Oblivious Evaluation of Multivariate Polynomials

Tamir Tassa * Ayman Jarrous † Yonatan Ben-Ya'akov ‡

Abstract

One of the fundamental problems of multi-party computation is Oblivious Polynomial Evaluation. In that problem, that was introduced by Naor and Pinkas, Alice has a polynomial $P(x)$ and Bob has a point α. The goal is to allow Bob to compute $P(\alpha)$ so that Alice remains oblivious of α and Bob of $P(x)$, apart from what is implied by $P(\alpha)$ and α. We introduce the multivariate version of this problem, where x and α are vectors, and offer an efficient secure protocol. In addition, we discuss several applications that may be solved efficiently using oblivious multivariate polynomial evaluation, such as private linear algebraic computations and private support vector machines (SVM).

MSC 2010 classification. 94A60 Cryptography

Keywords. Cryptography, Privacy, Secure two-party computation, Secure function evaluation, Oblivious transfer, Multivariate polynomials

1 Introduction

Consider a setting in which there are several parties, $P_1, \ldots, P_n$, where each party P_i holds a private value x_i. The goal is to compute the value $f(x_1, \ldots, x_n)$, where f is some publicly known function of n variables, so that each party does not learn anything about the private inputs of the other parties, except the information that is implied by his own input and the output result $f(x_1, \ldots, x_n)$. One way of accomplishing this task is by delegating the input values x_i to a trusted third party that can perform the computation on the inputs that he receives from the n parties and return the computation result. However, when such a trusted third party is not available, it is necessary to accomplish this task by a suitable protocol that the participating parties have to execute among themselves. Such protocols are called Multi-Party Computation (MPC hereinafter) protocols.

One of the basic primitives of MPC is Oblivious Transfer (OT). It was first introduced by Rabin [17]. Rabin considered a setting that involves two parties — Alice and Bob. Alice has an input bit and Bob wants to learn it. The goal is that Bob will receive that bit with probability

*Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel. tamirta@openu.ac.il

†Department of Computer Science, Haifa University, Israel. ayman@jarrous.net

‡Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel. yonatan1984@gmail.com

1/2, and will receive nothing with probability 1/2, while Alice remains oblivious of which of the two events happened. A different variant of OT, called "chosen 1-out-of-2 oblivious transfer", was later introduced and discussed by Even, Goldreich and Lempel [6]. In that setting, Alice has two bits, b_0 and b_1, and Bob has a selection bit s. The goal is for Bob to receive b_s and remain oblivious of b_{1-s} while Alice remains oblivious of s. In other words, the goal is to allow Bob to compute the following function of the joint inputs: $f(b_0, b_1; s) := (1-s)b_0 + sb_1$.

The generic problem of secure two-party computation was solved by Yao [19]. He gave a solution that accomplishes secure computation for every function f which can be represented by a binary or an algebraic circuit. His solution is based on the assumption that factoring integers is intractable. Later, Goldreich and Vainish [11] showed that the existence of OT is sufficient for this task. Kilian [14] showed, in a constructive manner, that OT is necessary and sufficient for general oblivious function evaluation, that is secure against *malicious* parties. The problem with these generic solutions is that their computational and communication complexities are very high. The aim of further studies in this field is to find more efficient solutions for specific problems of MPC.

One such problem of MPC is Oblivious Polynomial Evaluation, or OPE. It involves two parties: A sender, who has a univariate polynomial P over a finite field $\mathbb{F}$, and a receiver, who has a point $\alpha \in \mathbb{F}$. The goal is for the receiver to learn the value of $P(\alpha)$, without learning any other information about P, while the sender has to learn no information about α. This problem was introduced and solved by Naor and Pinkas in [15].

In this study we present the multivariate version of OPE, where P is an r-variate polynomial over a finite field $\mathbb{F}$ and $\boldsymbol{\alpha} \in \mathbb{F}^r$. The goal remains the same: The receiver wishes to learn the value of $P(\boldsymbol{\alpha})$, without learning any other information about P, while the sender needs to remain oblivious of the value of $\boldsymbol{\alpha}$. The extension to the multivariate case is interesting for its own sake, but also because it has applications that cannot be achieved by univariate OPE, e.g., private linear algebraic computations, or private support vector machines.

The paper is organized as follows. Section 2 sets the ground for the discussion that follows. Our main results — protocols for oblivious evaluation of multivariate polynomials and their analysis — are presented in Section 3. We conclude in Section 4 with applications of oblivious evaluation of both univariate and multivariate polynomials.

2 Preliminaries

In Section 2.1 we recall the basic definitions regarding the security of MPC protocols, and in Section 2.2 we provide a quick overview of oblivious transfer. Our discussion in these two sections is informal; the interested readers are referred to Goldreich [9] for a more thorough and formal discussion. In Section 2.3 we define the problem of Oblivious Multivariate Polynomial Evaluation (OMPE). Then, we discuss in Section 2.4 intractability assumptions that will play a significant role in the solution of that problem.

2.1 Security of MPC protocols

When analyzing the security of a protocol, it is customary to compare what an adversary can do in a real execution of the protocol to what he can do in an ideal execution scenario. In an ideal scenario, the participating parties send their inputs to a trusted third party that implements

the functionality of the protocol using the inputs that were received from all parties and then he returns to each of the parties their corresponding outputs.

To prove that the protocol is secure, it is needed to show that what an adversary can do in a real execution (namely, what he can cause or the information that he may extract from his view) is indistinguishable from what he can do in an ideal scenario. Towards that end, we build a simulator that simulates the execution of the protocol, where on one side, the simulator interacts with the adversary in the real world, and on the other side, it interacts with the trusted third party in the ideal world. In the simulation, we show that the interaction of the adversary with the simulator is indistinguishable from his interaction with the other parties in a real execution.

We consider two typical kinds of adversaries: semi-honest and malicious.

- *A semi-honest adversary* is an adversary that performs the steps of the protocol but, at the same time, tries to use the information that he gains during the execution of the protocol in order to deduce additional knowledge about the inputs of the other parties. A secure multi-party protocol guarantees that what the adversary may learn during a real execution is no more than what he can learn in an ideal scenario, namely, no more than what is implied by his own input and output.

- *A malicious adversary* can be viewed as a program that controls a number of the parties during the execution of the protocol. Such an adversary does not necessarily follow the steps of the protocol and he might behave arbitrarily, e.g. by even stopping performing his part in the protocol. When designing a secure protocol against a malicious adversary, the goal is to build a protocol that prevents the adversary from gaining forbidden information about the other parties' inputs, and from interrupting the execution of the protocol. It should be noted that there are some malicious conducts that cannot be prevented:

  – A malicious adversary may change his input or send incorrect values.
  – He may reject participating in some steps of the protocol.
  – He may stop participating in the execution of the protocol.

  Even though such conducts may affect that fairness of the protocol, it is important to maintain the ability of the other parties to detect such conducts and, consequently, recover and continue the execution of the protocol.

The hybrid model. When building a secure protocol that uses other secure subprotocols that are fully simulatable, it suffices to analyze the security of the entire protocol in the hybrid model [5]; namely, we may assume that the subprotocols are executed by a trusted third party. Since our proposed OMPE protocol is based on oblivious transfer (OT) subprotocols, we may analyze the security of the OMPE protocol in the hybrid model, by assuming that the OT subprotocols are executed by a trusted third party.

2.2 Oblivious Transfer (OT)

OT is a secure two-party protocol that was introduced by Rabin [17] and was afterwards extended by Even, Goldreich and Lempel [6] and Brassard, Crépeau and Robert [2]. In an OT protocol there are two parties, a sender with messages $\{m_1, \ldots, m_N\}$, and a receiver with a selection index

$i \in [N] := \{1, \ldots, N\}$. By the end of the protocol, the receiver learns m_i but learns nothing about the other messages, m_j, $j \in [N] \setminus \{i\}$. The sender, on the other hand, learns no information about the selection index i. OT protocols that are secure against malicious parties were presented recently in [4, 12, 16].

It was shown that the OT protocol is complete, i.e., that any secure protocol can be implemented using OT [10, 14].

2.3 Oblivious Multivariate Polynomial Evaluation (OMPE)

The OMPE problem concerns two parties, a receiver and a sender. The specifications of the problem are as follows.

- **Input:**
  - sender: An r-variate polynomial of degree (at most) d over a finite field $\mathbb{F}$,

$$P(\mathbf{y}) = \sum_{0 \le |\mathbf{k}| \le d} b_{\mathbf{k}} \mathbf{y}^{\mathbf{k}},$$

    where[1] $\mathbf{y} = (y_1, \ldots, y_r)$ is the vector of variables, $\mathbf{k} = (k_1, \ldots, k_r)$ are the exponent vectors, $\mathbf{y}^{\mathbf{k}} = \prod_{i=1}^{r} y_i^{k_i}$, $|\mathbf{k}| = \sum_{i=1}^{r} k_i$ is the order of the monomial $\mathbf{y}^{\mathbf{k}}$, and $b_{\mathbf{k}}$ are the scalar coefficients in $\mathbb{F}$.
  - receiver: A point $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r) \in \mathbb{F}^r$.
- **Output:**
  - sender: nothing.
  - receiver: $P(\boldsymbol{\alpha})$.

It is required that during the run of the protocol, the sender will acquire no information regarding the value of $\boldsymbol{\alpha}$ and that the receiver will acquire no more information about the polynomial P other than what is inferred by the output $P(\boldsymbol{\alpha})$.

2.4 The intractability assumption

The solution of Naor and Pinkas of the univariate OPE problem relied on either one of two intractability assumptions that were based on *noisy* polynomial reconstruction problems. Our solution in the multivariate case relies on corresponding multidimensional versions of those assumptions.

First, we define the following vector polynomial reconstruction problems. (Whenever we speak below on a polynomial function $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$ of degree at most k we mean that $\mathbf{S}(x) = (S_1(x), \ldots, S_r(x))$ and $S_i(x)$, $1 \le i \le r$, are polynomials of degree at most k.)

