver's privacy. The privacy goal of the receiver is to hide the value $\alpha = S(0)$ from the sender. We only need to show this for a semi-honest sender, since the protocol includes a single message from the receiver to the sender, which does not depend on the operation of the sender (and our security definition does not require stimulation of the output of the receiver together with the view of the sender). This property is guaranteed by the pseudorandomness assumptions stated in section 2.2. We prove this result for Protocol 3.2, based on pseudorandomness Assumption 1. The proof for Protocol 3.3 is identical and is based on pseudorandomness Assumption 2.

THEOREM 3.1. *If the sender can distinguish between two different inputs of the receiver in Protocol* 3.2*, then pseudorandomness Assumption* 1 *does not hold (namely, there is a distinguisher between the two probability ensembles stated in the assumption).*

Proof. The sender's view in the protocol contains the set of points that is given to him by the receiver, and the interaction between the two parties in the oblivious transfer protocol. Each instance of the sender's view therefore contains nm points, and in addition his view in an oblivious transfer protocol in which the receiver chooses to learn values associated with a set of n of the nm points. Recall also that we assume that the protocol uses an ideal OT functionality.

Let α_0, α_1 be any two values in $\mathcal{F}$. Consider the following probability distributions of instances of the sender's view in the protocol.

(i) $S_{OT,0}$: The set of nm points is chosen randomly from $A^{k,\alpha_0}_{n,m}$. The interaction of the receiver in the oblivious transfer protocol corresponds to it choosing to learn the n correct points of the polynomial among the nm points.

(ii) $S_{OT,1}$: Similar to $S_{OT,0}$, with the only difference being that the set of nm points is chosen randomly from $A^{k,\alpha_1}_{n,m}$.

(iii) $S_{R,0}$: The set of nm points is chosen randomly from $A^{k,\alpha_0}_{n,m}$. The interaction of the receiver in the oblivious transfer protocol corresponds to it choosing to learn n points sampled at random from the set of nm points.

(iv) $S_{R,1}$: Similar to $S_{R,0}$, with the only difference being that the set of nm points is chosen randomly from $A^{k,\alpha_1}_{n,m}$.

We would like to show that no algorithm D_{α_0,α_1} run by the sender can distinguish between an input sampled from $S_{OT,0}$ and an input sampled from $S_{OT,1}$. Define $D_{OT,0}$ as the probability that the output of D_{α_0,α_1} is 1 given an input sampled from $S_{OT,0}$. Similarly, define $D_{OT,1}$, $D_{R,0}$ and $D_{R,1}$. It holds that

$$|D_{OT,0} - D_{OT,1}| \leq |D_{OT,0} - D_{R,0}| + |D_{R,0} - D_{R,1}| + |D_{R,1} - D_{OT,1}|.$$

Assume to the contrary that $|D_{OT,0} - D_{OT,1}|$ is nonnegligible. In this case either the value $|D_{OT,0} - D_{R,0}| + |D_{R,1} - D_{OT,1}|$ is nonnegligible (option 1), or the value $|D_{R,0} - D_{R,1}|$ is nonnegligible (option 2).

If option 1 occurs, then either $|D_{R,0} - D_{OT,0}|$ or $|D_{R,1} - D_{OT,1}|$ is nonnegligible. This is a contradiction since in this case the sender can distinguish between different inputs of the receiver to the oblivious transfer protocol. (It is possible to show a reduction that uses the fact that, e.g., $|D_{R,0} - D_{OT,0}|$ is nonnegligible, to distinguish between different inputs of the receiver.)

If option 2 occurs then we have a distinguisher between inputs sampled from $A^{k,\alpha_0}_{n,m}$ and from $A^{k,\alpha_1}_{n,m}$. The distinguisher operates by receiving its input, adding to it an interaction for the oblivious transfer protocol in which the receiver learns a random set of n values, and forwarding the combined input to D_{α_0,α_1}. Since $|D_{R,0} - D_{R,1}|$ is nonnegligible there will be a nonnegligible difference between the output being 1 given inputs from the two distributions. ▯

3.3.2. The sender's privacy—the case of a semi-honest receiver. We first assume a semi-honest receiver whose operation follows the behavior that it should take according to the protocol. This is a good model for the case of a truthful party that executes the protocol, but at a later stage falls prey to an adversary that might examine the information learned during the protocol execution. The proof for the case of a semi-honest receiver, as we show in section 3.4, is also sufficient for the case of *linear* polynomials, even if the receiver is malicious. Later we show how to provide privacy in the general case of malicious receivers.

The proof that the sender's privacy is preserved against semi-honest receivers is identical for all protocols. The sender hides the polynomial P in a bivariate polynomial Q that is generated by adding to $P()$ a random polynomial in x, and the receiver obtains a system of $d_R + 1 = d + 1 = kd_P + 1$ values of Q, using different x values. Lemma 3.2 states that the receiver learns only a single linear combination of the coefficients of P. In particular this implies (Corollary 3.1) that a semi-honest receiver learns only a single value of the polynomial P.

LEMMA 3.2. *Let $Q(x, y)$ be a bivariate polynomial of the form*

$$Q(x,y) = P_x(x) + P(y) = \sum_{i=1}^{d} a_i x^i + \sum_{i=0}^{d_P} b_i y^i.$$

Then for any $d+1$ values $x_1, \ldots, x_{d+1}$, which are distinct and different from 0, and for any $d+1$ values $y_1, \ldots, y_{d+1}$, the distribution of $\{Q(x_j, y_j)\}_{j=1}^{d+1}$ is either independent of the coefficients $b_0, \ldots, b_{d_P}$, or depends on a single linear equation of these coefficients.

