Privacy-Preserving Set Operations

Lea Kissner and Dawn Song

Carnegie Mellon University, Pittsburgh PA 15213
leak@cs.cmu.edu, dawnsong@cmu.edu

Abstract. In many important applications, a collection of mutually distrustful parties must perform private computation over multisets. Each party's input to the function is his private input multiset. In order to protect these private sets, the players perform *privacy-preserving* computation; that is, no party learns more information about other parties' private input sets than what can be deduced from the result. In this paper, we propose efficient techniques for privacy-preserving operations on multisets. By building a framework of multiset operations, employing the mathematical properties of polynomials, we design efficient, secure, and composable methods to enable privacy-preserving computation of the *union, intersection,* and *element reduction* operations. We apply these techniques to a wide range of practical problems, achieving more efficient results than those of previous work.

1 Introduction

Private computation over sets and multisets is required in many important applications. In the real world, parties often resort to use of a trusted third party, who computes a fixed function on all parties' private input multisets, or forgo the application altogether. This unconditional trust is fraught with security risks; the trusted party may be dishonest or compromised, as it is an attractive target. We design efficient *privacy-preserving* techniques and protocols for computation over multisets by mutually distrustful parties: no party learns more information about other parties' private input sets than what can be deduced from the result of the computation.

For example, to determine which airline passengers appear on a 'do-not-fly' list, the airline must perform a set-intersection operation between its private passenger list and the government's list. This is an example of the Set-Intersection problem. If a social services organization needs to determine the list of people on welfare who have cancer, the union of each hospital's lists of cancer patients must be calculated (but not revealed), then an intersection operation between the unrevealed list of cancer patients and the welfare rolls must be performed. This problem may be efficiently solved by composition of our private union and set-intersection techniques. Another example is privacy-preserving distributed network monitoring. In this scenario, each node monitors anomalous local traffic, and a distributed group of nodes collectively identify popular anomalous behaviors: behaviors that are identified by at least a threshold t number of monitors. This is an example of the Over-Threshold Set-Union problem.

Contributions. In this paper, we propose efficient techniques for privacy-preserving operations on multisets. By building a framework of set operations using polynomial representations and employing the mathematical properties of polynomials, we design efficient methods to enable privacy-preserving computation of the *union, intersection, and element reduction*[1] multiset operations.

An important feature of our privacy-preserving multiset operations is that they can be composed, and thus enable a wide range of applications. To demonstrate the power of our techniques, we apply our operations to solve specific problems, including Set-Intersection (Section 5) and Over-Threshold Set-Union (Section 6). We also discuss a number of other applications in Section 7, such as constructing protocols for the composition of multiset operations, computing Threshold Set-Union and Cardinality Set-Intersection, and determining the Subset relation. Due to space constraints, we describe utilization of our techniques for efficiently and privately evaluating CNF boolean functions in [18].

Our protocols are more efficient than the results obtained from previous work. General multiparty computation is the best previous result for most of the problems that we address in this paper. Only the private Set-Intersection problem and two-party Cardinality Set-Intersection problem have been previously studied [12]. However, previous work only provides protocols for 3-or-more-party Set-Intersection secure only against honest-but-curious players; it is not obvious how to extend this work to achieve security against malicious players. Also, previous work focuses on achieving results for the Set-Intersection problem in isolation – these techniques cannot be used to compose set operations. In contrast, we provide efficient solutions for private multi-party Set-Intersection secure against malicious players [18], and our multiset intersection operator can be easily composed with other operations to enable a wide range of efficient private computation over multisets.

Our protocols are provably secure in the PPT-bounded adversary model. We consider both standard adversary models: honest-but-curious adversaries (HBC) and malicious adversaries. We prove the security of each of our protocols in the full version of this paper [18].

We discuss related work in Section 2. In Section 3, we introduce our adversary models, as well as our cryptographic tools. We describe our privacy-preserving set operation techniques in in Section 4. Section 5 gives a protocol and security analysis for the Set-Intersection problem, and Section 6 gives a protocol and security analysis for the Over-Threshold Set-Union problem. We discuss several additional applications of our techniques in Section 7.

2 Related Work

For most of the privacy-preserving set function problems we address in this paper (except for the Set-Intersection problem), the best previously known results

[1] The *element reduction* by d, $\mathrm{Rd}_d(A)$, of a multiset A is the multiset composed of the elements of A such that for every element a that appears in A at least $d' > d$ times, a is included $d' - d$ times in $\mathrm{Rd}_d(A)$.

are through general multiparty computation. General two-party computation was introduced by Yao [25], and general computation for multiple parties was introduced in [1]. In general multiparty computation, the players share the values of each input, and cooperatively evaluate the circuit. For each multiplication gate, the players must cooperate to securely multiply their inputs and re-share the result, requiring $O(n)$ communication for honest-but-curious players and $O(n^2)$ communication for malicious players [15]. Recent results that allow non-interactive private multiplication of shares [7] do not extend to our adversary model, in which any $c < n$ players may collude. Our results are more efficient than the general MPC approach; we compare communication complexity in [18].

The most relevant work to our paper is by Freedman, Nissim, and Pinkas (FNP) [12]. They proposed protocols for the problems related to Set-Intersection, based on the representation of sets as roots of a polynomial [12]. Their work does not utilize properties of polynomials beyond evaluation at given points. We explore the power of polynomial representation of multisets, using operations on polynomials to obtain composable privacy-preserving multisets operations.

Much work has been done in designing solutions for privacy-preserving computation of different functions. For example, private equality testing is the problem of set-intersection for the case in which the size of the private input sets is 1. Protocols for this problem are proposed in [9, 21, 19], and fairness is added in [2]. We do not enumerate the works of privacy-preserving computation of other functions here, as they address drastically different problems and cannot be applied to our setting.