The vector polynomial reconstruction problem

INPUT: Integers r, k, n, and N pairs $\{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{F}$ and $\mathbf{y}_i \in \mathbb{F}^r$.

[1] Hereinafter, bold face letters denote vectors and the corresponding regular letters denote their components.

OUTPUT: A polynomial function $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$ of degree at most k, such that $\mathbf{S}(x_i) = \mathbf{y}_i$ for at least n indices $i \in [N]$.

The vector polynomial list reconstruction problem

INPUT: Integers r, k, n, and N pairs $\{(x_i, \mathbf{y}_i)\}_{i=1}^N$, where $x_i \in \mathbb{F}$ and $\mathbf{y}_i \in \mathbb{F}^r$.

OUTPUT: All polynomial functions $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$ of degree at most k, such that $\mathbf{S}(x_i) = \mathbf{y}_i$ for at least n indices $i \in [N]$.

The intractability assumption, which is defined below, is the formalization of the idea that given an input to the vector polynomial reconstruction problem, the value of the polynomial vector at $x = 0$ is pseudo-random.

Definition 2.1 *Let r, k, n, m be integers, $\mathbb{F}$ be a finite field, and $\boldsymbol{\alpha}$ be a fixed vector in $\mathbb{F}^r$. Then $A_{n,m,r}^{k,\boldsymbol{\alpha}}$ denotes the probability distribution of the sets of pairs $\{(x_i, \mathbf{y}_i)\}_{i=1}^N$ that are generated in the following manner:*

1. *Pick at random a polynomial function $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$ of degree at most k such that $\mathbf{S}(0) = \boldsymbol{\alpha}$.*
2. *Generate a random set of nm distinct points $x_1, \ldots, x_{nm} \in \mathbb{F} \setminus \{0\}$.*
3. *Choose a random subset $T \subset [nm] = \{1, \ldots, nm\}$ of n distinct indices.*
4. *Set $\mathbf{y}_i = \mathbf{S}(x_i)$ for all $i \in T$ and $\mathbf{y}_i$ to be a random value in $\mathbb{F}^r$ otherwise.*
5. *Output the set $\{(x_i, \mathbf{y}_i)\}_{i=1}^{nm}$.*

Finally, we let $\mathcal{A}_{n,m,r}^{k,\boldsymbol{\alpha}}$ denote the random variable that is chosen according to the probability distribution $A_{n,m,r}^{k,\boldsymbol{\alpha}}$.

Let $\ell \in \mathbb{N}$ be a security parameter. Let $n(\ell), m(\ell), k(\ell)$ be any integer-valued polynomially-bounded functions of ℓ that define the values of the parameters n, m, k that appear in Definition 2.1. In addition, let $F(\ell)$ be any integer-valued polynomially-bounded function of ℓ, and $\mathbb{F}(\ell)$ be a field whose elements may be represented by $F(\ell)$ bits. Finally, let $\boldsymbol{\alpha}(\ell)$ and $\boldsymbol{\alpha}'(\ell)$ be any functions with values in $\mathbb{F}(\ell)^r$. For any fixed $r \geq 1$, we define the probability ensembles $\{\mathcal{A}^r(\ell)\}_{\ell \in \mathbb{N}}$ and $\{\mathcal{A}'^r(\ell)\}_{\ell \in \mathbb{N}}$ to be such that for every ℓ, $\mathcal{A}^r(\ell)$ is the random variable $\mathcal{A}_{n,m,r}^{k,\boldsymbol{\alpha}}$ and $\mathcal{A}'^r(\ell)$ is the random variable $\mathcal{A}_{n,m,r}^{k,\boldsymbol{\alpha}'}$, where $n = n(\ell), m = m(\ell), k = k(\ell), \mathbb{F} = \mathbb{F}(\ell), \boldsymbol{\alpha} = \boldsymbol{\alpha}(\ell)$, and $\boldsymbol{\alpha}' = \boldsymbol{\alpha}'(\ell)$.

The intractability assumption. *The probability ensembles $\{\mathcal{A}^r(\ell)\}$ and $\{\mathcal{A}'^r(\ell)\}$ are computationally indistinguishable; i.e., there does not exist a probabilistic polynomial-time algorithm that can distinguish between samples from $\{\mathcal{A}^r(\ell)\}$ and $\{\mathcal{A}'^r(\ell)\}$ with non-negligible success probability.*

The number of variables r is a constant in our discussion. It is easy to see that by setting $r = 1$ we recover the first intractability assumption of Naor and Pinkas in [15, Section 2.2]. We claim that if the intractability assumption holds for $r = 1$, it also holds for every fixed $r \geq 1$.

Theorem 2.1 *For any fixed $r \in \mathbb{N}$, $\{\mathcal{A}^r(\ell)\}$ and $\{\mathcal{A}'^r(\ell)\}$ are computationally indistinguishable if $\{\mathcal{A}^1(\ell)\}$ and $\{\mathcal{A}'^1(\ell)\}$ are computationally indistinguishable.*

Proof. Assume that $n = n(\ell)$, $m = m(\ell)$, $k = k(\ell)$, and $\mathbb{F} = \mathbb{F}(\ell)$, as described earlier, and let $\boldsymbol{\alpha} = \boldsymbol{\alpha}(\ell)$ and $\boldsymbol{\alpha}' = \boldsymbol{\alpha}'(\ell)$ be any two functions with values in $\mathbb{F}(\ell)^r$. For every $1 \leq i \leq r$, define $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i(\ell) = (\alpha'_1(\ell), \ldots, \alpha'_i(\ell), \alpha_{i+1}(\ell), \ldots, \alpha_r(\ell))$. Finally, we let $\mathcal{A}^r_i(\ell)$ denote the random variable $\mathcal{A}^{k,\boldsymbol{\alpha}_i}_{n,m,r}$.

We need to prove that there exists no probabilistic polynomial-time algorithm $D_{\boldsymbol{\alpha},\boldsymbol{\alpha}'}$ that can distinguish between $\mathcal{A}^r_0(\ell)$ and $\mathcal{A}^r_r(\ell)$ with a non-negligible success probability. Assume that such a distinguishing algorithm $D_{\boldsymbol{\alpha},\boldsymbol{\alpha}'}$ does exist. Say the algorithm's output is 0 if it decides that the input given to it was from $\mathcal{A}^r_0(\ell)$ and 1 if it decides that the input was from $\mathcal{A}^r_r(\ell)$.

Let p_i be the probability that the output of $D_{\boldsymbol{\alpha},\boldsymbol{\alpha}'}$ is 1 given an input sampled from $\mathcal{A}^r_i(\ell)$, $0 \leq i \leq r$. Then, by the triangle inequality,

$$|p_0 - p_r| \leq \sum_{i=1}^{r} |p_{i-1} - p_i| \,. \tag{1}$$

Since we assumed that $D_{\boldsymbol{\alpha},\boldsymbol{\alpha}'}$ is a successful distinguisher between $\mathcal{A}^r_0(\ell)$ and $\mathcal{A}^r_r(\ell)$, the difference $|p_0 - p_r|$ is non-negligible. Hence, at least one of the addends on the right-hand side of (1) must be non-negligible. Namely, there exists $1 \leq i \leq r$ such that $\mathcal{A}^r_{i-1}(\ell)$ can be distinguished from $\mathcal{A}^r_i(\ell)$ by a probabilistic polynomial time algorithm with a non-negligible success probability. That is impossible since even if we told the distinguisher that the distinction between the two samples lies in the i-th component, he would not be able to distinguish between the two samples with a non-negligible success probability, since we assumed that $\{\mathcal{A}^1(\ell)\}$ and $\{\mathcal{A}'^1(\ell)\}$ are computationally indistinguishable. Therefore, a distinguishing algorithm $D_{\boldsymbol{\alpha},\boldsymbol{\alpha}'}$ does not exist. □

A different way of showing that our intractability assumption is at least as hard as the one in [15] is the following. Let $S : \mathbb{F} \to \mathbb{F}$ be a random polynomial of degree at most k and let α and α' be two distinct values in $\mathbb{F}$. For any fixed integer $r > 1$, we define $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$ to be the function $\mathbf{S}(x) = (S(x), 0, \ldots, 0)$, and corresponding two points in $\mathbb{F}^r$, $\boldsymbol{\alpha} = (\alpha, 0, \ldots, 0)$ and $\boldsymbol{\alpha}' = (\alpha', 0, \ldots, 0)$. Denote by $\{\mathcal{A}^1(\ell)\}$ and $\{\mathcal{A}'^1(\ell)\}$ the two ensembles that correspond to S, α and α' through Definition 2.1, and by $\{\mathcal{A}^r(\ell)\}$ and $\{\mathcal{A}'^r(\ell)\}$ the two ensembles that correspond to $\mathbf{S}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$. Clearly, if the latter two ensembles were distinguishable, so would be the former two ensembles. But the former two ensembles are indistinguishable, according to the intractability assumption in [15], and therefore so are the two latter ensembles.

3 A protocol for the OMPE problem

In this section we describe the protocol for the OMPE problem (Section 3.1) and then discuss its security in the case of semi-honest parties (Section 3.2) and malicious parties (Section 3.3).

3.1 The protocol

The protocol is a generalization of the protocol in [15] to the multivariate case.

1. The sender hides $P(\cdot)$ in an $(r+1)$-variate polynomial. To that end, he generates a random univariate masking polynomial, $M(x)$, of degree sd, where s is a security parameter, such

that $M(0) = 0$. Namely,

$$M(x) = \sum_{1 \le j \le sd} a_j x^j \ .$$

The sender then defines the $(r+1)$-variate polynomial:

$$Q(x, \mathbf{y}) = M(x) + P(\mathbf{y})$$

for which it holds that $Q(0, \mathbf{y}) = P(\mathbf{y})$ for all $\mathbf{y}$.