Proof. The values of $Q(x_j, y_j)$ are equations of the form $Q(x_j, y_j) = \sum_{i=1}^{d} a_j \cdot (x_j)^i + \sum_{i=0}^{d_P} b_i \cdot (y_j)^i$. They correspond, therefore, to a set of $d+1$ equations of the following form, with different x_j's:

$$\underbrace{\begin{pmatrix} (x_{j_1})^d & (x_{j_1})^{d-1} & \cdots & x_{j_1} & y_{j_1}^{d_P} & \cdots & 1 \\ (x_{j_2})^d & (x_{j_2})^{d-1} & \cdots & x_{j_2} & y_{j_2}^{d_P} & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ (x_{j_{d+1}})^d & (x_{j_{d+1}})^{d-1} & \cdots & x_{j_{d+1}} & y_{j_{d+1}}^{d_P} & \cdots & 1 \end{pmatrix}}_{A} \begin{pmatrix} a_d \\ \vdots \\ a_1 \\ b_{d_P} \\ \vdots \\ b_0 \end{pmatrix} = \begin{pmatrix} Q(x_{j_1}, y_{j_1}) \\ Q(x_{j_2}, y_{j_2}) \\ \vdots \\ Q(x_{j_{d+1}}, y_{j_{d+1}}) \end{pmatrix}.$$

It should be shown that regardless of the values of the points (x_j, y_j), the rows of the matrix A do not span more than a single linear combination of the vectors

$$\{e_i = (\underbrace{0, \ldots, 0, 1}_{i}, 0, \ldots, 0) \mid d+1 \le i \le d + d_P + 1\}.$$

The matrix A has $d + d_P + 1$ columns and $d + 1$ rows. Consider the matrix A' with $d + d_P + 1$ rows that is formed by taking the first d rows of A and appending to them the vectors $e_{d+1}, \ldots, e_{d+d_p+1}$. The determinant of A' is different from 0, since its upper-left submatrix of size $d \times d$ is a Vandermonde matrix, and the lower-right submatrix of size $(d_p + 1) \times (d_p + 1)$ is an identity matrix. Therefore, the first d rows of A do not span any of $e_{d+1}, \ldots, e_{d+d_p+1}$, and the matrix A that has just a single additional row cannot span more than a single linear combination of these vectors. ∎

Note that we assume that the OT protocol implements an ideal functionality, and therefore the receiver (either semi-honest or malicious) learns $d + 1$ values from the set $\{Q(x_j, y_j)\}_{j=1}^{d+1}$. In the case of a semi-honest receiver these values are of the form $Q(x_j, P(x_j))$, whereas a malicious receiver can learn arbitrary values of $Q(x_j, y_j)$ subject to the constraint that the x coordinates are distinct and different from 0. Therefore Lemma 3.2 implies the following two corollaries.

Corollary 3.1. *A semi-honest receiver learns only a single value of the polynomial P.*

Corollary 3.2. *A malicious receiver learns only a single linear equation of the coefficients of the polynomial P.*

The latter corollary is used in section 3.4 to construct an OPE protocol which is secure against malicious adversaries.

3.4. Security against malicious behavior. As described in section 3.3.1, we should only consider a semi-honest server since the only message sent by the receiver is sent before it receives any information from the server. On the other hand, a malicious adversary that plays the receiver's role might run the protocol and ask to learn values $Q(x, y)$ that do not correspond to a polynomial $S(x)$, i.e., are not of the form $Q(x, S(x))$. By doing so it might learn a linear combination of the coefficients of P which does not correspond to any value of $P(x)$. Lemma 3.2 shows that the adversary can only learn a single such linear combination. This might be sufficient for some applications, but to conform to the "ideal model" security definition, or, in general, to security against malicious behavior, the protocol must limit the information that the receiver can learn to a linear combination corresponding to a value of the polynomial P. This is achieved by the Protocol 3.4 that is described in what follows.

Improved efficiency. Another advantage of Protocol 3.4 is that its computational overhead, as measured by the number of oblivious transfers that are executed, is independent of the degree of P, and is equal to the overhead of Protocols 3.2 and 3.3 for the case of a *linear* polynomial P.

Before describing Protocol 3.4, we first note that if the polynomial P is linear, then Protocols 3.1, 3.2, and 3.3 allow the receiver to learn only a single value of P and no other information about the coefficients (regardless of whether the receiver is malicious or not). This is proved for Protocol 3.2 in the following lemma.

Lemma 3.3. *When P is linear, the only information that the receiver can learn in Protocol* 3.2 *is a single value of $P(\cdot)$.*

Proof. Denote $P(x)$ as $P(x) = b_1 x + b_0$. Corollary 3.2 implies that the receiver can only learn a single linear combination of the form $b_1 \cdot \gamma_1 + b_0 \cdot \gamma_0$, where the receiver knows γ_1 and γ_0. If $\gamma_0 \neq 0$, then this is equivalent to the receiver learning $P(\gamma_1/\gamma_0) = b_1 \cdot (\gamma_1/\gamma_0) + b_0$.

We next claim that the adversary cannot learn any combination in which the coefficient γ_0 is 0. Consider the matrix A used in the proof of Theorem 3.2. The receiver can learn the scalar multiplication of the coefficient vector by a linear combination of the rows of this matrix. In the case of a linear polynomial P, the matrix has $d+1$ rows and $d+2$ columns, where the last two columns correspond to the coefficients b_1 and b_0 of P. Note that the $(d+1)\times(d+1)$ matrix that is composed of the first d columns together with the last column, is a Vandermonde matrix. Therefore no linear combination of the $d+1$ rows of the original matrix can generate the row $(0,\dots,0,1,0)$, which corresponds to the receiver learning a linear combination $b_1\cdot\gamma_1+b_0\cdot\gamma_0$ in which the coefficient γ_0 is 0. ∎

Using linearization to combat malicious receivers. Given Lemma 3.3 the major tool we use is a reduction from the OPE of a polynomial of degree z, to z OPEs of *linear* polynomials, due to Gilboa [27]. The reduction is stated in the following lemma, which is proven at the end of this section.

