# Thesis Proposal:
# Privacy Preserving Distributed Information Sharing

Lea Kissner
leak@cs.cmu.edu

July 5, 2005

## 1 Introduction

In many important applications, a collection of mutually distrustful parties must share information, without compromising their privacy. In order to protect this private data, the players perform *privacy-preserving* computation; that is, no party learns more information about other parties' private inputs than what can be deduced from the result. Currently, these applications are often performed by using some form of a trusted third party (TTP); this TTP receives all players' inputs, computes the desired function, and returns the result. However, the level of trust that must be placed in such a TTP is often inadvisable, undesirable, or even illegal. For example, in many countries there are complex and restrictive laws governing the use of personal medical data. In addition, the high level of trust placed in a TTP makes it an appealing target for attack by malicious parties. In order to make many applications practical and secure, we must remove the TTP, replacing it with efficient protocols for privacy-preserving distributed information sharing. My thesis will answer the question of whether it is possible for mutually distrustful parties to efficiently share information about large, distributed bodies of data, without compromising their privacy.

Large bodies of data are represented as sets or multisets, including many databases. As an example of privacy-preserving information sharing, we propose efficient techniques for privacy-preserving operations on multisets [28]. By building a framework of multiset operations, employing the mathematical properties of polynomials, we design efficient, secure, and composable methods to enable privacy-preserving computation of the *union, intersection,* and *element reduction* operations. (Each element $a$ that appears $b > 0$ times in $S$ appears only $b - 1$ times in the element reduction of $S$.) We apply these techniques to a wide range of practical problems, including the Set-Intersection, Over-Threshold Set-Union, Cardinality Set-Intersection, and Threshold Set-Union problems. Additionally, we address the problem of determining Subset relations, and even use our techniques to evaluate CNF boolean formulas.

The Over-Threshold Set-Union problem on sets (also known as the *hot item* identification problem) is one with particularly important applications. These applications include privacy-preserving distributed network monitoring (in which nodes attempt to identify widespread network attacks), management of content delivery networks (in which commonly-requested items are cached with higher priority than rarely-requested items), and various statistics-gathering applications (such as identifying common computer configurations to aid in troubleshooting). However, in many of these applications, protocols secure in the classical cryptographic sense are not sufficiently efficient. In addition, these applications are performed by a loosely organized group of players, and thus must be extremely flexible - nodes can join and leave at any time. In order to achieve more efficient and flexible results than our Over-Threshold Set-Union protocol, we define two new privacy properties: *owner privacy* and *data privacy.* Protocols that achieve these properties protect the privacy of each players' personal input set, as well as protecting information about the players' collective inputs. By designing our hot-item identification and publication protocols to achieve owner and data privacy, we are able to significantly increase efficiency over our privacy-preserving set operations, while still protecting the privacy of participants in important areas [29].

We intend to extend this work on privacy-preserving distributed information sharing in at least one of the following directions: extension of scope and increase of efficiency. The scope of our techniques for privacy-preserving multiset operations and hot item identification is limited to sets and multisets. Some data is not easily represented by a multiset; for example, probabilistic modeling data structures, such as Bayes nets and junction trees, are represented as labeled graphs. There is little previous work on privacy-preserving computation on these important structures, making this increase in scope a rich area for exploration. Secondly, we wish to design efficient protocols, as well as increasing the efficiency of existing protocols. Efficient protocols could include special-purpose protocols with new definitions of security or privacy. More generally, we intend to achieve more efficient protocols secure against malicious players by improving the tools used to construct such protocols. In particular, we have designed a framework in which more-efficient cryptographic tools, such as normal commitment schemes, may be securely substituted for less-efficient (but closely related) tools, such as equivocal or chameleon commitment schemes. This elegant and novel approach to increasing the efficiency of protocols secure against malicious players will both ease the process of developing new secure protocols and greatly increase the efficiency of previously-designed protocols.

To summarize, the contributions of this thesis will be as follows:

- Efficient, privacy-preserving, cryptographically-secure techniques for operations on multisets: intersection, union, and element reduction.
- Application of these techniques to a wide range of practical privacy-preserving distributed information sharing problems, including Set-Intersection, Cardinality Set-Intersection, Over-Threshold Set-Union, Threshold Set-Union, Subset, and evaluation of CNF boolean formulas.
- Extremely efficient protocols for hot-item identification and publication, utilizing new definitions of privacy, tailored for the application.
- A substitution framework, allowing more-efficient cryptographic tools to be substituted for less-efficient cryptographic tools in protocols secure against malicious players. This secure framework will increase the efficiency of our privacy-preserving distributed information sharing protocols, as well as the efficiency of many other protocols.

## 1.1 Impact and Applications

Efficient and secure privacy-preserving distributed information sharing has the potential to make a great impact in many applications. A weak form of the problem is being addressed by IBM [33], indicating that many organizations see a need for accommodating desires for privacy while taking advantage of distributed data. This demand is a natural result of privacy protection enforced by custom, law, and the marketplace. Many countries, including Canada and many European nations, have strict privacy laws that concern the acceptable dissemination of personal information. A larger number of countries have laws protecting certain aspects of personal information, such as the HIPAA laws in the United States [38]. These protections point to many natural applications of privacy-preserving distributed information sharing; our work can be used to cryptographically assure that the desired level of privacy has been enforced.

Currently, it is difficult to ascertain the level of privacy that data is given. In the absence of generally-known, efficient, secure protocols, applications have been implemented with use of a trusted third party (TTP) or foregone altogether. It is difficult to ensure that a TTP remains trustworthy, especially in the face of fiscal pressures. For example, a medical transcriber threatened to post medical records on the Internet, if she was not given more money [30]. While this is a dramatic violation of trust, there are many smaller betrayals every day, often unwitting. By reducing the need for direct human interaction with protected data, we may reduce the ability to compromise data. In addition, a TTP is an attractive target for attacks, including social engineering, network attacks, and leaks of data, because it holds so much data. By eliminating the need for a TTP, we eliminate some of the impetus for these attacks.

As data takes on many forms, there are correspondingly many possible applications for privacy-preserving distributed information sharing. In this section, we discuss only a very small number of potential applications. The first category of applications we discuss are related to statistics-gathering. If a social services organization needs to determine the number of people on welfare who have cancer, the union of each hospital's lists of cancer patients must be calculated (but not revealed), then an intersection operation between the unrevealed list of cancer patients and the welfare rolls must be performed, then the number of people in the resulting set must be determined. Many statistics-gathering applications fall into this sort of pattern. Another example is privacy-preserving distributed network monitoring. In this scenario, each node monitors anomalous local traffic, and a distributed group of nodes collectively identify popular anomalous behaviors: behaviors that are identified by at least a threshold $t$ number of monitors. Many other statistics-gathering applications fall into this pattern of hot item identification, such as distributed computer troubleshooting [20, 25, 50], and distributed cache management [7].

Another category of applications fall into the category of double-checking information, to ensure that certain security rules are observed. For example, to determine which airline passengers appear on a 'do-not-fly' list, the airline must perform a private set-intersection operation between its private passenger list and the government's list. An application with a more complex privacy restriction is in pharmacies' management of prescriptions. Pharmacies must ensure that they uncover any individuals who attempt to cheat by filling a prescription more than once; this requires sharing extremely sensitive medical and business data between organizations. To detect such cheaters, the pharmacies must determine (without revealing) which prescriptions appear more than once, and revealing that information only to the particular pharmacies that were cheated on a particular perscription. In this way, medical privacy rules can be followed, while enforcing medical security regulations.

We only give a short list of the possible applications of privacy-preserving distributed information sharing. As this work places guaranteed privacy within the practical reach of more individuals and organizations, we expect it to make a significant impact in every area of shared data. Not only does our work add a level of privacy to many interactions that already take place, but we expect it to make many more applications possible, by allowing participants to easily satisfy regulatory and cultural burdens.

## 2 Related Work

There has been a growing amount of research in the area of privacy-preserving distributed information sharing. Closely related to our work in [28] is work by Freedman, Nissim, and Pinkas on privacy-preserving set intersection [18]. Secure computation of the $k$th ranked element is another example of privacy-preserving distributed information sharing [1]. Another interesting area is in privacy-preserving data mining and classification [32, 41, 49], which can be viewed as an instance of privacy-preserving distributed information sharing. We do not enumerate the entire body of work on specialized protocols for cryptographically secure privacy-preserving computation, as much of it is only tangentially related to the topic of this thesis. All such results are improvements in efficiency on general privacy-preserving circuit evaluation. General two-party computation was introduced by Yao [51], and general computation for multiple parties was introduced in [4]. In general multiparty computation, the players share the values of each input, and cooperatively evaluate the circuit. For each multiplication gate, the players must cooperate to securely multiply their inputs and re-share the result, requiring $O(n)$ communication for honest-but-curious players and $O(n^2)$ communication for malicious players [23].

As many applications of privacy-preserving distributed information sharing demand extremely efficient solutions, several protocols have been designed with looser definitions of security than the accepted definitions of cryptographic security [23]. These protocols include one for distributed network monitoring [31]; this work did not, however, give a concrete definition of security, and require trusted central servers. Additionally, in many cases, significant breaches in privacy occur when outlier elements, that appear in very few other players' servers, are revealed; they do not assuage such concerns. Privacy-preserving collection of statistics about computer configurations has also been considered in previous work [25, 50]. Like the work in [31], they do not give a concrete definition of security, but instead a technique for heuristically confusing attackers. Their approach also relies on chains of trust between friends, unlike our approach, in which nodes may be arbitrarily malicious.

