Privacy-Preserving Set Operations

Lea Kissner **Dawn Song**

February 2005 – Last modified June 2005
CMU-CS-05-113

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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 employing the mathematical properties of polynomials, we build a framework of efficient, secure, and composable multiset operations: 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.

This research was supported in part by the Center for Computer and Communications Security at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, Carnegie Mellon University, or the U.S. Government or any of its agencies.

Keywords: private multiset operations, secure multi-party computation, privacy, intersection, threshold, union, element reduction

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, Cardinality Set-Intersection, Over-Threshold Set-Union, and Threshold Set-Union, as well as determining the Subset relation. Furthermore, we show that our techniques can be used to efficiently compute the output of any function over multisets expressed in the following grammar, where s represents any set held by some player and $d \geq 1$:

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

Note that any monotonic function over multisets[2] can be expressed using our grammar, showing that our techniques have truly general applicability. Finally, we show that our techniques are applicable even outside the realm of set computation. As an example, we describe how to utilize our techniques to efficiently and privately evaluate CNF boolean functions.

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 [13]. However, previous work only provides protocols for 3-or-more-party

[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 $\mathrm{Rd}_d(A)$.

[2]Any function computed with only intersection and union, without use of an inverse operation.

Problem	*Communication Complexity of our solution*	*Communication Complexity of previous solution*	*Communication Complexity of general MPC*
Set-Intersection (HBC)	$O(cnk \lg \lvert P \rvert)$	$O(n^2 k \lg \lvert P \rvert)$ [13]	$O(n^2 k \text{ polylog}(k) \lg \lvert P \rvert)$
Set-Intersection (Malicious)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^3 k \text{ polylog}(k) \lg \lvert P \rvert)$
Cardinality Set-Intersection (HBC)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^2 k \text{ polylog}(k) \lg \lvert P \rvert)$
Over-Threshold Set-Union (HBC)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^2 k \text{ polylog}(nk) \lg \lvert P \rvert)$
Threshold Set-Union (HBC)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^2 k \text{ polylog}(nk) \lg \lvert P \rvert)$
Subset (HBC)	$O(k \lg \lvert P \rvert)$	none	$O(k \text{ polylog}(k) \lg \lvert P \rvert)$

Table 1: Total communication complexity comparison for our multiparty protocols, previous solutions, and general multiparty computation. There are $n \geq 2$ players, $c < n$ dishonestly colluding, each with an input multiset of size k. The domain of the multiset elements is P. Security parameters are not included in the communication complexity.

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, and our multiset intersection operator can be easily composed with other operations to enable a wide range of efficient private computation over multisets. We compare the communication complexity of our protocols with previous work and solutions based on general multiparty communication in Table 1. Note that the techniques utilized to create the circuits for the general solution are both complex and incur very large constants, on top of the constants inherent in the use of general multiparty computation [1]; we thus achieve greater practical efficiency, as well as asymptotic efficiency.

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. For protocols secure in the HBC model, we prove that the information learned by any coalition of honest-but-curious players is indistinguishable from the information learned in the ideal model, where a trusted third party (TTP) calculates the function. For protocols secure in the malicious model, we provide simulation proofs showing that for any strategy followed by a malicious coalition Γ in the real protocol, there is a translated strategy they could follow in the ideal model, such that, to Γ, the real execution is computationally indistinguishable from ideal execution.

Outline. 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 protocols, secure against honest-but-curious players, and security analysis for the Set-Intersection and Cardinality Set-Intersection problems. Section 6 gives protocols, secure against honest-but-curious players, and security analysis for the Over-Threshold Set-Union problem, as well for several variants of the Threshold Set-Union problem. We introduce techniques and protocols secure against malicious players for the Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union problems in Section 7. Finally, we discuss several additional applications of our techniques in Section 8, including the subset protocol, general privacy-preserving computation over sets, and evaluation of CNF boolean formulas.

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 are through general multiparty computation. General two-party computation was introduced by Yao [28], and general computation for multiple parties was introduced in [2]. 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 [16]. Recent results that allow non-interactive private multiplication of shares [8] 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 Table 1.

The most relevant work to our paper is by Freedman, Nissim, and Pinkas (FNP) [13]. They proposed protocols for the problems related to Set-Intersection, based on the representation of sets as roots of a polynomial [13]. 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. We give a more detailed comparison of our Set-Intersection protocol with FNP in Table 1 and in Section 1.

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 [10, 24, 22], and fairness is added in [3]. Another related problem is in testing the disjointness of private input sets [20]; a restricted version of the Cardinality Set-Intersection problem. 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

The notation used in this paper is described in Appendix A. 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. We provide intuition and informal definitions of these models; formal definitions of these models can be found in [16].

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., [16]) 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 [17], 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 [15]; (2) efficient construction of zero-knowledge proofs of plaintext knowledge; (3) optionally, efficient construction of certain zero-knowledge proofs, as detailed inSection 7.1.

Note that Paillier's cryptosystem [26] satisfies each of our requirements: it is additively homomorphic, supports ciphertexts re-randomization and threshold decryption (secure in the malicious case) [11, 12], and allows certain efficient zero-knowledge proofs (standard constructions from [6, 4], and proof of plaintext knowledge [7]).

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

3.3 Shuffle Protocol

Each player i $(1 \le i \le n)$ has a private input multiset V_i. We define the Shuffle problem as follows: all players learn the joint multiset $V_1 \cup \cdots \cup V_n$, such that no player or coalition of players Γ can gain a non-negligible advantage in distinguishing, for each element $a \in V_1 \cup \cdots \cup V_n$, an honest player i $(1 \le i \le n,\ i \notin \Gamma)$ such that $a \in V_i$.

In several protocols in this paper, we will impose an additional privacy condition on the Shuffle problem; the multisets $V_1, \ldots, V_n$ are composed of ciphertexts, which must be re-randomized so that no player may determine which ciphertexts were part of his private input multiset. The revised problem statement is as follows: all players learn the joint multiset $V_1 \cup \cdots \cup V_n$, such that no player or coalition of players can gain a non-negligible advantage in distinguishing, for each element $a \in V_1 \cup \cdots \cup V_n$, an honest player i $(1 \leq i \leq n)$ such that $a \in V_i$.

Both variants of the Shuffle protocol can be easily accomplished with standard techniques [5, 18, 9, 14, 25], with communication complexity at most $O(n^2k)$.

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. 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 [13] (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 8.1), and constitute truly general techniques.

4.2.1 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, 3, and 5, 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 $1 \leq i \leq n$, 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 $\forall_{0 \leq i \leq \beta}\ r[i] \leftarrow R$, $\forall_{0 \leq i \leq \beta}\ s[i] \leftarrow R$.

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 3.

First, we must prove the following lemma:

Lemma 2. *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 $\forall_{0 \leq i \leq \beta}\ r[i] \leftarrow R$, $\forall_{0 \leq i \leq \beta}\ s[i] \leftarrow R$ (independently) and $\beta \geq \alpha$.*

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

We give a proof of Lemma 2 in Appendix B.

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 3. *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 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 $1 \leq i \leq n$, 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 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 4. *Let $f \in R[x]$, where R is a ring, $d \geq 1$.*

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 4 is a standard result [27]. 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 2, $f^{(d)} * F * r + f * s = \gcd(f^{(d)}, f) * u$, 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 5. *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 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 $g * u$. □

4.2.2 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 $E_{pk}(g[i]) := E_{pk}(f_1[i]) +_h E_{pk}(f_2[i])$ $(0 \leq i \leq \max\{\deg(f_1), \deg(f_2)\})$
- 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 $E_{pk}(g[i]) := (f_2[0] \times_h E_{pk}(f_1[i])) +_h (f_2[1] \times_h E_{pk}(f_1[i-1])) +_h \ldots +_h (f_2[i] \times_h E_{pk}(f_1[0]))$ $(0 \leq i \leq \deg(f_1) + \deg(f_2))$.
- 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 $E_{pk}(g[i]) := (i+1) \times_h E_{pk}(f_1[i+1])$ $(0 \leq i \leq \deg(f_1) - 1)$.
- 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 $E_{pk}(a) := (b^0 \times_h E_{pk}(f_1[0])) +_h (b^1 \times_h E_{pk}(f_1[1])) +_h \ldots +_h (b^{\deg(f_1)} \times_h E_{pk}(f_1[\deg(f_1)]))$.

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 4.2.1 without the trusted third party (TTP2) while enjoying the same security. We demonstrate this property with concrete examples detailed in the remainder of this paper.

4.3 Overview of Applications

The techniques we introduce for privacy-preserving computations of multiset operations have many applications. We give several concrete examples that utilize our techniques for specific privacy-preserving functions on multisets in the following sections.

