Private and Threshold Set-Intersection

Lea Kissner **Dawn Song**

November 2004
CMU-CS-04-182

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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: set-intersection, secure multi-party computation, privacy, cardinality, threshold

Abstract

In this paper we consider the problem of privately computing the intersection of sets (*set-intersection*), as well as several variations on this problem: cardinality set-intersection, threshold set-intersection, and over-threshold set-intersection. *Cardinality set-intersection* is the problem of determining the size of the intersection set, without revealing the actual threshold set. In *threshold set-intersection*, only the elements which appear at least a threshold number t times in the players' private inputs are revealed. *Over-threshold set-intersection* is a variation on threshold set-intersection in which not only the threshold set is revealed, but also the number of times each element in the threshold set appeared in the private inputs.

We propose protocols that are more efficient than those previously known for set-intersection in the malicious case, as well as new protocols for problems which had no previous solution that did not utilize general secure circuit computation:

- Set-intersection protocols for the case of: $n = 2$ malicious parties that do not utilize the cut-and-choose technique; $n \geq 3$ malicious parties, for which there was no previous efficient solution; and multisets, in which elements may appear more than once.
- Cardinality set-intersection protocols secure against $n \geq 2$ malicious parties and $n \geq 3$ honest-but-curious parties, for which there were no previous efficient solutions
- Threshold set-intersection protocols for $n \geq 2$ honest-but-curious parties, for which there was no previous efficient solution
- Over-threshold set-intersection protocols for the case of $n \geq 2$ honest-but-curious parties and the case of $n \geq 2$ malicious parties, for which there was no previous efficient solution
- Fair protocols for all problems (when fairness in decryption is enforced)
- Protocols which are secure against even $n - 1$ dishonest colluding parties

1 Introduction

In this paper we consider problems related to privately computing the intersection of sets: set-intersection, cardinality set-intersection, threshold set-intersection, and over-threshold set-intersection. Let each party hold a private input set. The *set-intersection* is the intersection of the private sets. *Cardinality set-intersection* is the problem of determining the size of the intersection set, without revealing the actual set. In *threshold set-intersection*, only the elements which appear at least a threshold number t times in the players' private inputs are revealed. A variation of threshold set-intersection is *threshold contribution*, in which each player learns the elements in the threshold set that also appear in their private inputs; these are the elements of the threshold set to which they have contributed. *Over-threshold set-intersection* is another variation on threshold set-intersection in which not only the threshold set is revealed, but also the number of times each element in the threshold set appeared in the private inputs.

These problems have many applications in structured sharing of personal or private information, such as medical databases, online dating profiles, and distributed network monitoring. We give several specific examples:

Before choosing a movie to attend, a group of people must determine which movies everyone would like to see. By using a private set-intersection protocol, someone who likes unusual movies can avoid embarrassment while still expressing their preference for that movie; if everyone does like it, they can attend.

Hackers may attack many organizations, causing damage to the networks of each; likewise, bulk email troubles many people. These organizations may determine which hackers or spammers have been damaging a large number of them by utilizing the threshold set-intersection protocol. This information may be used to guard against them, or to determine which are worth pursuing by law-enforcement or network authorities, without revealing sensitive, unrelated information about network traffic.

Pharmacies can use a variation on the threshold set-intersection protocol to catch people who are breaking the law and endangering their health by filling the same prescription at multiple pharmacies. The threshold contribution variant of threshold set-intersection lets only those pharmacies who filled a prescription for a cheating patient learn their identity, protecting the privacy of both honest and dishonest patients.

Airline security checks in the U.S. require that no passengers appear on a 'no-fly' list held by the government. By privately determining the set-intersection between the passenger list and the list of possible terrorists, airlines may protect innocent passengers' privacy, while preventing people on the list from boarding. A similar situation exists when police wish to check that the passengers of a car do not have outstanding warrants for their arrest. If there is no such warrant, their names are not revealed to the computer maintaining the warrant list.

Results of a survey to determine popular musical artists can be distorted by people's desire to avoid embarrassment, as they might not include the names of artists who they do not believe to be sufficiently 'cool'. Allowing people to profess an interest in a musical artist privately might well increase the honesty of participants. The over-threshold set-intersection protocol will reveal the popular artists, and how popular they are.

Governmental agencies have many regulatory barriers to protect privacy of citizens.

However, they often need statistics to ensure the efficiency of services. For example, US agencies might wish to learn how many people are receiving both welfare assistance and medicare assistance who are also seriously ill, as these people will very likely be unable to work regularly for a long period in the future. To determine this statistic without invading privacy, agencies and hospitals can perform the cardinality set-intersection protocol.

Our Contributions Freedman, Nissim, and Pinkas proposed protocols that improved substantially on the best known solution for the private matching problem [12]. Private matching is simply unfair set-intersection. We propose protocols that are more efficient in the malicious case for set-intersection, as well as new protocols for problems which had no previous solution that did not utilize general multi-party computation:

- Set-intersection protocols for the case of: $n = 2$ malicious parties that do not utilize the cut-and-choose technique; $n \geq 3$ malicious parties, for which there was no previous efficient solution; and using multisets, in which elements may appear more than once.
- Cardinality set-intersection protocols secure against $n \geq 2$ malicious parties and $n \geq 3$ honest-but-curious parties, for which there were no previous efficient solutions
- Threshold set-intersection protocols for $n \geq 2$ honest-but-curious parties, for which there was no previous efficient solution
- Over-threshold set-intersection protocols for the case of $n \geq 2$ honest-but-curious parties and the case of $n \geq 2$ malicious parties, for which there was no previous efficient solution
- Fair protocols for all problems (when fairness in decryption is enforced)
- Protocols which are secure against even $n - 1$ dishonest colluding parties

These results are summarized in Table 1 for both the malicious case and the honest-but-curious (HBC) case. The communication complexity of protocols is shown in terms of ciphertexts. The size of these ciphertexts is $\max\left\{\kappa, O\left(\lg|P| + \lg\left(\frac{1}{\epsilon}\right)\right)\right\}$, where P is the set of possible private input set elements, and κ represents the minimum size of the ciphertext domain used in the protocols, which is dependent on the security parameter for the cryptosystem.[1]

In the protocols secure against malicious parties presented in this paper, the overhead in communication complexity is a result of the size of the zero-knowledge proofs employed. These larger proofs, however, are generally built out of simple, efficient proofs, such as a proof of equality of discrete logarithms [4].

We offer security proofs for our protocols for both the honest-but-curious case (indistinguishability proofs) and the malicous case (simulation proofs). These proofs of the correctness and security of our protocols and the proofs are given in Appendix C and E.

Organization We discuss the definitions of the problems addressed in this paper, as well as adversary models in Section 2. In section 3, we introduce the mathematical and cryptographic tools used in our protocols, including an additively homomorphic cryptosystem, operations on encrypted polynomials, and polynomial factoring. Section 4 gives intuition

[1]If the technique of simulated computation over a field is employed, the ciphertexts will be larger, but this is not necessary. For details, see section 3.2.2.

Problem	*Setting*	*Complexity of our solution*	*Complexity of previous solution*
Set-Intersection	2-party HBC	$O(k)$	$O(k)$ [12]
	2-party malicious	$O(k^2)$	cut-and-choose [12]
	$n \geq 3$ party HBC	$O(cnk)$	$O(n^2k)$ [12]
	$n \geq 3$ party malicious	$O(n^2k^2)$	none
Cardinality Set-Intersection	2-party HBC	$O(k)$	$O(k)$ [12]
	$n \geq 3$ party HBC	$O(n^2k)$	none
	$n \geq 3$ party malicious	$O(n^2k^3)$	none
Over-Threshold Set-Intersection	$n \geq 2$ party HBC	$O(n^3k)$	none
	$n \geq 2$ party malicious	$O(n^4k^3)$	none
Threshold Set-Intersection (Threshold contribution or Semi-Perfect Variant)	$n \geq 2$ party HBC	$O(n^3k)$	none

Table 1: Communication complexity comparison for our protocols and previous solutions.

for our protocols, including the representation of sets by polynomials and operations on these polynomials. We propose protocols for the honest-but-curious case in section 5, and protocols for the malicious case in section 6. (Proofs of security are given in the appendices.) Related work is discussed in section 7, and we conclude in section 8.

2 Problems and Models

In this section we define the problems for which we propose protocols, and discuss the adversary models under which we secure these protocols.

2.1 Problem Defintions

Assume we have n parties; each has a private input set S_i ($1 \leq i \leq n$) of size k. By engaging in the protocols for the problems defined below, every player learns the specified answer set.