2. The receiver hides $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$ as follows: He selects r random (univariate) polynomials of degree s, $\{S_i(\cdot)\}_{1 \le i \le r}$, such that $S_i(0) = \alpha_i$, $1 \le i \le r$. Then, he defines $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$ in the following manner:

$$\mathbf{S}(x) = (S_1(x), \ldots, S_r(x)) \ .$$

The receiver's plan is to use the univariate polynomial

$$R(x) = Q(x, \mathbf{S}(x))$$

in order to learn $P(\boldsymbol{\alpha})$ without leaking information on the value of $\boldsymbol{\alpha}$. Specifically, since

$$R(0) = Q(0, \mathbf{S}(0)) = Q(0, \boldsymbol{\alpha}) = P(\boldsymbol{\alpha}) \ ,$$

if the receiver is able to recover R, he can deduce the value of $P(\boldsymbol{\alpha})$. The degree of R is sd since

$$R(x) = Q(x, \mathbf{S}(x)) = M(x) + P(\mathbf{S}(x)) = M(x) + P(S_1(x), \ldots, S_r(x))$$

and $\deg M = sd$, $\deg P = d$, and $\deg S_i = s$ for all i.

3. The receiver learns the value of R at $sd + 1$ points,

$$(x_i, R(x_i)) \ , \quad 1 \le i \le sd + 1 \ .$$

4. The receiver uses the values of R that he learned in order to interpolate R and deduce the sought-after value $R(0) = P(\boldsymbol{\alpha})$.

It remains to discuss the implementation of Step 3. It is carried out by the following sub-protocol:

(a) Let $n = sd + 1$ and $N = nm$, where m is a security parameter.

(b) The receiver selects N distinct random points $x_1, \ldots, x_N \in \mathbb{F} \setminus \{0\}$.

(c) He selects a random subset T of n indices $1 \le i_1 < \cdots < i_n \le N$.

(d) For each $i \in T$, he sets $\mathbf{y}_i := \mathbf{S}(x_i)$; for all $i \in [N] \setminus T$, $\mathbf{y}_i$ is selected randomly from $\mathbb{F}^r$.

(e) The receiver sends the N pairs $\{(x_i, \mathbf{y}_i)\}_{1 \le i \le N}$ to the sender.

(f) The sender computes $Q(x_i, \mathbf{y}_i)$ for all $i \in [N]$.

(g) The sender and receiver execute an n-out-of-N OT protocol in which the receiver learns the values $\{Q(x_i, \mathbf{y}_i) : i \in T\}$.

Note that the set of points that the receiver sends to the sender is drawn from the probability distribution $A^{s,\boldsymbol{\alpha}}_{n,m,r}$, see Definition 2.1.

3.2 Security in the case of semi-honest parties

Here we assume that both parties follow the protocol precisely. The correctness of the protocol in this case is trivial. Indeed, by the properties of OT, the receiver will learn the values of R at $n = sd+1$ points and, thus, he will be able to successfully interpolate R and calculate $R(0) = P(0)$. We proceed to prove the receiver's and sender's privacy in this setting.

Theorem 3.1 *Under the intractability assumption and the OT hybrid model assumption (i.e., that the OT protocol is secure), the OMPE protocol in Section 3.1 is secure against semi-honest adversaries.*

In proving Theorem 3.1, we analyze the security of the protocol in the hybrid model assuming that the OT protocol is executed by a trusted third party. In order to prove security, we construct simulators, SIM_R and SIM_S, that generate the view of the receiver and the sender given only their input and output in the ideal model. The proof that the protocol preserves the receiver's privacy is given in Section 3.2.1; in Section 3.2.2 we prove that it preserves the sender's privacy.

3.2.1 The protocol is secure against semi-honest senders

We need to show that the sender does not learn any information about the receiver's private input $\boldsymbol{\alpha}$. To that end, we build a simulator SIM_S that simulates the view of the sender given its input $P(\cdot)$ and output $\perp$ (empty output) in the ideal model. When simulating the protocol, we only need to simulate Step 3 of the protocol, since that is the only step during the protocol in which the sender receives information. In that step, the simulator SIM_S chooses N random pairs $\{(x_i^R, \mathbf{y}_i^R)\}_{1 \le i \le N}$ (corresponding to Steps 3(a)-3(d)) and sends them to the sender (Step 3(e)); then, both parties engage in an OT protocol, Step 3(g). Since the OT is assumed to be executed by a trusted third party, the view of the sender during the protocol consists only of the set of N pairs sent to him in Step 3(e). We proceed now to show that the view of the sender during the hybrid model protocol execution is indistinguishable from his view in a corresponding simulation.

Let $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ be any two vectors in $\mathbb{F}^r$. We consider four probability distributions of instances of the sender's view in the protocol. The differences between the four are in the choice of two things:

- The interaction of the receiver in the OT protocol can correspond to him choosing to learn either the n *correct* pairs, i.e. the pairs $(x_i, \mathbf{y}_i)$ for which $i \in T$, or to him choosing to learn n pairs sampled *at random* from the set of N pairs.
- The set of N pairs can be chosen randomly either from $\mathcal{A}_{n,m,r}^{s,\boldsymbol{\alpha}_0}$ or from $\mathcal{A}_{n,m,r}^{s,\boldsymbol{\alpha}_1}$.

We define the following probability distributions:

- $S_{OT,0}$: The sender's view if the receiver chooses to learn the n correct pairs, and the N pairs are picked from $\mathcal{A}_{n,m,r}^{s,\boldsymbol{\alpha}_0}$.
- $S_{OT,1}$: The sender's view if the receiver chooses to learn the n correct pairs, and the N pairs are picked from $\mathcal{A}_{n,m,r}^{s,\boldsymbol{\alpha}_1}$.
- $S_{R,0}$: The sender's view if the receiver chooses to learn n pairs sampled at random, and the N pairs are picked from $\mathcal{A}_{n,m,r}^{s,\boldsymbol{\alpha}_0}$.

- $S_{R,1}$: The sender's view if the receiver chooses to learn n pairs sampled at random, and the N pairs are picked from $\mathcal{A}^{s,\boldsymbol{\alpha}_1}_{n,m,r}$.

We proceed to prove that there is no probabilistic polynomial-time algorithm $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$ that enables the sender to distinguish between a view sampled from $S_{OT,0}$ and one that is sampled from $S_{OT,1}$, with a non-negligible success probability. Letting $\boldsymbol{\alpha}_0$ be the value of $\boldsymbol{\alpha}$ in a real application of the protocol, and $\boldsymbol{\alpha}_1$ be the value in a simulation of the protocol, we arrive at the conclusion that the simulation view is indistinguishable from the view in the real world. Therefore, the sender does not learn any information about the receiver's private input $\boldsymbol{\alpha}$.

Assume that a distinguishing algorithm $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$ does exist. Say the algorithm's output is 1 if it decides that the input given to it was from $S_{OT,1}$ and 0 if it decides that the input was from $S_{OT,0}$. Let $p_{OT,0}$ be the probability that the output of $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$ is 1 given an input sampled from $S_{OT,0}$. We similarly define $p_{OT,1}$, $p_{R,0}$ and $p_{R,1}$. By the triangle inequality,

$$|p_{OT,0} - p_{OT,1}| \leq |p_{OT,0} - p_{R,0}| + |p_{R,0} - p_{R,1}| + |p_{R,1} - p_{OT,1}|\,. \tag{2}$$

Since we assumed that $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$ is a successful distinguisher between $S_{OT,0}$ and $S_{OT,1}$, then $|p_{OT,0} - p_{OT,1}|$ is non-negligible. In that case, at least one of the three addends on the right-hand side of inequality (2) must be non-negligible.

If $|p_{OT,0} - p_{R,0}|$ or $|p_{R,1} - p_{OT,1}|$ are non-negligible, then it implies that the probabilistic polynomial-time algorithm $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$ can distinguish between different inputs of the receiver to the OT protocol. As we assumed that the OT functionality is secure, we are forced to discard these possibilities and deduce that $|p_{R,0} - p_{R,1}|$ is non-negligible. Alas, this option yields a contradiction to our intractability assumption, since such non-negligibility implies the existence of a polynomial-time algorithm Q that distinguishes between $\mathcal{A}^{s,\boldsymbol{\alpha}_0}_{n,m,r}$ and $\mathcal{A}^{s,\boldsymbol{\alpha}_1}_{n,m,r}$, as we proceed to demonstrate. The algorithm Q will simulate both the receiver and the sender. Let us denote the part of Q simulating the sender by Q_S and the part simulating the receiver by Q_R. If Q is given an input from one of those two distributions, it will forward that input to Q_S and then trigger both Q_S and Q_R to engage in an OT protocol where Q_R chooses to learn n points sampled at random from within the set of N points. At the end of that simulation of the OT protocol, Q_S will delegate its view to $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$ and will output the answer that the latter distinguisher outputs. As the success probability of Q equals that of $D_{\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1}$, and the latter is non-negligible, it contradicts our intractability assumption. That concludes the proof that the protocol preserves the receiver's privacy when the sender is semi-honest. □

3.2.2 The protocol is secure against semi-honest receivers

Here too we build a simulator, SIM_R, that simulates the view of the receiver given his input $\boldsymbol{\alpha}$ and output $P(\boldsymbol{\alpha})$ in the ideal model. As in the previous proof, we rely upon the OT hybrid model assumption that the OT protocol is implemented by a trusted third party. The simulator SIM_R operates as follows:

1. It builds a random polynomial $Q^R(x, \mathbf{y})$, where $x \in \mathbb{F}$ and $\mathbf{y} \in \mathbb{F}^r$, of degree sd, such that $Q^R(0, \boldsymbol{\alpha}) = P(\boldsymbol{\alpha})$.

2. It then generates N pairs, $\{(x_i, \mathbf{y}_i)\}_{1 \leq i \leq N}$, and a subset $T \subset [N]$ of n indices, as described in Steps 3(a)-3(e).

3. SIM_R and the sender invoke an OT protocol, which is executed by a trusted third party, after which SIM_R learns the pairs $\{Q^R(x_i, \mathbf{y}_i) : i \in T\}$.