First, we design efficient protocols for the Set-Intersection and Cardinality Set-Intersection problems, secure against honest-but-curious adversaries (Section 5). We then provide an efficient protocol for the Over-Threshold Set-Union problem, as well as three variants of the Threshold Set-Union problem, secure against honest-but-curious adversaries, in Section 6. We introduce tools and protocols, secure against malicious players, for the Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union problems in Section 7. We propose an efficient protocol for the Subset problem in Section 8.2.

More generally, our techniques allow private computation of functions based on composition of the union, intersection, and element reduction operators. We discuss techniques for this general private computation on multisets in Section 8.1.

Our techniques are widely applicable, even outside the realm of computation of functions over multisets. As an example, we show how to apply our techniques to private evaluation of boolean formulas in CNF form in Section 8.3.

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.
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. 4.2.2.
   (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.

Each player $i = 1, \ldots, n$ determines the intersection multiset as follows: 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.

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

5 Application I: Private Set-Intersection and Cardinality Set-Intersection

In this section, we design protocols for Set-Intersection and Cardinality Set-Intersection, secure against a coalition of honest-but-curious adversaries.

5.1 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 $S_1 \cap S_2 \cap \cdots \cap S_n$.

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 3, 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 6. *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 7. *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 6 and 7 in Appendix C.1.

5.2 Cardinality Set-Intersection

Problem Definition. We define the Cardinality Set-Intersection problem on sets as follows: each player learns the number of unique elements in $S_1 \cap \cdots \cap S_n$, without learning any other information. A variant of this problem is the Cardinality Set-Intersection problem on multisets, which we define as follows: all players learn $|S_1 \cap \cdots \cap S_n|$, as computed on multisets.

Our protocol for Cardinality Set-Intersection, given in Figure 2, proceeds as our protocol for Set-Intersection, 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 Shuffle protocol can be easily accomplished with standard techniques [5, 18, 9, 14, 25], with communication complexity at most $O(n^2k)$. The players then decrypt these ciphertexts, finding that nb of the decryptions are 0, implying that there are b unique elements in $S_1 \cap \cdots \cap S_n$. FNP utilize a variation of this technique [13], but it is not obvious how to construct a multiparty Cardinality Set-Intersection protocol from their techniques.

Variants. Our protocol can be simply extended to privately compute the Cardinality Set-Intersection problem on multisets, by utilizing an encoding as follows: any element a that appears b times in a multiset is encoded as the set: $\{a \mid\mid 1, \ldots, a \mid\mid b\}$, with element included only once. Note that this is a set of equivalent size as the original multiset representation, so this variant preserves the efficiency of our protocol.

Protocol: Cardinality-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 $c + 1$ random 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.
2. Player 1 sends the encrypted 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. 4.2.2.
   (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. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial p at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = p((S_i)_j)$, using the algorithm given in Sec. 4.2.2.
   (b) for each $j = 1, \ldots, k$ chooses a random number $r_{ij} \leftarrow R$ and calculates an encrypted element $(V_i)_j = r_{ij} \times_h E_{pk}(c_{ij})$
6. All players perform the Shuffle protocol on their private input sets V_i, obtaining a joint set V, in which all ciphertexts have been re-randomized.
7. All players $1, \ldots n$ decrypt each element of the shuffled set V

If nb of the decrypted elements from V are 0, then the size of the set intersection is b.

Figure 2: Cardinality set-intersection protocol for the honest-but-curious case.

Security Analysis. We show that our protocol is correct, as each player learns the size of the 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 8. *In the Cardinality Set-Intersection protocol of Fig. 2, every player learns the size of the intersection of all players' private inputs, $|S_1 \cap S_2 \cap \cdots \cap S_n|$, with overwhelming probability.*

Theorem 9. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure and that the Shuffle protocol is secure, with overwhelming probability, in the Cardinality Set-Intersection 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 8 and 9 in Appendix C.2.

5.3 Malicious Case.

We can extend our protocols in Figures 1 and 2, secure against honest-but-curious players, to protocols secure against malicious adversaries by adding zero-knowledge proofs or using cut-and-choose to ensure security. We give details of our protocols secure against malicious adversaries in Section 7.2. We prove security against malicious parties for these protocols in Appendices C.1 and C.2.

6 Application II: Private Over-Threshold Set-Union and Threshold Set-Union

In this section, we design protocols for the Over-Threshold Set-Union problem and several variations of the Threshold Set-Union problem, secure against a coalition of honest-but-curious adversaries.

6.1 Over-Threshold Set-Union Protocol

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 $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

We describe our protocol secure against honest-but-curious players for the Over-Threshold Set-Union problem in Fig. 3. 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. 3. 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 5, 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 [5, 18, 9, 14, 25], with communication complexity at most $O(n^2k)$. 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:

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. 4.2.2.
   (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. 4.2.2.
   (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 $i = 1, \ldots n$ perform the Shuffle protocol on the elements $u_{i,j}$ $(1 \leq j \leq k)$, such that each player obtains a joint set V.

Each element $a \in P$ that appears b times in V is an element in the threshold set that appears b times in the players' private inputs.

Figure 3: Over-Threshold Set-Union protocol for the honest-but-curious case.

Theorem 10. *In the Over-Threshold Set-Union protocol of Fig. 3, 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, with overwhelming probability.*

Theorem 11. *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. 3, 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 10 and 11 in Appendix D.1.

6.2 Threshold Set-Union

Problem Definition. We define the Threshold Set-Union problem as follows: all players learn which elements appear in the combined private input of the players at least a threshold number t times. For example, assume that a appears in the combined private input of the players 15 times. If $t = 10$, then all players learn a. However, if $t = 16$, then no player learns a. This problem differs from the Over-Threshold Set-Union problem in that each player learns the elements of $\mathrm{Rd}_{t-1}(S_1 \cap \cdots \cap S_n)$, without learning how often each element appears.

We offer protocols for several variants on Threshold Set-Union: threshold contribution, perfect, and semi-perfect. Threshold contribution allows for thresholds $t \geq 1$, and each player learns only those elements which appear both in his private input and the threshold set: player i ($1 \leq i \leq n$) learns the elements of $S_i \cap \mathrm{Rd}_{t-1}(S_1 \cap \cdots \cap S_n)$. Perfect threshold set-intersection allows for thresholds $t \geq 1$, and conforms exactly to the definition of threshold set-intersection. The semi-perfect variant requires for security that $t \geq 2$, and that the cheating coalition does not include any single element more than $t-1$ times in their private inputs. Note that the information illicitly gained by the coalition when they include more than $t-1$ copies of an element a is restricted to a possibility of learning that there exists some other player whose private input contains a. We do not consider the difference in security between the semi-perfect and perfect variants to be significant.

The protocols for the Threshold Set-Union problem, given in Figs. 4, 5, and 6, are identical to the protocol for Over-Threshold Set-Union (given in Fig. 3) from step 1-5. We explain the differences between the protocols for each variant: threshold contribution, semi-perfect, and perfect. Each player constructs encryptions of the elements $\Phi((S_i)_j)$ from his private input set in step 6, and continues as described below.

Threshold Contribution Threshold Set-Union. This protocol is given in Fig. 5. The players cooperatively decrypt the encrypted elements $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j})$. This decryption must take place in such a way that only player i learns the element $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j})$. Typically, parties produce decryption shares and reconstruct the element from them; player i simply retains his decryption share, so that only he learns the decryption. Thus each player learns which of his elements appear in the threshold set, since if $(S_i)_j$ appears in the threshold set, $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j}) = 0$. No player learns more information because if an element $(S_i)_j$ is not in the threshold set, $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j})$ is uniformly distributed.

Semi-Perfect Threshold Set-Union. This protocol is given in Fig. 4. The encrypted element $(U_i)_j$ calculated from the encrypted evaluation of $\Phi((S_i)_j)$ is either: (1) an encryption of the private input element $(S_i)_j$ (if $(S_i)_j$ is in the intersection set) or (2) an encryption of a random element (otherwise). However, the player also constructs a corresponding encrypted tag for each $(U_i)_j$, T_{ij}. We require that the cryptosystem used to construct these tags be key-private, so that the origin of ciphertext pairs T, U cannot be ascertained by the key used to construct the tags.