Set-Intersection All players learn the intersection of all private input sets S_i; that is, each player learns $S_1 \cap S_2 \cap \cdots \cap S_n$. This problem may be considered in two settings: S_i is either a set or a multiset (where elements may be included more than once). If S_i is a simple set, then set-intersection will return a simple set. If S_i is a multiset, then if an element a appears at least b times in each player's private input, then all players learn that a appears at least b times in each private input.

Cardinality Set-Intersection All players learn the size of the intersection set of all private input sets S_i; that is, each player learns $|S_1 \cap S_2 \cap \cdots \cap S_n|$. We require that each S_i be a simple set.

Threshold Set-Intersection 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. An element may appear in a player's private input more than once.

We offer protocols for several variants on threshold set-intersection: threshold contribution, perfect, and semi-perfect. Threshold contribution allows any threshold $t \geq 1$, and each player learns only those elements which appear both in his private input and the threshold set. Perfect threshold set-intersection allows any threshold $t \geq 1$, 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.

Over-Threshold Set-Intersection All players learn which elements appear in the combined private input of the players at least a threshold number t times, and the number of times these elements appeared in the players' private inputs. 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 or the number of times it has appeared. As in threshold set-intersection, an element may appear in a player's private input more than once.

2.2 Adversary Models

In this paper, we present protocols for the aforementioned problems in two standard adversary models. We give only an intuitive notion of each. These notions are formalized in [15].

Honest-But-Curious An honest-but-curious player is assumed to follow the protocol exactly. Security in this model is straightforward: no player or coalition of players (who cheat by sharing their private information) gains information which is not inherent in the output of the calculated function. Formally, consider an ideal scenario in which a trusted third party receives the input of each party, calculates the function output, and broadcasts it to each player. We require that when the parties perform the protocol, no party learns information it does not learn in the ideal case. Thus no player or cheating coalition can distinguish between possible sets of valid private input sets held by non-cheating players. A set of private input sets is valid if its produces the same answer set as the private inputs held by the players. For example, if the element a does not appear in the intersection set of **A** and **B**'s private inputs, then if a does not appear in **A**'s private input, she cannot determine whether a appears in **B**'s private input.

Malicious A malicious player (or coalition of such players) will misbehave in any way within their power to extract extra information in the course of the protocol. We cannot, however, prevent malicious players from choosing their 'private input' arbitrarily or refusing to participate in the protocol at any point. The standard definition of security in this case does not require enforcement against these actions [15]. Instead, the definition of security compares the ideal model (where the trusted third party computes the function, but malicious parties may submit any value as their 'private input') with performing the protocol in the real scenario. If the protocol is secure, we can construct a simulation that translates any strategy of the coalition of malicious players in the real model into the ideal model such that the coalition gains computationally indistinguishable information in the two scenarios.

3 Preliminaries

In this section, we introduce the mathematical and cryptographic tools that we use to construct our protocols.

3.1 Additively Homomorphic Cryptosystem

In this paper we utilize a semantically-secure [16], additively homomorphic public-key cryptosystem. Let $E_{pk}(\cdot)$ denote the encryption function with public key pk. The cryptosystem supports the following two 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 one transforms a ciphertext into a different ciphertext encrypting the same plaintext, in such a way that is it difficult to determine that it is a transformation of the original ciphertext.

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 the decryption is performed by all players acting together, but cannot be performed by fewer than n players. In our protocols for the malicious case, we require that the decryption protocol be secure against malicious players; typically, this is done by requiring each player to prove in zero-knowledge that he has followed the threshold decryption protocol correctly [14].

In the protocols we give for the malicious case, we also require that the homomorphic cryptosystem allow efficient zero-knowledge proofs of plaintext knowledge and zero-knowledge proofs for the correctness of certain operations, as detailed in Section 6.1.

Note that Paillier's cryptosystem [27] satisfies each of the aforementioned requirements: it is additively homomorphic, supports ciphertexts re-randomization and threshold decryption (secure in the malicious case) [10, 11], allows efficient zero-knowledge proofs required in Section 6.1 (standard constructions from [6, 4], and proof of plaintext knowledge [7]).

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

3.2 Polynomials Over Rings

Let R denote the plaintext domain of the homomorphic public key cryptosystem (in Paillier's cryptosystem, R is Z_N). We define the polynomial ring $R[x]$ as the polynomials with coefficients from R. Let f be a polynomial 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. We define the *formal derivative* of f as $\sum_{i=0}^{\deg(f)-1}(i+1)f[i+1]\, x^i$, and the dth formal derivative $f^{(d)}$ as the result of taking the formal derivative sequentially d times. We also define the *encryption of polynomial* f as the ordered list of the encryptions of its coefficients $E_{pk}(f[0]), \ldots, E_{pk}(f[\deg(f)])$ under the homomorphic cryptosystem.

3.2.1 Algorithms for Operations on Encrypted Polynomials

Let f, g, and h be polynomials in $R[x]$ such that $f(x) = \sum_{i=0}^{\deg(f)} f[i]\, x^i$, $g(x) = \sum_{i=0}^{\deg(g)} g[i]\, x^i$, and $h(x) = \sum_{i=0}^{\deg(h)} h[i]\, x^i$. Let a and b be elements 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:

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

Ciphertexts created through these operations should be re-randomized, for security, as specified in the cryptosystem definition.

3.2.2 Polynomial Factoring

Some protocols in this paper can yield better efficiency if efficient polynomial factoring is possible without knowledge of the private key. If R is a field of known order, then we can achieve efficient polynomial factoring [29]. If R is a ring with s subfields of known order, then we can achieve efficient polynomial factoring through polynomial factoring in the subfields. However, in this case, we may obtain a larger number of roots than the degree of the polynomial (in particular, there can be k^s roots for a polynomial of degree k), which will make polynomial factoring more computationally expensive.

However, we are not aware of an additively homomorphic cryptosystem whose domain is a field of publicly-known order of sufficient size. Note that we cannot simply use a single sub-field of the plaintext ring in the otherwise acceptable Paillier cryptosystem; doing so requires revealing that subfield, and thus the factorization of N from which the scheme's security is drawn. The Naccache-Stern cryptosystem [24] gives a ring with subfields of publicly-known orders, but cannot include a large subfield, making factoring inefficient.

Thus in this paper, when appropriate, we give protocols for both the cases where we assume efficient polynomial factoring is possible and where we do not use polynomial factoring. For protocols where we assume efficient polynomial factoring is possible, we currently use a technique called *simulated calculations over a field*. Instead of drawing coefficients directly from a field Z_p, we may instead choose the homomorphic cryptosystem to have a sufficiently large plaintext domain (for example, Z_m) so as to perform all calculations without ever "wrapping around" (being taken modulo m). Threshold decryption can be performed so as to only reveal the plaintext modulo p [20]. However, this is generally quite inefficient. For example, for the over-threshold set-intersection protocol (assuming polynomial factoring) described in this paper, the cryptosystem must have a domain that is of size approximately $n \lg \left(\frac{|P|}{\epsilon}\right) + kn^2$, where P is the valid input set, n the number of players, k the size of each player's private input, and ϵ the negligible probability that a random element will represent a member of P. The protocols that require polynomial factoring, however, may have certain advantages over the alternate protocols, such as removing the need for mix-nets and reducing the number of rounds in the protocol. Also, when new homomorphic cryptosystems are developed that enable efficient polynomial factoring without the private key, they can be directly plugged into our protocols to achieve better efficiency.

3.3 Other Tools

Key-Private Cryptosystem. Given a ciphertext, any player who does not hold the private key cannot distinguish which key was used to create the ciphertext [2].

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, where the simulator can open the 'envelope' to an arbitrary value without being detected by the adversary [19, 22].

Mix-Net and Shuffling Protocol. Either a standard mix-net [5, 17, 8, 13, 26] or the muti-party shuffling computation given in this paper (Figs. 11) can be used to distribute data to all players without revealing the origin.

Hash Function. In this paper, let $h(\cdot)$ denote a hash function from $\{0,1\}^*$ to $\{0,1\}^\ell$ ($\ell = \lg\left(\frac{1}{\epsilon}\right)$, where ϵ is a probability parameter chosen to be negligable). This hash function maps to each output bitstring with uniform and certain ω-wise independent probability (where ω is polynomial in nk). This can be approximated by a cryptographic hash function [23].