**Privacy-Preserving Multiset Operations.** For most of the privacy-preserving set function problems we address (except for the Set-Intersection problem), the best previously known results are through general multiparty computation. General two-party computation was introduced by Yao [51], and general computation for multiple parties was introduced in [4]. In general multiparty computation, the players share the values of each input, and cooperatively evaluate the circuit. For each multiplication gate, the players must cooperate to securely multiply their inputs and re-share the result, requiring $O(n)$ communication for honest-but-curious players and $O(n^2)$ communication for malicious players [23]. Recent results that allow non-interactive private multiplication of shares [13] do not extend to our adversary model, in which any $c < n$ players may collude. Our results are more efficient than the general MPC approach.

The most relevant work is by Freedman, Nissim, and Pinkas (FNP) [18]. They proposed protocols for the problems related to Set-Intersection, based on the representation of sets as roots of a polynomial [18]. Their work does not utilize properties of polynomials beyond evaluation at given points. We explore the power of polynomial representation of multisets, using operations on polynomials to obtain composable privacy-preserving multiset operations.

**Hot-Item Identification and Publication.** In a non-distributed context, [3, 22, 27] examine the identification of elements that appear often in a data stream, through the use of approximate counting.

Several applications of privacy-preserving hot item identification and publishing have been considered in previous work. Certain privacy-breaching attacks against distributed network monitoring nodes were described in [31]. They did not, however, give a concrete definition of security, and their techniques require central trusted servers. Additionally, in many cases, significant breaches in privacy occur when outlier elements, that appear in very few other players' servers, are revealed; they do not assuage such concerns.

Privacy-preserving collection of statistics about computer configurations has also been considered in previous work [25, 50]. Like the work in [31], they do not give a concrete definition of security, but instead a technique for heuristically confusing attackers. Their approach also relies on chains of trust between friends, unlike our approach, in which nodes may be arbitrarily malicious.

# 3 Preliminaries

In this section, we describe the cryptographic tools and models used in this thesis. For clarity, we first list the common tools and models, used in the work throughout this thesis, before proceeding to specialized tools for each particular application.

## 3.1 Common Tools and Models

In this section we describe cryptographic tools and adversary models used in the work throughout this thesis.

### 3.1.1 Cryptographic Tools

**Cryptographic Hash Function.** A function $h : \{0,1\}^* \to \{0,1\}^\kappa$ that is both preimage-resistant and collision-resistant [34].

- *Preimage resistant.* Given $h(\cdot)$ and $y \in \{0,1\}^\kappa$, it is infeasible for a probabilistic polynomial-time adversary to find an $x \in \{0,1\}^*$ such that $h(x) = y$
- *Collision-resistant.* Given $h(\cdot)$, it is infeasible for a probabilistic polynomial-time adversary to find $x_1, x_2 \in \{0,1\}^*$ such that $h(x_1) = h(x_2)$

In our work on identifying hot items, we also use the following hash functions, based on the common cryptographic hash function $h$: $h' : \{0,1\}^* \to [b]$ can be constructed from $h$, and a family of hash functions defined as $h'_x(m) := h'(x \mid\mid m)$ for $x \in \{0,1\}^{\kappa'}$.

**Commitment Scheme.** A commitment $y$ is probabilistically constructed from a value $v$. The party who constructed the commitment can later prove that $y$ was constructed from $v$, by *opening* the commitment. More specifically, we require that it be computationally binding and hiding [34].

- *Computationally binding.* No probabilistic polynomial time adversary can construct, with non-negligable probability, one commitment $y$ that can be opened to both $v$ and $v'$, for any $v \neq v'$ in the domain of the commitment scheme.
- *Computationally hiding.* The distribution of the commitments to each value $v$ (in the domain of the commitment scheme) are computationally indistinguishable.

### 3.1.2 Adversary Models

In this thesis, we consider two standard adversary models: honest-but-curious adversaries and malicious adversaries. We provide intuition and informal definitions of these models; formal definitions of these models can be found in [23].

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

**Malicious Adversaries.** In this model, an adversary may behave arbitrarily. In particular, we cannot hope to prevent malicious parties from refusing to participate in the protocol, choosing arbitrary values for its private input set, or aborting the protocol prematurely. Instead, we focus on the standard cryptographic security definition (see, e.g., [23]) which captures the correctness and the privacy issues of the protocol. Informally, the security definition is based on a comparison between the ideal model and a TTP, where a malicious party may give arbitrary input to the TTP. The security definition is also limited to the case where at least one of the parties is honest. Let $\Gamma$ be the set of colluding malicious parties; for any strategy $\Gamma$ can follow in the real protocol, there is a translated strategy that it could follow in the ideal model, such that, to $\Gamma$, the real execution is computationally indistinguishable from execution in the ideal model.

## 3.2 Tools For Privacy-Preserving Multiset Operations

In this section, we describe several cryptographic and probabilistic tools that we use in constructing our privacy-preserving multiset operations.

**Additively Homomorphic Cryptosystem.** We utilize a semantically secure [24], additively homomorphic public-key cryptosystem. Let $E_{pk}(\cdot)$ denote the encryption function with public key $pk$. The cryptosystem supports the following operations, which can be performed without knowledge of the private key: (1) Given the encryptions of $a$ and $b$, $E_{pk}(a)$ and $E_{pk}(b)$, we can efficiently compute the encryption of $a+b$, denoted $E_{pk}(a+b) := E_{pk}(a) +_h E_{pk}(b)$; (2) Given a constant $c$ and the encryption of $a$, $E_{pk}(a)$, we can efficiently compute the encryption of $ca$, denoted $E_{pk}(c \cdot a) := c \times_h E_{pk}(a)$. When such operations are performed, we require that the resulting ciphertexts be re-randomized for security. In re-randomization, a ciphertext is transformed so as to form an encryption of the same plaintext, under a different random string than the one originally used. We also require that the homomorphic public-key cryptosystem support secure $(n, n)$-threshold decryption, i.e., the corresponding private key is shared by a group of $n$ players, and decryption must be performed by *all* players acting together.

In our protocols for the malicious case, we require: (1) the decryption protocol be secure against malicious players, typically, this is done by requiring each player to prove in zero-knowledge that he has followed the threshold decryption protocol correctly [21]; (2) efficient construction of zero-knowledge proofs of plaintext knowledge; (3) optionally, efficient construction of certain zero-knowledge proofs, as detailed in [28].

Note that Paillier's cryptosystem [39] satisfies each of our requirements: it is additively homomorphic, supports ciphertexts re-randomization and threshold decryption (secure in the malicious case) [16, 17], and allows certain efficient zero-knowledge proofs (standard constructions from [5, 10], and proof of plaintext knowledge [12]).

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

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

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

Both variants of the Shuffle protocol can be easily accomplished with standard techniques [9, 14, 19, 26, 37], with communication complexity at most $O(n^2k)$.

## 3.3 Tools for Hot Item Identification and Publication

In this section, we describe several cryptographic and probabilistic tools that we use in constructing our hot item identification and publication protocols. We describe several additional tools in more detail in Section 5.2.2.

**Group Signatures.** A group signature scheme allows each member of group of players to sign messages on behalf of the group. Given these signatures, no player or coalition of players (except the trusted *group manager*) can distinguish the player that produced any signature, nor can they determine if two signatures were produced by the same group member [2].

**Merkle Hash Trees.** A Merkle hash tree is a data structure allowing a player to produce a constant-sized commitment to an ordered set $S$. Later any player holding $S$ may produce a verifiable proof (given the committment) that an element $a \in S$ [35].

**Anonymous Routing.** An anonymous routing scheme allows any player to send a message to any other player, without revealing the source to any intermediate routing node. Note that simple and lightweight schemes can be used, as we do not require return messages or other features. Examples of such networks include [8, 9, 11, 36, 40, 44, 45]. In particular, the scheme of [36] is well-suited to our needs, as there is no need for additional infrastructure. Their anonymous routing algorithm can be described as follows: on receiving a message to be anonymously delivered, with probability $p_f$, forward the message to a random neighbor; with probability $1 - p_f$ deliver the message to the intended destination.

# 4 Privacy-Preserving Set Operations

## 4.1 Techniques and Mathematical Intuition

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

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

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

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

### 4.1.1 Background: Polynomial Rings and Polynomial Representation of Sets

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

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

Note that previous work has proposed to use polynomials to represent sets [18] (as opposed to multisets). However, to the best of our knowledge, previous work has only utilized the technique of polynomial evaluation for privacy-preserving operations. As a result, previous work is limited to set intersection and cannot be composed with other set operators. In this section, we propose a framework to perform various set operations using polynomial representations and construct efficient privacy-preserving set operations using the mathematical properties of polynomials. By utilizing polynomial representations as the intermediate form of representations of sets, our framework allows arbitrary composition of set operators as outlined in our grammar.

### 4.1.2 Our Techniques: Privacy-Preserving Set Operations

In this section, we construct algorithms for computing the polynomial representation of operations on sets, including union, intersection, and element reduction. We design these algorithms to be privacy-preserving in the following sense: the polynomial representation of any operation result reveals no more information than the set representation of the result. First, we introduce our algorithms for computing the polynomial representation of set operations union, intersection, and element reduction (with a trusted third party). We then extend these techniques to encrypted polynomials, allowing secure implementation of our techniques without a trusted third party. Note that the privacy-preserving set operations defined in this section may be *arbitrarily composed* [28], and constitute truly general techniques.