The players then correctly obtain a decryption of each element in the threshold set exactly once. Any other time a ciphertext U for an element in the threshold set is decrypted, a player sabotages it. In group decryption schemes, players generally produce shares of the decrypted element; if one player sends a uniformly generated share instead of a valid one, the decrypted element is uniform. If the decrypted element is uniform, it conveys no information to the players. To ensure an encryption of an element in the threshold set is not decrypted once the element is known to be in the threshold set, a player sabotages the decryption under the following conditions: (1) he can decrypt the tag

Protocol: THRESHOLD-SEMIPERFECT-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. 4.2.2.
   (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) calculate the encryption of the $t-1$th derivative of p, denoted $p^{(t-1)}$, by repeating the algorithm given in Sec. 4.2.2.
   (b) choose random polynomials $r_i, s_i \leftarrow R^{nk}[x]$
   (c) calculate the encryption of the polynomial $p * r_i + F * p^{(t-1)} * s_i$ and send it to all other players
6. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial $\Phi = p * \left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} s_i\right)$ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = \Phi((S_i)_j)$, using the algorithm given in Sec. 4.2.2.
   (b) for each $j = 1, \ldots, k$ calculates an encrypted tag $T_{ij} = Enc_i(h((S_i)_j) \mid\mid (S_i)_j)$
   (c) for each $j = 1, \ldots, k$ chooses a random number $r_{ij} \leftarrow R$ and calculates an encrypted element $U_{ij} = (r_{ij} \times_h E_{pk}(c_{ij})) +_h E_{pk}((S_i)_j)$
   (d) constructs the set $V_i = \{(T_{ij} \mid\mid U_{ij}) \mid 1 \leq j \leq k\}$
7. By using the Shuffle protocol, players perform shuffling on their private input sets V_i.
8. For each shuffled element $T \mid\mid U$ in sorted order, each player $i = 1, \ldots, n$
   (a) if $D_i(T) = h(a) \mid\mid a$ for some a
      i. if a has previously been revealed to be in the threshold set, then calculate an incorrect decryption share of U, and send it to all other players
   (b) else calculate a decryption share of U, and send it to all other players
   (c) reconstruct the decryption of U. If the element $a \in P$, then a is in the threshold result set

Figure 4: Threshold Set-Union protocol for the honest-but-curious case (semi-perfect variant).

to $h(a) \mid\mid a$ for some a and (2) a has already been determined to be a member of the threshold set. All other ciphertexts should be correctly decrypted; either they are encryptions of elements in the threshold set which have not yet been decrypted, or they are encryptions of random elements.

Note that the protocol is the only protocol proposed in this paper with a non-constant number

Protocol: THRESHOLD-CONTRIBUTION-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. The threshold number of repetitions at which an element appears in the output is $t \geq 2$. 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. 4.2.2.
   (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) calculate the encryption of the $t-1$th derivative of p, denoted $p^{(t-1)}$, by repeating the algorithm given in Sec. 4.2.2.
   (b) choose random polynomials $r_i, s_i \leftarrow R^{nk}[x]$
   (c) calculate the encryption of the polynomial $p * r_i + F * p^{(t-1)} * s_i$ and send it to all other players
6. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial $\Phi = p * \left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} s_i\right)$ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = \Phi((S_i)_j)$, using the algorithm given in Sec. 4.2.2.
   (b) sends the ciphertexts c_{ij} $(1 \leq j \leq k)$ to all other players
   (c) chooses a random element $b_{i,j,\ell}$ $(1 \leq j \leq n,\ 1 \leq \ell \leq k)$
   (d) for each ciphertext $c_{j\ell}$, calculate $b_{i,j,\ell} \times_h c_{j\ell}$ $(1 \leq j \leq n,\ 1 \leq \ell \leq k)$
7. The players i $(1 \leq i \leq n)$ calculate $U_{jm} = \left(\sum_{\ell=1}^{n} b_{\ell,j,m}\right) \times_h c_{jm}$ $(1 \leq j \leq n,$ $1 \leq m \leq k)$
8. All players decrypt the ciphertexts U_{ij}, so that only player i learns the decryption $a_{i,j}$.

For each player i $(1 \leq i \leq n)$, if $a_{i,j} = 0$ $(1 \leq j \leq k)$, then $(S_i)_j$ is in his result set.

Figure 5: Threshold Set-Union protocol for the honest-but-curious case (threshold contribution variant).

of rounds. Because of the need to sabotage decryptions based on the results of past decryptions, there are $O(nk)$ rounds in this protocol.

Perfect Threshold Set-Union. This protocol is given in Fig. 6. Each player constructs the encrypted elements $(U_i)_j$ from the encrypted evaluation of $\Phi((S_i)_j)$ as written in step 6 of Figure 4.

Protocol: Threshold-Perfect-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. The threshold number of repetitions at which an element appears in the output is $t \geq 2$. F is a fixed polynomial of degree $t-1$ which has no roots representing elements of P. $\text{IsEq}(C, C') = 1$ if the ciphertexts C, C' encode the same plaintext, and 0 otherwise.

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. 4.2.2.
   (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) calculate the encryption of the $t-1$th derivative of p, denoted $p^{(t-1)}$, by repeating the algorithm given in Sec. 4.2.2.
   (b) choose random polynomials $r_i, s_i \leftarrow R^{nk}[x]$
   (c) calculate the encryption of the polynomial $p * r_i + F * p^{(t-1)} * s_i$ and send it to all other players
6. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial $\Phi = p * \left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} s_i\right)$ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = \Phi((S_i)_j)$, using the algorithm given in Sec. 4.2.2, and sends them to all players
   (b) for each $i' = 1, \ldots, n,\ j = 1, \ldots, k$ chooses a random number $r_{i'j} \leftarrow R$ and calculates an encrypted element $U_{ij} = (r_{i'j} \times_h E_{pk}(c_{i'j}))$, and sends it to player i'
   (c) calculates the elements for $j = 1, \ldots, k$
   $U_{ij} = (r_{1j} \times_h E_{pk}(c_{1j})) +_h \ \ldots \ +_h (r_{nj} \times_h E_{pk}(c_{nj})) +_h E_{pk}((S_i)_j)$
   (d) constructs the set $V_i = \{U_{ij} \mid 1 \leq j \leq k\}$
7. By using the Shuffle protocol, all players perform shuffling on their private input sets V_i, obtaining the set U'.
8. For each shuffled ciphertext U'_ℓ with arbitrary ordering index $\ell \in [nk]$, the players $i = 1, \ldots, n$
   (a) each player i chooses random elements $q_{i,\ell} \leftarrow R$
   (b) calculate $W_\ell = U'_\ell +_h E_{pk}\left(\left(\sum_{i=1}^{n} q_{i,\ell}\right)\left(\text{IsEq}(U'_\ell, U'_{\ell-1}) + \cdots + \text{IsEq}(U'_\ell, U'_1)\right)\right)$
9. All players $1, \ldots, n$ decrypt each ciphertext W_ℓ, obtaining an element a_ℓ $(1 \leq \ell \leq nk)$.

If $a_j \in P$ $(1 \leq j \leq k)$, then a_j is a member of the result set.

Figure 6: Threshold Set-Union protocol for the honest-but-curious case (perfect variant).

The players then utilize the Shuffle protocol to anonymously distribute these elements. If an element appears in the threshold set, then at least one encryption of it appears in the shuffled ciphertexts. The players ensure in step 8 that all duplicates (ciphertexts of the same element) except the first have a random element added to them. This disguises the number of players who have each element of the threshold set in their private input. Let the shuffled ciphertexts U have an arbitrary ordering $U'_1, \ldots, U'_{nk}$. $\mathrm{IsEq}(C, C') = 1$ if the ciphertexts C encode the same plaintext, and 0 otherwise. (This calculation can be achieved with the techniques in [21].) The players $i \in [n]$ then choose random elements $q_{i,\ell} \leftarrow R$ $(1 \leq \ell \leq nk)$ and decrypt the ciphertexts $W_\ell = U'_\ell +_h E_{pk}\left(\left(\sum_{i=1}^{n} q_\ell\right)\left(\mathrm{IsEq}(U'_\ell, U'_{\ell-1}) + \ldots \mathrm{IsEq}(U'_\ell, U'_1)\right)\right)$. Thus, if U'_ℓ is a duplicate (encryption of an element which also appeared early in the ordering), it has a uniformly distributed element added to it, and conveys no information. Each element of the threshold set is decrypted exactly once, and all players thus learn the threshold set.

Security Analysis. We show that our protocol is correct, as each player learns the appropriate result 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 12. *In the Threshold Contribution Threshold Set-Union protocol of Fig. 5, every player i ($1 \leq i \leq n$) learns the set $S_i \cap Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Theorem 13. *In the Semi-Perfect Threshold Set-Union protocol of Fig. 4, each player i ($1 \leq i \leq n$) learns the set $Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Theorem 14. *In the Perfect Threshold Set-Union protocol of Fig. 6, every player learns the set $Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Theorem 15. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure and that the Shuffle protocol is secure, with overwhelming probability, in the Threshold Set-Union protocols of Figs. 4, 5, and 6, 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 12, 13, 14, and 15 in Appendix D.2.

6.3 Malicious Case

By adding zero-knowledge proofs to our Over-Threshold Set-Union protocol secure against honest-but-curious adversaries, we extend our results to enable security against malicious adversaries. We provide details of our protocol secure against malicious adversaries in Section 7.4, and proof of security in Appendix D.1.