3.4 Notation

Our 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
- $h(\cdot)$ – hash function

Mathematical Notation

- $R^a[x]$ – 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
- $a \;||\; b$ denotes a concatenated with b
- $a \leftarrow S$ denotes that element a is sampled uniformly from set S
- $\deg(p)$ 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_j is the jth element of the set S under some arbitrary ordering
- $\mathrm{Dom}(q)$ denotes the domain of function q

4 Overview and Mathematical Intuition

In this section, we give the overview and the mathematical intuition of our protocols.

4.1 Polynomial Representation of Sets

In our problem setting, there are n players, each with a private input set S_i, where $|S_i| = k$, $1 \le i \le n$. The jth element of set i is denoted $(S_i)_j$. We denote the domain of the elements in these sets as P, $\forall_{i \in [n]}\; S_i \subseteq P$.

Let R denote the plaintext domain of the homomorphic cryptosystem we use in our protocols. R, however, must be larger than P, so that a random element drawn from the plaintext domain has only negligible probability of representing an element of P. For example, we could require that only elements of the form $b = a \;\;||\;\; h(a)$ could *represent an element in* P. If $|h(\cdot)| = \lg\left(\frac{1}{\epsilon}\right)$, then there is only ϵ probability that a random element from the plaintext domain is in P.

We represent a set of elements as a polynomial in $R[x]$ where the elements of the set are the roots of the polynomial. For example, given a set of k elements $S_i = \{(S_i)_j\}_{1 \le j \le k}$, we construct its polynomial representation as $f_i(x) = \prod_{1 \le j \le k}(x - (S_i)_j)$.

4.2 Mathematical Intuition

By representing sets of elements as polynomials, we can use mathematical properties of polynomials to help compute the different variations of the set-intersection and threshold set-intersection problems:

Intersection using polynomial addition. When two polynomials f and g are added, the shared roots of f and g will be preserved: $(f(a) = 0) \wedge (g(a) = 0) \rightarrow (f + g)(a) = 0$. This approximately represents an intersection operator.

Union using polynomial multiplication. Multiplying two such polynomials approximately represents a union operator that preserves duplicates: $(f(a) = 0) \wedge (g(b) = 0) \rightarrow ((f * g)(a) = 0) \wedge ((f * g)(b) = 0)$, and, as $f(a) = 0 \Leftrightarrow (x - a) \mid f$, $(f(a) = 0) \wedge (g(a) = 0) \rightarrow (x - a)^2 \mid (f * g)$.

Removing duplicate elements using polynomial derivative. To reduce the number of duplicate elements represented in a polynomial p by d, one may take the dth derivative of the polynomial p, denoted as $p^{(d)}$, as formalized by this theorem: (proof given in Section A)

Theorem 1. *Let $p = \prod_{i=1}^{nk}(x - a_i)$ where $\forall_{i \in [nk]}\ a_i \in R$ for a ring R, $t \geq 1$.*

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

Masking non-common factors of polynomials. To hide factors which are not shared between two polynomials f and g, we select two random polynomials r and s in $R[x]$ with sufficiently high degree, and calculate $f * r + g * s$. The resulting polynomial completely hides all information except for those factors shared between f and g, as formalized by this theorem: (proof given in Section A)

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

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

4.3 Overview of Protocols

In all of our protocols, there are n players, and each player has an input set of k elements $S_i = \{(S_i)_j\}_{1 \leq j \leq k}$. We call the polynomial representation of each player's input set S_i its *input polynomial*, $f_i(x) = \prod_{1 \leq j \leq k}(x - (S_i)_j)$. We give a brief overview of our protocols below.

In the set-intersection protocol (Fig. 1), the players wish to add all their input polynomials $f_i = \prod_{j=1}^{k}(x - (S_i)_j)$ $(1 \leq i \leq n)$ to obtain a polynomial that preserves the roots representing elements that appear in each private input set S_i, to obtain a polynomial representing the intersection set. By multiplying the random polynomials $r_{i+j,j}$ by the polynomials f_i before adding the products $f_i * r_{i+j,j}$ together to obtain $p = \sum_{i=1}^{n} f_i \left(\sum_{j=0}^{c} r_{i+j,j} \right)$, they also hide all information about the input sets except for the intersection set.

The cardinality set-intersection protocol (Fig. 7) calculates p in the same way as the set-intersection protocol, but instead of decrypting the polynomial, all players evaluate each element in their input sets, to determine the number of roots of the polynomial.

In the over-threshold set-intersection protocol with polynomial factoring (Fig. 2), the players multiply their input polynomials $f_i = \prod_{j=1}^{k}(x - (S_i)_j)$ to obtain $p = \prod_{i=1}^{n} f_i$, which represents a union of all private input elements, with duplicates preserved. Taking the derivative of a polynomial reduces the number of duplicates of each repeated root by one, so $p^{(t-1)}$ only retains representations of those input elements which appeared at least t times in p. By multiplying the random polynomials r_i and s_i by $p^{(t-1)} * F$ (where F is a padding polynomial) and p to obtain $p^{(t-1)} * F * (\sum_{i=1}^{n} r_i) + p\,(\sum_{i=1}^{n} s_i)$, all information about the input sets except for those polynomial factors that appear in both $p^{(t-1)} * F$ and p are hidden. The polynomial factors that so appear represent the threshold set, and how many times each element in the threshold set is repeated.

The over-threshold protocol without polynomial factoring (Fig. 3) and threshold set-intersection protocols (Figs. 4, 5, 6) calculate a polynomial representing the threshold set in the same way as the over-threshold protocol with polynomial factoring, but instead of decrypting the polynomial, all players evaluate the elements in their input sets, to determine which are roots of the polynomial.

5 Protocols for the Honest-But-Curious Case

We present, in this section, protocols for the problems of: set-intersection, over-threshold set-intersection, threshold set-intersection, and cardinality set-intersection. Each of these protocols is secure against any dishonest coalition of fewer than n honest-but-curious players, where there are n total players. A security analysis is given in section 5.6.

5.1 Set-intersection

The first protocol we present is for the set-intersection problem, given in Fig. 1. In step 1, each player i ($1 \le i \le n$) first calculates his input polynomial f_i and sends the encryption of its polynomial f_i to c other players (where c is the maximum dishonest coalition size), making $c+1$ players in all who have this encrypted polynomial. Each of these players $i+j$ ($0 \le j \le c$) chooses a random polynomial $r_{i+j,i}$, and computes the encryption of $f_i * r_{i+j,j}$. Note that, as no coalition of players can be of size $c+1$, not all of the random polynomials multiplied by any polynomial f_i can be known to the dishonest coalition. Thus, the effective random polynomial that is obtained by adding these polynomials $(\sum_{j=0}^{c} r_{i+j,j})$ is uniformly distributed and unknown to all players. Each player i ($1 \le i \le n$) adds their polynomials $f_{i-j} * r_{i,i-j}$ ($0 \le j \le c$) to produce ϕ_i, and these polynomials are then added in steps 2-3 to produce $p = \sum_{i=1}^{n} \phi_i = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j} \right)$.

If a is a root of some polynomial f_i, it is also a root of $f_i * r$ for a random polynomial

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 (Sec. 3.2.2).

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. 3.2.1.
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. 3.2.1.
   (c) sends the encryption of the polynomial λ_i to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$ to all other players.
5. All players perform a group decryption to obtain the polynomial p.
6. Each player $i = 1, \ldots, n$ may determine which of his items j are in the intersection of all private inputs as follows: if $p((S_i)_j) = 0$, then it is in the intersection; otherwise it is not.

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

r, and thus all members of the private input sets are preserved as roots when the random polynomials are multiplied in. If a is a root of the polynomials f and g, then it is also a root of the polynomial $f + g$. Thus, when all the polynomials $f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$ are added together, if an element was in every private input set, its representation will be a root of the final polynomial. If it was not, the use of the random polynomials ensures that it is both hidden and does not appear as a root with overwhelming probability (see Theorem 2). The players jointly decrypt this result polynomial and each player tests to see if the representation of each member of their private input set is a root of the polynomial. Thus every player learns the set-intersection of all players' private inputs.

Note that the protocol is identical when the private input sets S_i are either sets or multisets (as described in Section 2.1).

5.2 Over-Threshold Set-Intersection (With Polynomial Factoring)

A protocol for the over-threshold set-intersection is given in Fig. 2. To recover the answer set, the players must be able to factor polynomials (see Sec. 3.2.2). In this protocol the players calculate the product of the polynomials f_i created from their private inputs, creating a polynomial $p = \prod_{i=1}^{n} f_i$. Duplicate elements in players' input sets are represented by repeated factors; if a appears b times in the private inputs, then $(x-a)^b \mid p$. Players

Protocol: OverThreshold-Factor-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 (Sec. 3.2.2). 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. 3.2.1.
   (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. 3.2.1.
   (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. 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 factors Φ. Each factor of the form $x - a$, where a represents an element of P, indicates that a is in the set intersection. If the factor $x - a$ appears b times, a appeared $t + b - 1$ times in the players' private inputs.

