Secure Efficient Multiparty Computing of Multivariate Polynomials and Applications

Dana Dachman-Soled[1], Tal Malkin[1], Mariana Raykova[1], and Moti Yung[2]

[1] Columbia University
[2] Columbia University and Google Inc.

Abstract. We present a robust secure methodology for computing functions that are represented as multivariate polynomials where parties hold different variables as private inputs. Our generic efficient protocols are fully black-box and employ threshold additive homomorphic encryption. They do not assume honest majority, yet are robust and can detect any misbehavior. We achieve a solution that both, takes advantage of the algebraic structure of the polynomials, and is polynomial-time in all parameters (security parameter, polynomial size, polynomial degree, number of parties). It further exploits a "round table" communication paradigm to reduce the complexity in the number of parties.
A large collection of problems are naturally and efficiently represented as multivariate polynomials over a field or a ring: problems from linear algebra, statistics, logic, as well as operations on sets represented as polynomials. In particular, we present a new efficient solution to the multi-party set intersection problem, and a solution for a multi-party variant of the polynomial reconstruction problem.
Keywords: secure multiparty computation, multivariate polynomial evaluation, additive homomorphic encryption, threshold cryptosystems, secret sharing, multiparty set intersection.

1 Introduction

Secure multiparty computation (MPC) allows mutually distrustful parties to jointly compute a functionality while keeping their inputs private. Seminal feasibility results for the two-party and multi-party settings have been demonstrated in [3,8,23,28,37,38]. These results show that any functionality can be securely computed in time polynomial in the size of its Boolean [8, 23, 37, 38] or arithmetic [1, 3, 7, 28] circuit representation.

While the works above yield strong feasibility results, the generic approaches typically lead to inefficient implementations since the circuit size representation of a functionality may be very large. Thus, an important open problem in MPC is designing highly efficient protocols for smaller, yet large enough to be interesting, sets of functionalities, taking advantage of the domain specific mathematical structure.

Problem Statement. We consider the problem of secure multiparty computation functions that can be represented by *polynomial-size multivariate polynomials*. Each party's inputs correspond to some subset of the variables in the polynomial representation. There is a designated party receiving output that learns only the output of the polynomial evaluation while all other parties receive no output. [3] We assume a broadcast channel and that the private keys for the threshold encryption scheme distributed in a preprocessing stage.

Our Results: The General Protocol. We present a protocol that allows multiple parties to compute the above functionalities, assuring security against a dishonest majority and robustness (detection misbehavior). Our protocol is fully black-box assuming any threshold additive homomorphic encryption with a natural property that we specify later, (instantiated by Paillier scheme, say). The protocol utilizes a "round table" structure where parties are nodes in a ring network (which means that frequently a party only communicates with the consecutive parties around the table). This structure (employed already in past protocols) has two benefits: first, it allows each party to be offline for the majority of the execution of the protocol and to be involved only when it needs to contribute its inputs at its turn. Second, it allows a division of the communication complexity into two types: "round table" communication complexity including messages exchanged between two neighboring parties, and broadcast communication complexity including messages sent simultaneously to all parties. We give simulation-based proofs of security in the Ideal/Real (standard) Model as per definitions in [24].

To the best of our knowledge, the only paper that has considered secure computation of multivariate polynomials is [18]. This recent independent work has focused on multivariate polynomials of degree 3 but points out that the proposed protocols can be generalized to higher degree polynomials, however, with communication complexity that is no longer optimal, leaving as an open question improvements of this complexity. Their protocol is based on the compiler of [27], but with the difference being that the outer and the inner protocols inhere are instantiated with efficient constructions tailored for multivariate polynomials. The communication complexity of their protocol is (sub)-exponential in the number of variables t: $O(poly(k)d^{\lfloor t/2 \rfloor})$ for polynomials of degree d and security parameter k. Our work, in turn, improves their communication complexity to be fully polynomial (i.e., polynomial in all pa-

[3] We note that our protocol can be generalized to allow any subset of the parties to receive output.

rameters of the problem). Clearly, one can take a poly-size multivariate polynomial and translate it to a circuit with poly time secure computation soultion, but this will have a huge polynomial factor expansion and will lose the structure enabling the special-purpose speedups. We achieve "round-table" complexity $10kDn(m-1)$ and broadcast complexity $k(10D+1)(\sum_{j=1}^{m}\sum_{t=1}^{l_j}\log\alpha_{j,t}+1)$ for m parties where party i has l_i inputs of degrees $\alpha_{i,1},\ldots,\alpha_{i,l_i}$, D being the sum of the logarithms of the variable degrees for polynomials consisting of n monomials. Next, since every polynomial can be easily converted into an arithmetic circuit, our protocol is also a protocol for MPC of a subclass of all arithmetic circuits. From this point of view, the work of [28] addresses a comparable problem to ours (constructing a MPC protocol for all poly-size arithmetic circuits, using a black-box construction and assuming no honest majority). The work of [18] already improves in the worst case the complexity results of [28] (for proper set of multivariate polynomials), and as we noted above we bring additional improvement (intuitively our amortized broadcast complexity is linear in the size of the representation of the largest term of the polynomial, and does not depend on the number of terms in the representation, which contributes to the size of the arithmetic circuit). Further, the protocol of [28] requires as many rounds (involving all the parties) as the depth of the circuit and communication complexity depending on the size of the circuit. In contrast, we achieve a number of rounds independent of depth of the size of the arithmetic circuit of the polynomial (and our round-complexity is actually constant when either counting a round-table round as one round or when considering only a constant number of parties).