LEMMA 3.4 (see [27]). *For every polynomial P of degree z, there exist z* linear *polynomials $P_1,\dots,P_z$ such that an OPE of P can be reduced to a parallel execution of an OPE of each of $P_1,\dots,P_z$, where all the linear polynomials are evaluated at the same point.*

We first describe Protocol 3.4 that uses this reduction to provide security against a malicious receiver. We then show that the privacy of the server is preserved and prove the lemma. Protocol 3.4 itself ensures that the receiver evaluates all linear polynomials at the same point. Furthermore, the overhead, in terms of oblivious transfers, is the same as that of a *single* OPE of a linear polynomial, since the choices of the receiver in the oblivious transfer invocations in all OPEs are the same, and therefore it is possible to use parallel executions of OPE as described in section 3.2.4.

PROTOCOL 3.4 (oblivious polynomial evaluation against malicious receivers). *The sender's input is P and the receiver's input is a polynomial α. The protocol is composed of the following steps:*

1. *The sender generates the d_P linear polynomials $P_1,\dots,P_{d_P}$ that are used for reducing the OPE of the polynomial P to d_P OPEs of linear polynomials, by the method of Lemma* 3.4*.*
2. *The parties execute d_P instances of OPE in which the receiver evaluates the linear polynomials $P_1,\dots,P_{d_P}$ at the point α, under the following constraints:*
   (i) *The sender generates independent masking polynomials $P_{x,i}$, $1\le i\le d_P$, and obtains the resulting bivariate polynomials $Q_i(x,y)=P_{x,i}(x)+P_i(y)$, one for each of the d_P OPEs. (step* 1 *of Protocol* 3.1*.)*
   (ii) *The receiver generates* a single *polynomial S for use in all the OPEs (step* 2 *of Protocol* 3.1*). This step defines d_P polynomials $R_1(x)=Q_1(x,S(x)),\dots,R_{d_P}(x)=Q_{d_P}(x,S(x))$ such that for each i it holds that $R_i(0)=P_i(\alpha)$.*
   (iii) *The receiver learns d_R+1 tuples of the form $(x_j,R_1(x_j),\dots,R_{d_P}(x_j))$. These values enable it to interpolate $R_1(\alpha),\dots,R_{d_P}(\alpha)$ (steps* 3 *and* 4 *of Protocol* 3.1*).*
   The implementation of this step is done by executing the same number of oblivious transfers as is required for a single *OPE of a linear polynomial. In the protocol the receiver obtains values of all polynomials, namely, $(x_j,R_1(x_j),\dots,R_{d_P}(x_j))$, instead of a single value $(x_j,R(x_j))$.*
3. *The receiver uses $P_1(\alpha),\dots,P_{d_P}(\alpha)$ to compute $P(\alpha)$ according to the method of Lemma* 3.4*.*

It follows from Lemma 3.4 and from the correctness of Protocol 3.1 that this protocol is correct. Namely, that it enables the receiver to compute $P(\alpha)$ for every value α. As for efficiency, we measure the computation overhead by the number of oblivious transfers that are executed. The protocol requires the same number of oblivious transfers as is required by an OPE of a *linear* polynomial, regardless of the degree d_P.[1] It is therefore more efficient computationally than a direct OPE of the d_P degree polynomial P. The communication overhead is about $d_P + 1$ times larger than that of the OPE of a linear polynomial.

THEOREM 3.5. *Assuming the use of an ideal OT functionality, Protocol* 3.4 *is secure against malicious behavior. Namely,*

(i) *The sender cannot distinguish between two different inputs of the receiver.*

(ii) *The receiver learns only a single value of the polynomial P.*

Proof. The receiver's privacy is an immediate corollary of the receiver's privacy in the linear OPE protocol (Theorem 3.1), since the information that the receiver sends is the same as in a single OPE protocol where it asks to learn the value of a linear polynomial at $x = \alpha$. The sender's privacy follows from Lemma 3.4 and from Lemma 3.3 (which guarantees the sender's privacy against malicious receivers in the case of linear polynomials). ∎

We now turn to the proof of Lemma 3.4.

Proof of Lemma 3.4. The lemma follows from the following three claims. ∎

CLAIM 3.1. *For every polynomial P of degree z the server can define z* linear *polynomials $P_1, \dots, P_z$ such that given $P_1(\alpha), \dots, P_z(\alpha)$, it is possible to compute the value of $P(\alpha)$.*

Proof. Denote the polynomial P as $P(x) = \sum_{i=0}^{z} b_i x^i$, where z denotes the degree of the polynomial. The Horner representation of this polynomial is the following:

$$P(x) = (((b_z x + b_{z-1}) \cdot x + b_{z-2}) \cdot x) + \cdots + b_0.$$

The innermost linear polynomial of the Horner representation is $Q_z(x) = b_z x + b_{z-1}$. Define the linear polynomial $P_z(x) = b_z x + b_{z-1} - s_m$, where s_z is a random value chosen by the server. Of course, it holds that $P_z(x) + s_z = Q_z(x)$. Suppose that the client learns the value $P_z(\alpha)$ (say, by executing an OPE). Following this step, the client and server have two random shares, $P_z(\alpha)$ and s_z, that sum up to $Q_z(\alpha)$. Now, the innermost polynomial of degree 2 is the following:

$$Q_{z-1}(x) = Q_z(x) \cdot x + b_{z-2} = (P_z(x) + s_z) \cdot x + b_{z-2} = P_z(x) \cdot x + s_z \cdot x + b_{z-2}.$$

Define $P_{z-1}(x) = s_z \cdot x + b_{z-2} - s_{z-1}$, where s_{z-1} is randomly chosen by the server. Then,

$$Q_{z-1}(\alpha) = P_z(\alpha) \cdot \alpha + P_{z-1}(\alpha) + s_{z-1}.$$

Suppose now that the client learns $P_{z-1}(\alpha)$ (by executing an OPE). Now the parties know two random shares that sum up to $Q_{z-1}(\alpha)$: the client can compute the share $P_z(\alpha) \cdot \alpha + P_{z-1}(\alpha)$, and the server knows s_{z-1}.