Figure 2: Over-threshold set-intersection protocol for the honest-but-curious case (with polynomial factoring)

$1, \ldots, c+1$ then calculate the $t-1$th derivative of that polynomial, reducing the number of repetitions of *each* linear factor in p by $t-1$ (see Theorem 1). Each player i $(1 \leq i \leq c+1)$ chooses random polynomials r_i and s_i, and calculates $p * r_i + F * p^{(t-1)} * s_i$, where F is a polynomial used to pad $p^{(t-1)}$ to the same degree as p. All players then add these polynomials to obtain $\Phi = p * \left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} s_i\right)$. Each linear factor $(x-a)$ which appears $v \geq t$ times in p appears $v - t + 1$ in $p^{(t-1)}$, and for each element appearing in the threshold set which appeared $v \geq t$ times in the private inputs, $(x-a)^{v-t+1} \mid \Phi$. When Φ is factored, these linear factors can be recovered, so that every player learns the solution to the over-threshold problem. The random polynomials $\sum_{i=1}^{c+1} r_i$ and $\sum_{i=1}^{c+1} s_i$ ensure all information except for the answer set is hidden.

To see that, with overwhelming probability, no extra linear factors representing private set elements are included in Φ, we may note that unless a factor appears in both p and $p^{(t-1)}$, with overwhelming probability it will not appear in Φ (see Theorem 2). F is chosen so as to exclude possible factors of p, and the only factors that appear in both p and $p^{(t-1)}$

are those that represent the threshold set, with $v - t + 1$ repetitions in $p^{(t-1)}$ if there were $v \geq t$ repetitions in p (see Theorem 1).

This protocol requires polynomial factoring, but has two advantages over the version which does not: it does not need mix-nets and utilizes fewer rounds.

5.3 Over-Threshold Set-Intersection (Without Polynomial Factoring)

Another protocol for the over-threshold set-intersection problem is presented in Fig. 3. This protocol does not require polynomial factoring. In this protocol, each player creates an input polynomial f_i with roots representing the elements of their private input. The players then calculate the encrypted polynomial $\Phi = p * \left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} s_i\right)$ where $p = \prod_{i=1}^{n} f_i$ and r_i and s_i are chosen randomly. As described in Section 5.2, the roots of this polynomial are either: (1) exactly those represent elements in the threshold set, or (2) are random, and thus with overwhelming probability are irrelevent, as they do not represent elements from the valid set P.

Each player then computes the encrypted evaluation of Φ at the points that represent their private input. With overwhelming probability, each such encrypted evaluation is an encryption of 0 if that element is in the intersection set, and non-zero otherwise. The encrypted element $(V_i)_j$ calculated from this encrypted evaluation is thus 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). (This technique is related to the conditional disclosures of [1].) These ciphertexts are shuffled and then decrypted, revealing each element a that appears in the intersection set, with as many repetitions $c \geq t$ as appeared in the initial private inputs. The other decrypted elements are random, and thus with overwhelming probability do not represent members of the valid set P. This both hides unpopular elements from the players' private inputs and ensures that incorrect elements are not inserted into the answer set.

5.4 Threshold Set-Intersection

The protocols for the threshold set-intersection problem, given in Figs. 4, 5, and 6, are identical to the protocol for over-threshold set-intersection 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-Intersection 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.

Protocol: OverThreshold-NoFactor-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 (Sec. 3.2.2). 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 $i = 1, \dots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \dots (x - (S_i)_k)$
2. Player 1 sends the encryption of the polynomial $\lambda_1 = f_1$ to player 2
3. Each player $i = 2, \dots, n$
   (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. 3.2.1.
   (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 $2, \dots, c+1$
5. Each player $i = 1, \dots, c+1$
   (a) calculate the encryption of the $t-1$th derivative of p, denoted $p^{(t-1)}$, by repeating the algorithm given in Sec. 3.2.1.
   (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 $i = 1, \dots, n$
   (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. 3.2.1.
   (b) for each $j = 1, \dots, k$ chooses a random number $r_{ij} \leftarrow \mathrm{Dom}(E_{pk})$ and calculates an encrypted element $(V_i)_j = (r_{ij} \times_h E_{pk}(c_{ij})) +_h E_{pk}((S_i)_j)$
7. All players perform shuffling on their private input sets V_i either through use of a mix-net or by using the shuffling protocol given in Fig. 11, obtaining a joint set V.
8. All players
   (a) decrypt each element of the shuffled set V
   (b) for each element, if the element a represents an element of the valid set P, then a is in the threshold set. If a appears b times, then a appeared b times in the players' private inputs.

Figure 3: Over-threshold set-intersection protocol for the honest-but-curious case (Does not require polynomial factoring.)

Semi-Perfect Threshold Set-Intersection 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

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 to a homomorpic cryptosystem (Sec. 3.2.2). Each player has their own secret key to a key-private cryptosystem, where encryption and decryption are denoted (Enc_i, Dec_i). 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. 3.2.1.
   (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. 3.2.1.
   (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. 3.2.1.
   (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 $r_{ij} \leftarrow \mathrm{Dom}(E_{pk})$ 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. All players perform shuffling on their private input sets V_i either through use of a mix-net or by using the protocol given in Fig. 11.
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 represents an element of P, then a is in the threshold set

Figure 4: Threshold set-intersection protocol for the honest-but-curious case (semi-perfect variant). (Does not require polynomial factoring.)

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 to a homomorpic cryptosystem (Sec. 3.2.2). Each player has their own key to a key-private cryptosystem, where encryption and decryption are denoted (Enc_i, Dec_i). 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. 3.2.1.
   (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. 3.2.1.
   (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. 3.2.1.
   (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 $U_{jm} = (\sum_{\ell=1}^{n} b_{\ell,j,m}) \times_h c_{jm}$ $(1 \leq j \leq n,\ 1 \leq m \leq k)$
8. All players decrypt the ciphertexts U_{ij}, so that only player i learns the decryption a. If $a = 0$, then $(S_i)_j$ is in the threshold set.

Figure 5: Threshold set-intersection protocol for the honest-but-curious case (threshold contribution variant). (Does not require polynomial factoring.)

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 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 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-Intersection 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. The players then use a mix-net to shuffle them. 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 [20].) The players $i \in [n]$ then choose random elements $q_{i,j} \leftarrow R$ $(1 \leq j \leq nk)$ and decrypt the ciphertexts $W_j = U'_j +_h E_{pk}\left(\left(\sum_{i=1}^{n} q_j\right)\left(\mathrm{IsEq}(U'_j, U'_{j-1}) + \cdots + \mathrm{IsEq}(U'_j, U'_1)\right)\right)$. Thus, if U'_j 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.

5.5 Cardinality Set-Intersection

The cardinality set-intersection protocol, given in Fig. 7, is essentially a combination of the set-intersection protocol in Fig. 1 and the over-threshold protocol in Fig. 3. Jointly, the players calculate the polynomial p whose roots represent the set-intersection of the private inputs, as described in Section 5.1. Instead of decrypting the polynomial, like in the set-intersection protocol, the players evaluate the elements of their private input sets in this encrypted polynomial, like in the over-threshold protocol described in Section 5.3. After shuffling these encrypted results, the players decrypt them. If an element a is in the intersection set then both $p(a) = 0$ and n players have this element in their private input. If an element a' is not in the intersection set, then with overwhelming probability, $p(a') \neq 0$. Thus for each element in the intersection set there will be n elements which decrypt to 0, and elements in the intersection set will decrypt to uniformly distributed, non-zero elements. The number of elements in the intersection set can thus be determined by dividing the total number of 0 elements by the number of players, n.