7 Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union for Malicious Parties

We extend the protocols for the Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union problems given in Sections 5 and 6 to obtain security against adversaries in the malicious model. To obtain this result, we add zero-knowledge proofs, verified by all players, to ensure the correctness of all computation. In this section, we first introduce notation for zero-knowledge proofs, then give the protocols secure against malicious parties.

7.1 Tools

In this section, we describe cryptographic tools that we utilize in our protocols secure against malicious players.

Zero-Knowledge Proofs. We utilize several zero-knowledge proofs in our protocols for the malicious adversary model. We introduce the notation for these zero-knowledge proofs below; for additively homomorphic cryptosystems such as Paillier, we can efficiently construct these zero-knowledge proofs using standard constructions [6, 4].

- POPK$\{E_{pk}(x)\}$ denotes a zero-knowledge proof that given a public ciphertext $E_{pk}(x)$, the player knows the corresponding plaintext x [7].
- ZKPK$\{f \mid p' = f *_h \alpha\}$ is shorthand notation for a zero-knowledge proof of knowledge that the prover knows a polynomial f such that encrypted polynomial $p' = f *_h \alpha$, given the encrypted polynomials p' and α.
- ZKPK$\{f \mid (p' = f *_h \alpha) \wedge (y = E_{pk}(f))\}$ is the proof ZKPK$\{f \mid p' = f *_h \alpha\}$ with the additional constraint that $y = E_{pk}(f)$ (y is the encryption of f), given the encrypted polynomial p', y, and α.

Equivocal Commitment. A standard commitment scheme allows parties to give a "sealed envelope" that can be later opened to reveal exactly one value. We use an equivocal commitment scheme in our protocols secure against malicious players, such that the simulator can open the 'envelope' to an arbitrary value without being detected by the adversary [19, 23].

7.2 Set-Intersection Protocol for Malicious Adversaries

Our protocol for malicious parties performing Set-Intersection, given in Fig. 7, proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 1. The commitments to the data items $\Lambda(c_{i,j})$ are purely for the purposes of a simulation proof. We add zero-knowledge proofs to prevent three forms of misbehavior: choosing ciphertexts for the encrypted coefficients of f_i without knowledge of their plaintext, not performing the polynomial multiplication of $f_j * r_{i,j}$ correctly, and not performing decryption correctly. We also constrain the leading coefficient of f_i to be 1 for all players, to prevent any player from setting their polynomial to 0; if $f_i = 0$, every element is a root, and thus it can represent an unlimited number of elements. We can thus detect or prevent misbehavior from malicious players, forcing this protocol to operate like the honest-but-curious protocol in Fig. 1. The protocol can gain efficiency by taking advantage of the maximum coalition size c.

Our set-intersection protocol secure against malicious parties utilizes an expensive ($O(k^2)$ size) zero-knowledge proof to prevent malicious parties from cheating when multiplying the polynomial $r_{i,j}$ by the encryption of the polynomial f_j. Each player i must commit to each polynomial $r_{i,j}$ ($1 \leq i, j \leq n$), for purposes of constructing a zero-knowledge proof. We may easily replace this proof with use of the cut-and-choose technique, which requires only $O(k)$ communication.

Protocol: SET-INTERSECTION-MAL
Input: There are $n \geq 2$ players, 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. The commitment scheme used in this protocol is a equivocal commitment scheme.

All players verify the correctness of all proofs sent to them, and stop participating in the protocol if any are not correct.

Each player $i = 1, \ldots, n$:

1. (a) calculates the polynomial f_i such that the k roots of the polynomial are the elements of S_i, as $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends δ_i, the encryption of the polynomial f_i to all other players along with proofs of plaintext knowledge for all coefficients except the leading coefficient (POPK$\{(\delta_i)_j\}$, $0 \leq j < k$).
   (c) for $1 \leq j \leq n$
      i. chooses a random polynomial $r_{i,j} \leftarrow R^k[x]$
      ii. sends a commitment to $\Lambda(r_{i,j})$ to all players, where $\Lambda(r_{i,j}) = E_{pk}(r_{i,j})$
2. for $1 \leq j \leq n$
   (a) opens the commitment to $\Lambda(r_{i,j})$
   (b) verifies proofs of plaintext knowledge for the encrypted coefficients of f_j
   (c) sets the leading encrypted coefficient (for x^k) to a known encryption of 1
   (d) calculates μ, the encryption of the polynomial $p_{i,j} = f_j * r_{i,j}$ with proofs of correct multiplication ZKPK$\{r_{i,j} \mid (\mu = r_{i,j} *_h \delta_j) \wedge (\Lambda(r_{i,j}) = E_{pk}(r_{i,j}))\}$ and sends it to all other players
3. All players
   (a) calculate the encryption of the polynomial $p = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j} = \sum_{i=1}^{n} f_i * (r_{j,i})$ as in Sec. 4.2.2, and verifies all attached proofs
   (b) perform a group decryption to obtain the polynomial p, and distribute proofs of correct decryption

Each player $i = 1, \ldots, n$ determines the intersection multiset as follows: 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.

Figure 7: Set-Intersection protocol for the malicious case.

Security Analysis. We provide a simulation proof of this protocol's security; an intermediary G translates between the real wold with malicious, colluding PPT players Γ and the ideal world, where a trusted third party computes the answer set. Our proof shows that no Γ can distinguish between the ideal world and the real world, thus no information other than that in the answer set can be gained by malicious players. A formal statement of our security property is as follows:

Theorem 16. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Set-Intersection protocol for the malicious case in Fig. 7, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally* *indistinguishable from the views of the honest players and Γ in the real model.*

Proof of this theorem is given in Appendix C.1.

7.3 Cardinality Set-Intersection Protocol for Malicious Adversaries

We give a protocol, secure against malicious parties, to perform Cardinality Set-Intersection in Fig. 8. It proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 2. The commitments to the data items $\Lambda(r_{i,j})$ are purely for the purposes of a simulation proof. We add zero-knowledge proofs of knowledge to prevent five forms of misbehavior: choosing f_i without knowledge of its roots, choosing f_i such that it is not the product of linear factors, not performing the polynomial multiplication of $f_j * r_{i,j}$ correctly, not calculating encrypted elements $(V_i)_j$ correctly (either not from the data items $(S_i)_j$ or not evaluating the encrypted polynomial p), and not performing decryption correctly. We can thus detect or prevent misbehavior from malicious players, forcing this protocol to operate like the honest-but-curious protocol in Fig. 2.

Security Analysis. We provide a simulation proof of this protocol's security; an intermediary G translates between the real wold with malicious, colluding PPT players Γ and the ideal world, where a trusted third party computes the answer set. Our proof shows that no Γ can distinguish between the ideal world and the real world, thus no information other than that in the answer set can be gained by malicious players. A formal statement of our security property is as follows:

Theorem 17. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, the Shuffle protocol is secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Cardinality Set-Intersection protocol for the malicious case in Fig. 8, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and Γ in the real model.*

Proof of this theorem is given in Appendix C.2.

7.4 Over-Threshold Set-Union Protocol for Malicious Adversaries

We give a protocol, secure against malicious parties, to perform Over-Threshold Set-Union in Fig. 9. It proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 3. The commitments to the data items $\Lambda(r_{i,j})$ are purely for the purposes of a simulation proof. We add zero-knowledge proofs of knowledge to prevent six forms of misbehavior: choosing f_i without knowledge of its roots, choosing f_i such that it is not the product of linear factors, not performing the polynomial multiplication of $f_j * \lambda_{j-1}$ correctly, not calculating $\alpha_i = p * r_i$ or $\beta_i = p^{(t-1)} * s_i$ correctly, not calculating encrypted elements $(V_i)_j$ correctly (either not from the data items $(S_i)_j$ or not evaluating the encrypted polynomial Φ), and not performing decryption correctly. We can thus detect or prevent misbehavior from malicious players, forcing this protocol to operate like the honest-but-curious protocol in Fig. 3.

Protocol: CARDINALITY-MAL
Input: There are $n \geq 2$ players, 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. The commitment scheme used in this protocol is a equivocal commitment scheme.

All players verify the correctness of all proofs sent to them, and stop participating in the protocol if any are not correct.