3 Preliminaries

In this section, we describe our adversary models and the cryptographic tools used in this paper.

3.1 Adversary Models

In this paper, we consider two standard adversary models: honest-but-curious adversaries and malicious adversaries. Due to space constraints, we only provide intuition and informal definitions of these models. Formal definitions of these models can be found in [15].

Honest-But-Curious Adversaries. In this model, all parties act according to their prescribed actions in the protocol. Security in this model is straightforward: no player or coalition of $c < n$ players (who cheat by sharing their private information) gains information about other players' private input sets, other than what can be deduced from the result of the protocol. This is formalized by considering an ideal implementation where a trusted third party (TTP) receives the inputs of the parties and outputs the result of the defined function. We require that in the real implementation of the protocol—that is, one without a TTP—each party does not learn more information than in the ideal implementation.

Malicious Adversaries. In this model, an adversary may behave arbitrarily. In particular, we cannot hope to prevent malicious parties from refusing to participate in the protocol, choosing arbitrary values for its private input set, or aborting the protocol prematurely. Instead, we focus on the standard security definition (see, e.g., [15]) which captures the correctness and the privacy issues of the protocol. Informally, the security definition is based on a comparison between the ideal model and a TTP, where a malicious party may give arbitrary input to the TTP. The security definition is also limited to the case where at least one of the parties is honest. Let Γ be the set of colluding malicious parties; for any strategy Γ can follow in the real protocol, there is a translated strategy that it could follow in the ideal model, such that, to Γ, the real execution is computationally indistinguishable from execution in the ideal model.

3.2 Additively Homomorphic Cryptosystem

In this paper we utilize a semantically secure [16], additively homomorphic public-key cryptosystem. Let $E_{pk}(\cdot)$ denote the encryption function with public key pk. The cryptosystem supports the following operations, which can be performed without knowledge of the private key: (1) Given the encryptions of a and b, $E_{pk}(a)$ and $E_{pk}(b)$, we can efficiently compute the encryption of $a+b$, denoted $E_{pk}(a+b) := E_{pk}(a) +_h E_{pk}(b)$; (2) Given a constant c and the encryption of a, $E_{pk}(a)$, we can efficiently compute the encryption of ca, denoted $E_{pk}(c \cdot a) := c \times_h E_{pk}(a)$. When such operations are performed, we require that the resulting ciphertexts be re-randomized for security. In re-randomization, a ciphertext is transformed so as to form an encryption of the same plaintext, under a different random string than the one originally used. We also require that the homomorphic public-key cryptosystem support secure (n,n)-threshold decryption, i.e., the corresponding private key is shared by a group of n players, and decryption must be performed by *all* players acting together.

In our protocols for the malicious case, we require: (1) the decryption protocol be secure against malicious players, typically, this is done by requiring each player to prove in zero-knowledge that he has followed the threshold decryption protocol correctly [14]; (2) efficient construction of zero-knowledge proofs of plaintext knowledge; (3) optionally, efficient construction of certain zero-knowledge proofs, as detailed in [18].

Note that Paillier's cryptosystem [23] satisfies each of our requirements: it is additively homomorphic, supports ciphertexts re-randomization and threshold decryption (secure in the malicious case) [10, 11], and allows certain efficient zero-knowledge proofs (standard constructions from [5, 3], and proof of plaintext knowledge [6]).

In the rest of this paper, we simply use $E_{pk}(\cdot)$ to denote the encryption function of the homomorphic cryptosystem which satisfies all the aforementioned properties.

4 Techniques and Mathematical Intuition

In this section, we introduce our techniques for privacy-preserving computation of operations on multisets.

Problem Setting. Let there be n players. We denote the private input set of player i as S_i, and $|S_i| = k$ $(1 \leq i \leq n)$. We denote the jth element of set i as $(S_i)_j$. We denote the domain of the elements in these sets as P, $(\forall_{i \in [n], j \in [k]}\ (S_i)_j \in P)$.

Let R denote the plaintext domain $\mathrm{Dom}(E_{pk}(\cdot))$ (in Paillier's cryptosystem, R is Z_N). We require that R be sufficiently large that an element a drawn uniformly from R has only negligible probability of *representing an element of* P, denoted $a \in P$. For example, we could require that only elements of the form $b = a \mid\mid h(a)$ could represent an element in P, where $h(\cdot)$ denotes a cryptographic hash function [20]. That is, there exists an a of proper length such that $b = a \mid\mid h(a)$. If $|h(\cdot)| = \lg\left(\frac{1}{\epsilon}\right)$, then there is only ϵ probability that $a' \leftarrow R$ represents an element in P.

In this section, we first give background on polynomial representation of multisets, as well as the mathematical properties of polynomials that we use in this paper. We then introduce our privacy-preserving (TTP model) set operations using polynomial representations, then show how to achieve privacy in the real setting by calculating them using encrypted polynomials. Finally, we overview the applications of these techniques explored in the rest of the paper.

4.1 Background: Polynomial Rings and Polynomial Representation of Sets

The polynomial ring $R[x]$ consists of all polynomials with coefficients from R. Let $f, g \in R[x]$, such that $f(x) = \sum_{i=0}^{\deg(f)} f[i]x^i$, where $f[i]$ denotes the coefficient of x^i in the polynomial f. Let $f + g$ denote the addition of f and g, $f * g$ denote the multiplication of f and g, and $f^{(d)}$ denote the dth formal derivative of f. Note that the *formal derivative* of f is $\sum_{i=0}^{\deg(f)-1}(i+1)f[i+1]x^i$.