**Set Operations Using Polynomial Representations.** In this section, we introduce efficient techniques for set operations using polynomial representations. In particular, let $f, g$ be polynomial representations of the multisets $S, T$. We describe techniques to compute the polynomial representation of their union, intersection, and element reduction by $d$. We design our techniques so that the polynomial representation of any operation result reveals no more information than the set representation of the result. This privacy property is formally stated in Theorems 1, 3, and 5, by comparing to the ideal model.

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

$$f * g.$$

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

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

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

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

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

$$f * r + g * s$$

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

We show below that $f * r + g * s$ is a polynomial representation of $S \cap T$. In addition, we prove that, given $f * r + g * s$, one cannot learn more information about $S$ and $T$ than what can be deduced from $S \cap T$, as formally stated in Theorem 3.

First, we must prove the following lemma:

**Lemma 2.** *Let $\hat{f}, \hat{g}$ be polynomials in $R[x]$ where $R$ is a ring, $\deg(\hat{f}) = \deg(\hat{g}) = \alpha$, and $\gcd(\hat{f}, \hat{g}) = 1$. Let $r = \sum_{i=0}^{\beta} r[i]x^i$, and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \le i \le \beta}\ r[i] \leftarrow R$, $\forall_{0 \le i \le \beta}\ s[i] \leftarrow R$ (independently) and $\beta \ge \alpha$.*

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

We give a proof of Lemma 2 in [28].

By this lemma, $f * r + g * s = \gcd(f, g) * u$, where $u$ is distributed uniformly in $R^\gamma[x]$ for $\gamma = 2\deg(f) - |S \cap T|$. Note that $a$ is a root of $\gcd(f, g)$ and $(x - a)^{\ell_a} \mid \gcd(f, g)$ if and only if $a$ appears $\ell_a$ times in $S \cap T$. Moreover, because $u$ is distributed uniformly in $R^\gamma[x]$, with overwhelming probability the roots of $u$ do not represent any element from $P$ (as explained in the beginning of Section 4.1). Thus, the computed polynomial $f * r + g * s$ is a polynomial representation of $S \cap T$. Note that this technique for computing the intersection of two multisets can be extended to simultaneously compute the intersection of an arbitrary number of multisets in a similar manner. Also, given $f * r + g * s$, one cannot learn more information about $S$ and $T$ than what can be deduced from $S \cap T$, as formally stated in the following theorem:

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

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

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

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

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

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

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

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

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

Lemma 4 is a standard result [47]. By this lemma and $\gcd(F, f) = 1$, an element $a$ is a root of $\gcd(f^{(d)}, f)$ and $(x-a)^{\ell_a} \mid \gcd(f^{(d)}, f)$ if and only if $a$ appears $\ell_a$ times in $\mathrm{Rd}_d(S)$. By Lemma 2, $f^{(d)} * F * r + f * s = \gcd(f^{(d)}, f) * u$, where $u$ is distributed uniformly in $R^{\gamma}[x]$ for $\gamma = 2k - |\mathrm{Rd}_d(S)|$. Thus, with overwhelming probability, any root of $u$ does not represent any element from $P$. Therefore, $f^{(d)} * F * r + f * s$ is a polynomial representation of $\mathrm{Rd}_d(S)$, and moreover, given $f^{(d)} * F * r + f * s$, one cannot learn more information about $S$ than what can be deduced from $\mathrm{Rd}_d(S)$, as formally stated in the following theorem:

**Theorem 5.** *Let $F$ be a publicly known polynomial of degree $d$ such that $\forall_{a \in P}\ F(a) \neq 0$. Let* TTP1 *be a trusted third party which receives a private input multiset $S$, and then returns the reduction multiset $Rd_d(S)$ directly. Let* TTP2 *be another trusted third party, which receives a private input multiset $S$, and then: (1) calculates the polynomial representation $f$ of $S$; (2) chooses $r, s \leftarrow R^k[x]$; (3) computes and returns $f^{(d)} * F * r + f * s$.*

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

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

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

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

- Sum of encrypted polynomials: given the encryptions of the polynomial $f_1$ and $f_2$, we can efficiently compute the encryption of the polynomial $g := f_1 + f_2$, by calculating $E_{pk}(g[i]) := E_{pk}(f_1[i]) +_h E_{pk}(f_2[i])$ $(0 \leq i \leq \max\{\deg(f_1), \deg(f_2)\})$

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

It is easy to see that with the above operations on encrypted polynomials, we can allow the computation of the polynomial representations of set operations described in Section 4.1.2 without the trusted third party (TTP2) while enjoying the same security. We demonstrate this property with concrete examples detailed in the remainder of this section.

## 4.2 Sample Application: Private Set-Intersection

The techniques we introduce for privacy-preserving computations of multiset operations have many applications. In this section, we design a protocol for Set-Intersection, secure against a coalition of honest-but-curious adversaries.

In addition, in the full version of this work [28], we provide an efficient protocol for the Cardinality Set-Intersection problem, Over-Threshold Set-Union problem, and three variants of the Threshold Set-Union problem, secure against honest-but-curious adversaries. We also introduce tools and protocols, secure against malicious players, for the Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union problems. We propose an efficient protocol for the Subset problem. More generally, our techniques allow private computation of functions based on composition of the union, intersection, and element reduction operators. We discuss techniques for this general private computation on multisets. Our techniques are widely applicable, even outside the realm of computation of functions over multisets. As an example, we show how to apply our techniques to private evaluation of boolean formulas in CNF form.

### 4.2.1 Set-Intersection

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

Our protocol for the honest-but-curious case is given in Fig. 1. In this protocol, each player $i$ $(1 \le i \le n)$ first calculates a polynomial representation $f_i \in R[x]$ of his input multiset $S_i$. He then encrypts this polynomial $f_i$, and sends it to $c$ other players $i+1, \ldots, i+c$. For each encrypted polynomial $E_{pk}(f_i)$, each player $i+j$ $(0 \le j \le c)$ chooses a random polynomial $r_{i+j,j} \in R^k[x]$. Note that at most $c$ players may collude, thus $\sum_{j=0}^{c} r_{i+j,j}$ is both uniformly distributed and known to no player. They then compute the encrypted polynomial $\left(\sum_{j=0}^{c} r_{i+j,j}\right) *_h E_{pk}(f_i)$. From these encrypted polynomials, the players compute the en-

**Protocol:** Set-Intersection-HBC

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

1. Each player $i = 1, \ldots, n$
   (a) calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends the encryption of the polynomial $f_i$ to players $i + 1, \ldots, i + c$
   (c) chooses $c + 1$ polynomials $r_{i,0}, \ldots, r_{i,c} \leftarrow R^k[x]$
   (d) calculates the encryption of the polynomial $\phi_i = f_{i-c} * r_{i,i-c} + \cdots + f_{i-1} * r_{i,i-1} + f_i * r_{i,0}$, utilizing the algorithms given in Sec. 4.1.2.
2. Player 1 sends the encryption of the polynomial $\lambda_1 = \phi_1$, to player 2
3. Each player $i = 2, \ldots, n$ in turn
   (a) receives the encryption of the polynomial $\lambda_{i-1}$ from player $i - 1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} + \phi_i$ by utilizing the algorithms given in Sec. 4.1.2.
   (c) sends the encryption of the polynomial $\lambda_i$ to player $i + 1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$ to all other players.
5. All players perform a group decryption to obtain the polynomial $p$.

Each player $i = 1, \ldots, n$ determines the intersection multiset as follows: for each $a \in S_i$, he calculates $b$ such that $(x - a)^b | p \ \wedge \ (x - a)^{b+1} \not| p$. The element $a$ appears $b$ times in the intersection multiset.

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

cryption of $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$. All players engage in group decryption to obtain the polynomial $p$. Thus, by Theorem 3, the players have privately computed $p$, a polynomial representing the intersection of their private input multisets. Finally, to reconstruct the multiset represented by polynomial $p$, the player $i$, for each $a \in S_i$, calculates $b$ such that $(x - a)^b | p \ \wedge \ (x - a)^{b+1} \not| p$. The element $a$ appears $b$ times in the intersection multiset.

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

**Theorem 6.** *In the Set-Intersection protocol of Fig. 1, every player learns the intersection of all players' private inputs, $S_1 \cap S_2 \cap \cdots \cap S_n$, with overwhelming probability.*

**Theorem 7.** *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, with overwhelming probability, in the Set-Intersection protocol of Fig. 1, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

We provide proof sketches for Theorems 6 and 7 in [28].

# 5 Hot Item Identification and Publication

## 5.1 Model and Problem Definitions

In this section, we define the characteristics of secure, privacy-preserving HotItem-ID and HotItem-Pub protocols. These protocols are executed in the model described in Section 5.1.2, in which honest and malicious players are connected by a network.

### 5.1.1 Problem Definition

Each player $i$ ($1 \leq i \leq n$) holds a private input set $S_i$ of $m$ elements. Let a *$k$-threshold hot item* (referred to in this thesis as a hot item) be any element $a$ that appears in at least $k$ distinct players' private input sets. Every element is a member of some domain, denoted $M$.