Each player $i = 1, \ldots, n$:

1. (a) calculates the polynomial f_i such that the k roots of the polynomial are the elements of S_i, as $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends:
      i. encrypted elements $y_{i,1} = E_{pk}((S_i)_1), \ldots, y_{i,k} = E_{pk}((S_i)_k)$ to all other players, along with proofs of plaintext knowledge (POPK$\{E_{pk}(y_{i,j})\}$, $1 \leq j < k$)
      ii. sends δ_i, the encryption of the polynomial f_i to all other players, along with a proof of correct construction $\text{ZKPK}\left\{ a_1, \ldots, a_k \;\middle|\; \begin{array}{ll} & \tau_i = ((x - a_1) *_h \ \ldots \ *_h \ (x - a_{k-1}) *_h \alpha) \\ \wedge & y_{i,1} = E_{pk}(a_1) \wedge \cdots \wedge y_{i,k} = E_{pk}(a_k) \\ \wedge & \alpha = E_{pk}(x - a_k) \end{array} \right\}$
   (c) for $1 \leq j \leq n$
      i. chooses a random polynomial $r_{i,j} \leftarrow R^k[x]$
      ii. sends a commitment to $\Lambda(r_{i,j})$ to all players, where $\Lambda(r_{i,j}) = E_{pk}(r_{i,j})$
2. for $1 \leq j \leq n$
   (a) opens the commitment to $\Lambda(r_{i,j})$
   (b) verifies proofs of plaintext knowledge for the encrypted coefficients of f_j
   (c) sets the leading encrypted coefficient (for x^k) to a known encryption of 1
   (d) calculates $\tau_{i,j}$, the encryption of the polynomial $p_{i,j} = f_j * r_{i,j}$, with proofs of correct multiplication ZKPK$\{r_{i,j} \mid (\tau_{i,j} = r_{i,j} *_h \delta_j) \wedge (\Lambda(r_{i,j}) = E_{pk}(r_{i,j}))\}$ and sends it to all other players
3. Each player $i = 1, \ldots, n$:
   (a) calculates μ, the encryption of the polynomial $p = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j}$, as in Sec. 4.2.2, and verifies all attached proofs
   (b) evaluates the encryption of the polynomial p at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = p((S_i)_j)$, using the algorithm given in Sec. 4.2.2.
   (c) for each $j \in [k]$ chooses a random element r_{ij}, calculates an encrypted element $(V_i)_j = r_{ij} \times_h E_{pk}(c_{ij})$, with attached proof of correct construction ZKPK$\{(r_{ij}, z) \mid ((V_i)_j = r_{ij} \times_h \mu(z)) \wedge (y_{i,j} = E_{pk}(z))\}$, and sends the encrypted element $(V_i)_j$ and the proof of correct construction to all players
4. All players perform the Shuffle protocol on the sets V_i, obtaining a joint set V, in which all ciphertexts have been re-randomized.
5. All players $1, \ldots, n$ decrypt each element of the shuffled set V (and send proofs of correct decryption to all other players)

If nb of the decrypted elements from V are 0, then the size of the set intersection is b.

Figure 8: Cardinality set-intersection protocol for the malicious case.

Protocol: OverThreshold-Mal

Input: There are $n \geq 2$ players, $c < n$ maliciously 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. The commitment scheme used in this protocol is a equivocal commitment scheme. 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.

All players verify the correctness of all proofs sent to them, and refuse to participate in the protocol if any are not correct.

Each player $i = 1, \dots, n$:

1. Each player $i = 1, \dots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \dots (x - (S_i)_k)$
2. Players $1, \dots, c+1$ send commitments to $y_{i,1}, \dots, y_{i,k}$ to all players, where $y_{i,j} = E_{pk}((S_i)_j)$ $(1 \leq j \leq k)$. All players then open these commitments.
3. Player 1 sends to all other players: encrypted elements $y_{1,1} = E_{pk}((S_1)_1), \dots, y_{1,k} = E_{pk}((S_1)_k)$, along with proofs of plaintext knowledge (POPK$\{E_{pk}(y_{1,j})\}$, $1 \leq j < k$); τ_1, the encryption of the polynomial $\lambda_1 = f_1$ to all other players, along with a proof of correct construction ZKPK$\left\{ a_1, \dots, a_k \;\middle|\; \begin{array}{c} \tau_1 = ((x - a_1) *_h \dots *_h (x - a_{k-1}) *_h \alpha) \\ \wedge \;\; y_{1,1} = E_{pk}(a_1) \wedge \dots \wedge y_{1,k} = E_{pk}(a_k) \\ \wedge \;\; \alpha = E_{pk}(x - a_k) \end{array} \right\}$
4. Each player $i = 2, \dots, n$
   (a) receives τ_i, the encryption of the polynomial λ_{i-1}, from player $i-1$
   (b) sends to all other players: encrypted elements $y_{i,1} = E_{pk}((S_i)_1), \dots, y_{i,k} = E_{pk}((S_i)_k)$, along with proofs of plaintext knowledge (POPK$\{E_{pk}(y_{i,j})\}$, $1 \leq j < k$); τ_i, the encryption of the polynomial $\lambda_i = f_i * \lambda_{i-1}$, along with a proof of correct construction ZKPK$\left\{ a_1, \dots, a_k \;\middle|\; \begin{array}{c} \tau_i = ((x - a_1) *_h \dots *_h (x - a_k) *_h \tau_{i-1}) \\ \wedge \;\; y_{i,1} = E_{pk}(a_1) \wedge \dots \wedge y_{i,k} = E_{pk}(a_k) \end{array} \right\}$
5. Each player $i = 1, \dots, c+1$
   (a) choose random polynomials $r_i, s_i \leftarrow R^k[x]$
   (b) calculate α the encryption of the $t-1$th derivative of $p = \lambda_n$, denoted $p^{(t-1)}$, by repeating the algorithm given in Sec. 4.2.2.
   (c) calculate α_i, the encryptions of the polynomial $p * r_i$, and β_i, the encryption of the polynomial $p^{(t-1)} * s_i$ and send it to all other players, along with proofs of correct polynomial multiplication, ZKPK$\{r_i \mid \alpha_i = r_i *_h \tau_n\}$, ZKPK$\{s_i \mid \beta_i = s_i *_h \tau_n\}$
6. Each player $i = 1, \dots, n$:
   (a) calculates μ, the encryption of the polynomial $\Phi = F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right)$, as in Sec. 4.2.2, and verifies all attached proofs
   (b) evaluates the encryption of the polynomial Φ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = p((S_i)_j)$, using the algorithm given in Sec. 4.2.2.
   (c) for each $j \in [k]$ chooses a random element $r_{ij} \leftarrow R$, calculates an encrypted element $(V_i)_j = (r_{ij} \times_h E_{pk}(c_{ij})) + (S_i)_j$, with attached proof of correct construction ZKPK$\{(r_{ij}, z) \mid ((V_i)_j = (r_{ij} \times_h \mu(z)) + z) \wedge (y_{i,j} = E_{pk}(z))\}$, and sends the encrypted element $(V_i)_j$ and the proof of correct construction to all players
7. All players perform the Shuffle protocol on the sets V_i, obtaining a joint set V, in which all ciphertexts have been re-randomized, then jointly decrypt each element of the shuffled set V (and send proofs of correct decryption to all other players).

Each element $a \in P$ that appears b times in V is an element in the threshold set that appears b times in the players' private inputs.

Figure 9: Over-threshold set-intersection protocol for the malicious case.

Security Analysis. We provide a simulation proof of this protocol's security; an intermediary G translates between the real wold with malicious, colluding PPT players Γ and the ideal world, where a trusted third party computes the answer set. Our proof shows that no Γ can distinguish between the ideal world and the real world, thus no information other than that in the answer set can be gained by malicious players. A formal statement of our security property is as follows:

Theorem 18. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, the Shuffle protocol is secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Over-Threshold Set-Union protocol for the malicious case in Fig. 8, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and Γ in the real model.*

Proof of this theorem is given in Appendix D.1.

8 Other Applications

Our techniques for privacy-preserving computation of multiset operations have wide applicability beyond the protocols discussed earlier in Sections 5 and 6. We first discuss the composition of our techniques to compute arbitrary functions based on the intersection, union, and reduction operators. We also propose an efficient method for the Subset problem, determining whether $A \subseteq B$. As an example of the application of our techniques to problems outside the realm of set computation, we describe their use in evaluation of boolean formulas.

8.1 General Set Computation

Our techniques for privacy-preserving set operations can be arbitrarily composed to enable a wide range of privacy-preserving set computations. In particular, we give a grammar describing functions on multisets that can be efficiently computed using our privacy-preserving operations:

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

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.

It is worth noting that the above grammar only allows computation of the union operator when at least one of the two operands is a set known to some player. Although any monotone function on sets can be described by our grammar, in some cases it is desirable (or more efficient) to enable the calculation of the union operator on two sets calculated from other set operations, such that neither operand is known to any player. In this case, we could calculate the union operation in the following way. Let λ and $E_{pk}(f)$ be the encrypted polynomial representations of the two multisets. The players use standard techniques to privately obtain additive shares $f_1, \ldots, f_\nu$ of f, given $E_{pk}(f)$. Using these shares, they then calculate $(f_1 *_h \lambda) +_h \ldots +_h (f_\nu *_h \lambda) = f *_h \lambda$, the encryption of the polynomial representation of the union multiset.