5.6 Security and Correctness

A protocol is correct if each player learns the appropriate answer set at its termination. This is proved for our set-intersection, over-threshold set-intersection (with and without

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 to a homomorpic cryptosystem (Sec. 3.2.2). Each player has their own key to a key-private cryptosystem, where encryption and decryption are denoted (Enc_i, Dec_i). 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. $\mathrm{IsEq}(C, C') = 1$ 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. 3.2.1.
   (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. 3.2.1.
   (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. 3.2.1, and sends them to all players
   (b) for each $i' = 1, \ldots, n$, $j = 1, \ldots, k$ chooses a random number $r_{i'j} \leftarrow \mathrm{Dom}(E_{pk})$ 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. All players perform shuffling on their private input sets V_i either through use of a mix-net or by using the protocol given in Fig. 11.
8. For each shuffled ciphertext U'_j with arbitrary ordering index $j \in [nk]$, the players $i = 1, \ldots, n$
   (a) each player i chooses a random number $q_{i,j} \leftarrow R$
   (b) calculate $W_j = U'_j +_h E_{pk}\left(\left(\sum_{i=1}^{n} q_{i,j}\right)\left(\mathrm{IsEq}(U'_j, U'_{j-1}) + \cdots + \mathrm{IsEq}(U'_j, U'_1)\right)\right)$
9. All players decrypt the ciphertexts W_j. For each decrypted W_j, player i obtains an element a. If a represents an element of P, then a is in the threshold set.

Figure 6: Threshold set-intersection protocol for the honest-but-curious case (perfect variant). (Does not require polynomial factoring.)

Protocol: Cardinality-HBC

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(Sec. 3.2.2).

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. 3.2.1.
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. 3.2.1.
   (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. 3.2.1.
   (b) for each $j = 1, \ldots, k$ chooses a random number $r_{ij} \leftarrow \mathrm{Dom}(E_{pk})$ and calculates an encrypted element $(V_i)_j = r_{ij} \times_h E_{pk}(c_{ij})$
6. All players perform shuffling on their private input sets V_i by using mix-net, obtaining a joint set V, in which all ciphertexts have been re-randomized.
7. All players
   (a) decrypt each element of the shuffled set V
   (b) if na of the decrypted elements are 0, then the size of the set intersection is a

Figure 7: Cardinality set-intersection protocol for the honest-but-curious case. (Does not require polynomial factoring.)

polynomial factoring), and threshold set-intersection in Theorems 4, 9, 11, and 7, which are given in the appendix. Proof for cardinality set-intersection follows from the proofs for set-intersection and over-threshold set-intersection (without polynomial factoring).

Each of these protocols is secure in the honest-but-curious model; no player gains information that it would not gain when using its input in the ideal model. A formal statement of our security property is as follows:

In the protocol, any honest-but-curious player learns no more information than would be gained by using the same private input in an ideal setting.

Application and proof of this theorem to the set-intersection, over-threshold set-intersection (with and without polynomial factoring), and threshold set-intersection protocols is given in Theorems 5, 10, 12, and 8. Proof for cardinality set-intersection follows from the proofs for set-intersection and over-threshold set-intersection (without polynomial factoring). Proofs are given in the appendix.

6 The Malicious Case

Protocols secure against malicious players largely follow those secure against honest-but-curious players, given in Section 5. We have added zero-knowledge proofs, verified by all players, to ensure the correctness of all computation. In this section, we first introduce notations for the zero-knowledge proofs we use to ensure security of our protocols for the malicious case, then give the protocols secure against malicious parties. An analysis of security is given in section 6.5.

6.1 Zero-Knowledge Proofs

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

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

6.2 Set-Intersection

Our protocol for malicious parties performing set-intersection, given in Fig. 8, 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.

Protocol: 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(Sec. 3.2.2). 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. 3.2.1, and verifies all attached proofs
   (b) perform a group decryption to obtain the polynomial p, and distribute proofs of correct decryption
4. For each element $a \in S_i$, if $p(a) = 0$, then a is in the intersection set; otherwise, it is not.

Figure 8: Set-intersection protocol for the malicious case.

6.3 Cardinality Set-Intersection

We give a protocol, secure against malicious parties, to perform cardinality set-intersection in Fig. 9. It proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 7. 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

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 (Sec. 3.2.2). 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
      $$\text{ZKPK}\left\{ (S_i)_1, \ldots, (S_i)_k \;\middle|\; \begin{array}{ll} & \tau_i = ((x - (S_i)_1) *_h \ldots *_h (x - (S_i)_{k-1}) *_h \alpha) \\ \wedge & y_{i,1} = E_{pk}((S_i)_1) \wedge \cdots \wedge y_{i,k} = E_{pk}((S_i)_k) \\ \wedge & \alpha = E_{pk}(x - (S_i)_k) \end{array} \right\}$$
   (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. 3.2.1, 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. 3.2.1.
   (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 shuffling on the sets V_i through use of a mix-net, obtaining a joint set V, in which all ciphertexts have been re-randomized.
5. All players
   (a) decrypt each element of the shuffled set V (and send proofs of correct decryption to all other players)
   (b) if na of the decrypted elements are 0, then the size of the set intersection is a

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

Protocol: OverThreshold-Factor-Mal
Input: There are $n \geq 2$ players, $c < n$ malicious and 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 (Sec. 3.2.2). 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.

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
2. Players $1, \ldots, c+1$
   (a) choose random polynomials $r_i, s_i \leftarrow R^k[x]$
   (b) send a commitment to $\Lambda(r_i)$ and $\Lambda(s_i)$ to all players where $\Lambda(r_i) = E_{pk}(r_i)$ and $\Lambda(s_i) = E_{pk}(s_i)$
3. Player 1 sends the encryption of $\lambda_1 = f_1$ to all players, along with proofs of plaintext knowledge for all coefficients (POPK$\{E_{pk}((f_1)_j)\}$, $0 \leq j \leq k$).
4. 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. 3.2.1.
   (c) sends the encryption τ_i of the polynomial λ_i to all players, along with a proof of correct polynomial multiplication ZKPK$\{f_i \mid \tau_i = f_i *_h E_{pk}(\lambda_{i-1})\}$
5. Each player $i = 1, \ldots, c+1$
   (a) open the commitment to $\Lambda(r_i)$ and $\Lambda(s_i)$
   (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. 3.2.1.
   (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$\{r_i \mid (\alpha_i = r_i *_h \tau_n) \wedge (\Lambda(r_i) = E_{pk}(r_i))\}$, ZKPK$\{s_i \mid (\beta_i = s_i *_h \alpha) \wedge (\Lambda(s_i) = E_{pk}(s_i))\}$
6. All players perform a group decryption to obtain the polynomial $\Phi = F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right)$, and distribute proofs of correct decryption
7. Each player factors Φ. Each factor of the form $x - a$, where a represents an element of P, indicates that a is in the set intersection. If the factor $x - a$ appears b times, a appeared $t + b - 1$ times in the players' private inputs.

Figure 10: Over-threshold set-intersection protocol for the malicious case (with polynomial factoring).

operate like the honest-but-curious protocol in Fig. 7.

6.4 Over-Threshold Set-Intersection (With Polynomial Factoring)

The over-threshold set-intersection protocol for malicious parties uses polynomial factoring, and is given in Fig. 10. It is substantially similar to the protocol secure against honest-but-curious parties given in Fig. 2. We have added zero-knowledge proofs to prevent misbehavior in choosing the coefficients of polynomials f_i, polynomial multiplication, or computing the polynomial's derivitive, and the computations performed by the parties are broadcast, so the correctness of these proofs may be checked by all parties. Note that we do not prevent players from setting their polynomials to 0, like in the set-intersection protocol (Fig. 8); if a player sets $f_i = 0$, then $p = 0$, and no information is revealed by the protocol. We can thus detect and avert misbehavior by malicious players, enforcing that this protocol operates like the protocol secure against honest-but-curious parties.

6.5 Security Analysis

Each of these protocols is secure in the simulation model; an intermediary G translates between the real wold with malicious, colluding players Γ and the ideal world, where a trusted third party computes the answer set. This proof shows that 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:

In the protocol, 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.

Application and proof of this theorem for set-intersection, cardinality set-intersection, and over-threshold set-intersection are given in Theorems 6, 14, and 13. These proofs are given in the appendix.

7 Related Work

Freedman, Nissim, and Pinkas proposed protocols for the problems of set-intersection and cardinality set-intersection (for $n = 2$ players) [12]. They do not, however, address certain problems, such as set-intersection using multisets, $n \geq 3$ player set-intersection in the malicious case, $n \geq 3$ player cardinality set-intersection, threshold set-intersection, and over-threshold set-intersection, which are addressed in our paper. In addition to addressing and introducing many new problems, our complexity results are comparable or more efficient than those for the protocols proposed in their paper; a summary comparison of the results of our paper to theirs is given in Table 1.

Private equality testing is the problem of set-intersection for the limited case $k = 1$. Generalized circuit evaluation gives a protocol for privately computing equality with $O(\lg |P|)$ overhead, where P is the domain from which elements are chosen. Protocols for this problem are proposed in [9, 25, 21], and are approximately as expensive. Fairness is added in [3].