In the general case, the inner polynomial of degree $z - i$ can be represented as

$$\begin{aligned} Q_i(x) &= Q_{i+1}(x) \cdot x + b_{i-1} \\ &= P_z(x) \cdot x^{z-i} + P_{z-1}(x) \cdot x^{z-i-1} + \cdots + P_{i+1}(x) \cdot x + s_{i+1} \cdot x + b_{i-1}. \end{aligned}$$

[1]The length of the inputs of the oblivious transfer protocols is likely to be larger than the security parameter. This fact does not increase the number of oblivious transfers: the sender encrypts the long inputs with random keys, uses oblivious transfer to let the receiver learn one of the keys, and sends all encrypted inputs to the receiver. The receiver then uses the key to decrypt one of the inputs.

The server chooses a random value s_i and defines the linear polynomial $P_i(x) = s_{i+1} \cdot x + b_{i-1} - s_i$. It now holds that $(P_z(\alpha) \cdot \alpha^{z-i} + P_{z-1}(\alpha) \cdot \alpha^{z-i-1} + \cdots + P_{i+1}(\alpha) \cdot \alpha + P_i(\alpha))$ and s_i are two random shares, which can be computed by the client and the server, respectively, and which sum up to $Q_i(\alpha)$. This definition is used up to $P_2(x)$; for $P_1(x)$, the server defines $P_1(x) = s_2 \cdot x + b_0$ (without any random value m_1).

We get that $Q_1(x) = P(x)$, and therefore

$$P(\alpha) = P_z(\alpha) \cdot \alpha^{z-1} + P_{z-1}(\alpha) \cdot \alpha^{z-2} + \cdots + P_1(\alpha). \qquad (3.1)$$ □

Claim 3.2. *The computation of the z linear polynomials of Claim* 3.4 *can be done by z OPEs that are executed in parallel.*

Proof. Note that the polynomials $P_1, \ldots, P_z$ do not depend on α. Therefore, the server can define them in advance (by defining the values $s_2, \ldots, s_z$). This enables the parties to execute z parallel invocations of OPE, in which the client computes $P_1(\alpha), \ldots, P_z(\alpha)$. At the end of this stage, the client is able to compute $P(\alpha)$ using (3.1).

It follows that for every $\alpha \in \mathcal{F}$, the client can compute $P(\alpha)$ by learning the values $P_1(\alpha), \ldots, P_z(\alpha)$. If the parties use the OPE protocols described in this paper then the $z+1$ OPEs can be implemented with the client sending the same message in all invocations, which is the same information as in a single OPE of a linear polynomial at the point α. The server responds with the relevant answer for each of the z polynomials. □

In order to prove that the only information about P that can be computed given $P_1(\alpha), \ldots, P_z(\alpha)$ is $P(\alpha)$ we prove the following claim.

Claim 3.3. *Given $P(\alpha)$ it is possible to simulate the client's output in the z invocations of the OPE protocols. (Namely, the only information about P that can be computed given $P_1(\alpha), \ldots, P_z(\alpha)$ is $P(\alpha)$.)*

Proof. To prove the claim we show that for every $\langle P, \alpha \rangle$ it holds that the vector $\langle P_2(\alpha), \ldots, P_z(\alpha) \rangle$ is uniformly distributed in $\mathcal{F}^{z-1}$ given that $\langle s_2, \ldots, s_z \rangle$ are uniformly distributed in $\mathcal{F}^{z-1}$.

For every $2 \le i \le z$, observe that we can write $P_i(\alpha) = C_i(P, \alpha, s_{i+1}, \ldots, s_z) - s_i$, where $C_i(P, \alpha, s_{i+1}, \ldots, s_z)$ is a function of the coefficients of P, of α, and of $s_{i+1}, \ldots, s_z$. The claim is proved by induction, showing that $P_i(\alpha), \ldots, P_z(\alpha)$ is random, with i going from z down to 2. The base case, $i = z$ is clear. In every step, we observe that s_i, which is chosen at random, is used to define $P_i(\alpha)$ alone and is not involved in defining $P_{i+1}(\alpha), \ldots, P_z(\alpha)$. Therefore the vector $P_i(\alpha), \ldots, P_z(\alpha)$ is randomly distributed given that $P_{i+1}(\alpha), \ldots, P_z(\alpha)$ is randomly distributed.

Now, recall (3.1). $P_1(\alpha)$ is well defined given $\alpha, P(\alpha)$, and the values of $P_2(\alpha), \ldots, P_z(\alpha)$. Therefore to simulate the client's view we choose random values for $P_2(\alpha), \ldots, P_z(\alpha)$ and compute $P_1(\alpha)$ according to (3.1). □

Realizing oblivious transfer. The security proofs show a reduction of security to that of an ideal OT functionality. All implementations of OT which are described in section 1.3.1 provide security against semi-honest adversaries. In the malicious case, the security provided by Protocol 3.4 depends on the security of the protocol which is used to realize the OT functionality. Universally composable OT protocols obviously result in a secure OPE protocol, since the choices of the receiver in the invocations of the OT protocol define a single point in which it can evaluate the polynomial. Invoking an information-theoretic OT protocol results in an OPE protocol which provides information-theoretic security for the sender. In this case the receiver cannot learn more than a single point of the polynomial, but it might not be possible to extract it. Finally, it is also possible to use OT protocols that provide computational security for the sender, where the receiver's input is extractable if the OT invocations are done sequentially. In this case, however, the protocols must be invoked sequentially in order to enable extraction of the receiver's input from the OPE protocol.