8.2 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.

8.3 Computation of CNF Formulas

Finally, we show that our techniques on private set operations have applications outside of the realm of set computations. As a concrete example, we show that we can apply our techniques to efficient privacy-preserving evaluation of boolean formulas, in particular, the conjunctive normal form (CNF). A formula in CNF is a conjunction of a number of disjunctive clauses, each of which is formed of several variables (or their negations).

Problem Statement Let ϕ be a public CNF boolean formula on variables $V_1, \dots, V_\kappa$. Each player knows the truth assignment to some subset of $\{V_1, \dots, V_\kappa\}$, where each variable is known to at least one player. The players cooperatively calculate the truth value of ϕ under this assignment, without revealing any other information about the variable assignment.

We address this problem by introducing set representations of boolean formulas. Let TRUE, FALSE be distinct elements of R (e.g., 0 and 1). For each variable in the formula, let the set representation of the variable be { TRUE} if its value is true, and { FALSE} if its value is false. Then, replace each $\vee$ operator in ϕ with a $\cup$ operator, and each $\wedge$ operator with a $\cap$ operator. If TRUE is a member of the resulting set, then ϕ is true. The polynomial set representation of the CNF formula can now be evaluated by the players through use of our privacy-preserving multiset operations, as the function is described in the grammar given in Section 8.1.

We can also solve many variations of boolean formula evaluation using our techniques. For example, we might require, instead of using the boolean operations, that at least t of the variables in a clause be satisfied. Note that using our techniques can be more efficient than standard multiparty techniques, as they require an expensive multiplication operation, involving all players, to compute the $\wedge$ operator [2, 16].

Acknowledgments:

We extend our thanks to Dan Boneh, Benny Pinkas, and David Molnar for their invaluable help and comments on the content and presentation of this paper. We also extend our thanks to Luis von Ahn, Lujo Bauer, David Brumley, Bryan Parno, Alina Oprea , Mike Reiter, and anonymous reviewers for their comments.

References

[1] M. Ajtai, J. Komlos, and E. Szemeredi. An $o(n\ logn)$ sorting network. In *Proc. of STOC*, pages 1–9, 1983.

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

[3] Fabirce Boudot, Berry Schoenmakers, and Jacques Traore. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111:77–85, 2001.

[4] Jan Camenisch. Proof systems for general statements about discrete logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich, Mar 1997.

[5] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–8, 1981.

[6] 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.

[7] R. Cramer, I. Damgård, and J. Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of Eurocrypt*, pages 280–99. Springer-Verlag, 2001.

[8] 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.

[9] 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.

[10] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39:77–85, 1996.

[11] P. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting of lotteries. In *Proc. of Financial Cryptography*, 2000.

[12] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Proc. of Asiacrypt*, pages 573–84, 2000.

[13] 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.

[14] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of Crypto*, pages 368–87. Springer-Verlag, 2001.

[15] Rosario Gennaro and Victor Shoup. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15:75–96, 2002.

[16] Oded Goldreich. The foundations of cryptography – volume 2. http://www.wisdom.weizmann.ac.il/ oded/foc-vol2.html.

[17] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and Systems Science*, 28:270–99, 1984.

[18] M. Jakobsson. A practical mix. In *Proc. of Eurocrypt*, pages 448–61. Springer-Verlag, 1998.

[19] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Proc. of Crypto*. Springer-Verlag, 2004.

[20] Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness of private datasets. In *Proc. of Financial Cryptography*, 2005.

[21] Lea Kissner, Alina Oprea, Michael Reiter, Dawn Song, and Ke Yang. Private keyword-based push and pull with applications to anonymous communication. In *Applied Cryptography and Network Security*, 2004.

[22] Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Proc. of Asiacrypt*, pages 416–33, 2003.

[23] Philip MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *Proc. of Eurocrypt*, pages 382–400. Springer-Verlag, 2004.

[24] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. ACM Symposium on Theory of Computing*, pages 245–54, 1999.

[25] A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–25, 2001.

[26] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of Asiacrypt*, pages 573–84, 2000.

[27] Victor Shoup. A computational introduction to number theory and algebra. http://shoup.net/ntb/.

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

A Notation

- P – the set of elements which can be members of a private input set
- k – size of each private input set
- n – number of players participating in a protocol
- t – threshold number, an element must appear t times in the private input sets to be included in the threshold set
- $E_{pk}(\cdot)$ – encryption under the additively homomorphic, public key cryptosystem to which all players share a secret key
- $E_{pk}(a) +_h E_{pk}(b)$ – combination of two ciphertexts (under the homomorphic cryptosystem) to produce a re-randomized ciphertext which is the encryption of $a + b$
- $a \times_h E_{pk}(b)$ – combination of an integer and a ciphertext (under the homomorphic cryptosystem) to produce a re-randomized ciphertext which is the encryption of ab
- $f *_h E_{pk}(g)$ – combination of two polynomials (under the homomorphic cryptosystem) to produce a re-randomized encrypted polynomial which is the encryption of $f * g$
- $h(\cdot)$ – a cryptographic hash function from $\{0,1\}^*$ to $\{0,1\}^\ell$ ($\ell = \lg\left(\frac{1}{\epsilon}\right)$), where ϵ is negligible.
- $\mathrm{Rd}_d(S)$ denotes the element reduction by d of set S
- $R^a[x]$ denotes the set of all polynomials of degree between 0 and a with coefficients from R
- $[c]$ for an integer c denotes the set $\{1, \ldots, c\}$
- $a := b$ denotes that the variable a is given the value b
- $a \mid\mid b$ denotes a concatenated with b
- $a \leftarrow S$ denotes that element a is sampled uniformly from set S
- $f * g$ is the product of the polynomials f, g
- $\deg(p)$ is degree of polynomial p
- $p^{(d)}$ is the dth formal derivative of p
- $\gcd(p, q)$ is the greatest common divisor of p, q
- S_i is the ith player's private input set
- V_j is the jth element of the set V, under some arbitrary ordering

B Proof of Lemma

Theorem 2: *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

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

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. □

C Proofs for Set-Intersection and Cardinality Set-Intersection Protocols

C.1 Set-Intersection

In this section, we give proofs of security and correctness for our protocols for Set-Intersection in the honest-but-curious and malicious cases. For simplicity, we give proof sketches for these theorems.

C.1.1 Honest-But-Curious Case

Theorem 6: *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.*

Proof. Each player learns the decrypted polynomial $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j} \right)$. If $\forall_{i \in [n]}\ f_i(a) = 0$, then $p(a) = 0$. As no elements that are not in every players' private input can be in the set-intersection of all private inputs, all elements in the set-intersection can be recovered by each player. Each element in his private input that a root of p is a member of the intersection set.

We now show that, with high probability, erroneous elements are not inserted into the answer set. By Theorem 2, the decrypted polynomial is of the form $\left(\prod_{a \in I}(x - a)\right) * s$, where s is uniformly distributed over $R^{2k-|I|}[x]$. This random polynomial s is of polynomial size, and thus has a polynomial number of roots. Each of these roots is a representation of an element from P with only negligible probability. Thus, the probability that an erroneous element is included in the answer set is also negligible, and all players learn exactly the intersection set. □

Theorem 7: *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.*

Proof. We assume that the homomorphic cryptosystem (E, D) used in the protocol is in fact secure as we required. Thus, as the inputs of the other players are all encrypted until the decryption is performed, nothing can be learned by any player before that point. Each player j then learns only the summed polynomial $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$.

Note that to every coalition of c players, for every i, $\sum_{j=0}^{c} r_{i+j,j}$ is completely random, as at least one player in the $c+1$ players who chose that random polynomial is not a member of the coalition, and so $\sum_{j=0}^{c} r_{i+j,j}$ is uniformly distributed and unknown.