Determining whether input sets (subsets of $[|P|]$) are disjoint (without privacy) has communication overhead of $\Theta(|P|)$ [18, 28]. This implies that determining the cardinality of the set-intersection requires at least $\Theta(|P|)$ communication as well, and the communication complexity of cardinality set-intersection is proportional to the size of the input set k.

8 Conclusions

We present in this paper protocols for four problems and several variations on these problems: set-intersection, cardinality set-intersection, over-threshold set-intersection, and threshold set-intersection. These protocols are secure against honest-but-curious parties. We also include protocols for set-intersection, cardinality set-intersection, and over-threshold set-intersection (with polynomial factoring) secure against malicious players.

Comparison of communication complexity of our solutions with previous results is given in Table 1. We reiterate advantages of our solutions: they are fair (when fairness in decryption is enforced), efficient, include solutions secure against malicious parties, include solutions to problems for which the best previous solutions utilize general multi-party computation, and are secure against any dishonest coalition that does not include all players.

Acknowledgments: We extend our thanks to Dan Boneh, David Molnar, David Brumley, Alina Oprea, and Luis von Ahn for their invaluable comments.

Contact: Lea Kissner can be contacted at `leak@cs.cmu.edu`. Dawn Song can be contacted at `dawnsong@cmu.edu`.

Note: This paper last modified January 2005.

References

[1] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology – Eurocrypt 2001*, 2001.

[2] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology – Theory and Application of Cryptology and Information Security*, pages 566–82. Springer-Verlag, 2001.

[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, *Advances in Cryptology – Crypto 1986*, pages 200–212. Springer-Verlag, 1986.

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

[8] Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *Advances in Cryptology – Eurocrypt 2000*, pages 557–72. Springer-Verlag, 2000.

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

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

[11] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Proc. of Asiacrypt 2001*, pages 573–84, 2000.

[12] Michael Freedman, Kobi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – Eurocrypt 2004*, volume LNCS 3027, pages 1–19. Springer-Verlag, May 2004.

[13] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology – Crypto 2001*, pages 368–87. Springer-Verlag, 2001.

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

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

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

[17] M. Jakobsson. A practical mix. In *Advances in Cryptology – Eurocrypt 1998*, pages 448–61. Springer-Verlag, 1998.

[18] Bala Kalyanasundaram and Georg Schitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Mathematics*, 5:545–57, 1992.

[19] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology – Crypto 2004*. Springer-Verlag, 2004.

[20] 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.

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

[22] Philip MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *Advances in Cryptology – Eurocrypt 2004*, pages 382–400. Springer-Verlag, 2004.

[23] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[24] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM Conference on Computer and Communications Security*, pages 59–66, 1998.

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

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

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

[28] Alexander A. Razborov. Application of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10:81–93, 1990.

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

A Proofs of Polynomial Theorems

Theorem 1: *Let $p = \prod_{i=1}^{nk}(x - a_i)$ where $\forall_{i\in[nk]}\ a_i \in R$ for a ring R (or field F), $t \geq 1$.*

1. *If $(x-a)^t \mid p$, then $(x-a) \mid p^{(t-1)}$.*
2. *If $(x-a) \mid p$ and $(x-a)^t \not\mid p$, then $(x-a) \not\mid p^{(t-1)}$.*

Proof. This theorem follows from 9.16 in Shoup's Computational Introduction to Number Theory and Algebra. [29].

□

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

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

Proof. Firstly note that the number of possible r, s pairs is $|R|^{2\beta+2}$, and thus there are that many potentially unique mappings from f, g to u. However, $\deg(u) = \alpha+\beta$, and so there are only $|F|^{\alpha+\beta+1}$ result polynomials. We thus must show that the same number of r, s pairs map to each result polynomial, and each result polynomial can be mapped to by at least one choice of r, s. This implies that $f*r+g*s$ is distributed uniformly over all polynomials of $\deg(f*r+g*s)$.

Note that if two pairs of polynomials r, s, r', s' ($r \neq r'$, $s \neq s'$) that map f, g to the same output polynomial u:

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

Because $\gcd(f,g) = 1$, $g \mid (r-r')$ and $f \mid (s'-s)$, this transformation is valid. Thus $r - r' = g*p$ and $s'-s = f*p$ for some polynomial p such that $\deg(p) = \beta-\alpha$.

We wish to show that the number of pairs of such polynomials that map to the given result polynomial t is equal. Pick any pair of polynomials r, s such that $f*r+g*s = t$. Choose p such that $\deg(p) = \beta - \alpha$. This fixes $\frac{r-r'}{g}$ and $\frac{s'-s}{f}$. As we have already chosen r and s, this fixes r' and s'. We may then simply count how many polynomials p exist, as each choice counts exactly one pair r', s' such that $f*r'+g*s' = u$, and this counts every such pair, as shown above (our original pair r, s is counted with the polynomial $p = 0$). There are $|R|^{\beta-\alpha+1}$ such polynomials p, and thus an equal number of pairs of polynomials that map to any given output pair for which there exists at least one mapping.

We now show that every result polynomial must have these same number of polynomial pairs r, s that map to it. There are exactly $|R|^{\beta-\alpha+1}$ mappings to any polynomial that has at least one mapping. There are exactly $|R|^{\beta+\alpha+1}$ possible result polynomials t, as $\deg(u) = \beta+\alpha$. The mappings to each result polynomial multiplied by the number of result polynomials must exactly equal the total number of mappings $|R|^{2\beta+2}$.

$$|R|^{\beta-\alpha+1} * |R|^{\beta+\alpha+1} = |R|^{2\beta+2}$$

Protocol: Shuffle-Factor
Input: There are $n \geq 2$ players, each with a private input set V_i, such that $|V_i| = k$. The players share the secret key SK, to which PK is the corresponding public key to a homomorphic cryptosystem (Sec. 3.2.2).
Additional zero-knowledge proofs necessary to make the protocol secure against malicious adversaries are specified in parentheses.

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (V_i)_1) \ldots (x - (V_i)_k))$
2. Player 1 sends the encrypted polynomial $\lambda_1 = f_1$ to player 2 (with proofs of plaintext knowledge of the encrypted coefficients POPK$\{E_{pk}((f_i)_j)\}$, $0 \leq j \leq k$)
3. Each player $i = 2, \ldots, n$
   (a) receives τ_{i-1}, 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. 3.2.1
   (c) sends τ_i, the encrypted coefficients of the polynomial λ_i to player $i + 1 \bmod n$ (with proofs of correct polynomial multiplication ZKPK$\{f_i \mid \tau_i = f_i *_h E_{pk}(\lambda_{i-1})\}$, to all players)
4. All players perform decryption to obtain $p = \lambda_n = \prod_{i=1}^{n} f_i$, and factor p into a linear factors of the form $(x - a)$. For each such factor, a is a member of the answer set. (Each player proves the correctness of their decryption.)

Figure 11: Shuffling protocol for $n \geq 2$ parties. (Requires polynomial factoring.)

Thus every possible result polynomial must have $|R|^{\beta-\alpha+1}$ mappings to it, and the mappings are thus distributed uniformly over the entire space of polynomials of deg$(f * r + g * s)$. $\square$

Corollary 3. *This result holds for* $n \geq 2$ *fixed polynomials* f_i *of degree* α*, with corresponding randomly distributed polynomials* r_i *of uniform degree at least* $\beta \geq \alpha$ *for all* i*:* $\sum_{i=1}^{n} f_i * r_i$ *is uniformly distributed over the polynomials of degree* $\alpha + \beta$*.*

B Shuffling Protocols

Either this protocol or a standard mix-net may be used where tuple shuffling is required by the protocols presented in this paper. The shuffle protocol with polynomial factoring, given in Fig. 11, allows the players to learn all elements in the inputs of all players (and how many times those elements appear) without any dishonest players being able to discern the origin of any element that did not originate in their input. Each player creates a polynomial with roots representing their private input elements. The players then calculate the encrypted product of these polynomials. Note that if an element appears as a root of one player's polynomial, it appears as a root of the product polynomial; it also appears as many times as a root in the product polynomial as it appeared in the players' polynomials. Thus, when the polynomial is decrypted, each player can factor it and obtain all linear factors, and thus the private inputs. As polynomial multiplication is a commutative operation, no dishonest players can distinguish the origin of any elements other than their own.