4. Applications of oblivious evaluation of polynomials.

There are two types of applications in which oblivious polynomial evaluation is useful. The first is where the receiver obtains a value from a k-wise independent space. For many applications such values are as good as truly random or pseudorandom values. The second type of application is of cryptographic protocols in which users must obtain a value of a polynomial held by another party without revealing their input. We give examples of both types of applications below. We also give a short description of applying the protocols of this paper to obliviously computing $g^{P(x)}$, where P is a polynomial.

4.1. Obliviously computing a polynomial in the exponent.

In some scenarios it might be required to let the receiver compute the value of $g^{P(x)}$, where g is a generator of some group, P is known to the sender and x is known to the receiver. This computation is likely to be useful for cryptographic protocols that are based on the Diffie–Hellman assumption [19].

All the protocols described in this paper can be readily applied for this computation. The only change is for the sender to provide the receiver with values of $g^{R(x)} = g^{Q(x,S(x))}$, instead of values of $R(x) = Q(x, S(x))$. The receiver needs to interpolate these values to compute $g^{R(0)} = g^{P(\alpha)}$. She can do so by computing

$$g^{R(0)} = g \sum_{i=1}^{d_R+1} \lambda_i \cdot R(i) = \prod_{i=1}^{d_R+1} g^{\lambda_i \cdot R(i)} = \prod_{i=1}^{d_R+1} (g^{R(i)})^{\lambda_i},$$

where the values $\{\lambda_i\}_{i=1}^{d_R+1}$ are the appropriate Lagrange coefficients.

4.2. Comparing information without leaking it.

Imagine two parties, A and B. Each of the parties holds a particular name (e.g., of a "suspect" from a small group of people). The two parties would like to check whether they both have the same input, under the condition that if the inputs are different they do not want to reveal any information about them (except for the fact that they are different). The main obstacle in designing a protocol for this problem is that the domain of inputs, e.g., names, is probably small enough to enable a brute force search over all possible inputs. This problem was thoroughly discussed by Fagin, Naor, and Winkler [22], with subsequent constructions by Crépeau and Salvail [16].

To specify the function more accurately, we require that if the parties' inputs are (α, β), respectively, then their outputs are $(1, 1)$ if $\alpha = \beta$, and $(0, 0)$ otherwise. In the case of malicious parties we relax the requirement and say that while for $\alpha = \beta$ the outputs can be arbitrary (since a malicious party can always change its input or output), in the case of inputs $\alpha \neq \beta$, the output of the honest party must be 0. Namely, the malicious party cannot convince the honest party that the inputs are equal. (This requirement is natural in the context of password verification, discussed below, in which if one party does not provide the right password then the other party must reject.)

Since we cannot prevent a malicious party from choosing or guessing its input, we refer to the common security notion of simulation: for every behavior of the malicious party it should be possible to construct a simulator which has access to the trusted third party (TTP) in the ideal model (but has no access to the input of the honest party) and can interact with the malicious party. The simulator should simulate the output distribution of the two parties, where the output of the honest party satisfies the requirement above. (In our case, the simulator will operate by extracting the input that the malicious party provides to the OPE and sending it as an input to the TTP.)

Oblivious evaluation of linear polynomials can be used to construct a very simple solution to the comparison of information problem.

PROTOCOL 4.1 (privacy preserving comparison of information).

(i) *Input: Denote A's input as α and B's input as β.*

(ii) *Party A generates a random* linear *polynomial $P_A(\cdot)$.*

(iii) *Party B generates a random* linear *polynomial $P_B(\cdot)$.*

(iv) *The parties execute the oblivious polynomial evaluation twice, switching roles.*

(i) *In the first invocation, A obliviously learns a value of B's polynomial (it should choose to learn $P_B(\alpha)$).*

(ii) *In the second invocation, B obliviously learns a value of A's polynomial (it should choose to learn $P_A(\beta)$).*

(iii) *The parties compute and compare the two values, $P_A(\alpha)+P_B(\alpha)$ (computed by A), and $P_A(\beta)+P_B(\beta)$ (computed by B).*

(iv) *If these values are equal then each party outputs 1. Otherwise the parties output 0. (If $\alpha=\beta$, then the two values are the same, otherwise they are different with probability $1/|\mathcal{F}|$, where $\mathcal{F}$ is the field over which the polynomial is defined.)*

For semi-honest parties, privacy is preserved since the parties compare values of the function $P_A(x)+P_B(x)$ that has the following properties (which are trivial to prove):

(i) The function is pairwise independent.

(ii) Each party only computes $P_A(x)+P_B(x)$ once.

(iii) Each party computes $P_A(x)+P_B(x)$ without revealing x to the other party.

In the case of malicious parties, the proof is simple if the protocol uses an OPE protocol which enables the extraction of the receiver's input. In this case, given a TTP which computes the function in the ideal model, we can simulate the joint distribution of the malicious party and the output of the other party: assume that Alice is malicious. We extract her input α from her invocations of the OPE protocol and provide α to the TTP. If the answer is 1, we continue the protocol by evaluating her polynomial at $\beta=\alpha$, sending the value $P_A(\alpha)+P_B(\alpha)$ to the comparison, and fixing Bob's output based on the result of the comparison. If the answer of the TTP is 0 then we provide a random value to the comparison. (Note that in the case of a malicious Bob, who computes a value of Alice's polynomial *after* letting her evaluate his, we cannot extract Bob's input before evaluating $P_B()$. We can, however, execute the OPE of Bob's polynomial twice in the simulation, learn $P_B()$ completely, and then be able to compute any value of $P_B()$ and use it to provide the right value to the comparison.)