By Theorem 2, $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right) = \left(\prod_{a \in I}(x-a)\right) * s$, where I is the intersection set and s is uniformly distributed over the polynomials of appropriate degree. Thus no information about the private inputs of the honest players can be recovered from p, other than that given by revealing the intersection set. □

C.1.2 Malicious Case

Theorem 16: *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Set-Intersection protocol for the malicious case in Fig. 7, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and Γ in the real model.*

Proof. In this simulation proof, we give an algorithm for a player G in the ideal model. This player communicates with the malicious players Γ, pretending to be one or more honest players in such a fashion that Γ cannot distinguish that he is not in the real world. We assume that all malicious players can collude. The trusted third party takes the input from G and the honest parties, and gives both G and the honest parties the intersection set. G then communicates with the malicious players Γ, so they also learn the intersection set. A graphical representation of these players is given in Figure 10

We give a sketch of how the player G operates (note that G can prevaricate when opening commitments, as we use an equivocal commitment scheme, and can extract plaintext from proofs of plaintext knowledge):

1. For each simulated honest player i, G:
   (a) chooses a polynomial f_i such that each such polynomial is relatively prime and has leading coefficient 1 (for randomly generated polynomials with leading coefficient 1, this is true with overwhelming probability)
   (b) chooses arbitrary polynomials $r_{i,1}, \ldots, r_{i,n}$ and creates encryptions $\Lambda(r_{i,j})$ from them (in the case of Paillier, specially construct encryptions of those polynomials, and proofs of knowledge of each coefficient, see Section 7.1)
2. Performs step 1 of the protocol:
   (a) sends the encryption of f_i to all malicious players Γ, along with proofs of plaintext knowledge and commitments to $\Lambda(r_{i,j})$ $(1 \leq j \leq n)$
   (b) sends data items $\Lambda(r_{i,j})$ $(1 \leq j \leq n)$ to all malicious players Γ
   (c) Receives from each malicious player $\alpha \in \Gamma$:
      i. encryption of a polynomial f_α and proofs of plaintext knowledge for its coefficients

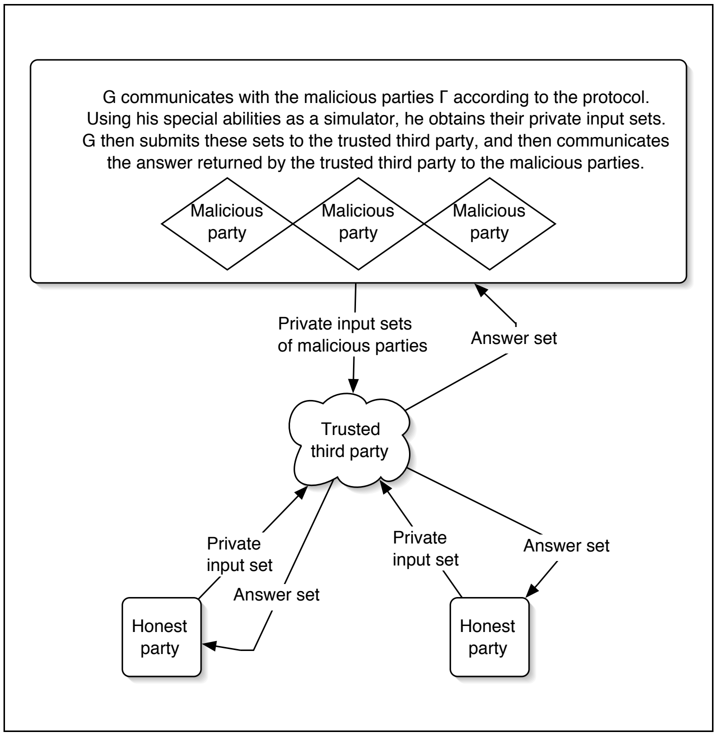

Figure 10: A simulation proof defines the behavior of the player G, who translates between the malicious players Γ, who believe they are operating in the real model, and the ideal model, in which the trusted third party computes the desired answer.

  ii. trapdoor commitments to data items $\Lambda(r_{\alpha,j})$ for each random polynomial $r_{\alpha,j}$, $1 \leq j \leq n$

3. The player G extracts from the proofs of plaintext knowledge and trapdoor commitments to $\Lambda(r_{i,j})$ (in the case of Paillier, the extraction is from the proof of knowledge of the discrete logarithm), the polynomials f_α, and the random polynomials $r_{\alpha,j}$ the malicious players Γ have chosen.
4. G obtains the roots of each polynomial f_α (as these exactly determine, for the purposes of the protocol, his set):
   - If polynomial factoring is possible, G may factor f_α. $f_\alpha(a) = 0 \Leftrightarrow (x-a)|f_\alpha$, so all roots of f_α may be determined by examining the linear factors.
   - If we are working in the random oracle model, then, with overwhelming probability, to correctly represent any element of the valid set P, a player must consult the random oracle. As there can be only a polynomial number of such queries, for each query a, G may check if $f_\alpha(a \mid\mid h(a)) = 0$.
   - If neither of these routes are feasible, then a proof that f_α was constructed by multiplying k linear factors of the form $x - a$ may be added to the protocol instead of proofs of plaintext knowledge. This proof is of size $O(k^3)$, and is constructed by using proofs of plaintext knowledge for some linear factors, and layering proofs of correct multiplication to obtain the complete polynomial f_α. From this proof, each linear factor of f_α can be

obtained, and thus all roots of f_α.

5. G submits the sets represented by these roots to the trusted third party. The honest player submit their private input sets to the trusted third party. The trusted third party returns the intersection set I to G and the honest players.
6. G prepares to reveal the intersection set to the malicious players Γ:
   (a) selects a target polynomial $p = \left(\prod_{a\in I}(x-a)\right) * s$, where s is chosen uniformly from those polynomials of degree $2k - |I|$. (note that, by Theorem 2, this is exactly the polynomial calculated by simply running the protocol)
   (b) chooses a set of polynomials $r_{i,j}$ (where i is one of the simulated honest players) such that $\sum_{i=1}^{n} f_i \left(\sum_{j=1}^{n} r_{i,j}\right) = p$ (from the proof of Theorem 2, we know that such polynomials exist, and can be determined through simple polynomial manipulation)
7. G follows the rest of the protocol with the malicious players Γ as written, except that he opens the trapdoor commitment to reveal an appropriate $\Lambda(r_{i,j})$ for the new chosen $r_{i,j}$. In this way, the players calculate an encryption of the polynomial p chosen by G, and then decrypt it. The coalition players thus learn the intersection set.

Note that the dishonest players cannot distinguish that they are talking to G (who is working in the ideal model) instead of other clients (in the real world), and the correct answer is learned by all parties, in both the real and ideal models.

□

C.2 Cardinality Set-Intersection

In this section, we give proofs of security and correctness for our protocols for Set-Intersection in the honest-but-curious and malicious cases. For simplicity, we give proof sketches for these theorems.

C.2.1 Honest-But-Curious Case

Theorem 8: *In the Cardinality Set-Intersection protocol of Fig. 2, every player learns the size of the intersection of all players' private inputs,* $|S_1 \cap S_2 \cap \cdots \cap S_n|$, *with overwhelming probability.*

Proof. Note that, following the proof of Theorem 6, p is a polynomial representation of the intersection multiset, with overwhelming probability. Each player evaluates p (encrypted) at each of their inputs, then blinds it by homomorphically multiplying a random element by the encrypted evaluation. Thus each resulting encrypted element $(V_i)_j$ $(1 \le i \le n,\ 1 \le j \le k)$ is either 0, representing some element of a private input set in the intersection set, or uniformly distributed, representing some element not in the intersection set. An element is a member of $S_1 \cap \cdots \cap S_n$ if and only if each player holds it as part of their private input set, for each element of $S_1 \cap \cdots \cap S_n$, there are n encrypted evaluations that are 0. Thus, when the encrypted evaluations $(V_i)_j$ $(1 \le i \le n,$ $1 \le j \le k)$ are shuffled and decrypted, there are exactly $n|S_1 \cap \cdots \cap S_n|$ 0s, and thus all players learn the size of the intersection set. □

Theorem 9: *Assuming that the additively homomorphic, threshold cryptosystem* $E_{pk}(\cdot)$ *is semantically secure and that the Shuffle protocol is secure, with overwhelming probability, in the Cardinality Set-Intersection 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.*

Proof. We assume that the cryptosystem $E_{pk}(\cdot)$ and Shuffle protocol are secure, so we may note that no player or coalition of players learns any information from the protocol except the decryption of the randomly-ordered set $\{(V_i)_j\}_{i\in[n],j\in[k]}$. As each element of that set is either 0 or a uniformly distributed element, it conveys no information other than the statement 'some player had an element in their private input set that was/was not in the intersection set'. As this information precisely constitutes the result of the Cardinality Set-Intersection problem, no additional information is revealed. □

C.2.2 Malicious Case

Theorem 17: *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, the Shuffle protocol is secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Cardinality Set-Intersection protocol for the malicious case in Fig. 8, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and Γ in the real model.*

Proof. The simulation proof of this theorem follows the proof of Theorem 16 with only small changes; the additional zero-knowledge proofs in the protocol are generally irrelevant to the simulator. □

D Proofs for the Over-Threshold Set-Union and Threshold Set-Union Protocols

D.1 Over-Threshold Set-Union

D.1.1 Honest-But-Curious Case

Theorem 10: *In the Over-Threshold Set-Union protocol of Fig. 3, 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, with overwhelming probability.*

Proof. All players calculate and decrypt $\Phi = F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right)$. As $\sum_{i=1}^{c+1} r_i$ and $\sum_{i=1}^{c+1} s_i$ are distributed uniformly over all polynomials of approximate size nk, Theorem 2 tells us that $\Phi = \gcd\left(p^{(t-1)}, p\right) * r$, where r is a random polynomial of the appropriate size. As r has only a polynomial number of roots, each of which has a negligable probability of representing a member of P, when Φ is factored, $\gcd\left(p^{(t-1)}, p\right)$ can be recovered.