Our Results: Special Cases. The class of polynomial size multivariate polynomials contains a wide range of efficiently representable functionalities with special structure that enables further optimizations. Most of the commonly used statistics functions can either be represented as polynomials or approximated with polynomials using Taylor series approximation for trigonometric functions, logarithms, exponents, square, etc. Examples include average, standard deviation, variance, chi-square test, Pearson's correlation coefficients, and central moment of statistical distributions. Matrix operations (i.e., linear algebra) can also be translated to polynomial evaluations.

In particular, as a special case of the general protocol, we implement *secure multiparty set intersection* against a malicious adversary controlling a majority of the parties; we note that the set intersection question in the two party case has been addressed in many works [11,12,20,26,29,30]

while there are fewer works that have considered the multiparty version. Two works adress the issue in the computational protocol setting. First, Kissner et al. [30] present a semi-honest protocol and suggests using generic zero communication complexity $O(m^2d^2)$ for m parties with input sets of size d. The work of [35] improves this complexity by a factor of $O(m)$ for m party protocols, using more efficient ZK based on pairings. In addition, relatively inefficient information theoretic solutions are presented in [33, 34]). Our protocol achieves communication complexity $O(md + 10d \log^2 d)$ improving the existing works. We achieve linear complexity in the number of parties m due to the round table communication paradigm, whereas even the recent unpublished work [9] is quadratic in the number of parties. We note that our scheme extends the approach of representing a set as the zeroes of a polynomial as in [2, 5, 14, 20].

Finally, when polynomial's coefficients correspond to the input of the designated receiver, we obtain a *multi-party oblivious multivariate polynomial evaluation*, a generalization of the problem of oblivious polynomials evaluation [31] to inputs from multiple parties.

Techniques. Many of our techniques exploit the "nice structure" of multivariate polynomials as well as various interactions of this structure with other algebraic and cryptographic primitives. First, we crucially utilize the fact that multivariate polynomials are linear operators when combined with additive homomorphic encryption and polynomial secret sharing. We formalize this property by presenting a commutativity property between the evaluation of multivariate polynomials and reconstruction of Shamir's secret sharing [36]. Intuitively, this allows us to evaluate a given polynomial on multiple (modified) Shamir secret shares in parallel and obtain the final evaluatation of the polynomial by reconstructing the resulting secret shares. This technique allows us to apply (black box) "cut-and-choose" techniques to verify the correctness of the evaluation, without revealing information about the shared inputs or outputs. We note that analogous techniques were used in a different context by [10,12].

A second property of multivariate polynomials is that they can be computed over additive homomorphic encryption non-interactively in a round-table type protocol where each participant incrementally contributes his inputs to the encryption of a monomial outputted by the previous participant (note that a participant's contribution to a given monomial amounts to multiplication of the encrypted monomial by a scalar).

We additionally use the polynomial structure of a variant of Shamir's threshold sharing in zero knowledge protocols proving that inputs were shared correctly and committed under homomorphic encryption. We uti-

lize Lagrange interpolation combined with what we call vector homomorphic encryption (where the homomorphic properties hold for both the plaintexts and the encryption randomness; which is true for many of the known homomorphic encryption schemes [15,17,25,32]) to verify that inputs were shared correctly by interpolating over encrypted values. This verifies that inputs were shared and encrypted correctly, provided that the randomness for the encryptions was chosen in a specific way. This encrypted interpolation technique combined with the large minimum distance of Reed-Solomon codes allows us to guarantee the correctness of an entire computation on encrypted codewords based on the verification that a small random subset of shares were computed correctly. Finally, we use the linear operator properties of the sharing polynomials for share re-randomization under additive homomorphic encryption.

We note that when we instantiate our protocol with homomorphic encryption over a ring, we apply the technique of Feldman ([16]) and also used, e.g., in Fouque et al. ([17]) for Paillier sharing that transforms interpolation over an RSA-composite ring to an interpolation over the integers (where computing inverses, i.e., division, is avoided and finding uninvertible elements is hard, assuming factoring is hard).

2 Protocol Overview

Semi-honest structure: As described above, multivariate polynomials can be computed over additive homomorphic encryption by a round-table protocol. This constitutes the underlying semi-honest evaluation protocol.
Robustness idea: To achieve security against malicious adversaries, we employ the commutativity between evaluation of multivariate polynomials and Shamir's secret sharing reconstruction described above. Consider the following simplified example that illustrates a our basic techniques. Let us say that we have m parties that wish to evaluate the univariate polynomial $\mathbf{Q}(x) = x^5+10x^3+6x+9$, at point x, where x is the committed input of Party 1. Note that allowing Party 1 to do the entire computation will not ensure that the outcome is consistent with the committed input. One possible solution is to require Party 1 to commit to its input x by encrypting x with a homomorphic encryption scheme, and have all parties compute the encrypted result using the homomorphic properties of the encryption, which is then decrypted. However, in order to compute all polynomial functions we will need a threshold doubly-homomorphic encryption scheme. Although Gentry, [22], recently introduced the first known doubly-homomorphic encryption scheme, a threshold analogue is not yet known.