Note that for all OPE protocols it holds that if the evaluation point does not match the input of the honest party, then with high probability the output of the protocol is 0. If the OPE protocol does not enable extraction, then we don't know, however, how to extract the evaluation point. (It might even be possible that the malicious party has some computational representation of the other party's input such that it can evaluate the OPE in the right point but does not have an explicit knowledge of the value of its input.)

Application to passwords. The protocol can serve as a basis for a mutually authenticated key exchange based on (possibly weak) passwords. Consider a user who wishes to log in to a remote server over an insecure network. She does not want to send her password in the clear, and is not even certain 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 PKI (public key infrastructure), and that the user does not carry with her a public key of the remote server. There have been several initial solutions to this problem (see, e.g., [5, 47]), for which it seems that a security proof must postulate 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 using Protocol 4.1. Therefore, if it is assumed that there is no active adversary, and an adversary only listens to the communication between the two parties and then tries to impersonate the user, then Protocol 4.1 can be used for password authentication. Furthermore, it can be used to generate a session key for the two parties, whose entropy does not depend on the entropy of the passwords.

In more detail, each of the parties, the user and the remote server, chooses a random linear polynomial and (obliviously) computes 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 the adversary can also be active, i.e., change the communication sent between the two parties, then the above protocol is insufficient. Although the adversary cannot decrypt the messages sent between the parties (e.g., in the invocations of the oblivious transfer protocol), it can change them and cause the output of the oblivious transfer protocol to be different, but related, to its legitimate output. The adversary can use this feature to attack different invocations of the protocol that are being executed in parallel. Our protocol can serve as a basis for a protocol that prevents such attacks, which must address delicate issues such as the nonmalleability [18] of the oblivious transfer protocols. Elaborate definitions and constructions of such protocols were given in subsequent work (see, e.g., [30, 55]).

4.3. Secure computation using randomizing polynomials. Ishai and Kushilevitz showed that secure evaluation of arbitrary functions can be reduced to secure evaluation of (perhaps several) degree-3 polynomials [36]. The inputs of these polynomials are the input variables of the original function and additional variables which must take random values. The size of the polynomials is quadratic in the size of a branching program computing the original function (therefore, this approach is efficient for functions that have an efficient branching program representation).

An examination of the structure of the polynomials generated by the constructions in [36] shows that each monomial is of the form $r_i x_j r_k$, where x_j is an original input variable and r_i and r_k are variables which must be assigned random values. Each of the variables is assigned to a party that must instantiate it with its corresponding input value (in the case of x_j) or with a random value (in the case of r_j and r_k). In the case of secure two-party computation, the computation of the randomizing polynomials can be easily reduced to OPE of *linear* polynomials. Namely, the sender assigns values to each of the variables that it controls and consequently each monomial becomes either a constant or a linear polynomial of an input known to the receiver. The two parties then do a shared output OPE for each of these polynomials and compute the sum of all the shares they learned.

The original work of [36] reduced the computation of the OPE to generic secure computation (the focus of that work was on information-theoretic security). However, a more efficient solution could use the linear OPE described in this paper. Furthermore, this results in a strongly polynomial secure computation protocol for a more general class of functions—namely, functions with efficient branching program representation.

4.4. Anonymous initialization for metering. A scheme for efficient and secure metering of clients' visits to servers was suggested in [50]. This scheme involves *clients* which send requests to *servers*, and a trusted, off-line, *audit authority.* Servers prove to the audit authority that they fulfilled requests (e.g., served web pages) of a certain number of clients. On a very high level, the operation of the scheme is as follows:

(i) The audit authority generates a random bivariate polynomial $P(x, y)$, where the degree of x is $k - 1$.

(ii) Each client u receives from the audit authority initialization data, which contains a univariate polynomial $P(u, \cdot)$.

(iii) When u sends a request to a server S (e.g., visits its web site), it sends it the value $P(u, S)$.

(iv) After k requests of different clients, S can interpolate the value $P(0, S)$ which serves as a proof for serving k requests.

A modification of this scheme described in [50] preserves the anonymity of clients towards servers. However, if the audit agency cooperates with a server S they are able to identify clients which access the server, since the audit agency knows which polynomial is owned by every client. Combining this scheme with oblivious polynomial evaluation enables the client to learn the initialization data from the audit agency without revealing to the agency which initialization value (i.e., polynomial) was learned by the client. If this method is employed, then even a coalition of the audit agency and a server is not able to identify the client.

4.5. Anonymous coupons. Consider the following scenario. An organization wants to set up an *anonymous* complaint box for its personnel, but would like to ensure that each person complains at most once (for example, if there are ten complaints about the quality of the coffee machine, they should be from ten different people and not ten complaints of the same person). A solution for this problem is to give each person an anonymous coupon that he or she should attach to any submitted complaint. When a complaint is received the coupon is checked for validity and freshness (namely, it is verified that it is a valid coupon that was not previously used).

Anonymous coupons can be constructed using oblivious polynomial evaluation.[2] The coupon manager sets up a polynomial P of degree d. Each person A obtains a coupon by choosing a random secret value R_A and obliviously computing $P(R_A)$, using an oblivious polynomial evaluation protocol in which the coupon manager is the sender. The coupon is the pair $\langle R_A, P(R_A)\rangle$. It is easy for the coupon manager to verify that a coupon $\langle X, Y\rangle$ is valid by simply testing whether $Y = P(X)$. The coupon manager also keeps a list of the coupons that were received and compares each new coupon to that list in order to verify that it was not used before.