Polynomial Representation of Sets. In this paper, we use polynomials to represent multisets. Given a multiset $S = \{S_j\}_{1 \leq j \leq k}$, we construct a polynomial representation of S, $f \in R[x]$, as $f(x) = \prod_{1 \leq j \leq k}(x - S_j)$. On the other hand, given a polynomial $f \in R[x]$, we define the multiset S represented by the polynomial f as follows: an element $a \in S$ if and only if (1) $f(a) = 0$ and (2) a represents an element from P. Note that our polynomial representation naturally handles multisets: The element a appears in the multiset b times if $(x-a)^b \mid f \wedge (x-a)^{b+1} \not| f$.

Note that previous work has proposed to use polynomials to represent sets [12] (as opposed to multisets). However, to the best of our knowledge, previous work has only utilized the technique of polynomial evaluation for privacy-preserving operations. As a result, previous work is limited to set intersection and cannot be composed with other set operators. In this paper, we propose a framework to perform various set operations using polynomial representations and construct efficient privacy-preserving set operations using the mathematical properties of polynomials. By utilizing polynomial representations as the intermediate form of representations of sets, our framework allows arbitrary composition of set operators as outlined in our grammar.

4.2 Our Techniques: Privacy-Preserving Set Operations

In this section, we construct algorithms for computing the polynomial representation of operations on sets, including union, intersection, and element reduction. We design these algorithms to be privacy-preserving in the following sense: the polynomial representation of any operation result reveals no more information than the set representation of the result. First, we introduce our algorithms for computing the polynomial representation of set operations union, intersection, and element reduction (with a trusted third party). We then extend these techniques to encrypted polynomials, allowing secure implementation of our techniques without a trusted third party. Note that the privacy-preserving set operations defined in this section may be *arbitrarily composed* (see Section 7), and constitute truly general techniques.

Set Operations Using Polynomial Representations. In this section, we introduce efficient techniques for set operations using polynomial representations. In particular, let f, g be polynomial representations of the multisets S, T. We describe techniques to compute the polynomial representation of their union, intersection, and element reduction by d. We design our techniques so that the polynomial representation of any operation result reveals no more information than the set representation of the result. This privacy property is formally stated in Theorems 1, 2, and 3, by comparing to the ideal model.

Union. We define the union of multisets $S \cup T$ as the multiset where each element a that appears in S $b_S \geq 0$ times and T $b_T \geq 0$ times appears in the resulting multiset $b_S + b_T$ times. We compute the polynomial representation of $S \cup T$ as follows, where f and g are the polynomial representation of S and T respectively:

$$f * g.$$

Note that $f * g$ is a polynomial representation of $S \cup T$ because (1) all elements that appear in either set S or T are preserved: $(f(a) = 0) \wedge (g(b) = 0) \rightarrow ((f*g)(a) = 0) \wedge ((f*g)(b) = 0)$; (2) as $f(a) = 0 \Leftrightarrow (x-a) \mid f$, duplicate elements from each multiset are preserved: $(f(a) = 0) \wedge (g(a) = 0) \rightarrow (x-a)^2 \mid (f * g)$. In addition, we prove that, given $f * g$, one cannot learn more information about S and T than what can be deduced from $S \cup T$, as formally stated in the following theorem:

Theorem 1. *Let* TTP1 *be a trusted third party which receives the private input multiset* S_i *from player* i *for* $1 \leq i \leq n$, *and then returns to every player the union multiset* $S_1 \cup \cdots \cup S_n$ *directly. Let* TTP2 *be another trusted third party,* *which receives the private input multiset S_i from player i for $1 \le i \le n$, and then: (1) calculates the polynomial representation f_i for each S_i; (2) computes and returns to every player $\prod_{i=1}^{n} f_i$.*

There exists a PPT translation algorithm such that, to each player, the results of the following two scenarios are distributed identically: (1) applying translation to the output of TTP1*; (2) returning the output of* TTP2 *directly.*

Proof. Theorem 1 is trivially true. (This theorem is included for completeness.)

Intersection. We define the intersection of multisets $S \cap T$ as the multiset where each element a that appears in S $b_S > 0$ times and T $b_T > 0$ times appears in the resulting multiset $\min\{b_S, b_T\}$ times. Let S and T be two multisets of equal size, and f and g be their polynomial representations respectively. We compute the polynomial representation of $S \cap T$ as:

$$f * r + g * s$$

where $r, s \leftarrow R^{\deg(f)}[x]$, where $R^b[x]$ is the set of all polynomials of degree $0, \ldots, b$ with coefficients chosen independently and uniformly from R: $r = \sum_{i=0}^{\beta} r[i]x^i$ and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \le i \le \beta}\ r[i] \leftarrow R$, $\forall_{0 \le i \le \beta}\ s[i] \leftarrow R$.

We show below that $f * r + g * s$ is a polynomial representation of $S \cap T$. In addition, we prove that, given $f * r + g * s$, one cannot learn more information about S and T than what can be deduced from $S \cap T$, as formally stated in Theorem 2.

First, we must prove the following lemma:

Lemma 1. *Let $\hat{f}, \hat{g}$ be polynomials in $R[x]$ where R is a ring, $\deg(\hat{f}) = \deg(\hat{g}) = \alpha$, and $\gcd(\hat{f}, \hat{g}) = 1$. Let $r = \sum_{i=0}^{\beta} r[i]x^i$, and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \le i \le \beta}\ r[i] \leftarrow R$, $\forall_{0 \le i \le \beta}\ s[i] \leftarrow R$ (independently) and $\beta \ge \alpha$.*

Let $\hat{u} = \hat{f} * r + \hat{g} * s = \sum_{i=0}^{\alpha+\beta} u[i]x^i$. Then $\forall_{0 \le i \le \alpha+\beta}\ \hat{u}[i]$ are distributed uniformly and independently over R.