C Proofs of Correctness and Security for Set-Intersection Protocols

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 Honest-But-Curious Case

Theorem 4. *In the set-intersection protocols of Fig. 1 and 8, every player learns the intersection of all players' private inputs, $I = S_1 \cap S_2 \cap \cdots \cap S_n$, 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 5. *In the set-intersection protocol of Fig. 1, any honest-but-curious player learns no more information than would be gained by using the same private input in an ideal setting.*

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.2 Malicious Case

Theorem 6. *In the 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*

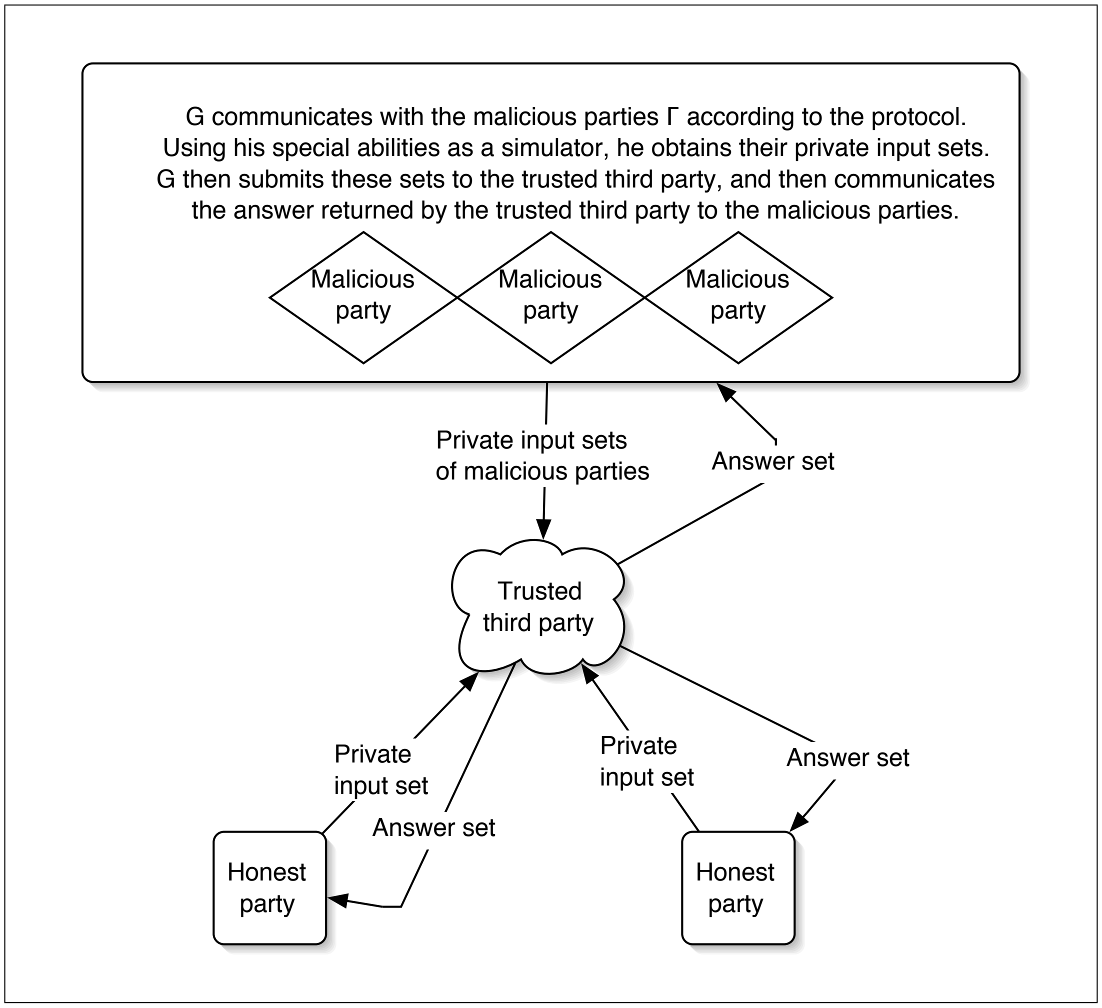
G communicates with the malicious parties Γ according to the protocol.
Using his special abilities as a simulator, he obtains their private input sets.
G then submits these sets to the trusted third party, and then communicates
the answer returned by the trusted third party to the malicious parties.
Malicious party
Malicious party
Malicious party
Private input sets of malicious parties
Answer set
Trusted third party
Private input set
Answer set
Private input set
Answer set
Honest party
Honest party

Figure 12: 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.

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 12

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 6.1)
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
      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.

□

D Proofs of Correctness and Security for Threshold Set-Intersection Protocols

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

D.1 Honest-But-Curious Case

Theorem 7. *In the threshold set-intersection protocol of Fig. 4 (semi-perfect variant), every player learns each element a which appears at least t times in the n players' private inputs.*

Proof. As shown in Theorem 1, if an element a appears at least t times in the players' private inputs, it is a root of both the polynomial $p^{(t-1)}$, as $(x-a)^t \mid p$, and p. If an element is a root of p but does not appear at least t times, it is not a root of $p^{(t-1)}$. Thus, if an element a is a root of both p and $p^{(t-1)}$, it is in the threshold intersection set. The polynomial evaluated to create conditional disclosures is $\Phi = p * \left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)} \left(\sum_{i=1}^{c+1} s_i\right)$. As not all players in $[c+1]$ can be members of the dishonest coalition, the polynomials $\sum_{i=1}^{c+1} r_i$ and $\sum_{i=1}^{c+1} s_i$ are both uniformly distributed and unknown to any coalition of players. Thus, by Theorem 2, $\Phi = \left(\prod_{a\in I}(x-a)\right) s$, where I is the threshold intersection set and s is a polynomial uniformly distributed over those of the appropriate size. As s has only a polynomial number of uniformly distributed roots, and each root has negligible probability of representing an element from the valid set P, with overwhelming probability, the only roots representing elements of P are those in the threshold set. Thus, if the element $(S_i)_j$ appears at least t times in the private input sets of the players, the conditional disclosure $U_{ij} = E_{pk}((S_i)_j)$, and if it does not, U_{ij} is the encryption of a uniformly distributed element in R.

At this point in the protocol, all players engage in shuffling of inputs. As explained in Section 5.4, the players obtain a decryption of each element in the threshold set exactly once. All players therefore learn the answer set. □

Theorem 8. *In the threshold set-intersection protocol of Fig. 4 (semi-perfect variant), any honest-but-curious player learns no more information than would be gained by using the same private input in an ideal setting, assuming that $t \geq 2$ and that the cheating coalition does not include any single element more than $t-1$ times in their private inputs, as per the definition of security.*

Proof. All polynomials representing private input are encrypted. We will assume that the cryptosystem we use is semantically secure, so no information is revealed about the players' private inputs when calculating $\Phi = p\left(\sum_{i=1}^{c+1} r_i\right) + F * p^{(t-1)}\left(\sum_{i=1}^{c+1} s_i\right)$.

First we observe that, by Theorem 2, $\Phi = \gcd\left(p, p^{(t-1)}\right) * s$ where s is a random polynomial. (Note that F is chosen to not share factors with p, and so is irrelevant to determining factors shared between p and $p^{(t-1)}$.) By Theorem 1, the only factors shared between p and $p^{(t-1)}$ are those in the threshold set.

Each player constructs tag/disclosure pairs T, U from all of his inputs. At this point in the protocol, all players engage in shuffling of inputs (either through a separate protocol or trusted third party). Thus all players receive all tag/disclosure pairs T, U, without any indication as to the origin. As the cryptosystem used to encrypt the tags is key-private, no player can gain information about which honest player created any tag. As the disclosures are all encrypted under the same key, no information can be gleaned about their origin.

Let $a = (S_i)_j$ for some i, j be the representation of the private input element used to create the encrypted element U. If $\Phi(a) = 0$, then, with overwhelming probability, a is a representation of an element in the threshold intersection set. Then U is an encryption of a. If a is not in the threshold intersection set, then, with overwhelming probability, $\Phi(a) \neq 0$, and U is thus an encryption of a uniformly distributed element, which does not reveal any information.

When the players consider each T, U pair, they either correctly decrypt it, or they do not. If they do not correctly decrypt it, the decryption shares do not reveal information about the contents of the encryption U. If they do correctly decrypt it, the tag indicates that the element a used to create the ciphertext U has not been determined to be in the threshold intersection set. If the decryption of U is not a member of the valid set P, then the player who created it learns that his input element is not in the intersection set; this is information he can gain directly from the answer set. If the decryption of U is a representation of an element a from P, then all players learn that a is in the answer set. Thus, no player learns any information except the answer set.