We define the HotItem-ID problem as follows: each player $i$ ($1 \leq i \leq n$) learns the set of hot items that appear in their private input set, denoted $P_i$. We define the HotItem-Pub problem as follows: each player learns the complete set of hot items, denoted $P = P_1 \cup \cdots \cup P_n$, from all honest players' input sets.

**Hot Item Identification.** A secure $k$-HotItem-ID protocol has the following properties:

- **Correctness.**
  - For each player $i$ ($1 \leq i \leq n$), if element $a \in S_i$ is a hot item, then at the conclusion of the HotItem-ID protocol player $i$ learns that $a \in P_i$ with probability at least $1 - \delta_-$,
  - For each player $i$ ($1 \leq i \leq n$), if element $a \in S_i$ is not a hot item, then at the conclusion of the HotItem-ID protocol player $i$ learns that $a \notin P_i$ with probability at least $1 - \delta_+$.
- **Data Privacy.** No coalition of at most $\lambda$ malicious adversaries can: (1) learn more than an expected $I$ bits of information about the honest players' private input sets, (2) distinguish an element $a$, which appears in $f_a$ players' private input sets, from an expected indistinguishable set of $\Phi(f_a)$ elements.
- **Owner Privacy.** No coalition of at most $\lambda$ malicious probabilistic polynomial-time (PPT) adversaries can gain more than a negligible advantage in associating an item $a$ with a honest player $i$ ($1 \leq i \leq n$) such that $a \in S_i$, over a situation in which all players learn the set of hot items $P$ and their frequencies.

**Hot Item Publication.** A secure $k$-HotItem-Pub protocol has the following properties:

- **Correctness.**
  - For each player $i$ ($1 \leq i \leq n$), if element $a \in S_i$ is a hot item, then at the conclusion of the HotItem-Pub protocol all players learn that $a \in P$ with probability at least $1 - \delta_-$,
  - For each player $i$ ($1 \leq i \leq n$), if element $a \in S_i$ is not a hot item, then at the conclusion of the HotItem-Pub protocol player $i$ learns that $a \notin P_i$ with probability at least $1 - \delta_+$.

- **Data Privacy.** No coalition of at most $\lambda$ PPT malicious adversaries can: (1) learn more than an expected $I$ bits of information about the honest players' private input sets, (2) distinguish a non-hot element $a$, which appears in $f_a$ players' private input sets, from an expected $\Phi(f_a)$ other elements.
- Optionally, HotItem-Pub protocols may have one of the following forms of owner privacy:
  - **Correlated Owner Privacy.** No coalition of at most $\lambda$ malicious PPT adversaries can gain more than a negligible advantage in associating an item $a$ with a honest player $i$ ($1 \le i \le n$) such that $a \in S_i$, over a situation in which all players learn the set of hot items $P$, and their frequencies.
  - **Uncorrelated Owner Privacy.** The protocol has the property of correlated owner privacy. Additionally, no coalition of at most $\lambda$ malicious PPT adversaries can gain more than a negligible advantage in associating two items $a_1, a_2$ such that $a_1, a_2 \in S_i$ for some honest player $i$ ($1 \le i \le n$), over a situation in which all players learn the set of hot items $P$, and their frequencies.

### 5.1.2 Model

**Communication Model** We require the players in our protocols to be connected via a network represented as a graph. Each player is represented as a node, with edges over which messages can be sent to other nodes. The *neighbors* of a player $V$ are those players $W$ such that $(V, W)$ is an edge in the network graph. As, in our protocols, messages are broadcast from a node to all of its neighbors, it is advisable that the number of neighbors remain reasonably small. We require that there exist a routing system to allow non-adjacent nodes to exchange messages, by following multiple edges in a path between the nodes. As we place very few restrictions on the networking scheme employed, many efficient P2P network protocols may be employed [43, 46, 48].

We assume that the honest players form a connected component in the network graph. When the players communicate via an overlay network, this may be easily achieved, with high probability. For example, a random graph with degree $\psi = O(\log n)$ is connected with high probability.

**Adversary model.** In this chapter, we prove security of our protocol against standard cryptographic adversaries [23]. Most players are considered to be *honest-but-curious*, that is, they: (1) do not collude with other players to gain information and (2) follow the protocol as specified. At most $\lambda$ of the players are *malicious* and will collude and act arbitrarily to violate the data and owner privacy properties.

For example, a malicious player might attempt to inject non-hot items, claiming they are hot, to fool the other players. If the malicious players succeeded at injecting an erroneous item, they might convince organizations to filter legitimate network traffic, thinking it has been identified as an attack by the distributed network monitor. Malicious players might also attempt to forge cryptographic signatures or wrongly route messages in order to learn honest players' private data. By claiming to have many copies of an item, the malicious players could attempt to cause a non-hot item to be identified as hot. They then could learn during publication if any honest players had that item in their private input sets. Note that our protocols defend against all of these attacks, and we prove its security in the standard model.

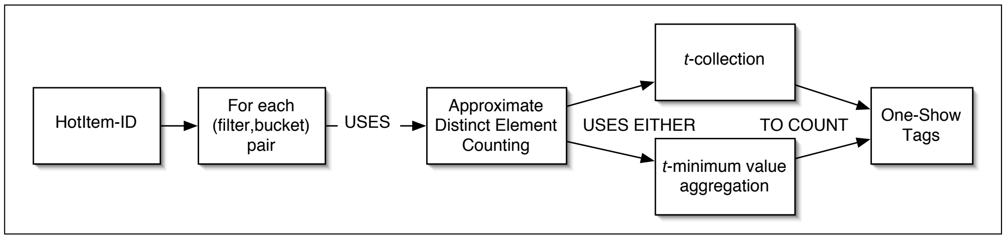
HotItem-ID
For each (filter,bucket) pair
USES
Approximate Distinct Element Counting
USES EITHER
t-collection
t-minimum value aggregation
TO COUNT
One-Show Tags

Figure 2: An outline of the building blocks of our HotItem-ID protocol.

## 5.2 Our Approach

In this section, we begin by giving an overview of our protocols, before explaining the tools we use to construct them. We then design protocols for efficient privacy-preserving distributed hot item identification (HotItem-ID) and publishing (HotItem-Pub).

### 5.2.1 Intuition and Overview

Once our protocols are complete each player has a set of global filters which identify hot items. Each global filter is an array of $b$ bits, which probabilistically indicate hot elements. Bit $b$ is set to 1 if a hot item hashs to location $b$. The players construct the global filters by combining local filters; each local filter is an array of $b$ bits which probabilistically indicate the elements of that player's private input set. If at least $k$ players set a particular bit to 1 in their local filters, then the corresponding bit is set to 1 in the global filters as well.

The global filters do not maintain an exact count of how many players may have an item, since preventing manipulation by an unknown set of malicious players is difficult and expensive. Instead, we take the more efficient approach of employing a secure approximate counting algorithm to decide whether each bit was set to 1 by at least $k$ players. An approximate counting algorithm estimates the number of distinct elements in a set. This counting algorithm is run once to construct each bit in each global filter: if a player sets the corresponding bit to 1 in their local filter it constructs a unique set element (known as a *one-show tag*). The counting algorithm then determines if the set of one-show tags contributed to by all players has at least $k$ elements; if the set does contain at least $k$ elements then the corresponding bit is set to 1 in the global filter.

By using the approximate distinct-element counting algorithm of [3], each player may accurately estimate the number of one-show tags produced by all players with only a small amount of information. This is done by the $t$-minimum aggregation protocol (see Section 5.2.2). However, the players do not need an exact count: to save bandwith they can simply determine if the set size is at least $k$. This task may be achieved with a $t$-collection protocol (see Section 5.2.2).

To ensure that malicious players cannot manipulate the count of one-show tags, each player must only be able to contribute a single value to each count. We design anonymous one-show tags (Section 5.2.2) such that each player can produce only one valid tag per specific bit in the filters, and so they cannot be associated with the player that constructed them. We thus will approximately count the number of valid one-show tags for each specific bit in the filters. Due to the one-show property of the tags, malicious players cannot cause an element to be identified as a hot item with greater probability than if they simply held that item in their private input sets.

### 5.2.2 Building Blocks

We now discuss in more detail several of the building blocks that we use to construct our protocols. An illustration of how these tools are used is given in Figure 2.

**Approximate Heavy-Hitter Identification.** An approximate heavy-hitter identification algorithm produces a data structure that can be used to identify elements that appeared in a large number of distinct players' private input sets; the HotItem-ID protocol is focused on constructing and using such a structure. In this chapter, we use a filter scheme that identifies hot items, those elements that appeared in at least $k$ players' private input sets.

First, each player constructs a set of $T$ *local filters*, which approximately represent his private input set. Let $h_1, \ldots, h_T : \{0,1\}^\kappa \rightarrow [b]$ be cryptographic hash functions. Each filter $q$ ($1 \leq q \leq T$) for player $i$ ($1 \leq i \leq n$) is an array of $b$ buckets, represented by the bit array $\{w_{i,q,j}\}_{j \in [b]}$. Each bit set to 1 in a local filter indicates that at least one element in the player's private input set hashes to that bucket; we refer to such buckets as *hit*. Formally, player $i$ ($1 \leq i \leq n$) computes each bit $w_{i,q,j} = 1 \Leftrightarrow \exists_{a \in S_i} h_q(a) = j$.