We give a proof of Lemma 1 in Appendix A.

By this lemma, $f * r + g * s = \gcd(f, g) * u$, where u is distributed uniformly in $R^{\gamma}[x]$ for $\gamma = 2\deg(f) - |S \cap T|$. Note that a is a root of $\gcd(f, g)$ and $(x - a)^{\ell_a} \mid \gcd(f, g)$ if and only if a appears ℓ_a times in $S \cap T$. Moreover, because u is distributed uniformly in $R^{\gamma}[x]$, with overwhelming probability the roots of u do not represent any element from P (as explained in the beginning of Section 4). Thus, the computed polynomial $f*r+g*s$ is a polynomial representation of $S \cap T$. Note that this technique for computing the intersection of two multisets can be extended to simultaneously compute the intersection of an arbitrary number of multisets in a similar manner. Also, given $f * r + g * s$, one cannot learn more information about S and T than what can be deduced from $S \cap T$, as formally stated in the following theorem:

Theorem 2. *Let* TTP1 *be a trusted third party which receives the private input multiset S_i from player i for $1 \le i \le n$, and then returns to every player the* *intersection multiset $S_1 \cap \cdots \cap S_n$ directly. Let* TTP2 *be another trusted third party, which receives the private input multiset S_i from player i for $1 \le i \le n$, and then: (1) calculates the polynomial representation f_i for each S_i; (2) chooses $r_i \leftarrow R^k[x]$; (3) computes and returns to each player $\sum_{i=1}^n f_i * r_i$.*

There exists a PPT translation algorithm such that, to each player, the results of the following two scenarios are distributed identically: (1) applying translation to the output of TTP1*; (2) returning the output of* TTP2 *directly.*

Proof (Proof sketch). Let the output of TTP1 be denoted T. The translation algorithm operates as follows: (1) calculates the polynomial representation g of T; (2) chooses the random polynomial $u \leftarrow R^{2k-|T|}[x]$; (3) computes and returns $g * u$.

Element Reduction. We define the operation of element reduction (by d) of multiset S (denoted $\mathrm{Rd}_d(S)$) as follows: for each element a that appears b times in S, it appears $\max\{b-d, 0\}$ times in the resulting multiset. We compute the polynomial representation of $\mathrm{Rd}_d(S)$ as:

$$f^{(d)} * F * r + f * s$$

where $r, s \leftarrow R^{\deg(f)}[x]$ and F is any polynomial of degree d, such that $\forall_{a \in P}\ F(a) \neq 0$. Note that a random polynomial of degree d in $R[x]$ has this property with overwhelming probability.

To show that formal derivative operation allows element reduction, we require the following lemma:

Lemma 2. *Let $f \in R[x]$, where R is a ring, $d \ge 1$.*

1. *If $(x-a)^{d+1} \mid f$, then $(x-a) \mid f^{(d)}$.*
2. *If $(x-a) \mid f$ and $(x-a)^{d+1} \nmid f$, then $(x-a) \nmid f^{(d)}$.*

Lemma 2 is a standard result [24]. By this lemma and $\gcd(F, f) = 1$, an element a is a root of $\gcd(f^{(d)}, f)$ and $(x-a)^{\ell_a} \mid \gcd(f^{(d)}, f)$ if and only if a appears ℓ_a times in $\mathrm{Rd}_d(S)$. By Lemma 1, $f^{(d)} * F * r + f * s = \gcd(f^{(d)}, f) * u$, where u is distributed uniformly in $R^{\gamma}[x]$ for $\gamma = 2k - |\mathrm{Rd}_d(S)|$. Thus, with overwhelming probability, any root of u does not represent any element from P. Therefore, $f^{(d)} * F * r + f * s$ is a polynomial representation of $\mathrm{Rd}_d(S)$, and moreover, given $f^{(d)} * F * r + f * s$, one cannot learn more information about S than what can be deduced from $\mathrm{Rd}_d(S)$, as formally stated in the following theorem:

Theorem 3. *Let F be a publicly known polynomial of degree d such that $\forall_{a \in P}\ F(a) \neq 0$. Let* TTP1 *be a trusted third party which receives a private input multiset S, and then returns the reduction multiset $Rd_d(S)$ directly. Let* TTP2 *be another trusted third party, which receives a private input multiset S, and then: (1) calculates the polynomial representation f of S; (2) chooses $r, s \leftarrow R^k[x]$; (3) computes and returns $f^{(d)} * F * r + f * s$.*

There exists a PPT translation algorithm such that the results of the following two scenarios are distributed identically: (1) applying translation to the output of TTP1*; (2) returning the output of* TTP2 *directly.*

Proof (Proof sketch). Let the output of TTP1 be denoted T. The translation algorithm operates as follows: (1) calculates the polynomial representation g of T; (2) chooses the random polynomial $u \leftarrow R^{2k-|T|}[x]$; (3) computes and returns $g * u$.

Operations with Encrypted Polynomials. In the previous section, we prove the security of our polynomial-based multiset operators when the polynomial representation of the result is computed by a trusted third party (TTP2). By using additively homomorphic encryption, we allow these results to be implemented as protocols in the real world without a trusted third party (i.e., the polynomial representation of the set operations is computed by the parties collectively without a trusted third party). In the algorithms given above, there are three basic polynomial operations that are used: addition, multiplication, and the formal derivative. We give algorithms in this section for computation of these operations with encrypted polynomials.