Instead, we take the following approach: Party 1 computes a Shamir secret-sharing of its input x by choosing a polynomial P_x of degree k uniformly at random conditioned on $P_x(0) = x$. Now, instead of committing to the value x, Party 1 commits to, say, $20k$ input shares of $P_x : P_x(1), \ldots, P_x(20k)$. Next, Party 1 commits to $20k$ output shares of $\mathbf{Q} \circ P_x(i) : \mathbf{Q}(P_x(1)), \ldots, \mathbf{Q}(P_x(20k))$. Notice that $\mathbf{Q} \circ P_x(i)$ is a polynomial of degree $5k$ and that $\mathbf{Q} \circ P_x(0) = Q(P_x(0)) = Q(x)$. Thus, by reconstructing $\mathbf{Q} \circ P_x(0)$ we obtain the output value $\mathbf{Q}(x)$. After Party 1 sends the input and output commitments, the parties verify efficiently that the input and output shares indeed lie on a polynomial of degree k and $5k$ respectively using an interpolation algorithm we define below. Now, the parties run a cut-and-choose step where a set $I \subset [20k]$ of size k is chosen at random. For each index $i \in I$, Party 1 must open the commitments to reveal $P_x(i)$ and $\mathbf{Q} \circ P_x(i)$. The remaining parties now verify privately that $\mathbf{Q} \circ P_x(i)$ was computed correctly. Note that due to the secret-sharing properties of the commitment scheme, the cut-and-choose step reveals no information about $P_x(0) = x$ or $\mathbf{Q} \circ P_x(0) = \mathbf{Q}(x)$. Now, let's assume that Party 1 acted maliciously. Since the set I was chosen at random, and due to the large distance of Reed-Solomon codes, we show that if Party 1 is able to open all the shares corresponding to I correctly, then with very high probability Party 1 must have computed *all* of the output shares correctly. We note that the above description leaves out important **re-randomization techniques** (that are described in the full protocol) whose goal is to prevent parties from learning during the incremental evaluation and robustness checking.

Efficient Robustness: Although the technique described above is sufficient to ensure that the parties behave honestly, it induces a huge blow-up in the number of required shares. Indeed, in order to reconstruct the zero coefficient of a polynomial of degree deg, we must have at least $deg + 1$ secret shares. Thus, when evaluating a polynomial such as $\mathbf{Q} = x^{2^n}$, we would require an exponential number of shares. To prevent this blow-up, we employ an input preprocessing step (described in Section 3).

Secure output reconstruction: Finally, we use a threshold decryption algorithm to ensure that no subset of the parties can decrypt the intermediate messages exchanged. The threshold decryption is needed in the case that more than one party contributes its inputs to the polynomial (and is actually not necessary in our toy example above). Any additive homomorphic threshold encryption scheme (with one additional natural property, which we describe later) would suffice for the correctness of our protocol. Examples of such schemes are the El Gamal threshold encryption scheme

[21] and the Paillier threshold encryption scheme [17]. Note that additive El Gamal does not allow efficient decryption over a large domain, but it sufficies for our Set Intersection applications. We use the Paillier threshold encryption scheme to instantiate our general polynomial evaluation protocols. To obtain the final output, the designated party reconstructs the encryption of the final output value using Lagrange interpolation over encrypted values and decrypts with the help of the other parties.

3 Definitions and Building Block Protocols

We use a standard simulation-based definition of security see [6], and follow the definitions of zero knowledge proofs of knowledge and commitment schemes presented in [24].

Notation: We denote by Com_B a perfectly binding commitment scheme and by Com_H a perfectly hiding commitment scheme. Given $d+1$ evaluation points $(x_0, y_0), \ldots (x_d, y_d)$ on a polynomial of degree d, we denote the interpolation value at the point x as $L_{x_0,\ldots,x_d}(y_0, \ldots, y_d, x)$.

3.1 Vector Homomorphic Encryption

We require threshold additive homomorphic encryption scheme with the following additional property, capturing the fact that the homomorphism applies also to the randomness.[4] These property is satisfied by most known homomorphic encryption schemes: Paillier [32] and threshold Paillier [17], ElGamal [15], and Goldwasser-Micali [25].

Property 1. Let $E = (\mathrm{GEN}, \mathrm{ENC}, \mathrm{DEC})$ be an encryption scheme where the plaintexts come from a ring R_1 with operations $(+, \cdot)$, the randomness comes from a ring R_2 with operations $(\oplus, \odot)$, and the ciphertexts come from a ring R_3 with operations $(\otimes, \hat{})$. We say that E is *vector homomorphic* if the following holds: $\mathrm{ENC}(m_1; r_1) \otimes \mathrm{ENC}(m_2; r_2) = \mathrm{ENC}(m_1 + m_2; r_1 \oplus r_2)$ and $\mathrm{ENC}(m; r)^c = \mathrm{ENC}(c \cdot m; r \odot c)$.

3.2 Polynomial Code Commutativity

Shamir secret sharing [36]/Reed-Solomon codes are commutative with respect to polynomial evaluations, which we formalize as follows:

[4] We actually only need a slightly weaker property, but to simplify the presentation we assume our encryption scheme possesses the stronger property defined here.

Property 2 (Polynomial Code Commutativity). Let $\mathbf{Q}(x_1, \ldots, x_m)$ be a multivariate polynomial. Let $P_x(1), \ldots, P_x(t+1)$ be Shamir secret shares of a value x where P_x is a polynomial of degree t such $P_x(0) = x$. We can reconstruct x from its secret shares using Lagrange interpolation L. The evaluation of $\mathbf{Q}$ commutes with L in the sense that we can compute the value $\mathbf{Q}(x_1, \ldots, x_m)$ with either of the following algorithms:

$$\begin{aligned}
&(\mathbf{Q} * L)(P_{x_1}(1), \ldots P_{x_1}(t+1), \ldots P_{x_m}(1), \ldots P_{x_m}(t+1), 0) = \\
&= \mathbf{Q}((L(P_{x_1}(1), \ldots P_{x_1}(t+1), 0), \ldots, L(P_{x_m}(1), \ldots P_{x_m}(t+1), 0))) = \\
&= \mathbf{Q}(x_1, \ldots x_m),
\end{aligned}$$

where we first use L to retrieve the secrets and then evaluate $\mathbf{Q}$, or

$$\begin{aligned}
&(L * \mathbf{Q})(P_{x_1}(1), \ldots P_{x_1}(t+1), \ldots P_{x_m}(1), \ldots P_{x_m}(t+1), 0) = \\
&= L(\mathbf{Q}(P_{x_1}(1), \ldots, P_{x_m}(1)), \ldots, \mathbf{Q}(P_{x_1}(t+1), \ldots, P_{x_m}(t+1)), 0) = \\
&= L(w_1, \ldots, w_{t+1}, 0) = \mathbf{Q}(x_1, \ldots x_m),
\end{aligned}$$

where we evaluate $\mathbf{Q}$ on each set of shares of $x_1, \ldots, x_m$ to obtain shares of $\mathbf{Q}(x_1, \ldots, x_m)$ and then use L to reconstruct the final secret.