The players then combine their local filters into a set of *global filters*, approximately representing the players' combined private input sets. We will represent each global filter as the bit array $\{x_{q,j}\}_{j \in [b]}$ ($1 \leq q \leq T$). If at least $k$ players marked bucket $j$ of filter $q$ as hit, then let the bit in the global filter $x_{q,j} = 1$ ($1 \leq q \leq T$, $1 \leq j \leq b$). Otherwise, let $x_{q,j} = 0$.

Note that, if $a$ is a hot item, then by definition $x_{1,h_1(a)} = 1, \ldots, x_{T,h_T(a)} = 1$. However, as the range $[b]$ of each hash function is relatively small, collisions may well occur between elements, e.g. $h_1(a) = h_1(a')$. This does not pose a problem, however, if probability of an element colliding with others in every filter $1, \ldots, T$ is relatively small. By choosing appropriate values of the number of buckets per filter, $b$, and the number of filters, $T$ [29], we may ensure that for $a \in \cup_{i=1}^n S_i$, with high probability, if $x_{1,h_1(a)} = 1, \ldots, x_{T,h_T(a)} = 1$, then $a$ is indeed a hot item.

**Anonymous One-Show Tags.** We now describe the construction of one-show tags, which players use to perform approximate counting for each (filter,bucket) pair when constructing the global filters. To prevent malicious players from constructing multiple distinct tags to manipulate the counting, we require that the tag have a *one-show value*. Each player cannot construct multiple valid tags for any (filter,bucket) pair with different one-show values. For each (bucket,filter) pair, the one-show values of different players are distinct, with overwhelming probability. Thus, when performing approximate counting, the players may estimate the number of distinct one-show values to determine whether at least $k$ players have hit a certain bucket. The validity of one-show tags for any (filter,bucket) pair may be verified by any player.

We also require that the tags preserve players' anonymity. Given a valid one-show tag, no probabilistic polynomial time adversary can distinguish the player that constructed it, with probability non-negligibly different than $\frac{1}{n}$, where $n$ is the number of honest players. Additionally, the tags must be *unlinkable*, so that no adversary can distinguish whether two tags were constructed by the same player.

In this section, we describe a one-show tag scheme obtained through the modification of the Ateniese et al. group signature scheme [2]. In this group signature scheme, the group public key is denoted $pk = (\eta, z_1, z_2, z_0, \chi, g_1, g_2)$, where $\eta = r_1 r_2$, $r_1 = 2r_1' + 1$, $r_2 = 2r_2' + 1$ (where $r_1, r_2, r_1', r_2'$ are large primes), $\chi = g_1^s \mod \eta$ (where $g_1$ is a generator of the quadratic residues $QR(\eta)$ of $Z_\eta$ and $s \leftarrow Z^*_{r_1' r_2'}$), $z_1, z_2, z_0$ are generators of $QR(\eta)$. The group private key,

$(r'_1, r'_2, s)$, can be held by a trusted group manager or distributed among multiple parties. Each user $i$ $(1 \leq i \leq n)$ has a private key $s_i$ (a bitstring of sufficient length) and a group certificate $(A_i, e_i)$ (constructed by the group manager), such that $z_1^{s_i} z_0 \equiv A_i^{e_i} \pmod{\eta}$. Using his private key and group certificate, he may sign a message by proving knowledge of values $(\beta, \phi, \zeta)$, using Fiat-Shamir's heuristics [15], such that $z_1^{\beta} z_0 \equiv \phi^{\zeta} \pmod{\eta}$. This group signature scheme is relatively efficient.

We may modify this group signature scheme to include provably correct one-show values, making each signature a one-show tag. In this scheme, each user $i$ $(1 \leq i \leq n)$ holds a secret key $sk_i = (s_i, o_i)$ (both bitstrings of sufficient length) and a group certificate $(A_i, e_i)$ (constructed by the group manager), such that $z_1^{s_i} z_2^{o_i} z_0 \equiv A_i^{e_i} \pmod{n}$. Each user $i$ $(1 \leq i \leq n)$ may construct a one-show tag for bucket $j$ of filter $q$ $(1 \leq q \leq T,\ 1 \leq j \leq b)$ by: (1) signing the message $q \mid\mid j$ as in the original signature scheme; (2) generating the *one-show value* $g_{q,j}^{o_i} \in QR(\eta)$ (see [29] for a specific construction of $g_{q,j}$) and proving in zero-knowledge that this one-show value is constructed correctly. In addition, we define this tag's *hash value* as $h(g_{q,j}^{o_i})$.

The additional zero-knowledge proof required for the one-show tag construction is efficient, and thus our one-show tag construction and verification is nearly as efficient as the original group signature scheme. Note that these one-show tags are unlinkable, anonymous, and can be verified by all players who hold the group public key. This construction is similar to the techniques for the construction of one-show credentials in [6].

We denote the algorithm for construction of a one-show tag: `OST`(group public key, private signing key, message, one-show parameter).

**t-Collection and t-Minimum Value Aggregation.** In order to construct the global filters, we must determine whether, for each bucket of each filter, at least $k$ players marked that bucket as 'hit'. In order to determine this, we may use approximate distinct element counting to count the number of one-show tags constructed for that specific bucket.

**Approximate Distinct Element Counting.** To count the number of distinct tags for each (filter,bucket) pair, we use an approximate distinct element counting algorithm [3]. For each (filter,bucket) pair, we collect the $t$ smallest hash values of the tags to be counted. Let $h(v_t)$ be the $t$th smallest hash value. The estimate of the number of distinct tags is $\frac{t2^{\kappa}}{h(v_t)}$. To increase the accuracy of this approximation, one may choose $\alpha$ independent hash functions and perform this estimation with each such function; the resulting estimate is the median of the approximation obtained with each hash function. We have found $\alpha := 1, t := 25$ sufficient in practice; see [29] for details.

***t*-Minimum Value Aggregation.** We may use a $t$-minimum value aggregation protocol, described in Figure 3, to ensure that all players efficiently gain the information necessary to approximately count the set size.The protocol is simple: each player maintains a shortlist of the $t$ tags with the smallest hash values among the tags. When a player learns new tags to be added to the shortlist, he sends those new tags to all of his neighbors. This cycle repeats until it converges, which is at most $D$ times, where $D$ is the diameter of the network. This protocol is based on that of [42].

***t*-Collection.** In most situations, specific estimations of the number of players who hit any given bucket are not needed; to construct the global filters, players only need to determine

**Protocol:** $t$-Minimum Aggregation
**Input:** There are $n$ players, $\lambda$ maliciously colluding. Each $h$ is a cryptographic hash function with range $\{0,1\}^\kappa$. All players know the parameter $t$ and the diameter of the network, $D$. Each player $i$ $(1 \leq i \leq n)$ holds a set $U'_{i,0} = U_{i,0}$ of one-show tags for some specific bucket.
**Output:** Each player learns the set $U'$, those $t$ tags with the smallest hashes of their one-show values.

For $\ell = 1, \ldots, D$:

1. Each player $i$ $(1 \leq i \leq n)$ sends the set of new small tags $U_{i,\ell-1} \cap U'_{i,\ell-1}$ to all of his neighbors.
2. As a result of Step 1, each player $i$ $(1 \leq i \leq n)$ receives a set of tags – let those which are valid one-show tags for the correct bucket form the set $U_{i,\ell}$.
3. Each player $i$ $(1 \leq i \leq n)$ calculates $U'_{i,\ell}$ to be those $t$ tags with one-show values $v_1, \ldots, v_t$ such that $h(v_1) < \cdots < h(v_t)$ and $\forall_{w \in (U_{i,\ell} \cup U'_{i,\ell-1}) \cap \overline{U'_{i,\ell}}}\ h(w) > h(v_t)$.

Each player $i$ $(1 \leq i \leq n)$ outputs the set $U' = U'_{i,D}$.

Figure 3: $t$-minimum value aggregation protocol,

**Protocol:** $t$-Collection
**Input:** There are $n$ players, $\lambda$ maliciously colluding. Each $h$ is a cryptographic hash function with range $\{0,1\}^\kappa$. All players know the parameters $t, k$ and the diameter of the network, $D$. Each player $i$ $(1 \leq i \leq n)$ holds a set $U'_{i,0} = U_{i,0}$ of one-show tags for some specific bucket.
**Output:** Each player outputs `success` if they collected $t$ valid one-show tags with one show value $v$ such that $h(v) \leq \frac{t2^\kappa}{k}$; otherwise, they output `failure`.

For $\ell = 1, \ldots, D$:

1. Each player $i$ $(1 \leq i \leq n)$ sends the set of new small tags $U_{i,\ell-1} \cap U'_{i,\ell-1}$ to all of his neighbors.
2. For each player $i$ $(1 \leq i \leq n)$, if $|U_{i,\ell-1}| = t$ then player $i$ **stops participating** in the protocol and outputs `success`; he has now collected a sufficient number of small tags, and sent them to his neighbors.
3. As a result of Step 1, each player $i$ $(1 \leq i \leq n)$ receives a set of tags – let the set $U_{i,\ell}$ be those which are valid tags for the correct bucket, with one show value $v$ such that $h(v) \leq \frac{t2^\kappa}{k}$.
4. Each player $i$ $(1 \leq i \leq n)$ calculates $U'_{i,\ell} = U'_{i,\ell-1} \cup U_{i,\ell}$.

Each player $i$ $(1 \leq i \leq n)$ outputs `success` if $|U_{i,\ell}| = t$, and otherwise outputs `failure`.