For $f \in R[x]$, we represent the *encryption of polynomial* f, $E_{pk}(f)$, as the ordered list of the encryptions of its coefficients under the additively homomorphic cryptosystem: $E_{pk}(f[0]), \ldots, E_{pk}(f[\deg(f)])$. Let f_1, f_2, and g be polynomials in $R[x]$ such that $f_1(x) = \sum_{i=0}^{\deg(f_1)} f_1[i]x^i$, $f_2(x) = \sum_{i=0}^{\deg(f_2)} f_2[i]x^i$, and $g(x) = \sum_{i=0}^{\deg(g)} g[i]x^i$. Let $a, b \in R$. Using the homomorphic properties of the homomorphic cryptosystem, we can efficiently perform the following operations on encrypted polynomials without knowledge of the private key:

- Sum of encrypted polynomials: given the encryptions of the polynomial f_1 and f_2, we can efficiently compute the encryption of the polynomial $g := f_1 + f_2$, by calculating $E_{pk}(g[i]) := E_{pk}(f_1[i]) +_h E_{pk}(f_2[i])$ $(0 \le i \le \max\{\deg(f_1), \deg(f_2)\})$
- Product of an unencrypted polynomial and an encrypted polynomial: given a polynomial f_2 and the encryption of polynomial f_1, we can efficiently compute the encryption of polynomial $g := f_1 * f_2$, (also denoted $f_2 *_h E_{pk}(f_1)$) by calculating the encryption of each coefficient $E_{pk}(g[i]) := (f_2[0] \times_h E_{pk}(f_1[i])) +_h \ldots +_h (f_2[i] \times_h E_{pk}(f_1[0]))$ $(0 \le i \le \deg(f_1) + \deg(f_2))$.
- Derivative of an encrypted polynomial: given the encryption of polynomial f_1, we can efficiently compute the encryption of polynomial $g := \frac{d}{dx} f_1$, by calculating the encryption of each coefficient $E_{pk}(g[i]) := (i+1) \times_h E_{pk}(f_1[i+1])$ $(0 \le i \le \deg(f_1) - 1)$.
- Evaluation of an encrypted polynomial at an unencrypted point: given the encryption of polynomial f_1, we can efficiently compute the encryption of $a := f_1(b)$, by calculating $E_{pk}(a) := (b^0 \times_h E_{pk}(f_1[0])) +_h \ldots +_h (b^{\deg(f)} \times_h E_{pk}(f_1[\deg(f_1)]))$.

It is easy to see that with the above operations on encrypted polynomials, we can allow the computation of the polynomial representations of set operations described in Section 4.2 without the trusted third party (TTP2) while enjoying the same security. As an example, we design in Section 5, a protocol for the Set-Intersection problem, and discuss several other selected applications in Section 7.

5 Application I: Private Set-Intersection

Problem Definition. Let there be n parties; each has a private input set S_i $(1 \leq i \leq n)$ of size k. We define the *Set-Intersection* problem as follows: all players learn the intersection of all private input multisets without gaining any other information; that is, each player learns $S_1 \cap S_2 \cap \cdots \cap S_n$.

We design a protocol, secure against a coalition of honest-but-curious adversaries, in Section 5.1. We then describe variations of the problem and how to extend this protocol to be secure against malicious adversaries in Section 7.

5.1 Set-Intersection Protocol

Protocol: Set-Intersection-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key to a homomorpic cryptosystem.

1. Each player $i = 1, \ldots, n$
   (a) calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends the encryption of the polynomial f_i to players $i+1, \ldots, i+c$
   (c) chooses $c+1$ polynomials $r_{i,0}, \ldots, r_{i,c} \leftarrow R^k[x]$
   (d) calculates the encryption of the polynomial $\phi_i = f_{i-c} * r_{i,i-c} + \cdots + f_{i-1} * r_{i,i-1} + f_i * r_{i,0}$, utilizing the algorithms given in Sec. 4.2.
2. Player 1 sends the encryption of the polynomial $\lambda_1 = \phi_1$, to player 2
3. Each player $i = 2, \ldots, n$ in turn
   (a) receives the encryption of the polynomial λ_{i-1} from player $i-1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} + \phi_i$ by utilizing the algorithms given in Sec. 4.2.
   (c) sends the encryption of the polynomial λ_i to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$ to all other players.
5. All players perform a group decryption to obtain the polynomial p.
6. Each player $i = 1, \ldots, n$ determines the intersection multiset: for each $a \in S_i$, he calculates b such that $(x-a)^b | p \ \wedge \ (x-a)^{b+1} \not| p$. The element a appears b times in the intersection multiset.

Fig. 1. Set-Intersection protocol for the honest-but-curious case.

Our protocol for the honest-but-curious case is given in Fig. 1. In this protocol, each player i $(1 \leq i \leq n)$ first calculates a polynomial representation

$f_i \in R[x]$ of his input multiset S_i. He then encrypts this polynomial f_i, and sends it to c other players $i+1, \ldots, i+c$. For each encrypted polynomial $E_{pk}(f_i)$, each player $i+j$ $(0 \leq j \leq c)$ chooses a random polynomial $r_{i+j,j} \in R^k[x]$. Note that at most c players may collude, thus $\sum_{j=0}^{c} r_{i+j,j}$ is both uniformly distributed and known to no player. They then compute the encrypted polynomial $\left(\sum_{j=0}^{c} r_{i+j,j}\right) *_h E_{pk}(f_i)$. From these encrypted polynomials, the players compute the encryption of $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$. All players engage in group decryption to obtain the polynomial p. Thus, by Theorem 2, the players have privately computed p, a polynomial representing the intersection of their private input multisets. Finally, to reconstruct the multiset represented by polynomial p, the player i, for each $a \in S_i$, calculates b such that $(x-a)^b | p \wedge (x-a)^{b+1} \not| p$. The element a appears b times in the intersection multiset.