3.3 Incremental Encrypted Polynomial Evaluation

We will use homomorphic encryption to allow multiple parties to evaluate a multivariate polynomial depending on their inputs by incrementally contributing their inputs to partial encrypted evaluations of its monomials. This is facilitated by the following property:

Property 3 (Incremental Encrypted Polynomial Evaluation). Let m be the number of parties evaluating a multivariate polynomial $\mathbf{Q}$ defined by

$$\mathbf{Q}(x_{1,1}, \ldots, x_{1,l_1}, \ldots, x_{m,1}, \ldots, x_{m,l_m}) = \sum_{s=1}^{n} c_s(\prod_{j=1}^{m} h_{j,s}(x_{j,1}, \ldots, x_{j,l_j})),$$

where $h_{j,s}$ represents the inputs of party j to the s-th monomial of $\mathbf{Q}$. Let $E = (\mathrm{GEN}, \mathrm{ENC}, \mathrm{DEC})$ be an additive homomorphic encryption. We define the partial evaluations $b_{j,s}$ (including the contributions of parties 1,. . . , j) of the monomials $s, 1 \leq s \leq n$ of $\mathbf{Q}$ as follows:

$$b_{0,s} = \mathrm{ENC}(c_j) \text{ for } 1 \leq j \leq n, \text{ and } b_{j,s} = b_{j-1,s}^{h_{j,s}(x_{j,1}, \ldots, x_{j,l_j})} \text{ for } 1 \leq j \leq m$$

3.4 Polynomial Interpolation Over Encrypted Values

In this section we present a protocol that allows a verifier to verify (without help from the prover) that the prover's encrypted points lie on a polynomial of low degree, assuming the prover constructed the encryptions in a predetermined manner. Recall that Lagrange interpolation allows us, given $d+1$ points, to reconstruct the polynomial of degree d that interpolates the given points. In the following, we use the fact that Lagrange interpolation can, in fact, be carried out over encrypted points when the known encryption used possesses the vector homomorphic Property 1. Since the encryption is over a ring we use Feldman's technique for shift interpolation by factorial [16].

Lagrange Interpolation Protocol Over Encrypted Values (LIPEV)

Input: $(1, \mathrm{ENC}_{pk}(y_1, r_1)), \ldots (A, \mathrm{ENC}_{pk}(y_A, r_A))$, d where $d+1 < A$,
Output: Verifier outputs Accept if there are polynomials $P_1 \in R_1[x], P_2 \in R_2[x]$ of degree at most d such that $y_j = P_1(j)$ for $1 \leq j \leq A$ and $r_j = P_2(j)$ (P_1 and P_2 are defined with respect to the operations in the respective rings) for $1 \leq j \leq A$.
Verification Protocol:

1. Let $\Delta = A!$.
2. Let $l_j(x) = \Delta \cdot \prod_{i=1, i \neq j}^{d+1} \frac{x-i}{j-i}$ for $1 \leq j \leq d+1$.
3. Verifier checks whether $\mathrm{ENC}_{pk}(y_i, r_i)^{\Delta} = \prod_{j=1}^{d+1} (\mathrm{ENC}_{pk}(y_j, r_j))^{l_j(i)}$, and rejects otherwise.

Using the LIPEV protocol, a prover can prove to a verifier that A encrypted points lie on one polynomial of degree d, provided that the randomness for the encryptions was chosen in a specific way; namely, the random values chosen must also lie on a polynomial of degree d. For completeness, we describe next how to compute the random values for the encryptions so that they lie on a polynomial $P_2 \in R_2[x]$ of degree d. We note that even though the randomness for all A encrypted points are not chosen uniformly at random, semantic security is still preserved since the randomness for $d+1$ of the points is chosen uniformly at random and the remaining $A-d-1$ encryptions can be computed given only the first $d+1$ encryptions due to Property 1.

Randomness Interpolation

Input: $(1, y_1), \ldots, (A, y_A), r_1, \ldots, r_{d+1}$ for $d+1 < A$
Output: $r_{d+2}^{\Delta}, \ldots, r_A^{\Delta}$ such that the Lagrange Interpolation Protocol Over Encrypted Values outputs accept on the input: $[i, \mathrm{ENC}_{pk}(y_i, r_i)^{\Delta}]_{1 \le i \le A}, d$.

Protocol:

1. Compute $l_j(x) = \Delta \prod_{i=0, i \neq j}^{d+1} \frac{x-i}{j-i}$ for $1 \le j \le d+1$.
2. Compute $r_i^{\Delta} = \prod_{j=0}^{d+1} (r_j)^{l_j(i)}$ for $d+2 \le i \le A$.

4 Multiparty Polynomial Evaluation

The multiparty polynomial evaluation has the following setup:

- Each party T_j has l_j inputs $\overline{X}_j = \{x_{j,1}, \ldots, x_{j,l_j}\}$ for $1 \le j \le m$.
- A designated output receiver T^* (one of the parties $T_1, \ldots, T_m$).
- A polynomial $\mathbf{Q}(x_{1,1}, \ldots, x_{1,l_1}, \ldots, x_{m,1}, \ldots, x_{m,l_m})$, which depends on the inputs of all parties.