Figure 4: $t$-collection protocol,

whether at least $k$ players hit a bucket. As we prove in the expanded version of this work [29], a player who collects $t$ valid one-show tags such that the hash of each one show value is at most $\frac{t2^\kappa}{k}$ may conclude that there are at least $k$ tags in the full set. We describe a $t$-collection protocol in Figure 4 that ensures all players determine if the set has at least $k$ elements. This protocol is similar to the $t$-minimum value aggregation protocol: each player maintains a set of at most $t$ valid tags which have hash values at most $\frac{t2^\kappa}{k}$. Upon learning a new tag that will be added to his set, a player sends the new tag to all of his neighbors. However, unlike the $t$-minimum value aggregation protocol, when a player has collected $t$ tags hash values at most $\frac{t2^\kappa}{k}$, and sent each of these tags to his neighbors, he ends his participation in the protocol.

**Protocol:** HotItem-ID
**Input:** There are $n$ players, $\lambda$ maliciously colluding, each with a private input set $S_i$. Each player $i \in [n]$ holds a private key $sk_i = (s_i, o_i)$ and a public key $pk$ allowing him to construct and verify one-show tags. $h_1, \dot{;} h_T$ are independently chosen cryptographic hash functions with range $[b]$. $h$ is a cryptographic hash function with range $\{0,1\}^\kappa$. $\rho, b, T, t, \alpha$ are parameters known to all participants.
**Output:** Each player $i$ $(1 \le i \le n)$ obtains the set $P_i \subseteq S_i$, such that each element $a \in P_i$ is a hot item. All players hold the approximate heavy-hitter filters $\{x_{q,j}\}_{j \in [b]}$ $(1 \le q \le T)$.

1. Each player $i$ $(1 \le i \le n)$ constructs $T$ local approximate heavy-hitter identification filters $\{w_{i,q,j}\}_{j \in [b]}$ $(1 \le q \le T)$ from their private input set $S_i$, by setting each bit corresponding to a bucket that was hit to 1: $w_{i,q,j} = 1 \Leftrightarrow \exists_{a \in S_i} h_q(a) = j$.
2. Each player $i$ $(1 \le i \le n)$, for each bucket $j$ of filter $q$ $(1 \le j \le b,\ 1 \le q \le T)$ that was hit in the construction of the multistage filter $(w_{i,q,j} = 1)$:
   (a) constructs a one-show tag $\texttt{OST}(pk, sk_i, q \mid\mid j, g_{q,j}^{o_i})$
   (b) if $h\left(g_{q,j}^{o_i}\right) \le \frac{t2^\kappa}{k}$, player $i$ anonymously routes it to $\rho$ randomly chosen players
3. For each bucket $j$ of filter $q$ $(1 \le j \le b,\ 1 \le q \le T)$ , all players $1, \ldots, n$:
   (a) perform $t$-collection, with $\alpha$ independent hash functions, on the tags received in Step 2, attempting to collect, for each hash function $h$, $t$ valid one-show tags such that each one-show value $v \le \frac{t2^\kappa}{k}$
   (b) if $t$-collection was successfully performed for the majority of hash functions, set $x_{q,j} = 1$. Otherwise, $x_{q,j} = 0$.
4. Each player calculates his output set $P_i$: for each element $a \in S_i$, if $\forall_{q \in [T]}\ x_{q,h_q(a)} = 1$, then $a \in P_i$.

Figure 5: HotItem-ID protocol, for identifying the hot items in each players' private input.

**Redundant Element Distribution.** In the HotItem-Pub protocol, all players publish the hot items from their private input sets. Hot items are, by definition, held by a large number of players; distributing $k$ or more copies of each hot element is inefficient and unnecessary. Instead, we employ a *redundant element distribution* protocol, ElementDist. Such a protocol follows from the following rule: when a player receives a previously-unseen hot item, he sends it to each of his neighbors. By following this simple protocol, duplicate publications of hot items are efficiently suppressed, while ensuring that all players receive each hot item.

### 5.2.3 Hot Item Identification Protocol

Our HotItem-ID protocol allows each player $i$ $(1 \le i \le n)$ to identify the set $P_i$ of hot items in his private input set $S_i$ (Figure 5). Ultimately, each player will use global filters (as described in Section 5.2.2) to identify their hot items; construction and use of these filters is the goal of our protocol.

First, each player $i$ $(1 \le i \le n)$ constructs a set of $T$ local filters, each with $b$ buckets, each denoted $\{w_{i,q,j}\}_{q \in [b]}$. Each filter starts out with all buckets marked as un-hit $(w_{i,q,j} = 0)$. Then, for each element $a \in S_i$, player $i$ lets $w_{i,q,h_q(a)} := 1$ for each filter $q$ $(1 \le q \le T)$, marking that bucket as hit. Note that a bucket may be marked as hit more than once; the result is the same as marking it once.

The $T$ global filters are denoted $\{x_{q,j}\}_{j \in [b]}$ for $1 \le q \le T$. Each bucket in the global filters is marked as 1 (instead of 0) if at least $k$ players hit the corresponding bucket in their local

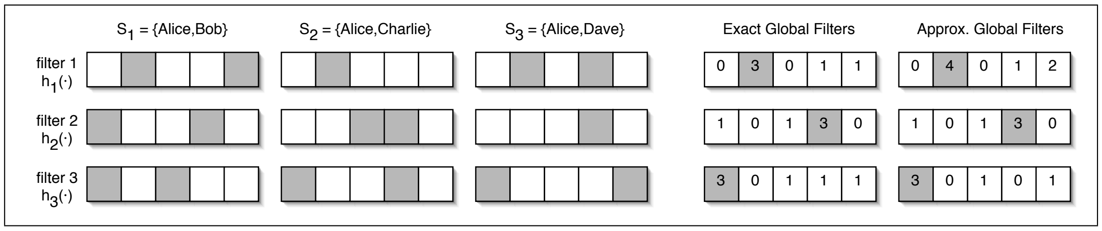

Figure 6: In the HotItem-ID protocol, each player constructs a local filter from his private input set, where $h_1(Alice) = 2, h_2(Alice) = 4, h_3(Alice) = 1, h_1(Bob) = 5, h_2(Bob) = 1, h_3(Bob) = 3, h_1(Charlie) = 2, h_2(Charlie) = 3, h_3(Charlie) = 4, h_1(Dave) = 4, h_2(Dave) = 4, h_3(Dave) = 5$. (We mark the hit buckets as grey.) The global filters are constructed by approximately counting the number of players who marked each bucket as hit; if at least $k = 3$ players have marked that bucket as hit in the local filters, then it is marked as hit in the global filter. Note that, even with small errors in counting, the final global filters are correct.

filter. For each bucket $j$ in filter $q$ ($1 \leq j \leq b$, $1 \leq q \leq T$), each player $i$ ($1 \leq i \leq n$) constructs a one-show tag with one-show value $g_{q,j}^{o_i}$ if $w_{i,q,j} = 1$.

As we prove in the expanded version of this work [29], any player who collects $t$ valid tags, where the hash of each one-show value is at most $\frac{t2^\kappa}{k}$, may conclude that there are at least $k$ distinct tags, with high probability. Thus, any one-show tag with value $v$ such that $h(v) > \frac{t2^\kappa}{k}$ is irrelevant to the estimation algorithm; any player with such a tag discards it. Each player sends any remaining one-show tags anonymously to $\rho$ random players. Beginning with the tags anonymously received, the players use the $t$-collection protocol to determine whether there exist $t$ valid tags, such that the hash of each one-show value is at most $\frac{t2^\kappa}{k}$. If $t$-collection succeeds for bucket $j$ of filter $q$ ($1 \leq j \leq b$, $1 \leq q \leq T$), then all players set $x_{q,j} = 1$; otherwise $x_{q,j} = 0$.

Each player $i$ ($1 \leq i \leq n$) determines their output set $P_i \subseteq S_i$ of hot items by testing each element against the global filters. We choose the number of filters $T$ and number of buckets per filter $b$, as detailed in citehotitem, so that, with high probability, if $\forall_{q \in [T]}\ x_{q,h_q(a')} = 1$, $a'$ is a hot item.

### 5.2.4 Hot Item Publication Protocol

When players publish hot items, we must prevent malicious players from fooling honest players into accepting non-hot items as hot. After constructing global filters to identify hot items in the HotItem-ID protocol, malicious players may find elements that pass the filters, but are not in the private input sets of any other players. Honest players must be able to distinguish between truly hot items and those erroneously injected by malicious players.

Our HotItem-Pub protocol, for securely publishing all hot items in each honest player $i$'s ($1 \leq i \leq n$) private input set $S_i$, is described in Figure 7. For the sake of clarity, we will first explain the elements of HotItem-Pub common to the non-owner-private, correlated owner-private, and uncorrelated owner-private variants, and then describe how each variant differs.

In all variants of the HotItem-Pub protocol, the same basic procedure is followed by the participants. In Step 1, each player $i$ ($1 \leq i \leq n$) commits to the elements of their private input set $S_i$, and sends this commitment to all other players. Then, using the HotItem-ID protocol, each player $i$ ($1 \leq i \leq n$) obtains: (a) his set $P_i \subseteq S_i$ of hot items, (b) a set of approximate heavy-hitter identification filters $\{x_{q,j}\}_{j \in [b]}$ ($1 \leq q \leq T$). All players $i$ ($1 \leq i \leq n$)

then use a redundant element distribution protocol, ElementDist, to efficiently distribute the elements of each set $P_i$, along with the opening of the commitment to $a \in P_i$ from Step 1. To remove erroneous non-hot elements published by malicious players, each player checks that each element's commitment opening is correct, and that the element $a$ is identified as a hot item by the approximate heavy-hitter identification filters: $\forall_{q \in [T]}\ w_{q,h_q(a)} = 1$.