We will show that the receiver does not learn any information about the polynomial P other than its value at the point $\boldsymbol{\alpha}$, whence it cannot distinguish between the real execution using P and the simulation.

We will show this by first showing that a semi-honest receiver can learn at most a single linear combination of the coefficients of the polynomial P. Because of correctness, we know that he will learn the value $P(\boldsymbol{\alpha})$ for some $\boldsymbol{\alpha}$. As $P(\boldsymbol{\alpha})$ is a linear combination of the coefficients of P, such a receiver cannot learn any additional information about P.

Let Q be the polynomial that is defined in the protocol, i.e.

$$Q(x, \mathbf{y}) = M(x) + P(\mathbf{y}) = \sum_{1 \le j \le sd} a_j x^j + \sum_{0 \le |\mathbf{k}| \le d} b_{\mathbf{k}} \mathbf{y}^{\mathbf{k}} . \tag{3}$$

Proposition 3.2 *Let $x_1, \ldots, x_n$ be $n = sd+1$ distinct nonzero values in $\mathbb{F}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_n$ be any n points in $\mathbb{F}^r$. Then the values $\{Q(x_i, \mathbf{y}_i)\}_{i=1}^n$ are either independent of the coefficients $b_{\mathbf{k}}$ or depend on a single linear combination of these coefficients.*

Proof. Let κ_r^d be the set of all r-dimensional multi-indices of order d at most. (The second sum on the right hand side of Eq. (3) is for all multi-indices $\mathbf{k} \in \kappa_r^d$.) The size of κ_r^d is $t := \binom{d+r}{r}$. Let

$$\kappa_r^d = \{\mathbf{k}_0, \ldots, \mathbf{k}_{t-1}\}$$

be an ordering of all multi-indices in κ_r^d, where $\mathbf{k}_0 = \mathbf{0} := (0, \ldots, 0)$. With these notations, the values $\{Q(x_i, \mathbf{y}_i)\}_{i=1}^n$ yield a set of n linear equations

$$Q(x_i, \mathbf{y}_i) = \sum_{1 \le j \le sd} a_j x_i^j + \sum_{0 \le j \le t-1} b_{\mathbf{k}_j} \mathbf{y}_i^{\mathbf{k}_j}$$

in the $n - 1 + t$ unknown coefficients $a_1, \ldots, a_{sd}, b_{\mathbf{k}_0}, \ldots, b_{\mathbf{k}_{t-1}}$ (recall that $sd = n - 1$). These equations may be summarized in the following matrix form:

$$A \cdot \begin{pmatrix} a_{sd} \\ \vdots \\ a_1 \\ b_{\mathbf{k}_{t-1}} \\ \vdots \\ b_{\mathbf{k}_1} \\ b_{\mathbf{k}_0} \end{pmatrix} = \begin{pmatrix} Q(x_1, \mathbf{y}_1) \\ Q(x_2, \mathbf{y}_2) \\ \vdots \\ Q(x_n, \mathbf{y}_n) \end{pmatrix}, \quad \text{where} \quad A = \begin{pmatrix} x_1^{sd} & \cdots & x_1^1 & \mathbf{y}_1^{\mathbf{k}_{t-1}} & \cdots & \mathbf{y}_1^{\mathbf{k}_1} & 1 \\ x_2^{sd} & \cdots & x_2^1 & \mathbf{y}_2^{\mathbf{k}_{t-1}} & \cdots & \mathbf{y}_2^{\mathbf{k}_1} & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^{sd} & \cdots & x_n^1 & \mathbf{y}_n^{\mathbf{k}_{t-1}} & \cdots & \mathbf{y}_n^{\mathbf{k}_1} & 1 \end{pmatrix} .$$

We aim at showing that the rows of A do not span more than a single linear combination of the vectors $\{\mathbf{e}_i\}_{i=n}^{n-1+t}$ where $\mathbf{e}_i = (\delta_{i,1}, \delta_{i,2}, \ldots, \delta_{i,n-1+t})$, and $\delta_{i,j}$ is the Kronecker delta. To that end, we consider the following square matrix:

$$B = \begin{pmatrix} x_1^{n-1} & \cdots & x_1^1 & 1 \\ x_2^{n-1} & \cdots & x_2^1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^{n-1} & \cdots & x_n^1 & 1 \end{pmatrix} .$$

As $x_1, \ldots, x_n$ are distinct, the determinant $|B|$ of B is nonzero. Let B_i be the sub-matrix of B that is obtained by removing its i-th row and the last column. Expanding the determinant of B by the last column, we infer that

$$|B| = \sum_{i=1}^{n} (-1)^{n+i} |B_i| \,.$$

Since $|B| \neq 0$, it follows that there exists an index $1 \leq i \leq n$ for which $|B_i| \neq 0$. Without loss of generality, we assume that $|B_n| \neq 0$. Returning now to the matrix A, we consider the square matrix C of order $n-1+t$ that is formed by taking the first $n-1$ rows of A (namely, all rows of A except the last one) and appending to them the t row vectors $\mathbf{e}_n, \ldots, \mathbf{e}_{n-1+t}$, i.e.,

$$C = \left(\begin{array}{ccc|cccc} x_1^{n-1} & \cdots & x_1^1 & \mathbf{y}_1^{\mathbf{k}_{t-1}} & \mathbf{y}_1^{\mathbf{k}_{t-2}} & \cdots & \mathbf{y}_1^{\mathbf{k}_1} & 1 \\ x_2^{n-1} & \cdots & x_2^1 & \mathbf{y}_2^{\mathbf{k}_{t-1}} & \mathbf{y}_2^{\mathbf{k}_{t-2}} & \cdots & \mathbf{y}_2^{\mathbf{k}_1} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n-1}^{n-1} & \cdots & x_{n-1}^1 & \mathbf{y}_{n-1}^{\mathbf{k}_{t-1}} & \mathbf{y}_{n-1}^{\mathbf{k}_{t-2}} & \cdots & \mathbf{y}_{n-1}^{\mathbf{k}_1} & 1 \\ \hline 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right) .$$

Since the lower-right $(t \times t)$-block of C is the identity matrix, and the lower-left block is zero, the determinant of C equals the determinant of its upper-left block of size $(n-1) \times (n-1)$. Since that block is exactly B_n, that was assumed to have a nonzero determinant, it follows that $|C| \neq 0$. This implies that the first $n-1$ rows of A do not span any linear combination of the vectors $\mathbf{e}_n, \ldots, \mathbf{e}_{n-1+t}$. Therefore, the matrix A, that has only one row in addition to those $n-1$ rows, cannot span more than a single linear combination of the vectors $\mathbf{e}_n, \ldots, \mathbf{e}_{n-1+t}$. That concludes the proof that the protocol preserves the sender's privacy. □

The only information that the receiver gets about the polynomial P in the algorithm is during the OT stage, in which he learns point values of R which is defined by $R(x) = Q(x, \mathbf{S}(x))$. By Proposition 3.2, those values can disclose only a single linear combination of the coefficients of P.

Corollary 3.3 *A semi-honest receiver that follows the protocol does not learn any information about the coefficients of P besides $P(\boldsymbol{\alpha})$ for some point $\boldsymbol{\alpha}$.*

Proof. The receiver cannot learn more than a single linear combination of the coefficients of P. By correctness, we know that a semi-honest party, following the protocol exactly, and using a suitable $\mathbf{S}(x)$ that satisfies $\mathbf{S}(0) = \boldsymbol{\alpha}$, will indeed learn $P(\boldsymbol{\alpha})$. As $P(\boldsymbol{\alpha})$ is a linear combination of the coefficients of P, we conclude that he can not learn any more information about P other than $P(\boldsymbol{\alpha})$. □

3.3 Security against malicious behavior

Here we consider the case where one of the parties is malicious. In that case it is important to protect the privacy of the other party.

The protocol preserves the receiver's privacy even when the sender is malicious, since the only message sent from the receiver to the sender is sent before any message is sent from the sender to the receiver, and the OT functionality is assumed to be secure against malicious parties. On the other hand, the protocol is vulnerable to an attack by a malicious *receiver*, since he may choose to learn a linear combination of the coefficients of P that does not correspond to a value of the polynomial. To protect from such malicious behavior, we use a technique similar to the one used in the univariate case, and that is breaking the polynomial into linear polynomials and evaluating them instead.

Lemma 3.4 *When the polynomial P is linear, i.e., $P(x_1, \ldots, x_r) = b_r x_r + \ldots + b_1 x_1 + b_0$, then with probability $1 - 1/|\mathbb{F}|$, the only information that the receiver may learn is a single value of $P(\cdot)$.*

Proof. As shown in Proposition 3.2, the receiver can learn at most a single linear combination of the coefficients, $b_0, \ldots, b_r$, i.e., a value of the form $\alpha_0 \cdot b_0 + \ldots + \alpha_r \cdot b_r$, where α_i are known to the receiver. There are two cases to consider:

If $\alpha_0 \neq 0$ then such a linear combination is just $\alpha_0 \cdot (b_0 + (\alpha_1/\alpha_0)b_1 + \ldots + (\alpha_r/\alpha_0)b_r) = \alpha_0 \cdot P(\alpha_1/\alpha_0, \ldots, \alpha_r/\alpha_0)$. Hence, in this case the receiver learns a point value of the polynomial P.