By Theorem 5, $\gcd\left(p^{(t-1)}, p\right)$ has roots which are exactly those that appear at least t times in the players' private inputs (the threshold set). The players calculate elements $u_{i,j}$, which are uniformly distributed if $(S_i)_j$ is not a member of the threshold set, and $(S_i)_j$ if it does appear in the threshold set. These elements are shuffled and distributed to all players. Each reveals an element of the private input, if that element is in the threshold set, and nothing otherwise. Thus each element in the threshold intersection set is revealed as many times as it appeared in the private inputs. □

Theorem 11: *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. 3, 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.*

Proof. We assume that the cryptosystem employed is semantically secure, and so players learn only the formula $\Phi = F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right)$. Note that both $\sum_{i=1}^{c+1} r_i$ and $\sum_{i=1}^{c+1} s_i$ are uniformly distributed and unknown to all players, as the maximum coalition size is smaller than $c+1$. Thus, by Theorem 2, $\Phi = \gcd\left(p, p^{(t-1)} * F\right) * s$, for some uniformly distributed polynomial s. As s is uniformly distributed for any player inputs, no player or coalition can learn more than $\gcd\left(p, p^{(t-1)} * F\right)$. F is chosen such that $\gcd(p, F) = 1$, and so $\gcd\left(p, p^{(t-1)} * F\right) = \gcd\left(p, p^{(t-1)}\right)$. As was observed in Theorem 10, this information exactly represents the threshold set, and can thus be derived from the answer that would be returned by a trusted third party. Thus no player or coalition of at most c players can learn more than in the ideal model.

Neither do the shuffled elements reveal additional information. As we assume the shuffling protocol is secure, the origin of any element is not revealed. The elements revealed are exactly those in the threshold set, each included as many times as it was included in the private inputs, and thus also do not reveal information to any adversary. □

D.1.2 Malicious Case

Theorem 18: *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, the Shuffle protocol is secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Over-Threshold Set-Union protocol for the malicious case in Fig. 8, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and Γ in the real model.*

Proof. In this simulation proof, we give an algorithm for a player G in the ideal model. This player communicates with the malicious players Γ, pretending to be one or more honest players in such a fashion that Γ cannot distinguish that he is not in the real world. We assume that all malicious players can collude. The trusted third party takes the input from G and the honest parties, and gives both G and the honest parties the intersection set. G then communicates with the malicious players Γ, so they also learn the intersection set. A graphical representation of these players is given in Figure 10.

We give a sketch of how the player G operates (note that G can prevaricate when opening commitments, as we use an equivocal commitment scheme, and can extract plaintext from proofs of plaintext knowledge):

1. For each simulated honest player i, G:
   (a) chooses a set S'_i of arbitrary elements $(S'_i)_1, \ldots, (S'_i)_k \in R$
   (b) Performs steps $1-2$ of the protocol, sending equivocal commitments to the set S_i for each simulated honest player.
2. The player G extracts the private input sets chosen by Γ, for each malicious player, from the equivocal commitments sent in step 2 of the protocol. G submits the sets extracted from

these commitments to the trusted third party. The honest player submit their private input sets to the trusted third party. The trusted third party returns the result set I to G and the honest players.

3. G prepares to reveal the intersection set to the malicious players Γ: G chooses new sets S_i to replace the sets S'_i used to construct the commitment. These sets are chosen to contain the following elements:
   (a) for each element a that appears $b > 0$ in I, and b_Γ times in the private input multisets of the malicious players (Γ), the element a is included $b + t - 1 - b_\Gamma$ times in the multisets S_i
   (b) all elements not specified by the prior rule are chosen uniformly from R
4. G follows the rest of the protocol with the malicious players Γ as written. The coalition players thus learn the result set.

Note that the dishonest players cannot distinguish that they are talking to G (who is working in the ideal model) instead of other clients (in the real world), and the correct answer is learned by all parties, in both the real and ideal models.

□

D.2 Threshold Set-Union

Theorem 12: *In the Threshold Contribution Threshold Set-Union protocol of Fig. 5, every player i ($1 \leq i \leq n$) learns the set $S_i \cap Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Proof. Note that the encrypted computation is performed in accordance with Theorems 3 and 5, and thus the polynomial Φ is a polynomial representation of the multiset $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability. Each player i ($1 \leq i \leq n$) constructs encrypted evaluations of each $a \in S_i$, which are them homomorphically multiplied by a uniformly distributed element by all players. Thus, each ciphertext constructed in this fashion is either 0 (meaning $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$) or uniformly distributed (meaning $a \notin \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$). These ciphertexts are then decrypted; thus, each player i learns which elements of his private input appear in the threshold set $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability. □

Theorem 13: *In the Semi-Perfect Threshold Set-Union protocol of Fig. 4, each player i ($1 \leq i \leq n$) learns the set $Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Proof. Following the proof of Theorem 12, the polynomial Φ is a polynomial representation of the multiset $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability and each shuffled element $T \mid\mid U$ is of one of the following forms:

- For some $a \in S_1 \cup \cdots \cup S_n$, $1 \leq i \leq n$, $T = Enc_i(h(a) \mid\mid a)$, U is an $E_{pk}(a)$ – thus, $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$
- For some $a \in S_1 \cup \cdots \cup S_n$, $1 \leq i \leq n$, $T = Enc_i(h(a) \mid\mid a)$, U is not an $E_{pk}(a)$ – thus, $a \notin \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$

The operation of Step 8 assures that for each $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, a corresponding U is correctly decrypted exactly once – all other decryptions of a are sabotaged to appear uniformly distributed. Thus, all players learn the elements of the set $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability. □

Theorem 14: *In the Perfect Threshold Set-Union protocol of Fig. 6, every player learns the set* $Rd_{t-1}(S_1 \cup \cdots \cup S_n)$*, with overwhelming probability.*

Proof. Following the proof of Theorem 12, the polynomial Φ is a polynomial representation of the multiset $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability and each shuffled (encrypted) element U'_ℓ $(1 \le \ell \le nk)$ is of one of the following forms: $a \in P$ (indicating that $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$), or a uniformly distributed element (which can be distinguished from a representation of an element of P with overwhelming probability). Note that, if U'_ℓ is an encryption of an element a, and $\neg\exists_{\ell' \in [\ell-1]}\, U'_{\ell'}$ such that $U'_{\ell'}$ is also an encryption of a, then W_ℓ is also an encryption of a. (Otherwise, W_ℓ is an encryption of a uniformly distributed element.)

This calculation results in a list of encrypted elements W_ℓ, each of which is of one of the following forms: $a \in P$ (indicating that both: $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, and W_ℓ is with overwhelming probability the only encryption of a in the list), or a uniformly distributed element. Thus, when the players decrypt the list W_ℓ, they learn all elements of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$ exactly once, with overwhelming probability. □

Theorem 15: *Assuming that the additively homomorphic, threshold cryptosystem* $E_{pk}(\cdot)$ *is semantically secure and that the Shuffle protocol is secure, with overwhelming probability, in the Threshold Set-Union protocols of Figs. 4, 5, and 6, 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.*

Proof. Note that in the threshold contribution and perfect variants of Threshold Set-Union, all data is encrypted until the final result sets are revealed through joint decryption. As shown in Theorems 12 and 14, the final sets correspond exactly to the elements revealed (all elements that are not in the result set are uniformly distributed over R, and thus hold no information), no information except the result set is revealed to the players.

In the protocol for semi-perfect Threshold Set-Union, the result set is not decrypted all-at-once, but one element at a time. Theorem 13 shows the the resulting elements correspond exactly to the desired result set, but we must show that the behavior of each player during the process of decryption yields no disallowed information. Note that we require for the security of this protocol that a dishonest coalition hold no more than $t-1$ copies of any given element in their private input sets.

When performing the decryption process, each player learns two pieces of information when a result set element is revealed: the element, and whether the element revealed came from that player's own private input multiset. Each ciphertext is 'tagged', so each player can easily decide whether they constructed that ciphertext. Thus, if a dishonest coalition held at least t copies of any given element, they could determine that at least one other player also held a copy of that element, revealing forbidden information. However, as we have precluded this situation, no information is revealed; if a dishonest coalition holds $t-1$ copies of an element which appears in the result set, they already know that at least one other player holds it (otherwise it would not appear in the result set!). □