**Owner Privacy.** Certain aspects of commitment and hot item publication depend on the chosen degree of owner privacy; more privacy requires more bandwidth [29]. In the commitment phase of the non-owner-private protocol variant, each player $i$ $(1 \leq i \leq n)$ sends $y_i$, the root of the Merkle tree commitment to $S_i$, to all other players. When publishing his set of hot items $P_i$, he opens each commitment by including a proof of inclusion in $S_i$.

To ensure correlated owner privacy, in the commitment phase of the protocol, each player $i$ $(1 \leq i \leq n)$ anonymously sends $y_i$, the root of the Merkle tree commitment to $S_i$, to all other players. When publishing his set of hot items $P_i$, he anonymously routes his hot elements, and the opened commitments to them, to $\rho$ randomly selected players. However, the use of a single commitment $y_i$ to all elements of the set $S_i$ may allow players to determine whether two hot items originated in the same private input set. To prevent this, we design a protocol that enforces uncorrelated owner privacy.

In the uncorrelated owner-private HotItem-Pub protocol, each player $i$ $(1 \leq i \leq n)$ computes a separate commitment for each element $a \in S_i$ and anonymously routes these commitments to all players. Like in the correlated owner-private variant, each player anonymously routes each hot item to $\rho$ players, along with the opening of the commitment to that item. As each element is committed to separately, no player can associate elements from the same private input set.

**Protocol:** HotItem-Pub
**Input:** There are $n$ players, $\lambda$ maliciously colluding, each with a private input set $S_i$. Each player $i \in [n]$ holds a private key $sk_i$, allowing him to construct one-show tags. $h_1, \dot{;} h_T$ are independently chosen cryptographic hash functions. $\rho, b, T, t$ are parameters known to all participants.
**Output:** The set $P$ of hot items published by honest players.

1. Each player $i$ $(1 \leq i \leq n)$ commits to their private input set $S_i$:
   - If the players are not concerned with Owner-Privacy: constructs a hash tree from his randomly-permuted private input set $S_i$, with root hash-value $y_i$, and sends $y_i$ to all players
   - If the players wish to ensure *Correlated Owner-Privacy*: constructs a hash tree from his randomly-permuted private input set $S_i$, with root hash-value $y_i$, and anonymously routes $y_i$ to all players
   - If the players wish to ensure *Uncorrelated Owner-Privacy*, for each element $a \in S_i$, he constructs a commitment to $a$, and anonymously routes it to every other player
2. All players $1, \ldots, n$ execute the HotItem-ID protocol (Figure 5), so that each player $i$ $(1 \leq i \leq n)$ obtains: (a) the set $P_i \subseteq S_i$ of hot items in that player's private input set and (b) approximate heavy-hitter filters $\{x_{q,j}\}_{j \in [b]}$ $(1 \leq q \leq T)$
3. All players $1, \ldots, n$ publish the hot items in their private input sets:
   - If the players are not concerned with Owner-Privacy: for each element $a \in P_i$, constructs a proof of inclusion in $S_i$, showing the path from the leaf $a$ to the root value $y_i$ and distributes the elements of the set $P_i$ to all other players, along with their proofs of correctness, using the ElementDist protocol, such that all connected honest players learn the set $P = P_1 \cup \cdots \cup P_n$.
   - If the players wish to ensure *Correlated Owner-Privacy*: (1) for each element $a \in P_i$, constructs a proof of inclusion in $S_i$, showing the path from the leaf $a$ to the root value $y_i$ and anonymously routes it to $\rho$ randomly chosen players; (2) the players distribute all the elements and proofs received in Step (1) using the ElementDist protocol, such that all connected honest players learn the set $P = P_1 \cup \cdots \cup P_n$.
   - If the players wish to ensure *Uncorrelated Owner-Privacy*: (1) for each element $a \in P_i$, opens his commitment to $a$, and anonymously routes it to $\rho$ randomly chosen players; (2) the players distribute all the elements and proofs received in Step (1) using the ElementDist protocol, such that all connected honest players learn the set $P = P_1 \cup \cdots \cup P_n$.
4. Each player $i = 1, \ldots, n$ verifies that each element $a \in P$: was committed to with a valid committment by some player in Step 1 and passes the filter – $\forall_{q \in [T]}\ x_{q,h_q(a)} = 1$

Figure 7: HotItem-Pub protocol, for publishing the hot items in each players' private input.

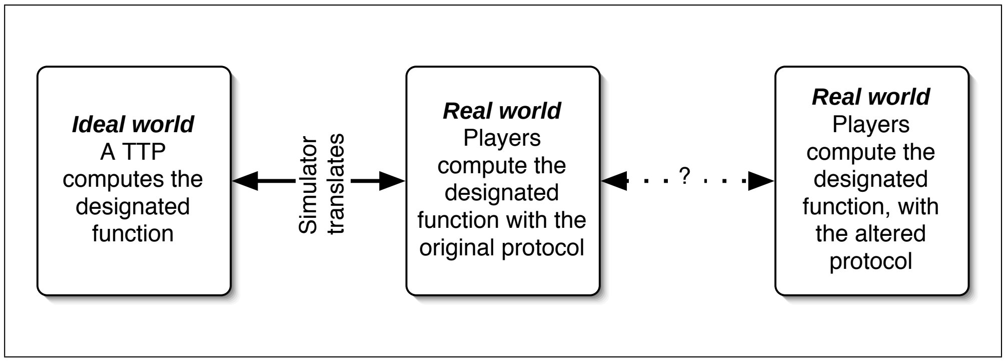
Ideal world
A TTP computes the designated function
Simulator translates
Real world
Players compute the designated function with the original protocol
?
Real world
Players compute the designated function, with the altered protocol

Figure 8: A simulation proof defines the behavior of the simulator, who translates between the malicious players $\Gamma$, who believe they are operating in the real model, and the ideal model, in which the trusted third party computes the desired answer. We wish to construct an algorithm to perform a similar task, translating from execution of the more-efficient protocol to execution of the original protocol.

## 6 Proposed Future Work

There are many promising areas of exploration in privacy-preserving distributed information sharing. In this section, we detail several interesting problems in which we expect we can obtain results. We will pursue at least one of the following areas.

### 6.1 Improving Efficiency

In order to improve the efficiency of protocols, we must improve the tools with which they are constructed. Protocols secure against malicious players often include inefficient steps or tools, which do not intuitively increase the security of the protocol. For example, an equivocal or chameleon commitment is no different from a normal commitment in a very important sense: it 'acts the same' to all players. These techniques are utilized only because they allow security against malicious players to be proved through use of a simulator. It is only to this simulator that the use of a chameleon or equivocal commitment is qualitatively different than use of a normal commitment; the definitions of security for such tools require this. Another example of an inefficient technique that appears in certain protocols secure against malicious players is what we refer to as the *no-key box*. Players construct a no-key box by encrypting data (and proving the validity of the encryption) under an unknown secret key. Note that *no* real player can know this secret key, if the cryptosystem is secure. The player then sends the no-key box to certain other players, who check the validity of the proof, but do not further utilize the no-key box. Intuitively, the no-key box cannot ensure any type of security, as its data is hidden from all players, but its use allows a proof of security to be constructed.

We have performed preliminary work on replacing inefficient tools, such as those described above, with their more efficient counterparts. The results of this work are promising; we have constructed a framework in which secure substitutions can be proven, without abandoning the paradigm of the simulation proof. In this way, we gain assurance that we are retaining security, while greatly increasing the efficiency of a wide variety of protocols secure against malicious players.

In a simulation proof, we define the behavior of a *simulator*, who translates between malicious players (who believe they are acting in the real world) and the trusted third party (of the ideal world). It is this simulator that makes such tools as chameleon commitments necessary, as he must leverage some sort of advantage in order to successfully translate between the real and ideal worlds. We wish to prove the security of protocols in which certain expensive tools have been replaced, negating the advantage on which the simulator relies.

First, we must define the circumstances under which one cryptographic tool may be substituted for another. To this end, we define each tool in terms of a set of *interfaces*. Each interface has a specified input and output; the tool is secure if no information can be gained, except as defined in the interface specifications. Note that an interface may be access-restricted. For example, a chameleon commitment is essentially a normal commitment with an added interface, accessible only to the simulator. A no-key box has no interfaces available to real players, but has a decryption interface available to the simulator; it is interface-indistinguishable from a randomly generated string, known as a *handle*. Certain interfaces may output or take as input a handle. Handles are chosen randomly from some specified distribution, and have no other purpose except as input to an interface of the same tool. Commitments are handles, for instance, as well as ciphertexts. One tool may be substituted for another if they are *interface-indistinguishable*; that is, each corresponding interface is computationally indistinguishable to all real players, with the exception of handles. There are several small details of this definition, important only once composed substitutions are introduced, that we do not explain here. We will merely claim that this theorem (and its proof) generalizes naturally to situations in which multiple cryptographic tools are substituted. This model of substitution has a compelling benefit: with the use of composition, interface-indistinguishability must be proven only once for each pair of tools, allowing unlimited substitution.