The case that worries us is the case where $\alpha_0 = 0$ and at least one of $\alpha_1, \ldots, \alpha_r$ is nonzero. Here, the receiver learns the value of a linear combination of the coefficients that does not correspond to any point value of P. We proceed to show that the protocol does not allow the learning of such a linear combination. Going back to the proof of Proposition 3.2, when the polynomial P is of degree $d = 1$, the matrix A takes the form

$$A = \begin{pmatrix} x_1^s & \cdots & x_1^1 & y_{1,r} & \cdots & y_{1,1} & 1 \\ x_2^s & \cdots & x_2^1 & y_{2,r} & \cdots & y_{2,1} & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^s & \cdots & x_n^1 & y_{n,r} & \cdots & y_{n,1} & 1 \end{pmatrix},$$

where $n = s + 1$ and $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,r})$, $1 \leq i \leq n$. It has n rows and $n + r$ columns. In order to prove that the receiver cannot learn the value of any nontrivial linear combination with $\alpha_0 = 0$, we need to show that no $(n + r)$-dimensional vector of the form $(0, \ldots, 0, \beta_r, \ldots, \beta_1, 0)$, where at least one of the β_i is nonzero, is spanned by the rows of A. Assume, towards contradiction, that such a vector is indeed spanned by the rows of A. Then, by focusing on the first $n-1$ components and the last one, we infer that there exists a non-trivial linear combination of the rows of the matrix

$$A' = \begin{pmatrix} x_1^{n-1} & \cdots & x_1^1 & 1 \\ x_2^{n-1} & \cdots & x_2^1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^{n-1} & \cdots & x_n^1 & 1 \end{pmatrix}$$

that yields the zero vector. But that is impossible since, owing to our assumption that $x_1, \ldots, x_n$ are distinct, the matrix A' is non-singular. Therefore, the only linear combination of the rows of A' that gives the zero vector is the trivial linear combination. Such a linear combination, when applied to the original matrix A, cannot yield a vector $(0, \ldots, 0, \beta_r, \ldots, \beta_1, 0)$ in which at least one of the β_i is nonzero. $\square$

We now proceed to generalize a theorem due to Gilboa [8] to r dimensions:

Theorem 3.5 *For every r-variate polynomial $P(\mathbf{x})$ of degree d, there exist $\binom{d-1+r}{r}$ linear polynomials $\{P_{\mathbf{j}}(\mathbf{x}) : 0 \leq |\mathbf{j}| \leq d-1\}$, such that an OMPE of P at a point $\boldsymbol{\alpha}$ can be reduced to a parallel execution of OMPEs of each of $P_{\mathbf{j}}$, where all the linear polynomials are evaluated at the same point $\boldsymbol{\alpha}$.*

We split the proof of Theorem 3.5 into Lemmas 3.6 and 3.7.

Lemma 3.6 *For every r-variate polynomial $P(\mathbf{x})$ of degree d there exist $\binom{d-1+r}{r}$ linear polynomials $\{P_{\mathbf{k}}(\mathbf{x}) : 0 \leq |\mathbf{k}| \leq d-1\}$ such that*

$$P(\boldsymbol{\alpha}) = \sum_{0 \leq |\mathbf{k}| \leq d-1} P_{\mathbf{k}}(\boldsymbol{\alpha}) \cdot \boldsymbol{\alpha}^{\mathbf{k}} \qquad \text{for all } \boldsymbol{\alpha} \in \mathbb{F}^r \,. \tag{4}$$

Proof. Let

$$P(\mathbf{x}) = \sum_{0 \leq |\mathbf{k}| \leq d} a_{\mathbf{k}} \mathbf{x}^{\mathbf{k}} \tag{5}$$

be an r-variate polynomial of degree d. Let $\{s_{\mathbf{k}} \in \mathbb{F} : 1 \leq |\mathbf{k}| \leq d-1\}$ be a set of random values. We define $\binom{d-1+r}{r}$ linear polynomials, $\{P_{\mathbf{k}} : 0 \leq |\mathbf{k}| \leq d-1\}$, in the following manner. The first polynomial, corresponding to the multi-index $\mathbf{0} = (0, \ldots, 0)$, is

$$P_{\mathbf{0}}(\mathbf{x}) = a_{\mathbf{0}} + \sum_{|\mathbf{j}|=1} s_{\mathbf{j}} \mathbf{x}^{\mathbf{j}} \,,$$

where $a_{\mathbf{0}}$ is the free coefficient in $P(\mathbf{x})$ (see Eq. (5)). Then, for multi-indices $\mathbf{k}$ with $1 \leq |\mathbf{k}| \leq d-2$, we define

$$P_{\mathbf{k}}(\mathbf{x}) = -s_{\mathbf{k}} + a_{\mathbf{k}} + \sum_{|\mathbf{j}|=1} \frac{s_{\mathbf{k}+\mathbf{j}}}{w(\mathbf{k}+\mathbf{j})} \mathbf{x}^{\mathbf{j}} \,,$$

where, for any multi-index $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_r)$, $w(\boldsymbol{\ell}) := \#\{\ell_i \neq 0 : 1 \leq i \leq r\}$. Finally, for multi-indices $\mathbf{k}$ with $|\mathbf{k}| = d-1$, we define

$$P_{\mathbf{k}}(\mathbf{x}) = -s_{\mathbf{k}} + a_{\mathbf{k}} + \sum_{|\mathbf{j}|=1} \frac{a_{\mathbf{k}+\mathbf{j}}}{w(\mathbf{k}+\mathbf{j})} \mathbf{x}^{\mathbf{j}} \,.$$

We claim that $P(\mathbf{x}) = \sum_{0 \leq |\mathbf{k}| \leq d-1} P_{\mathbf{k}}(\mathbf{x}) \cdot \mathbf{x}^{\mathbf{k}}$. Indeed,

$$\sum_{0 \leq |\mathbf{k}| \leq d-1} P_{\mathbf{k}}(\mathbf{x}) \cdot \mathbf{x}^{\mathbf{k}} = a_{\mathbf{0}} + \sum_{|\mathbf{j}|=1} s_{\mathbf{j}} \mathbf{x}^{\mathbf{j}} + \sum_{1 \leq |\mathbf{k}| \leq d-2} \left(-s_{\mathbf{k}} \mathbf{x}^{\mathbf{k}} + a_{\mathbf{k}} \mathbf{x}^{\mathbf{k}} + \mathbf{x}^{\mathbf{k}} \sum_{|\mathbf{j}|=1} \frac{s_{\mathbf{k}+\mathbf{j}}}{w(\mathbf{k}+\mathbf{j})} \mathbf{x}^{\mathbf{j}} \right) \quad +$$

$$+ \sum_{|\mathbf{k}|=d-1} \left(-s_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} + a_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} + \mathbf{x}^{\mathbf{k}} \sum_{|\mathbf{j}|=1} \frac{a_{\mathbf{k}+\mathbf{j}}}{w(\mathbf{k}+\mathbf{j})}\mathbf{x}^{\mathbf{j}} \right) .$$

After rearrangement, we find that

$$\sum_{0\le|\mathbf{k}|\le d-1} P_{\mathbf{k}}(\mathbf{x})\cdot\mathbf{x}^{\mathbf{k}} = a_{\mathbf{0}} + \sum_{|\mathbf{j}|=1} s_{\mathbf{j}}\mathbf{x}^{\mathbf{j}} + \sum_{1\le|\mathbf{k}|\le d-1} \left(-s_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} + a_{\mathbf{k}}\mathbf{x}^{\mathbf{k}}\right) +$$

$$+ \sum_{1\le|\mathbf{k}|\le d-2} \sum_{|\mathbf{j}|=1} \frac{s_{\mathbf{k}+\mathbf{j}}}{w(\mathbf{k}+\mathbf{j})}\mathbf{x}^{\mathbf{k}+\mathbf{j}} + \sum_{|\mathbf{k}|=d-1} \sum_{|\mathbf{j}|=1} \frac{a_{\mathbf{k}+\mathbf{j}}}{w(\mathbf{k}+\mathbf{j})}\mathbf{x}^{\mathbf{k}+\mathbf{j}} .$$

We observe that for every multi-index $\boldsymbol{\ell}$, there are exactly $w(\boldsymbol{\ell})$ pairs of multi-indices $(\mathbf{k},\mathbf{j})$ such that $\mathbf{k}+\mathbf{j}=\boldsymbol{\ell}$ and $|\mathbf{j}|=1$. Hence, the last two sums above become

$$\sum_{2\le|\mathbf{k}|\le d-1} s_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} + \sum_{|\mathbf{k}|=d} a_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} .$$

Consequently, three of the sums above cancel out,

$$\sum_{|\mathbf{j}|=1} s_{\mathbf{j}}\mathbf{x}^{\mathbf{j}} + \sum_{1\le|\mathbf{k}|\le d-1} -s_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} + \sum_{2\le|\mathbf{k}|\le d-1} s_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} = 0 ,$$

and we conclude that

$$\sum_{0\le|\mathbf{k}|\le d-1} P_{\mathbf{k}}(\mathbf{x})\cdot\mathbf{x}^{\mathbf{k}} = a_{\mathbf{0}} + \sum_{1\le|\mathbf{k}|\le d-1} a_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} + \sum_{|k|=d} a_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} = \sum_{0\le|\mathbf{k}|\le d} a_{\mathbf{k}}\mathbf{x}^{\mathbf{k}} = P(\mathbf{x}) .$$

□

The polynomials $P_{\mathbf{k}}$, $0 \le |\mathbf{k}| \le d-1$, are independent of $\boldsymbol{\alpha}$. Hence, the sender may define them upfront by selecting the random scalars $\{s_{\mathbf{k}} : 1 \le |\mathbf{k}| \le d-1\}$. Then, the two parties may execute $t := \binom{d-1+r}{r}$ OMPEs in which the receiver learns the values $P_{\mathbf{k}}(\boldsymbol{\alpha})$, $0 \le |\mathbf{k}| \le d-1$. After that stage, the receiver may deduce the value $P(\boldsymbol{\alpha})$ through Eq. (4). It remains only to show that such a protocol allows the receiver to learn only the value $P(\boldsymbol{\alpha})$ and no other information about the coefficients of P. This is proved in the following lemma.

Lemma 3.7 *Given $\boldsymbol{\alpha}$ and $P(\boldsymbol{\alpha})$, it is possible to simulate the receiver's output in the t invocations of the OMPE protocols.*

Since $\boldsymbol{\alpha}$ and $P(\boldsymbol{\alpha})$ alone are sufficient in order to simulate the entire output of the receiver, it follows that no other information on the polynomial may be inferred from the receiver's view.