Security Analysis. We show that our protocol is correct, as each player learns the appropriate answer set at its termination, and secure in the honest-but-curious model, as no player gains information that it would not gain when using its input in the ideal model. A formal statement of these properties is as follows:

Theorem 4. *In the Set-Intersection protocol of Fig. 1, every player learns the intersection of all players' private inputs, $S_1 \cap S_2 \cap \cdots \cap S_n$, with overwhelming probability.*

Theorem 5. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, with overwhelming probability, in the Set-Intersection protocol of Fig. 1, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

We provide proof sketches for Theorems 4 and 5 in [18].

6 Application II: Private Over-Threshold Set-Union

Problem Definition. Let there be n players; each has a private input set S_i $(1 \leq i \leq n)$ of size k. We define the *Over-Threshold Set-Union* problem as follows: all players learn which elements appear in the union of the players' private input multisets at least a threshold number t times, and the number of times these elements appeared in the union of players' private inputs, without gaining any other information. For example, assume that a appears in the combined private input of the players 15 times. If $t = 10$, then all players learn a has appeared 15 times. However, if $t = 16$, then no player learns a appears in any player's private input. This problem can be computed as $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

In Section 6.1, we design a protocol for the Over-Threshold Set-Union problem, secure against honest-but-curious adversaries. This protocol is significantly more efficient than utilizing general multiparty computation (the best previous solution for this problem). We describe a variation of the problem and security against malicious adversaries in Section 7.

6.1 Over-Threshold Set-Union Protocol

Protocol: Over-Threshold Set-Union-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key for a homomorphic cryptosystem. The threshold number of repetitions at which an element appears in the output is t. F is a fixed polynomial of degree $t-1$ which has no roots representing elements of P.

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
2. Player 1 sends the encryption of the polynomial $\lambda_1 = f_1$ to player 2
3. Each player $i = 2, \ldots, n$
   (a) receives the encryption of the polynomial λ_{i-1} from player $i-1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} * f_i$ by utilizing the algorithm given in Sec. 4.2.
   (c) sends the encryption of the polynomial λ_i to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \prod_{i=1}^{n} f_i$ to players $2, \ldots, c+1$
5. Each player $i = 1, \ldots, c+1$
   (a) calculates the encryption of the $t-1$th derivative of p, denoted $p^{(t-1)}$, by repeating the algorithm given in Sec. 4.2.
   (b) chooses random polynomials $r_i, s_i \leftarrow R^{nk}[x]$
   (c) calculates the encryption of the polynomial $p * s_i + F * p^{(t-1)} * r_i$ and sends it to all other players.
6. All players perform a group decryption to obtain the polynomial $\Phi = F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right)$.
7. Each player $i = 1, \ldots, n$, for each $j = 1, \ldots, k$
   (a) chooses a random element $b_{i,j} \leftarrow R$
   (b) calculates $u_{i,j} = b_{i,j} \times \Phi((S_i)_j) + (S_i)_j$
8. All players distribute/shuffle the elements $u_{i,j}$ $(1 \leq i \leq n,\ 1 \leq j \leq k)$ such that each player learns all of the elements, but does not learn their origin. Each element $a \in P$ that appears b times in the shuffled elements is an element in the threshold set that appears b times in the players' private inputs.

Fig. 2. Over-Threshold Set-Union protocol for the honest-but-curious case.

We describe our protocol secure against honest-but-curious players for the Over-Threshold Set-Union problem in Fig. 2. In this protocol, each player i $(1 \leq i \leq n)$ first calculates f_i, the polynomial representation of its input multiset S_i. All players then compute the encryption of polynomial $p = \prod_{i=1}^{n} f_i$, the polynomial representation of $S_1 \cup \cdots \cup S_n$. Players $i = 1, \ldots, c+1$ then each chooses random polynomials r_i, s_i, and calculates the encryption of the polynomial $F * p^{(t-1)} * r_i + p * s_i$ as shown in Fig. 2. All players then calculate the encryption of the polynomial $\Phi = F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right)$ and perform a group decryption to obtain Φ. As at most c players may dishonestly collude, the polynomials $\sum_{i=1}^{c+1} r_i$, $\sum_{i=1}^{c+1} s_i$ are uniformly distributed and known to no player. By Theorem 3, Φ is a polynomial representation of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

Each player $i = 1, \ldots, n$ then chooses $b_{i,j} \leftarrow R$ and computes $u_{i,j} = b_{i,j} \times \Phi((S_i)_j) + (S_i)_j$ $(1 \leq j \leq k)$. Each element $u_{i,j}$ equals $(S_i)_j$ if $(S_i)_j \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, and is otherwise uniformly distributed over R. The players then shuffle these elements $u_{i,j}$, such that each player learns all of the elements, but does not learn which player's set they came from. The shuffle can be easily accomplished with standard techniques [4, 17, 8, 13, 22], with communication complexity at most $O(n^2 k)$. The multiset formed by those shuffled elements that represent elements of P is $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

Security Analysis We show that our protocol is correct, as each player learns the appropriate answer set at its termination, and secure in the honest-but-curious model, as no player gains information that it would not gain when using its input in the ideal model with a trusted third party. A formal statement of these properties is as follows:

Theorem 6. *In the Over-Threshold Set-Union protocol of Fig. 2, with overwhelming probability, every honest-but-curious player learns each element a which appears at least t times in the union of the n players' private inputs, as well as the number of times it so appears.*

Theorem 7. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, with overwhelming probability, in the Over-Threshold Set-Union protocol of Fig. 2, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

We provide proof sketches for Theorems 6 and 7 in [18].