We use the following representation of the polynomial $\mathbf{Q}$:

$$\mathbf{Q}(x_{1,1}, \ldots, x_{1,l_1}, \ldots, x_{m,1}, \ldots, x_{m,l_m}) =$$
$$= \sum_{s=1}^{n} c_s x_{1,1}^{\alpha_{1,1,s}} \ldots x_{1,l_1}^{\alpha_{1,l_1,s}} \ldots x_{m,1}^{\alpha_{m,1,s}} \ldots x_{m,l_m}^{\alpha_{m,l_m,s}} = \sum_{s=1}^{n} c_s (\prod_{j=1}^{m} h_{j,s}).$$

where $h_{j,s}(x_{j,1}, \ldots, x_{j,l_j}) = x_{j,1}^{\alpha_{j,1,s}} x_{j,2}^{\alpha_{j,2,s}} \ldots x_{j,l_j}^{\alpha_{j,l_j,s}}$ and c_s is a known coefficients for $1 \le j \le m$. If $x_{j,v}$ does not participate in the s-th monomial of $\mathbf{Q}$, then $\alpha_{j,v,s} = 0$. Alternatively, we view $h_{j,s}$ for $1 \le j \le m$, $1 \le s \le n$ in the following way:

$$h_{j,s}(x_{j,1}, x_{j,1}^2, \ldots, x_{j,1}^{2^{\lfloor \log \alpha_{j,1,s} \rfloor}}, \ldots, x_{j,l_j}, x_{j,l_j}^2, \ldots, x_{j,l_j}^{2^{\lfloor \log \alpha_{j,l_j,s} \rfloor}}) \qquad (1)$$

in which each variable is of degree at most one.

Notation. In the protocol we will use the following variables: $D_{h,s} = \sum_{j=1}^{m} deg(h_{j,s})$ for $1 \le s \le n$; $D_{h,j,s} = k \sum_{v=1}^{j} deg(h_{v,s})$ for $1 \le j \le m$ where $h_{v,s}$ is defined as in Equation 1 (variables of degree at most 1); $D = \max_{s=1}^{n} D_{h,s}$. Also we let $\Delta = 10kD!$ be a public parameter, and $E = (\mathrm{GEN}, \mathrm{ENC}, \mathrm{DEC})$ be a threshold encryption scheme that possesses Property 1 with public key pk and secret keys $sk_1, \ldots, sk_m$ for the m parties $T_1, \ldots, T_m$.

Protocol Intuition. The protocol consists of four phases: *Input Preprocessing, Round-Table Step, Re-randomization, Verification and Reconstruction.* During the *Input Preprocessing* phase, the parties use techniques from [12] to transform $\mathbf{Q}$ from polynomial over the variables $x_{1,1}$, $\ldots, x_{1,l_1}, \ldots, x_{m,1}, \ldots, x_{m,l_m}$ into a polynomial of lower degree over the variables $x_{j,1}, x_{j,1}^2, \ldots, x_{j,1}^{2^{\max_{s=1}^{n} \lfloor \log \alpha_{j,1,s} \rfloor}}, \ldots, x_{j,l_j}, x_{j,l_j}^2, \ldots, x_{j,l_j}^{2^{\max_{s=1}^{n} \lfloor \log \alpha_{j,l_j,s} \rfloor}}$ for $1 \leq j \leq m$. Each party T_i commits to shares of its new inputs via the Efficient Preprocessing protocol described in the full version of the paper [13]. In the *Round Table Step* the parties compute the encrypted evaluations of the monomials in $\mathbf{Q}$ in a round-table fashion. Next, in the *Re-Randomization* phase, each party helps re-randomize the output shares. Honest behavior of the parties is checked during the *Verification* step via cut-and-choose and *Preprocessing Verification* protocol for the committed inputs, which is described in the full version of the paper [13]. If the verification passes, the parties jointly decrypt the output shares and the output receiver reconstructs the final polynomial evaluation result in the *Reconstruction* phase. We now present the detailed protocol and state our main theorem [5].

Multiparty Polynomial Evaluation Protocol Π_{poly_eval}

Inputs: $T_1 : \overline{X}_1, sk_1;\ \ldots;\ T_m : \overline{X}_m, sk_m$
Outputs: $T^* : \mathbf{Q}(x_{1,1}, \ldots, x_{1,l_1}, \ldots, x_{m,1}, \ldots, x_{m,l_m});\ \{T_1, \ldots, T_m\} \backslash T^* : \perp$

Input Preprocessing:

1. For $1 \leq j \leq m$, Party T_j converts each $h_{j,s}$ for $1 \leq s \leq n$ in the form of Equation 1 (each variable is of degree at most one).
2. For $1 \leq j \leq m$, Party T_j runs the Efficient Preprocessing protocol (see [13]) to generate $10kD$ shares for each of its new inputs $x_{j,1}, x_{j,1}^2, \ldots, x_{j,1}^{2^{\lfloor \log \alpha_{j,1} \rfloor}}, \ldots, x_{j,l_j}, x_{j,l_j}^2, \ldots, x_{j,l_j}^{2^{\lfloor \log \alpha_{j,l_j} \rfloor}}$ where $\alpha_{j,t} = \max_{i=1}^{n} \alpha_{j,t,i}$ for $1 \leq t \leq l_j$ and commits to the shares.

[5] We note that for each intermediate monomial $h_{j,s}$ passed between the parties in the round-table step, each Party j needs to transmit only $D_{h,j,s} + 1$ shares to Party $j+1$ since the rest of the shares may be constructed by the receiving party via Lagrange interpolation over committed values. This may yield significant savings in the communication complexity, which we assumed in our discussion in the introduction.