[2]Another construction for this problem can be based on using blind signatures: Each user prepares in advance coupons by letting the central server sign them using a blind signature scheme (e.g., Chaum's scheme [12], or the scheme of Juels et al. that can be based on standard hardness assumptions [37]). A complaint must be accompanied by a signed coupon. The server verifies the signature, and also verifies that this coupon has not been used before.

The system is secure against a coalition of corrupt users that attempt to generate fake coupons, as long as the coalition has at most d different coupons. The degree of the polynomial, d, should therefore be set in accordance with the potential size of a corrupt coalition of users.

A corrupt coupon manager might attempt to identify the owners of coupons by using a different polynomial for each person. Namely, for every user A it might use a different polynomial P_A for the oblivious evaluation protocol. When it later receives a coupon $\langle X, Y\rangle$ it attempts to identify its owner by checking for which polynomials P_A it holds that $Y = P_A(X)$. This attack can be prevented by ensuring that the sender uses the same polynomial in all oblivious evaluation protocols, for example, using the verifiable secret sharing techniques of Feldman [23] or Pedersen [57].

Acknowledgments. We would like to thank Sanjeev Arora, Daniel Bleichenbacher, Dan Boneh, Oded Goldreich, Yuval Ishai, Amit Klein, Ronitt Rubinfeld, Madhu Sudan, and the anonymous referees that reviewed this paper.

REFERENCES

[1] B. Aiello, Y. Ishai, and O. Reingold, in *Priced oblivious transfer: How to sell digital goods*, Advances in Cryptology–Eurocrypt 2001, LNCS 2045, Springer-Verlag, Berlin, 2001, pp. 119–135.

[2] D. Beaver, *Correlated pseudorandomness and the complexity of private computations*, in Proceedings of the 28th ACM Symposium on the Theory of Computing, 1996, pp. 479–488.

[3] D. Beaver and J. Feigenbaum, *Hiding instances in multioracle queries*, in Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, 1990, pp. 37–48.

[4] M. Bellare and S. Micali, *Non-interactive oblivious transfer and applications*, in Advances in Cryptology–CRYPTO 1989, Springer-Verlag, LNCS 435, 1990, pp. 547–557.

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

[6] D. Bleichenbacher and P. Nguyen, *Noisy polynomial interpolation and noisy Chinese remaindering*, in Advances in Cryptology–EUROCRYPT 2000, Springer-Verlag, LNCS 1807, pp. 53–69.

[7] D. Boneh, *Finding smooth integers in short intervals using CRT decoding*, J. Comput. System Sci., 64 (2002), pp. 768–784.

[8] G. Brassard, C. Crépeau, and J.-M. Robert, *Information theoretic reduction among disclosure problems*, in Proceedings of the 27th FOCS, IEEE, 1986, pp. 168–173.

[9] G. Brassard, C. Crépeau, and J.-M. Robert, *All-or-nothing disclosure of secrets*, in Advances in Cryptology–CRYPTO 1986, LNCS 263, Springer Verlag, 1987, pp. 234–238.

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

[11] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, *Universally composable two party computation*, in Proceedings of the 34th ACM Symposium on the Theory of Computing, 2002, pp. 494–503.

[12] D. Chaum, *Blind signatures for untraceable payments*, CRYPTO 1982, Plenum Press, New York, pp. 199–203.

[13] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, *Private information retrieval*, J. ACM, 45 (1998), pp. 965–981. Earlier version appeared in Proceedings of the 36th FOCS, IEEE, 1995, pp. 41–50.

[14] D. Coppersmith and M. Sudan, *Reconstructing curves in three (and higher) dimensional space from noisy data*, in Proceedings of the 35th ACM Symposium on the Theory of Computing, 2003, pp. 136–142.

[15] C. Crépeau and J. Kilian, *Achieving oblivious transfer using weakened security assumptions*, in Proceedings of the 29th FOCS, IEEE, 1988, pp. 42–52.

[16] C. Crépeau and L. Salvail, *Oblivious verification of common string*, CWI Quarterly, special issue for the Crypto Course 10th Anniversary, 8 (1995), pp. 97–109.

[17] Y. Dodis and S. Micali, *Lower bounds for oblivious transfer reductions*, Advances in Cryptology–EUROCRYPT 1999, LNCS, Springer-Verlag, 1999, pp. 42–55.