Proof. Recall that in the definition of the polynomials $P_{\mathbf{k}}$, as described in the proof of Lemma 3.6, we used random values $\{s_{\mathbf{k}} : 1 \le |\mathbf{k}| \le d-1\}$. In particular, each $P_{\mathbf{k}}(x)$, $1 \le |\mathbf{k}| \le d-1$, includes a random addend, $-s_{\mathbf{k}}$. Since $s_{\mathbf{k}}$ distributes uniformly in $\mathbb{F}$, then for any random variable X that takes values in $\mathbb{F}$, $X - s_{\mathbf{k}}$ also distributes uniformly in $\mathbb{F}$. Since for any $\boldsymbol{\alpha} \in \mathbb{F}^r$, $P_{\mathbf{k}}(\boldsymbol{\alpha})$ is of the form $X - s_{\mathbf{k}}$, $1 \le |\mathbf{k}| \le d-1$, we infer that $P_{\mathbf{k}}(\boldsymbol{\alpha})$ distributes uniformly in $\mathbb{F}$.

As in the proof of Proposition 3.2, let κ_r^{d-1} be the set of all r-dimensional multi-indices of order $d-1$ at most. The size of κ_r^{d-1} is t. Let

$$\kappa_r^{d-1} = \{\mathbf{k}_0, \ldots, \mathbf{k}_{t-1}\}$$

be an ordering of all multi-indices in κ_r^{d-1}, where $\mathbf{k}_0 = \mathbf{0} = (0, \ldots, 0)$. Then the linear polynomials used in this protocol are $P_{\mathbf{k}_0}, \ldots, P_{\mathbf{k}_{t-1}}$. Since we have shown that $P_{\mathbf{k}}(\boldsymbol{\alpha})$ distributes uniformly in $\mathbb{F}$ for any $1 \leq |\mathbf{k}| \leq d-1$, we conclude that the vector $(P_{\mathbf{k}_1}(\boldsymbol{\alpha}), \ldots, P_{\mathbf{k}_{t-1}}(\boldsymbol{\alpha}))$ distributes uniformly in $\mathbb{F}^{t-1}$. Invoking Eq. (4), we see that

$$P_{\mathbf{0}}(\boldsymbol{\alpha}) = P(\boldsymbol{\alpha}) - \sum_{1 \leq |\mathbf{k}| \leq d-1} P_{\mathbf{k}}(\boldsymbol{\alpha}) \boldsymbol{\alpha}^{\mathbf{k}} \,. \tag{6}$$

Hence, given $\boldsymbol{\alpha}$ and $P(\boldsymbol{\alpha})$ we can simulate the receiver's output in all of the t OMPEs by selecting uniformly and independently values for $\{P_{\mathbf{k}}(\boldsymbol{\alpha}) : 1 \leq |\mathbf{k}| \leq d-1\}$, and then computing $P_{\mathbf{0}}(\boldsymbol{\alpha})$ through Eq. (6). □

We now describe the new protocol. As in the previous OMPE protocols, the sender's input is an r-variate polynomial P of degree d over $\mathbb{F}$, and the receiver's input is a value $\boldsymbol{\alpha} \in \mathbb{F}^r$. The protocol is composed of the following steps:

1. The sender generates the $t = \binom{d-1+r}{r}$ polynomials $\{P_{\mathbf{k}} : 0 \leq |\mathbf{k}| \leq d-1\}$ as described in the proof of Lemma 3.6.
2. The parties execute a slight variation on the OMPE protocol, in which the receiver evaluates the linear polynomials $P_{\mathbf{k}}$ at the point $\boldsymbol{\alpha}$ in the following manner:
   - The sender generates t independent masking polynomials, $\{M_{\mathbf{k}}(x) : 0 \leq |\mathbf{k}| \leq d-1\}$, of degree sd (where s is a security parameter) and then the corresponding $(r+1)$-variate polynomials, $Q_{\mathbf{k}}(x, \mathbf{y}) = M_{\mathbf{k}}(x) + P_{\mathbf{k}}(\mathbf{y})$, $0 \leq |\mathbf{k}| \leq d-1$.
   - The receiver generates r random univariate polynomials of degree s, $\{S_i(\cdot)\}_{1 \leq i \leq r}$, such that $S_i(0) = \alpha_i$, $1 \leq i \leq r$, and then defines $\mathbf{S} : \mathbb{F} \to \mathbb{F}^r$, by $\mathbf{S}(x) = (S_1(x), \ldots, S_r(x))$. Note that this defines t polynomials $R_{\mathbf{k}}(x) = Q_{\mathbf{k}}(x, \mathbf{S}(x))$ such that $R_{\mathbf{k}}(0) = P_{\mathbf{k}}(\boldsymbol{\alpha})$, $0 \leq |\mathbf{k}| \leq d-1$. The degree of $R_{\mathbf{k}}$ is sd for all $0 \leq |\mathbf{k}| \leq d-1$.
   - The sender and receiver perform the OT stage of the OMPE protocol, at the end of which the receiver learns $sd+1$ tuples of the form

     $$(x_i, R_{\mathbf{k}_0}(x_i), \ldots, R_{\mathbf{k}_{t-1}}(x_i)) \,.$$

     To that end, each value held by the sender in the OT stage will be of the form

     $$(Q_{\mathbf{k}_0}(x_i, \mathbf{y}_i), \ldots, Q_{\mathbf{k}_{t-1}}(x_i, \mathbf{y}_i))$$

     (instead of simply $Q(x_i, \mathbf{y}_i)$ as in the original OMPE protocol).
   - The receiver then interpolates each $R_{\mathbf{k}}$ to learn $R_{\mathbf{k}}(0) = P_{\mathbf{k}}(\boldsymbol{\alpha})$, for $0 \leq |\mathbf{k}| \leq d-1$.
   - The receiver uses the values $P_{\mathbf{k}}(\boldsymbol{\alpha})$ to compute $P(\boldsymbol{\alpha})$ by Eq. (4).

The correctness of this protocol is trivial, hence, we concentrate on proving that it preserves the privacy of the sender even when the receiver is malicious.

Theorem 3.8 *Assuming the use of an ideal OT functionality, specifically one that is secure against malicious behavior, the OMPE protocol presented above is secure against malicious behavior. Namely:*

receiver's privacy: *A malicious sender cannot distinguish between two different inputs of the receiver.*

sender's privacy: *With probability* $1 - 1/|\mathbb{F}|$*, a malicious receiver learns only a single value of the polynomial* P *(or nothing at all).*

Proof.

receiver's privacy: The only information that the receiver sends is the same as in the original OMPE protocol, so that privacy is surely maintained in the semi-honest case. As for the malicious case, the sender still has no way to affect what the receiver sends, except for the OT stage, for the same reason as in the original protocol: The only message that the receiver sends to the sender is sent before any message is sent from the sender to the receiver.

sender's privacy: We know from Lemma 3.4 that an OMPE of a linear polynomial is secure against malicious receivers, with probability $1 - 1/|\mathbb{F}|$. We also know, from Theorem 3.5, and most specifically from Lemma 3.7, that the parallel OMPE of linear polynomials, as done in this protocol, does not reveal any more information than what it should.

□

4 Applications

In this section we describe some applications of oblivious evaluation of multivariate polynomials.

4.1 Measuring the distance between two parties

Assume that Alice and Bob have positions in a Euclidean space. The following method allows them to compute their distance without revealing their exact position.

Let us assume that Alice and Bob are located in a two-dimensional plane. We may assume that all possible positions consist of integer coordinates in the interval $[0, M]$, for a suitably selected M. Let $p > 2M^2$ be a prime number, and let $\mathbb{F} = \mathbb{F}_p$ be the finite field of size p. Say Alice is located at $(x_A, y_A) \in \mathbb{F}^2$ and Bob is located at $(x_B, y_B) \in \mathbb{F}^2$. Then Alice defines the bivariate polynomial $P(x, y) = (x - x_A)^2 + (y - y_A)^2$, and Bob sets $\boldsymbol{\alpha} = (x_B, y_B)$. They proceed to execute OMPE where Bob learns $P(\boldsymbol{\alpha}) = P(x_B, y_B) = (x_B - x_A)^2 + (y_B - y_A)^2$. Since $p > 2M^2$, the value $P(\boldsymbol{\alpha})$ may be interpreted as an integer denoting the square of the distance between Alice and Bob.

This example may be useful, say, in a dating website where two people want to initiate contact if and only if their locations are sufficiently close. The above protocol allows the two parties to

check whether they are close enough without disclosing their exact location. In this context it may be also useful to consider distance in higher dimensions. Suppose that each user of the dating website holds a vector describing their interests or attributes, and that Alice and Bob would like to get to know each other if and only if their characteristic vectors are sufficiently close. Assume that the characteristic vector is r-dimensional, and that each component is an integer in the range $[0, M]$. In addition, let us assume that the components of the vector are associated with weights, $w_1, \ldots, w_r$, which are integers in the range $(0, W]$. If Alice's vector is $(a_1, \ldots, a_r)$ and Bob's is $(b_1, \ldots, b_r)$, the distance that they aim to compute is $\sum_{i=1}^{r} w_i(a_i - b_i)^2$. This can be easily accomplished using OMPE as follows: Let $p > rWM^2$ be a prime number and let $\mathbb{F} = \mathbb{F}_p$. Alice defines the r-variate polynomial $P(x_1, ..., x_r) = \sum_{i=1}^{r} w_i(x_i - a_i)^2$ and Bob sets $\boldsymbol{\alpha} = (b_1, \ldots, b_r)$. At the end of the OMPE protocol, Bob will have learned the value of the integer $P(\boldsymbol{\alpha}) = \sum_{i=1}^{r} w_i(a_i - b_i)^2$. If that value is smaller than the thresholds that Alice and Bob determined, then Alice and Bob may pursue their acquaintance.