7 Other Applications

Using the encrypted polynomial techniques of Section 4, we may construct efficient protocols for functions composed of multiset intersection, union, and element reduction. These functions are described by the following grammar:

$$\Upsilon ::= s \mid \mathrm{Rd}_d(\Upsilon) \mid \Upsilon \cap \Upsilon \mid s \cup \Upsilon \mid \Upsilon \cup s,$$

where s represents any multiset held by some player, and $d \geq 1$. Note that any monotone function on multisets can be expressed using the grammar above, and thus our techniques for privacy-preserving set operations are truly general. Additional techniques allow arbitrary composition of multiset operations are described in [18].

We design a protocol for Cardinality Set-Intersection, using polynomial evaluation. We describe a protocol for the Threshold Set-Union problem, a variant of Over-Threshold Set-Union. We also design protocols for several more variations on the Over-Threshold Set-Union problem, determining the subset relation, and for evaluation of boolean CNF formulas using our techniques; constructions and proofs are given in [18].

7.1 Cardinality Set-Intersection

We may use the technique of polynomial evaluation to design protocols for variants on the multiset functions previously described; this is closely related to techniques utilized in [12]. In the Cardinality Set-Intersection problem, each player learns $|S_1 \cap \cdots \cap S_n|$, without learning any other information. Our protocol for Cardinality Set-Intersection on sets proceeds as our protocol for Set-Intersection, given in Section 5, until the point where all players learn the encryption of p, the polynomial representation of $S_1 \cap \cdots \cap S_n$. Each player $i = 1, \ldots, n$ then evaluates this encrypted polynomial at each unique element $a \in S_i$, obtaining β_a, an encryption of $p(a)$. He then blinds each encrypted evaluation $p(a)$ by calculating $\beta'_a = b_a \times_h \beta_a$. All players then distribute and shuffle the ciphertexts β'_a constructed by each player, such that all players receive all ciphertexts, without learning their source. The players then decrypt these ciphertexts, finding that nb of the decryptions are 0, implying that there are b unique elements in $S_1 \cap \cdots \cap S_n$. Due to space constraints, we describe the details of our protocols for this problem in [18].

7.2 Threshold Set-Union

Using our techniques, we design efficient solutions to variations of the Over-Threshold Set-Union problem, including the *Threshold Set-Union* problem, where each player learns which elements appear in $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$ without gaining any other information. Note that this differs from the Over-Threshold Set-Union problem in that the players do not learn the number of times any element appears in $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$. Our protocol for the Threshold Set-Union problem follows our protocol for Over-Threshold Set-Union until all players have learned the encryption of the polynomial Φ, the polynomial representation of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$. Each player $i = 1, \ldots, n$ then evaluates this encrypted polynomial at each element $a = (S_i)_j$, obtaining $U_{i,j}$, an encryption of $\Phi(a)$. He then chooses $b_{i,j} \leftarrow R$, and calculates $U'_{i,j} = b_{i,j} \times_h U_{i,j} +_h (S_i)_j$. Each element $U'_{i,j}$ is an encryption of $(S_i)_j$ if $(S_i)_j \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, and is otherwise uniformly distributed over R. All players then shuffle and distribute $U'_{i,j}$ $(1 \leq i \leq n, 1 \leq j \leq k)$, such that all players receive all ciphertexts, without learning their source. Shuffling can be easily accomplished with standard techniques [4, 17, 8, 13, 22], with communication complexity at most $O(n^2 k)$. The players then perform a calculation so that if any two shuffled ciphertexts are encryptions of the same plaintext, one will reveal the correct plaintext element, and the other will yield a uniformly distributed element of R. Thus each element of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$ is revealed exactly once. Due to space constraints, we describe the details of our protocols for the Threshold Set-Union problem and several other variants in [18].

7.3 Private Subset Relation

Problem Statement. Let the set A be held by Alice. The set B may be the result of an arbitrary function over multiple players' input sets (for example as calculated using the grammar above). The *Subset* problem is to determine whether $A \subseteq B$ without revealing any additional information.

Let λ be the encryption of the polynomial p representing B. Note that $A \subseteq B \Leftrightarrow \forall_{a \in A}\ p(a) = 0$. Alice thus evaluates the encrypted polynomial λ at each element $a \in A$, homomorphically multiplies a random element by each encrypted evaluation, and adds these blinded ciphertexts to obtain β'. If β' is an encryption of 0, then $A \subseteq B$. More formally:

1. For each element $a = A_j$ $(1 \leq j \leq |A|)$, the player holding A:
   (a) calculates $\beta_j = \lambda(a)$
   (b) chooses a random element $b_j \leftarrow R$, and calculates $\beta'_j = b_j \times_h \beta_j$
2. The player holding A calculates $\beta' = \beta'_1 +_h \ \dots\ +_h \beta'_{|A|}$
3. All players together decrypt β' to obtain y. If $y = 0$, then $A \subseteq B$.

This protocol can be easily extended to allow the set A to be held by multiple players, such that $A = A_1 \cup \cdots \cup A_\nu$, where each set A_i is held by a single player.

7.4 Security Against Malicious Parties

We can extend our Set-Intersection protocol in Figure 1, secure against honest-but-curious players, to one secure against malicious adversaries by adding zero-knowledge proofs or using cut-and-choose to ensure security. By adding zero-knowledge proofs to our Over-Threshold Set-Union and Cardinality Set-Intersection protocols secure against honest-but-curious adversaries, we extend our results to enable security against malicious adversaries. Due to space constraints, we provide details of these protocols, as well as security analysis, in [18].

Acknowledgments: We extend our thanks to Dan Boneh, Benny Pinkas, David Molnar, David Brumley, Alina Oprea, Luis von Ahn, and anonymous reviewers for their invaluable comments. We especially thank Benny Pinkas for suggestions on the presentation of this paper.