Note that if Alice and Bob both have some element a in their private inputs, each may learn that at least one other player holds the same element. However, as $t \geq 2$, if they have fewer than t copies of that element in their private input, they could determine that directly from their private input and the answer set. If they have at least t copies of that element in their private input, they may also learn that there exists some other player(s) holding that element, which they cannot learn directly from their input and the answer set. We do not consider this leak of information to be a problem in most circumstances, and the situation is precluded by the definition of security. □

E Proofs of Correctness and Security for Over-Threshold Set-Intersection Protocols

E.1 Honest-But-Curious Case (With Polynomial Factoring)

Theorem 9. *In the over-threshold set-intersection protocol of Fig. 2 (with polynomial factoring), every honest-but-curious player learns each element a which appears at least t times in the n players' private inputs, as well as the number of times it so appears.*

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 the valid set P, when Φ is factored, $\gcd\left(p^{(t-1)}, p\right)$ can be recovered.

By Theorem 1, $\gcd\left(p^{(t-1)}, p\right)$ has roots which are exactly the threshold set (see the proof of Theorem 7). By applying Theorem 1 we may also observe that if $(x-a)^v \mid p$, then $(x-a)^{(v-t+1)} \mid p^{(t-1)}$. Thus each linear factor repeated $v \geq t$ times in p (representing an element repeated v times in the private inputs) is repeated $v-t+1$ times in $p^{(t-1)}$, and thus in $\gcd\left(p^{(t-1)}, p\right)$. When the players factor Φ, they recover these linear factors and can thus learn both the threshold intersection set, and how many times each element in the threshold intersection set appeared in the original private inputs. □

Theorem 10. *In the over-threshold set-intersection protocol of Fig. 2 (with polynomial factoring), any honest-but-curious player learns no more information than would be gained by using the same private input in an ideal setting.*

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 9, 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. □

E.2 Honest-But-Curious Case (Without Polynomial Factoring)

Theorem 11. *In the over-threshold set-intersection protocol of Fig. 3 (without polynomial factoring), every honest-but-curious player learns each element a which appears at least t times in the n players' private inputs, as well as the number of times it so appears.*

Proof. The proof of correctness of this protocol closely follows that of the threshold set-intersection protocol given in Theorem 7. We simply observe that all the encrypted elements U are decrypted. 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 in the decryptions of the encrypted elements U as many times as it appeared in the private inputs. □

Theorem 12. *In the over-threshold set-intersection protocol of Fig. 3 (without polynomial factoring), any honest-but-curious player learns no more information than would be gained by using the same private input in an ideal setting.*

Proof. We will assume that the cryptosystem employed is semantically secure and shuffling is secure, and no player gains information except for the shuffled decrypted elements $(V_i)_j$, for unknown i and j. (If the element was constructed by a player, that player may recognize it. This does not impact out proof of security, and so we do not consider it.) As observed in Theorem 11, each such element $(V_i)_j$ is either a representation of an element from a private input or a uniformly distributed element. The uniformly distributed elements do not reveal information, and the information revealed by the private inputs revealed is that they are part of the threshold set, and how many times they appeared in private inputs. This is precisely the function output, and so no extra information is gained by any party or coalition of dishonest parties. □

E.3 Malicious Case (With Polynomial Factoring)

Theorem 13. *In the over-threshold set-intersection protocol for the malicious case in Fig. 10, 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 over-threshold set. A graphical representation of these players is given in Figure 12

Note that by definition, $|\Gamma| \leq c$, and so at least one player in the set $[c+1]$ is honest.

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 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 uniformly chosen from all polynomials of degree k (for randomly generated polynomials, this is true with overwhelming probability)

(b) if $i \in [c+1]$, G chooses arbitrary polynomials r_i, s_i and creates ciphertexts $\Lambda(r_i)$ and $\Lambda(s_i)$ from them (see Section 6.1)

2. G performs steps $2, 3, 4$ of the protocol with the dishonest players Γ
   (a) Players $i \in [c+1]$ send commitments to $\Lambda(r_i)$ and $\Lambda(s_i)$ to all other players
   (b) Players calculate the encryption of the polynomial $p = \prod_{i=1}^{n} f_i$ (note that for all i, a proof of knowledge of the polynomial f_i is given)
3. The player G extracts from the proofs of plaintext knowledge and commitments $\Lambda(r_i)$ and $\Lambda(s_i)$ (in the case of Paillier, the extraction is from the proof of knowledge of the discrete logarithm), the polynomials representing each player's private input f_α, and the random polynomials r_i, s_i 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, the set of the coalition player α) by factoring 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.
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 over-threshold multiset I to G and the honest players.
6. G prepares to reveal the over-threshold set to the coalition players Γ:
   (a) selects a target polynomial $q = \left(\prod_{a \in I}(x-a)\right) * s$, 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) Note that $\gcd\left(p, p^{(t-1)}\right) \mid \prod_{a \in I}(x-a)$. As the polynomials f_i chosen by G for the simulated honest players are generated randomly, with overwhelming probability they do not share any factors with those chosen by the coalition players Γ or F. By Theorem 1, the only factors to 'survive' the derivatives appear at least t times in p; as random polynomials of the simulated honest players do not share factors with the coalition players (with overwhelming probability), they must have been included at least t times by players in Γ. (If a factor does not include a root, it will not add a member to the intersection set, but will appear in the final polynomial. This is not a contradiction, as it does not include any information about any player's set, by definition.)
   (c) chooses a set of polynomials r_i, s_i (where i is one of the simulated honest players in $[c+1]$) such that $F * p^{(t-1)} * \left(\sum_{i=1}^{c+1} r_i\right) + p * \left(\sum_{i=1}^{c+1} s_i\right) = q$ (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 coalition players Γ as written, except that he opens the trapdoor commitment to reveal an appropriate $\Lambda(r_i)$ and $\Lambda(s_i)$ for the new chosen r_i and s_i polynomials. In this way, the players calculate an encryption of the polynomial q chosen by G, and then decrypt it. The coalition players thus learn the over-threshold 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. □

F Proof of Security for Cardinality Set-Intersection

In this section, we give proofs of security for our protocol for cardinality set-intersection in the malicious case. For simplicity, we give proof sketches for this theorem.

Theorem 14. *In the cardinality set-intersection protocol for the malicious case in Fig. 9, 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 size of the intersection set. A graphical representation of these players is given in Figure 12

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):

1. G chooses a common "private input" set S to be used for each simulated honest player (the k elements of this set can be chosen uniformly)
2. For each simulated honest player i, G chooses arbitrary polynomials $r_{i,1}, \dots, r_{i,n}$ and creates data items $\Lambda(r_{i,j})$ from them (see Section 6.1)
3. Performs step 1 of the protocol:
   (a) sends the encryptions of the elements of the "private input set" $y_{i,j}$ $(1 \leq j \leq k)$ to all malicious players Γ, along with proofs of plaintext knowledge
   (b) sends the encryption δ_i of the polynomial f_i constructed from the "private input set", along with a proof of correct construction
   (c) sends commitments to data items $\Lambda(r_{i,j})$ to all malicious players Γ
   (d) Receives from each malicious player $\alpha \in \Gamma$:
      i. encryption of a polynomial f_α and proofs of plaintext knowledge for its coefficients
      ii. trapdoor commitments to data items $\Lambda(r_{\alpha,j})$ for each random polynomial $r_{\alpha,j}$, $1 \leq j \leq n$
4. The player G (who knows the trapdoor commitment information) 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.
5. G obtains the roots of each polynomial f_α (as these exactly determine, for the purposes of the protocol, his set S_α), from the proof of correct construction of f_α.
6. 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 size of the intersection set $|I|$ to G and the honest players.
7. G prepares to reveal the size of the intersection set to the malicious players Γ:

   (a) construct the set A from any $|I|$ elements from the set $\bigcap_{\alpha \in \Gamma} S_\alpha$ and any $|I|$ elements from the common 'private input set' S of the n' simulated honest players (note that $\left|\bigcap_{\alpha \in \Gamma} S_\alpha\right| \geq |I|$, by the definition of set-intersection.)
   (b) selects a target polynomial $p = \left(\prod_{a \in A}(x-a)\right) * s$, where s is chosen uniformly from those polynomials of degree $2k - 2|I|$. (note that, by Theorem 2, this is exactly the polynomial calculated by simply running the protocol, and thus when the players evaluate it at their inputs, exactly $n|I|$ of the evaluations will be 0)
   (c) 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).

8. 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, evaluate it at their inputs (which are proved to be the same as those used to construct the polynomials f_i), and decrypt indications as to whether each element appears in the intersection set. The coalition players thus learn the size of 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. □