4.2 Linear algebra computations

4.2.1 Computing the scalar product of two vectors

Say Alice has a vector $\mathbf{a} = (a_1, \ldots, a_r) \in \mathbb{F}^r$ and Bob has a vector $\mathbf{b} = (b_1, \ldots, b_r) \in \mathbb{F}^r$ and they wish to compute the scalar product $\mathbf{a} \cdot \mathbf{b}$ without revealing additional information about $\mathbf{a}$ and $\mathbf{b}$. Then Alice defines an r-variate polynomial $P(x_1, \ldots, x_r) = \sum_{j=1}^{r} a_j x_j$ and Bob sets $\boldsymbol{\alpha} = (b_1, \ldots, b_r)$. If Alice and Bob perform OMPE using P and $\boldsymbol{\alpha}$ as inputs, Bob will learn the value of $P(\boldsymbol{\alpha}) = \sum_{j=1}^{r} a_j b_j = \mathbf{a} \cdot \mathbf{b}$.

There are other known protocols for oblivious evaluation of a scalar product, e.g., [1] and [18]. One of the protocols, by Atallah and Du [1], includes the following steps: The input vector of Alice, $\mathbf{a}$, is broken into a sum $\mathbf{a} = \sum_{i=1}^{m} \mathbf{a}_i$ where the first $m - 1$ vectors in that sum are chosen uniformly and independently at random from the vector space. Bob chooses m random numbers that sum up to zero, $r_1, \ldots, r_m$. Alice then hides the vector $\mathbf{a}_i$, $1 \leq i \leq m$, within a set of p vectors, where the other $p - 1$ vectors in that set are random vectors, and the position of $\mathbf{a}_i$ in them is random and known only to Alice. Alice sends each such set of p vectors to Bob, and Bob computes for each vector $\mathbf{h}$ in the set, the product $\mathbf{h} \cdot \mathbf{b} + r_i$. They then use chosen 1-out-of-p OT where Alice chooses to learn the computed value for the correct position amongst the p values, whence Alice learns $\mathbf{a}_i \cdot \mathbf{b} + r_i$. After all m iterations, Alice can compute

$$\sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{b} + r_i = \mathbf{b} \cdot \sum_{i=1}^{m} \mathbf{a}_i + \sum_{i=1}^{m} r_i = \mathbf{b} \cdot \mathbf{a}\,.$$

The parameters p and m are security parameters, chosen so that p^{-m} — the probability of guessing correctly all positions of the vectors $\mathbf{a}_i$ — is sufficiently small.

A slightly modified version of our protocol above may be used in order to allow Alice and Bob to check whether their input vectors are orthogonal or not, without revealing any other information about their secret vectors. Using the protocol for scalar product, if Alice and Bob find out that $\mathbf{a} \cdot \mathbf{b} = s \neq 0$, then Alice learns that Bob's vector is on the flat $\{\mathbf{x} \in \mathbb{F}^r : \mathbf{a} \cdot \mathbf{x} = s\}$, while Bob learns that $\mathbf{a} \in \{\mathbf{x} \in \mathbb{F}^r : \mathbf{b} \cdot \mathbf{x} = s\}$. However, if Alice multiplies her input vector $\mathbf{a}$ with a secret random nonzero scalar, $r_A \in_R \mathbb{F}^*$, and Bob multiplies $\mathbf{b}$ with $r_B \in_R \mathbb{F}^*$, then the

only information that is revealed by $r_A r_B \cdot \mathbf{a} \cdot \mathbf{b}$ is the orthogonality or non-orthogonality of $\mathbf{a}$ and $\mathbf{b}$.

4.2.2 The inclusion of a vector is in a subspace

Consider a setting where the receiver has an input vector $\mathbf{x} \in \mathbb{F}^r$ and the sender has a subspace $W \subseteq \mathbb{F}^r$. The receiver, who knows $k = \dim W$, would like to know whether $\mathbf{x} \in W$. The goal is to allow him to learn this information without learning any further information about W and without allowing the sender to learn any further information about $\mathbf{x}$.

To that end, the sender randomly chooses $r-k$ base vectors of $W^\perp$ (the orthogonal complement of W), say $\mathbf{y}_1, \ldots, \mathbf{y}_{r-k}$. Then $\mathbf{x} \in W$ if and only if $\mathbf{x} \cdot \mathbf{y}_i = 0$ for all $1 \leq i \leq r-k$.

We make a small alteration to the scalar product protocol that was described in Section 4.2.1. We add a random value to the polynomial, so that at the end of the protocol, the receiver learns $\mathbf{a} \cdot \mathbf{b} + s$ while s is known only to the sender. We can now use this altered protocol to compute the $r-k$ values $\{\mathbf{x} \cdot \mathbf{y}_i + s_i\}_{1 \leq i \leq r-k}$, where s_i are scalars that the sender selects randomly and independently. At this stage, the receiver has the vector $(\mathbf{x} \cdot \mathbf{y}_1 + s_1, \ldots, \mathbf{x} \cdot \mathbf{y}_{r-k} + s_{r-k})$ and the sender has the vector $(s_1, \ldots, s_{r-k})$. We have $\mathbf{x} \in W$ if and only if these vectors are identical, and that can be checked using protocols for oblivious string comparison, e.g. [7].

4.2.3 Sets of vectors of full rank

A set of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{F}^r$ is said to be of full rank if they span a subspace of $\mathbb{F}^r$ of dimension $\min\{n, r\}$. Assume that Alice holds a set of k independent vectors in $\mathbb{F}^r$, $A = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, and Bob holds a set of ℓ independent vectors in $\mathbb{F}^r$, $B = \{\mathbf{y}_1, \ldots, \mathbf{y}_\ell\}$. They wish to determine whether $A \cup B$ is of full rank without revealing to each other additional information about the vectors that they possess.

Let us assume first that $k + \ell = r$. Then if M is the matrix whose rows are the vectors in $A \cup B$, the set $A \cup B$ is of full rank if and only if $|M| \neq 0$. Hence, Alice may define the $r\ell$-variate polynomial of degree ℓ,

$$P_A(\mathbf{y}_1, \ldots, \mathbf{y}_\ell) = P_A(y_{1,1}, \ldots, y_{1,r}; \ldots; y_{\ell,1}, \ldots, y_{\ell,r}) = \begin{vmatrix} x_{1,1} & \cdots & x_{1,r} \\ \vdots & & \\ x_{k,1} & \cdots & x_{k,r} \\ y_{1,1} & \cdots & y_{1,r} \\ \vdots & & \\ y_{\ell,1} & \cdots & y_{\ell,r} \end{vmatrix} .$$

Alice and Bob may then engage in an OMPE protocol in order to evaluate P_A at the point that corresponds to the vectors that Bob possesses. If the result is zero, they may conclude that $A \cup B$ is not of full rank. If, on the other hand, the result is nonzero, then $A \cup B$ is of full rank.

Note that the actual value of the determinant reveals some information on the input vectors in case $|M| \neq 0$. However, this problem may be easily solved by having Alice and Bob multiply one of their vectors by a random nonzero scalar. Then any nonzero result for $|M|$ may be attained by any two sets of vectors A and B whose union is of full rank. Hence, if $|M| \neq 0$, the only information that Alice and Bob may deduce is that $A \cup B$ is of full rank.

The case where $k + \ell \neq r$ is harder, since in that case the matrix M is not square. The above described solution is restricted to the case where M is square since it uses its determinant, which is a polynomial function of the matrix entries for which the value is nonzero if and only if the matrix is of full rank. We are not aware of such a polynomial function, which is efficiently computable, in the case where the matrix is not square. Such a polynomial function exists over fields of characteristic zero[2]. Over finite fields, on the other hand, there exists a polynomial with the desired property but it is not efficiently computable[3]. Instead, we propose the following alternative solution.

In case $d := k + \ell - r > 0$, Alice may define the $r(\ell - d)$-variate polynomial of degree $\ell - d$,

$$P_A(\mathbf{y}_1, \ldots, \mathbf{y}_{\ell-d}) = P_A(y_{1,1}, \ldots, y_{1,r}; \ldots; y_{\ell-d,1}, \ldots, y_{\ell-d,r}) = \begin{vmatrix} x_{1,1} & \cdots & x_{1,r} \\ \vdots & & \\ x_{k,1} & \cdots & x_{k,r} \\ y_{1,1} & \cdots & y_{1,r} \\ \vdots & & \\ y_{\ell-d,1} & \cdots & y_{\ell-d,r} \end{vmatrix} . \quad (7)$$

Alice and Bob may then engage in a sequence of OMPE protocols in order to evaluate P_A at the point that corresponds to a selection of $\ell - d$ vectors from among the ℓ vectors that Bob possesses. If in one of those instantiations of the OMPE protocol the result is nonzero, then $A \cup B$ is of full rank. If, on the other hand, in all $\binom{\ell}{d}$ possible selections the result was zero, $A \cup B$ is not of full rank. (Clearly, in each such instantiation of the protocol, Alice and Bob should multiply one of their input vectors by a new random nonzero scalar.) A similar solution may be utilized when $d = k + \ell - r < 0$.

4.2.4 Creating coalitions in linear secret sharing

The concept of linear secret sharing schemes was introduced by Brickell [3] in the ideal setting and was later generalized to non-ideal schemes. Linear schemes are equivalent to monotone span programs [13]. Let us recall the basic definitions of linear secret sharing (for simplicity we concentrate on ideal linear schemes).

The domain of secrets in such schemes is some finite field $\mathbb{F}$. Every participant u_i is identified by a unique vector $\mathbf{x}_i \in \mathbb{F}^r$, $1 \leq i \leq n$ (n is the number of participants), where r is some dimension that is determined by the dealer. The dealer also defines a so-called target vector $\mathbf{t} \in \mathbb{F}^r$ and a random vector $\mathbf{a}$ for which $\mathbf{a} \cdot \mathbf{t} = S$, where S is the secret. Finally, the dealer assigns to participant u_i the share $\mathbf{a} \cdot \mathbf{x}_i$. A subset of participants is able to recover the secret S if and only if their corresponding vectors span the target vector $\mathbf{t}$. If the vectors of such a subset do not span the target vector then the shares held by the participants in that subset reveal no information about the value of S.