Round-Table Step:

3. Party T_1 computes encryptions of the polynomials coefficients $b_{0,i} = \mathrm{ENC}_{pk}(c_s; 1)$ for $1 \leq s \leq n$.
4. For $1 \leq s \leq n$, T_1 chooses $D_{h,1,s} + 1$ random numbers $r_1^{1,s}, \ldots, r_{D_{h,1,s}+1}^{1,s}$. T_1 uses the Randomness Interpolation protocol to compute $(r_{D_{h,1,s}+2}^{1,s})^{\Delta}, \ldots, (r_{10kD}^{1,s})^{\Delta}$.
5. For $1 \leq i \leq 10kD$, $1 \leq s \leq n$ T_1 uses the values chosen above to compute
$h_{1,s}(i) = h_{1,s}(P_{x_{1,1}}(i), \ldots, P_{x_{1,1}^{2^{\lfloor \log \alpha_{1,1,j} \rfloor}}}(i), \ldots, P_{x_{1,l_1}}(i), \ldots, P_{x_{1,l_1}^{2^{\lfloor \log \alpha_{1,l_1,s} \rfloor}}}(i))$
and $b_{1,s,i} = b_{0,j}^{\Delta \cdot h_{1,s}(i)} \cdot \mathrm{ENC}pk(0; r_i^1)^{\Delta}$, which he sends to party T_2.
6. For each $2 \leq j \leq m$:
   (a) Party T_j receives from party T_{j-1} coefficients $b_{j-1,1,i}, \ldots, b_{j-1,n,i}$ for $1 \leq i \leq 10kD$.
   (b) For $1 \leq s \leq n$, T_j chooses $D_{h,j,s} + 1$ random numbers $r_1^{j,s}, \ldots, r_{D_{h,j,s}+1}^{j,s}$. P_j uses the Randomness Interpolation protocol to compute $(r_{D_{h,j,s}+2}^{j,s})^{\Delta}, \ldots, (r_{10kD}^{j,s})^{\Delta}$.
   (c) For $1 \leq s \leq n$ T_j uses the values $r_i^{j,s}$ chosen above to compute
   $h_{j,s}(i) = h_{j,s}(P_{x_{j,1}}(i), \ldots, P_{x_{j,1}^{2^{\lfloor \log \alpha_{j,1} \rfloor}}}(i), \ldots, P_{x_{j,l_j}}(i), \ldots, P_{x_{j,l_j}^{2^{\lfloor \log \alpha_{j,l_j} \rfloor}}}(i))$
   and $b_{j,s,i} = b_{j-1,s,i}^{\Delta \cdot h_{j,s}(i)} \cdot \mathrm{ENC}_{pk}(0; r_i^{j,s})^{\Delta}$.
   (d) If $j < m$ T_j sends all $b_{j,s,i}$ to T_{j+1}. If $j = m$ for each $1 \leq i \leq 10kD$ T_m computes $S_i' = \Pi_{s=1}^{n} b_{m,s,i}$ and sends them to all parties on the broadcast channel.

Re-Randomization Step:

7. For $1 \leq j \leq m$, Party T_j computes polynomial $P_{j,0}$ of degrees kD such that $P_{j,0}(0) = 0$.
8. For $1 \leq j \leq m$, Party T_j chooses $kD + 1$ random values $r_{j,1}, \ldots, r_{j,kD+1}$ and uses the Randomness Interpolation protocol to compute $r_{j,kD+2}^{\Delta}, \ldots, r_{j,10kD}^{\Delta}$. T_j commits to shares $Z_{j,0} = \mathrm{ENC}_{pk}(P_{j,0}(i); r_{j,i})^{\Delta}$ for $1 \leq i \leq 10kD$
9. All parties run the LIPEV protocol and a zero knowledge proof protocol (HEPKPV, see [13]) to ensure that each $[Z_{j,i}]_{1 \leq i \leq 10kD}$ is an encryption of a polynomial with constant coefficient 0.
10. The final encryptions are: $S_i = S_i' \cdot \Pi_{j=1}^{m} Z_{j,0}$ for $1 \leq i \leq 10kD$.

Verification:

11. All parties verify using the Lagrange encrypted interpolation protocol that the values S_i lie on a polynomial of degree kD. Otherwise reject.
12. All parties run a multi-party coin-tossing protocol (see [13]) to choose a random subset I of size k from $[1, 10kD]$.
13. For each $i \in I$ parties $T_1, \ldots, T_m$ decommit their corresponding shares from the Efficient Input Preprocessing.
14. All parties run the Preprocessing Verification for their inputs (see [13]).
15. For each $i \in I$ each party T_j decommits the i-th shares of its inputs as well as the i-th share of the polynomials $P_{j,0}$. Additionally, each party T_j reveals the randomness $r_i^{j,s}$ for $1 \leq s \leq n$ and $r_{j,i}$ used for the corresponding shares. To verify, each party recomputes the entire share S_i^*, using the inputs and randomness revealed and checks that $S_i = S_i^*$. If any verification fails the protocol is aborted.

Reconstruction:

16. For each $1 \leq i \leq 10kD$ each party computes its partial decryption $s_{i,j}$ of S_i and sends it to the designated output receiver T^*.
17. Party T^* uses the partial decryptions $s_{i,j}$ for $1 \leq j \leq m$ to completely decrypt S_i. T^* reconstructs the value of $\mathbf{Q}(x_{1,1}, \ldots, x_{1,l_1}, \ldots, x_{m,1}, \ldots, x_{m,l_m})$ via interpolation and division by Δ^m.