References

1. M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of STOC*, 1988.

2. Fabirce Boudot, Berry Schoenmakers, and Jacques Traore. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111:77–85, 2001.
3. Jan Camenisch. Proof systems for general statements about discrete logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich, Mar 1997.
4. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–8, 1981.
5. David Chaum, Jan-Hendrick Evertse, Jeroen van de Graaf, and Rene Peralta. Demonstrating possession of a discrete log without revealing it. In A.M. Odlyzko, editor, *Proc. of Crypto*, pages 200–212. Springer-Verlag, 1986.
6. R. Cramer, I. Damgård, and J. Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of Eurocrypt*, pages 280–99. Springer-Verlag, 2001.
7. Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret sharing scheme. In *Proc. of Eurocrypt*. Springer-Verlag, May 2000.
8. Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *Proc. of Eurocrypt*, pages 557–72. Springer-Verlag, 2000.
9. Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39:77–85, 1996.
10. P. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting of lotteries. In *Proc. of Financial Cryptography*, 2000.
11. Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Proc. of Asiacrypt*, pages 573–84, 2000.
12. Michael Freedman, Kobi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Proc. of Eurocrypt*, volume LNCS 3027, pages 1–19. Springer-Verlag, May 2004.
13. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of Crypto*, pages 368–87. Springer-Verlag, 2001.
14. Rosario Gennaro and Victor Shoup. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15:75–96, 2002.
15. Oded Goldreich. The foundations of cryptography – volume 2. http://www.wisdom.weizmann.ac.il/ oded/foc-vol2.html.
16. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and Systems Science*, 28:270–99, 1984.
17. M. Jakobsson. A practical mix. In *Proc. of Eurocrypt*, pages 448–61. Springer-Verlag, 1998.
18. Lea Kissner and Dawn Song. Private and threshold set-intersection. Technical Report CMU-CS-05-113, Carnegie Mellon University, February 2005.
19. Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Proc. of Asiacrypt*, pages 416–33, 2003.
20. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.
21. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. ACM Symposium on Theory of Computing*, pages 245–54, 1999.
22. A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–25, 2001.
23. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of Asiacrypt*, pages 573–84, 2000.
24. Victor Shoup. A computational introduction to number theory and algebra. http://shoup.net/ntb/.

25. Andrew C-C Yao. Protocols for secure computations. In *Proc. of FOCS*, 1982.

A Proof of Lemma

Theorem 1: *Let f, g be polynomials in $R[x]$ where R is a ring,* $\deg(f) = \deg(g) = \alpha$*, and* $\gcd(f, g) = 1$*. Let $r = \sum_{i=0}^{\beta} r[i]x^i$ and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \le i \le \beta}\ r[i] \leftarrow R$, $\forall_{0 \le i \le \beta}\ s[i] \leftarrow R$ (independently) and $\beta \ge \alpha$.*

Let $u = f * r + g * s = \sum_{i=0}^{\alpha+\beta} u[i]x^i$. Then $\forall_{0 \le i \le \alpha+\beta}\ u[i]$ are distributed uniformly and independently over R.

Proof. For clarity, we give a brief outline of the proof before proceeding to the details. Given any fixed polynomials f, g, u, we calculate the number z of r, s pairs such that $f * r + g * s = u$. We may then check that, given any fixed polynomials f, g, the total number of possible r, s pairs, divided by z, is equal to the number of possible result polynomials u. This implies that, if $\gcd(f, g) = 1$ and we choose the coefficients of r, s uniformly and independently from R, the coefficients of the result polynomial u are distributed uniformly and independently over R.

We now determine the value of z, the number of r, s pairs such that $f * r + g * s = u$. Let us assume that there exists at least one pair $\hat{r}, \hat{s}$ such that $f * \hat{r} + g * \hat{s} = u$. For any pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$, then

$$f * \hat{r} + g * \hat{s} = f * \hat{r}' + g * \hat{s}'$$
$$f * (\hat{r} - \hat{r}') = g * (\hat{s}' - \hat{s})$$

As $\gcd(f, g) = 1$, we may conclude that $g | (\hat{r} - \hat{r}')$ and $f | (\hat{s}' - \hat{s})$. Let $p * g = \hat{r} - \hat{r}'$ and $p * f = \hat{s}' - \hat{s}$. We now show that each polynomial p, of degree at most $\beta - \alpha$, determines exactly one unique pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$. Note that $\hat{r}' = \hat{r} - g * p$, $\hat{s}' = \hat{s} + f * p$; as we have fixed $f, g, \hat{r}, \hat{s}$, a choice of p determines both $\hat{r}', \hat{s}'$. If these assignments were not unique, there would exist polynomials p, p' such that either $\hat{r}' = \hat{r} - g * p = \hat{r} - g * p'$ or $\hat{s}' = \hat{s} + f * p = \hat{s} + f * p'$; either condition implies that $p = p'$, giving a contradiction. Thus the number of polynomials p, of degree at most $\beta - \alpha$, is exactly equivalent to the number of r, s pairs such that $f * r + g * s = u$. As there are $|R|^{\beta-\alpha+1}$ such polynomials p, $z = |R|^{\beta-\alpha+1}$.

We now show that the total number of r, s pairs, divided by z, is equal to the number of result polynomials u. There are $|R|^{2\beta+2}$ r, s pairs. As $\frac{|R|^{2\beta+2}}{z} = \frac{|R|^{2\beta+2}}{|R|^{\beta-\alpha+1}} = |R|^{\alpha+\beta+1}$, and there are $|R|^{\alpha+\beta+1}$ possible result polynomials, we have proved the theorem true.