[18] D. Dolev, C. Dwork, and M. Naor, Non-malleable cryptography, SIAM J. Comput., 30 (2000), pp. 391–437. Earlier version appeared in Proceedings of the 23rd ACM Symposium on Theory of Computing, 1991, pp. 542–552.
[19] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, 22 (1976), pp. 644–654.
[20] P. Elias, *List decoding for noisy channels*, TR 335, Research Laboratory for Electronics, MIT, Cambridge, MA, 1957.
[21] S. Even, O. Goldreich, and A. Lempel, *A randomized protocol for signing contracts*, Commun. of ACM 28 (1985), pp. 637–647.
[22] R. Fagin, M. Naor, and P. Winkler, *Comparing information without leaking it*, Commun. of ACM 39 (1996), pp. 77-85.
[23] P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, in Proceedings of the 28th FOCS, IEEE, 1987, pp. 427–437.
[24] J. Garay and P. Mackenzie, *Concurrent oblivious transfer*, in Proceedings of the 41st FOCS, IEEE, 2000, pp. 314–324.
[25] J. Garay, P. MacKenzie, and K. Yang, *Efficient and universally composable committed oblivious transfer and applications*, in Proceedings of the 1st Theory of Cryptography Conference, LNCS 2951, Springer-Verlag, 2004, pp. 297–316.
[26] N. Gilboa, *Two party RSA key generation*, in Advances in Cryptology–CRYPTO 1999, LNCS 1666, Springer-Verlag, Berlin, 1999, pp. 116–129.
[27] N. Gilboa, *Topics in private information retrieval*, DSc. dissertation, Technion - Israel Institute of Technology, Haifa, Israel, 2000.
[28] O. Goldreich, *The Foundations of Cryptography Basic Tools*, Cambridge University Press, New York, 2001.
[29] O. Goldreich, *The Foundations of Cryptography - Basic Applications*, Cambridge University Press, New York, 2004.
[30] O. Goldreich and Y. Lindell, *Session key generation using human passwords only*, in Advances in Cryptology–CRYPTO 2001, LNCS 2139, Springer-Verlag, Berlin, 2001, pp. 408–432.
[31] O. Goldreich, S. Micali, and A. Wigderson, *How To Play Any Mental Game*, in Proceedings of the 19th ACM Conference on Theory of Computing, New York, New York, 1987, pp. 218–229.
[32] O. Goldreich, M. Sudan, and R. Rubinfeld, *Learning polynomials with queries: The highly noisy case*, in Proceedings of the 36th FOCS, IEEE, 1995, pp. 294–303.
[33] O. Goldreich and R. Vainish, *How to solve any protocol problem - an efficiency improvement*, in Advances in Cryptology–CRYPTO 1987, LNCS 293, Springer-Verlag, Berlin, 1988, pp. 297–316.
[34] V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and algebraic-geometric codes*, IEEE Trans. Inform. Theory, 45 (1999), pp. 1757–1767. Earlier version appeared in 39th FOCS, 1998.
[35] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, *Extending oblivious transfers efficiently*, in Advances in Cryptology–CRYPTO 2004, LNCS 2729, Springer-Verlag, Berlin, 2003, pp. 145–161.
[36] Y. Ishai and E. Kushilevitz, *Randomizing polynomials: A new representation with applications to round-efficient secure computation*, ini Proceedings of the 41st FOCS, IEEE, 2000, pp. 294–304.
[37] A. Juels, M. Luby, and R. Ostrovsky, *Security of blind signatures*, in Advances in Cryptology–CRYPTO 1997, LNCS 1294, Springer-Verlag, Berlin, 1997, pp. 150–164.
[38] R. Impagliazzo and S. Rudich, *Limits on the provable consequences of one-way permutations,* in Proceedings of the ACM Symposium on the Theory of Computing, 1989, pp. 44–61.
[39] Y. T. Kalai, *Smooth projective hashing and two-message oblivious transfer*, in Advances in Cryptology–EUROCRYPT 2005, Springer-Verlag, LNCS 3494, 2005, pp. 78–95.
[40] A. Kiayias and M. Yung, *Cryptographic hardness based on the decoding of reed-solomon codes*, in Proceedings of ICALP 2002, LNCS 2380, Springer-Verlag, Berlin, 2002, pp. 232–243.
[41] A. Kiayias and M. Yung, *Directions in polynomial reconstruction based cryptography (invited survey)*, IEICE Trans., E87-A (2004), pp. 978–985.
[42] J. Kilian, *Founding cryptography on oblivious transfer*, 20th STOC, (1988), pp. 20–31.
[43] E. Kushilevitz and R. Ostrovsky, *Replication is not needed: single database, computationally-private information retrieval*, in Proceedings of the 38th FOCS, IEEE, 1997, pp. 364–373.
[44] A. K. Lenstra, H. W. Lenstra, and L. Lovasz, *Factoring polynomials with rational coefficients*, Mathematische Ann., 261 (1982), pp. 513–534.

[45] Y. Lindell and B. Pinkas, *Privacy preserving data mining*, J. Cryptology, 15 (2002), pp. 177–206.

[46] R. J. Lipton, *Efficient checking of computations*, in Proceedings of STACS '90, 1990, pp. 207–215.

[47] S. Lucks, *Open key exchange: How to defeat dictionary attacks without encrypting public keys*, in Proceedings of Security Protocol Workshop '97, LNCS 1361, Springer-Verlag, Berlin, 1997, pp. 79–90.

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

[49] M. Naor and K. Nissim, *Communication preserving protocols for secure function evaluation*, in Proceedings of the 33rd ACM Symposium on the Theory of Computing, 2001, pp. 590–599.

[50] M. Naor and B. Pinkas, *Secure and efficient metering*, in Advances in Cryptology–EUROCRYPT 1998, Springer-Verlag, LNCS 1403, 1998, pp. 576–590.

[51] M. Naor and B. Pinkas, *Oblivious transfer and polynomial evaluation*, in Proceedings of the 31st ACM Symposium on the Theory of Computing, 1999, pp. 245–254.

[52] M. Naor and B. Pinkas, *Efficient oblivious transfer protocols*, in Proceedings of the SIAM Symposium on Discrete Algorithms, 2001, pp. 448–457.

[53] M. Naor and B. Pinkas, *Computationally secure oblivious transfer*, J. Cryptology, 18 (2005), pp. 1–35.

[54] M. Naor and O. Reingold, *Number-theoretic constructions of efficient pseudo-random functions*, in Proceedings of the 38th FOCS, IEEE, 1997, pp. 458–467.

[55] M.-H. Nguyen and S. Vadhan, *Simpler session-key generation from short random passwords*, in Proceedings of the Theory of Cryptography Conference, LNCS 2951, Springer-Verlag, Berlin, 2004, pp. 428–445.

[56] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, in *Advances in Cryptology–EUROCRYPT '99*, LNCS 1592, Springer-Verlag, (1999), pp. 223–238.

[57] T. P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Crypto '91, LNCS 576, (1991), pp. 129–140.

[58] M. O. Rabin, *How to exchange secrets by oblivious transfer*, Technical memo TR-81, Aiken Computation Laboratory, Harvard University, Cambridge, MA, 1981.

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

[60] A. C. Yao, *How to generate and exchange secrets*, in Proceedings of the 27th FOCS, IEEE, 1986, pp. 162–167.