Theorem 1. *If the Decisional Composite Residuosity problem is hard in $\mathbf{Z}_{n^2}^*$, where n is a product of two strong primes, and protocol Π_{poly_eval} is instantiated with the threshold Paillier encryption scheme TP_{enc}^m such that $E = TP_{enc}^m$, then Π_{poly_eval} securely computes the Polynomial Evaluation functionality in the presence of malicious adversaries.*

5 Communication and Computational Complexity

Our protocol computes the polynomial functionality in a constant number of rounds (counting round-table rounds as one, or with a constant number of players). The communication complexity of the protocol can be divided into two types: messages that are broadcast to all parties and the "round-table" communication that is passed between two consecutive parties. We note that the "round-table" communication can be done off-line. The broadcast communication consists of the commitments

of the inputs shares, the decommitments used in the final verification phase, the encrypted and decrypted output shares as well as the messages used in the coin tossing and HEPKPV protocols. These messages add up to $k(10D+1)(\sum_{j=1}^{m}\sum_{t=1}^{l_j}\log\alpha_{j,t}+1)$. Note that the communication complexity may be much smaller than the size of the polynomial representation. For example, if party P_j with input $x_{j,1}$ must contribute $\alpha_{j,t}$ consecutive powers of x_i: $x_i^1,\ldots,x_i^{\alpha_{j,t}}$ to $\alpha_{j,t}$ different terms, the broadcast communication complexity for this party will still only be $k(10D+1)\log\alpha_{j,t}+1$. The round-table messages passed between consecutive parties include all intermediate messages in the computation that are sent by the all the parties except the last one, which in total are $10kDn(m-1)$. The computational complexity (where we count number of exponentiations) for all m parties in total is $O(kDnm)$. Further if we apply the share packing optimization from Section 6 over k executions of the protocol we can drop k factor for the new amortized complexities.

Our protocol runs in constant number of "round table" rounds, in which every party in involved in order, while the protocol for secure computation of arithmetic circuits [28] requires as many rounds as the depth of the arithmetic circuit. It also requires fewer broadcasted messages compared to techniques proving the polynomial evaluation via zero knowledge proofs such as [4] since any ZK proof will have to be broadcasted. Additionally a ZK protocol will require runs of a multiparty coin tossing protocol to generate randomness for each ZK proof.

6 Protocol Optimizations and Application to Multiparty Set Intersection

6.1 Optimizations

We apply several optimizations to the protocol given in Section 4 for polynomials with specific structures. First, if we have a monomial that is computed only from the inputs of a subset of the parties, then clearly, we can evaluate it in a round-table fashion that only includes parties in this subset and proceed to the Re-Randomization Step.

Additionally, in some cases, we can remove the requirement of a party to share all of its inputs. Recall that we require the input-sharing in order to enable the cut-and-choose verification of honest behavior of the parties In the case when an input is used only once in the polynomial, this type of proof may not be necessary. We can avoid sharing an input if it belongs to the first party in the round table computation of the corresponding monomial as long as we can verify that the encryption

itself is valid with a ZKPOK and extract the encrypted value. We notice that the requirements imposed on the structure of the polynomial in order to be able to apply this optimization substantially limit the range of the possible polynomials. However, in the next section we will see how the problem of multiparty set intersection can be reduced to the evaluation of exactly this type of polynomials.

Finally, we use the approach of multi-secret sharing from [19] that allows us to use the same polynomials to share the input values for multiple parallel executions of the protocol, which lowers the amortized communication complexity of our protocol. Intuitively, we choose a set of points on the sharing polynomials to represent the input values for each of the different executions of the protocol, say points 1 to k for each of k different executions. The shares that will be used in the computation will be those corresponding to points not in this set. As a result, the final output polynomial will evaluate to each of the different output values corresponding to each execution at the points 1 to k respectively.

In the setting of our protocol in Section 4 we assume that the multivariate polynomial are known to all parties. By removing this requirement and assuming that the polynomial coefficients are the inputs of one of the parties, we reduce the problem to oblivious multivariate polynomial evaluation (introduced by [31] in the single-variable case) for a small class of multivariate polynomials.

6.2 Multiparty Set Intersection

We apply the techniques introduced in Section 4 to the problem of multiparty set intersection. We give here a brief sketch. In the multiparty set intersection problem, there are m parties $T_1, \ldots, T_m$ who have input sets $X_1, \ldots, X_m$ and wish to jointly compute $X_1 \bigcap \ldots \bigcap X_m$. While there are several papers that address the set intersection problem in the two-party case [2, 5, 20, 26, 29], the generalization to the multiparty setting has been considered only in [14]. Recall that a set $X = \{x_1, \ldots, x_d\}$ can be represented as a polynomial $P(x) = (x - x_1) \ldots (x - x_d)$. Now if we consider the polynomial $P'(x) = r \cdot P(x) + x$, where r is random, we have that if $x' \in X$ then $P'(x') = x'$ and if $x' \notin X$ then $P'(x')$ is uniformly distributed (see [20]). In the multiparty case we have m parties with input sets $X_1, \ldots, X_m$, represented by polynomials $P_{X_1}(x), \ldots, P_{X_m}(x)$. Thus the polynomial $\mathbf{R}(x) = \mathbf{r} \cdot \sum_{i=1}^{m-1} P_{X_i}(x) + x$, where $\mathbf{r} = r_1 + r_2 + \cdots + r_m$ and each r_i is a randomly chosen input contributed by Party i, will have the same property mentioned above: if $x' \in X_1 \bigcap \ldots \bigcap X_m$ then $\mathbf{R}(x') = x'$ and if $x' \notin X_1 \bigcap \ldots \bigcap X_m$ then $\mathbf{R}(x')$ is uniformly distributed. Now in the

setting of polynomial evaluation we let a designated party P_m evaluates $\mathbf{R}(x)$ on its own inputs, and thus the output is exactly the intersection of all sets with some additional random values. This problem now reduces to the Multiparty Polynomial Evaluation problem.