Consider such a linear secret sharing setting and assume that Alice and Bob are two malicious parties that wish to get hold of the secret S. To that end, they corrupt participants and buy their

[2] An $s \times r$ matrix M over a field of characteristic zero is of full rank if either $s \geq r$ and $|M^t M| \neq 0$ or $s \leq r$ and $|MM^t| \neq 0$.

[3] Let M be an $s \times r$ matrix over a finite field of a prime order p, and assume that $s > r$. Let $\{d_1, \ldots, d_t\}$, where $t := \binom{s}{r}$, be the set of all $r \times r$ minors of M. Define $P_M := 1 - \prod_{i=1}^{t}(1 - d_i^{p-1})$. Then P_M, which is a polynomial in the entries of M, equals one if M is of full rank, and zero otherwise.

shares. As a result, Alice has k shares that correspond to the vectors $A = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, and Bob has ℓ shares that correspond to the vectors $B = \{\mathbf{y}_1, \ldots, \mathbf{y}_\ell\}$. We may assume that the vectors in A are independent and so are the vectors in B. (If, for example, one of the vectors in A was a linear combination of the other vectors in A, Alice would not have invested the time and effort to get hold of the share that corresponds to that vector.)

Assume that the vectors in A do not span the target vector $\mathbf{t}$ and neither do the vectors in B. Alice and Bob have no more resources to corrupt additional participants and they contemplate the possibility of colluding and unifying their vectors. To that end, both want to know upfront, before exposing the identities of the participants that they had corrupted, whether the union $A \cup B$ spans the target vector $\mathbf{t}$. That is another problem of multi-party computation that may be solved using OMPE.

Assume that Alice and Bob implemented the OMPE protocol from the previous section and found out that the unified set of vectors $A \cup B$ is of full rank. If $k + \ell \geq r$ then that implies that $\text{Span}(A \cup B) = \mathbb{F}^r$ and, consequently, $\mathbf{t} \in \text{Span}(A \cup B)$. If, on the other hand, $k + \ell < r$, Alice and Bob may repeat the protocol, where Alice uses this time $A \cup \{\mathbf{t}\}$ instead of A. Clearly, $\mathbf{t} \in \text{Span}(A \cup B)$ if and only if the augmented set $A \cup \{\mathbf{t}\} \cup B$ is no longer of full rank.

The question of whether $\mathbf{t} \in \text{Span}(A \cup B)$ becomes harder when $A \cup B$ is not of full rank. In that case, Alice and Bob have first to extract a subset of $A \cup B$ which is of full rank and spans the same subspace as $A \cup B$. Only then they may implement the above described procedure. Assuming that each of the two sets, A and B, is a set of full rank on its own, it is easy to see that there exists a subset $B' \subset B$ such that $A \cup B'$ is of full rank and $\text{Span}(A \cup B') = \text{Span}(A \cup B)$. The following protocol computes such a subset B' and then proceeds to determine whether $\mathbf{t}$ is spanned by $A \cup B$.

1. Bob sets $B' = \emptyset$ and $i = 0$.
2. While $i < \ell$ do:
   (a) $i = i + 1$; $B' = B' \cup \{\mathbf{y}_i\}$.
   (b) Alice and Bob implement the protocol from the previous section to determine whether $A \cup B'$ has a full rank.
   (c) If the answer is negative, set $B' = B' \setminus \{\mathbf{y}_i\}$.
   (d) Else, if $|A| + |B'| = r$ stop and output "$\mathbf{t} \in \text{Span}(A \cup B)$".
3. Alice and Bob check whether $A \cup B' \cup \{\mathbf{t}\}$ is of full rank. If it is, output "$\mathbf{t} \notin \text{Span}(A \cup B)$". Else output "$\mathbf{t} \in \text{Span}(A \cup B)$".

The above protocol finds a subset B' such that $A \cup B'$ is of full rank either in Step 2(d) or in Step 3. Step 2(d) corresponds to the case where $A \cup B$ spans the entire space $\mathbb{F}^r$. The algorithm reaches Step 3 in case $A \cup B$ spans only a subspace of $\mathbb{F}^r$. In that stage, B' is a subset of B for which $\text{Span}(A \cup B') = \text{Span}(A \cup B)$. The protocol then proceeds to determine whether $\mathbf{t}$ is spanned by $A \cup B'$ in the manner that we discussed earlier.

The above protocol discloses, in addition to the inclusion of $\mathbf{t}$ in $\text{Span}(A \cup B)$, also the dimension of $\text{Span}(A \cup B)$, which is $|A| + |B'|$. That information is useful, since if Alice and Bob find out that $\mathbf{t} \notin \text{Span}(A \cup B)$, the dimension of $\text{Span}(A \cup B)$ quantifies the advantage that each of them gains by forming the coalition between them and how "far" they are from being able to span the target vector.

4.3 Private support vector machine

Support vector machines (SVM) are supervised machine learning methods that analyze data and recognize patterns. SVM is used for classification or regression analysis in numerous applications, such as chemistry, bioinformatics, handwriting recognition, text and data mining. Given points in space that have binary labels, the SVM algorithm computes a binary classifier by finding a hyperplane that optimally separates the data into two classes. If it finds no linear separation in the original space where the points are, the algorithm maps the points into a higher dimensional space where linear separation is possible. Since mapping the points into a higher-dimensional space has a large computational toll, the mappings used by SVM schemes are designed to ensure that inner products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $K(\mathbf{x}_1, \mathbf{x}_2)$.

Assume that the training set consists of n points, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, in an r-dimensional space X. Each of these points has a binary label $y_i \in \{-1, 1\}$. Let $\langle \cdot, \cdot \rangle$ be an inner product in X. Then if the SVM algorithm was able to find a linear separation of the training points within the original space X, it issues a binary classifier of the form

$$C(\mathbf{x}) = \mathsf{sign}(f(\mathbf{x})) \tag{8}$$

where the separating function is

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b\,,$$

and α_i and b are parameters that it learned from the training data. If it did not find a linear separation, it constructs a kernel function $K(\cdot, \cdot)$ on $X \times X$, and then the separating function takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\,. \tag{9}$$

4.3.1 Private SVM

Assume that Alice has n pairs of training points $(\{\mathbf{x}_1, y_1\}, \ldots, \{\mathbf{x}_n, y_n\})$, as described above. Assume that based on the training data, Alice learns a binary classifier of the form (8)+(9), where $K(\cdot, \cdot)$ is a kernel function which is a polynomial of degree d. Therefore, the separating function $f(\mathbf{x})$ is a polynomial of degree d in $\mathbf{x}$. Next, assume that Bob has a point $\boldsymbol{\alpha} \in X$ and he wishes to classify it, using the classifier that Alice learnt from her training data, without revealing $\boldsymbol{\alpha}$ to Alice. On the one hand, Alice does not want to send the separation function $f(\cdot)$ to Bob, since she charges fees for providing classification services and then, if she sent the function to Bob, Bob would be able to start his own business that would compete with hers; she wants to enable Bob to compute $f(\boldsymbol{\alpha})$ without learning the separation function f. On the other hand, Bob wishes to keep private the point $\boldsymbol{\alpha}$ which he needs to classify.

If we relax the requirements by allowing Bob to learn the value of $f(\boldsymbol{\alpha})$ rather than the final class label $C(\boldsymbol{\alpha}) = \mathsf{sign}(f(\boldsymbol{\alpha}))$, the problem may be solved by invoking an OMPE protocol, where Alice's input is $f(\cdot)$ and Bob's input is $\boldsymbol{\alpha}$. It should be noted that $f(\cdot)$ is typically a real-valued function, while the OMPE framework is defined over finite fields. Hence, all real values should

be discretized according to the required level of precision, and then they can be easily embedded within a finite field $\mathbb{F}$.

The above described relaxation enables Bob to learn more information than just the class lable; e.g., he may learn the distance of $f(\boldsymbol{\alpha})$ from the hyperplane, a value that provides an indication of how far $\boldsymbol{\alpha}$ is from the boundary that separates between the two classes. In order to achieve full privacy, a slight modification of the basic OMPE protocol is needed. Assume that the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, as well as Bob's point $\mathbf{x}$, are confined to some large box B in X, and let $M := \max_{\mathbf{x} \in B} |f(\mathbf{x})|$. Let p be a large prime which is larger than $2M$. Then Alice and Bob perform the OMPE protocol over a finite field $\mathbb{F}$ of size p. Alice selects at random any integer $0 \leq r < p/2$ and then sets her input to the protocol to be $f_r(\cdot) = f(\cdot) + r$. Consequently, Bob learns $f_r(\boldsymbol{\alpha})$. Then, Alice and Bob invoke a secure comparison protocol [19] where Bob learns whether $f_r(\boldsymbol{\alpha}) > r$ or not.

References

[1] M.J. Atallah and W. Du. Secure multi-party computational geometry. In *WADS*, pages 165–179, 2001.

[2] G. Brassard, C. Crépeau, and J.M. Robert. All-or-nothing disclosure of secrets. In *CRYPTO*, pages 234–238, 1986.

[3] E.F. Brickell. Some ideal secret sharing schemes. *J. of Combin. Math. and Combin. Comput.*, 6:105–113, 1989.

[4] J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.

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

[6] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28:637–647, 1985.

[7] R. Fagin, M. Naor, and P. Winkler. Comparing information without leaking it. *Communications of the ACM*, 39:77–85, 1996.

[8] N. Gilboa. *Topics in Private Information Retrieval.* PhD thesis, Technion - Israel Institute of Technology, 2000.

[9] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.

[10] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[11] O. Goldreich and R. Vainish. How to solve any protocol problem — an efficiency improvement. In *CRYPTO*, volume 293, pages 73–86, 1988.

[12] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, pages 265–282, 2007.

[13] M. Karchmer and A. Wigderson. On span programs. In *CCC*, pages 102–111, 1993.

[14] J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[15] M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35:1254–1281, 2006.

[16] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[17] M.O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[18] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644, 2002.

[19] A.C. Yao. Protocols for secure computation. In *FOCS*, pages 160–164, 1982.