Theorem 2. *If the Decisional Composite Residuosity problem is hard in $\mathbf{Z}^*_{n^2}$, where n is a product of two strong primes, protocol Π_{poly_eval} is instantiated with the threshold Paillier encryption scheme TP^m_{enc} such that $E = TP^m_{enc}$, and $\mathbf{Q} = \mathbf{R}$, then Π_{poly_eval} securely computes the Set Intersection functionality* [6] *in the presence of malicious adversaries.*

Using the optimizations described in this section, we have that the broadcast communication complexity of the Set Intersection protocol is $O(md+10d\log^2 d)$ (there is no round-table communication) and the computational complexity is $O(md^2)$, where $d >> k$ is the maximum input set size of each party.

References

1. Abadi, M., Feigenbaum, J.: Secure circuit evaluation. J. Cryptol. 2(1), 1–12 (1990)
2. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. pp. 86–97. ACM, New York, NY, USA (2003)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (1988)
4. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: In EUROCRYPT 1999. pp. 107–122 (1999)
5. Camenisch, J., Zaverucha, G.: Private intersection of certified sets. In: Proceedings of Financial Cryptography '09 (2009)
6. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13, 2000 (2000)
7. Canetti, R., Ishai, Y., Kumar, R., Reiter, M.K., Rubinfeld, R., Wright, R.N.: Selective private function evaluation with applications to private statistics. In: PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing. pp. 293–304. ACM, New York, NY, USA (2001)
8. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 11–19. ACM, New York, NY, USA (1988)

[6] We consider here a slight variant of the Set Intersection functionality where Party i for $1 \leq i \leq m-1$ submits the polynomial P_{X_i} to the Trusted Party, Party m submits X_m and the Trusted Party returns the intersection of $X_1, \ldots, X_m$. In order to compute the standard Set Intersection functionality, we must use the threshold El Gamal encryption scheme.

9. Cheon, J.H., Jarecki, S., Seo, J.H.: Multi-party privacy-preserving set intersection with quasi-linear complexity. Cryptology ePrint Archive, Report 2010/512 (2010), http://eprint.iacr.org/
10. Choi, S., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC (2008)
11. Cristofaro, E.D., Kim, J., Tsudik, G.: Linear-complexity private set intersection protocols secure in malicious model. In: ASIACRYPT. pp. 213–231 (2010)
12. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. In: ACNS. pp. 125–142 (2009)
13. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Multiparty secure computation over multivariate polynomials. Technical Report CUCS-024-10 (2010)
14. Dawn, L.K., Song, D.: Privacy-preserving set operations. In: in Advances in Cryptology - CRYPTO 2005, LNCS. pp. 241–257. Springer (2005)
15. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Proceedings of CRYPTO 84 on Advances in cryptology. pp. 10–18. Springer-Verlag New York, Inc., New York, NY, USA (1985)
16. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS. pp. 427–437. ACM, New York, NY, USA (1987)
17. Fouque, P.A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: FC '00: Proceedings of the 4th International Conference on Financial Cryptography. pp. 90–104. Springer-Verlag, London, UK (2001)
18. Franklin, M., Mohassel, P.: Efficient and secure evaluation of multivariate polynomials and applications. In: Applied Cryptography and Network Security. vol. 6123, pp. 236–254 (2010)
19. Franklin, M., Yung, M.: Communication complexity of secure computation (extended abstract). In: STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing. pp. 699–710 (1992)
20. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Proceedings of EUROCRYPT'04 (2004)
21. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. 20(1), 51–83 (2007)
22. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing. pp. 169–178. ACM, New York, NY, USA (2009)
23. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing. pp. 218–229. ACM, New York, NY, USA (1987)
24. Goldreich, O.: Foundations of cryptography: a primer. Found. Trends Theor. Comput. Sci. 1(1), 1–116 (2005)
25. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing. pp. 365–377. ACM, New York, NY, USA (1982)
26. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: TCC. pp. 155–175 (2008)
27. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer — efficiently. In: CRYPTO 2008: Proceedings of the 28th Annual conference on Cryptology. pp. 572–591 (2008)

28. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: TCC '09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography. pp. 294–314 (2009)
29. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In: TCC. pp. 577–594 (2009)
30. Kissner, L., Song, D.X.: Privacy-preserving set operations. In: CRYPTO. pp. 241–257 (2005)
31. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006)
32. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: In Proceedings of EUROCRYPT 1999. pp. 223–238. Springer-Verlag (1999)
33. Patra, A., Choudhary, A., Rangan, C.: Information theoretically secure multi party set intersection re-visited. In: Jacobson, M., Rijmen, V., Safavi-Naini, R. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 5867, pp. 71–91. Springer Berlin / Heidelberg (2009)
34. Patra, A., Choudhary, A., Rangan, C.P.: Round efficient unconditionally secure mpc and multiparty set intersection with optimal resilience. In: Proceedings of the 10th International Conference on Cryptology in India: Progress in Cryptology. pp. 398–417. INDOCRYPT '09 (2009)
35. Sang, Y., Shen, H.: Efficient and secure protocols for privacy-preserving set operations. ACM Trans. Inf. Syst. Secur. 13, 9:1–9:35 (November 2009)
36. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
37. Yao, A.C.C.: Protocols for secure computations. In: FOCS. pp. 160–164 (1982)
38. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS. pp. 162–167 (1986)