There are several ways in which we may prove this substitution rule is secure. The first, and simplest, is to define a *handle translator*. This translator has the ability, on input of a handle, to output a corresponding handle in the substituted tool, and vice-versa. In this way, the handle translator may translate naturally between players using the original protocol and players using the altered protocol. As illustrated in Figure 8, we may then utilize the original simulation proof, along with our translator, to prove the altered protocol is secure. The combination of the simulator and the translator give us an algorithm that translates between malicious parties (using the altered protocol) and the real world; thus we have designed a new simulation proof.

However, such a translator is non-uniform, leading us to a search for a different proof. There are several possibilities that we are currently exploring. We may avoid the need for a translator by operating directly in the real world with the original protocol, using an adversary designed for the altered protocol. This can be performed using non-black-box techniques: every time the adversary wishes to perform an operation on a handle from the substituted tool, the corresponding operation is performed instead on the original tool. The use of sampling allows the adversary to still believe he is operating under the altered protocol. Still, non-black-box techniques are not ideal, so we are exploring other possible proofs.

## 6.2 Extending Scope

All of our previous results have dealt with operations on sets and multisets. While these are very common structures for data, there are several more directions in which we wish to expand the scope of our techniques for privacy-preserving distributed information sharing. Primarily, we wish to focus on graphs; there is currently little research on privacy-preserving computation on graphs. An example application is in determining routing in a private network. Each player has secret ways to deliver messages to certain other players; these deliveries, however, are expensive.

However, not all players are directly connected, and thus must route their messages indirectly. Without knowledge of the network structure, it is difficult to efficiently route these messages. In a variant of this problem, different links between players have different costs. These costs can represent several things: probability of a courier being caught, the time for a message to traverse an edge, or the monetary cost involved in forwarding a message.

We have performed some preliminary research in this area, which has allowed us to develop several tools for use with undirected, directed, and labeled graphs. An important application of these techniques is in privacy-preserving use of probabilistic inference structures, such as Bayes nets and junction trees. These structures are represented as labeled graphs; we believe that the use of our techniques on these structures will yield interesting results. This area has been previously explored by Vaidya [49]. However, his techniques often require the disclosure of a large amount of information about private data, making them unacceptable in certain situations.

## 6.3 Timeline

We outline the thesis-related timeline, leading up to a thesis defense scheduled in approximately August 2006.

- *September 2005.* Complete proofs of security for cryptographic tool substitution framework in non-uniform and non-black box models. Begin search for proofs in alternate models.
- *November 2005.* Formalize and write proof of security for cryptographic tool substitution framework.
- *December 2005.* Begin exploration of other problems in privacy-preserving distributed information sharing, as outlined in this section.
- *May 2006.* Begin writing of draft of thesis document.
- *July 2006.* Draft thesis document completed.
- *August 2006.* Thesis defense.

## References

[1] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k'th-ranked element. In *Proc. of Eurocrypt*, May 2004. 2

[2] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270, 2000. URL `citeseer.ist.psu.edu/ateniese00practical.html`. 3.3, 5.2.2

[3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 1–10, London, UK, 2002. Springer-Verlag. ISBN 3-540-44147-6. 2, 5.2.1, 5.2.2

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

[5] Jan Camenisch. Proof systems for general statements about discrete logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich, Mar 1997. 3.2

[6] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 93–118, London, UK, 2001. Springer-Verlag. ISBN 3-540-42070-3. 5.2.2

[7] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 206–215, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-802-4. 1.1

[8] D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, 1988. ISSN 0933-2790. 3.3

[9] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–8, 1981. 3.2, 3.3

[10] David Chaum, Jan-Hendrick Evertse, Jeroen van de Graaf, and Rene Peralta. Demonstrating possession of a discrete log without revealing it. In A.M. Odlyzko, editor, *Proc. of Crypto*, pages 200–212. Springer-Verlag, 1986. 3.2

[11] D. A. Cooper and K. P. Birman. Preserving privacy in a network of mobile computers. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, page 2638, May 1995. 3.3

[12] R. Cramer, I. Damgård, and J. Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of Eurocrypt*, pages 280–99. Springer-Verlag, 2001. 3.2

[13] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret sharing scheme. In *Proc. of Eurocrypt*. Springer-Verlag, May 2000. 2

[14] Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *Proc. of Eurocrypt*, pages 557–72. Springer-Verlag, 2000. 3.2

[15] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag. ISBN 0-387-18047-8. 5.2.2

[16] P. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting of lotteries. In *Proc. of Financial Cryptography*, 2000. 3.2

[17] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Proc. of Asiacrypt*, pages 573–84, 2000. 3.2

[18] Michael Freedman, Kobi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Proc. of Eurocrypt*, volume LNCS 3027, pages 1–19. Springer-Verlag, May 2004. 2, 2, 4.1.1

[19] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of Crypto*, pages 368–87. Springer-Verlag, 2001. 3.2

[20] Archana Ganapathi and David Patterson. Crash data collection: A windows case study. In *To Appear in the Proceedings of the International Conference on Dependable Systems and Networks*, June 2005. 1.1

[21] Rosario Gennaro and Victor Shoup. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15:75–96, 2002. 3.2

[22] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 331–342, June 1998. 2

[23] Oded Goldreich. The foundations of cryptography – volume 2. http://www.wisdom.weizmann.ac.il/ oded/foc-vol2.html. 2, 2, 3.1.2, 3.1.2, 5.1.2

[24] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and Systems Science*, 28:270–99, 1984. 3.2

[25] Qiang Huang, Helen Wang, and Nikita Borisov. Privacy-preserving friends troubleshooting network. In *Proceedings of the Symposium on Network and Distributed Systems Security*, February 2005. 1.1, 2, 2

[26] M. Jakobsson. A practical mix. In *Proc. of Eurocrypt*, pages 448–61. Springer-Verlag, 1998. 3.2

[27] R. Karp, S. Shenker, and C. Papadimitriou. A simple slgorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28(1):51–55, 2003. 2

[28] Lea Kissner and Dawn Song. Private and threshold set-intersection. Technical Report CMU-CS-05-113, Carnegie Mellon University, February 2005. 1, 2, 3.2, 4.1.2, 4.1.2, 4.2, 4.2.1

[29] Lea Kissner, Dawn Song, Oren Dobzinski, and Anat Talmy. Hot-item identification and publication. Submitted to CCS '05. URL `http://www.cs.cmu.edu/~leak/hot-id.pdf`. 1, 5.2.2, 5.2.2, 5.2.2, 5.2.2, 5.2.3, 5.2.4

[30] David Lazarus. A tough lesson on medical privacy: Pakistani transcriber threatens ucsf over back pay. *San Francisco Chronicle*, October 2003. URL `http://www.sfgate.com/cgi-bin/article.cgi?file=/chronicle/archive/2003/10/22/MNGCO2FN8G1.DTL`. 1.1

[31] Patrick Lincoln, Phillip Porras, and Vitaly Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *Proceedings of the 13th USENIX Security Symposium*, page 239254, August 2004. 2, 2

[32] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proc. of CRYPTO*, August 2000. 2

[33] Steve Lohr. I.b.m. software aims to provide security without sacrificing privacy. *The New York Times*, May 2005. URL `http://query.nytimes.com/gst/abstract.html?res=F30714F8395D0C778EDDAC0894DD404482`. 1.1

[34] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, October 1996. 3.1.1, 3.1.1

[35] Ralph C. Merkle. A certified digital signature. In *Proc. of Advances in Cryptology*, pages 218–238. Springer-Verlag New York, Inc., 1989. ISBN 0-387-97317-6. 3.3

[36] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. Ap3: Cooperative, decentralized anonymous communication. In *11th ACM SIGOPS European Workshop*, September 2004. 3.3

[37] A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–25, 2001. 3.2

[38] United States Department of Heath & Human Services. Medical privacy - national standards to protect the privacy of personal health information. http://www.hhs.gov/ocr/hipaa/. 1.1

[39] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of Asiacrypt*, pages 573–84, 2000. 3.2

[40] A Pfitzmann and M Waidner. Networks without user observability. *Comput. Secur.*, 6(2): 158–166, 1987. ISSN 0167-4048. 3.3

[41] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explorations, the newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, January 2003. 2

[42] Bartosz Przydatek, Dawn Song, and Adrian Perrig. Sia: secure information aggregation in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265. ACM Press, 2003. ISBN 1-58113-707-9. 5.2.2

[43] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000. URL `citeseer.ist.psu.edu/ratnasamy01scalable.html`. 5.1.2

[44] M G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication, Special Issue on Copyright and Privacy Protection*, 1998. 3.3

[45] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. Technical report, 1997. 3.3

[46] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001. 5.1.2

[47] Victor Shoup. A computational introduction to number theory and algebra. http://shoup.net/ntb/. 4.1.2

[48] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. on Networking*, 11:17–32, February 2003. 5.1.2

[49] Jaideep Vaidya. *Privacy Preserving Data Mining over Vertically Partitioned Data*. PhD thesis, Purdue University, 2004. URL `http://cimic.rutgers.edu/~jsvaidya/pub-papers/thesis.pdf`. 2, 6.2

[50] Helen J. Wang, Yih-Chun Hu, Chun Yuan, Zheng Zhang, and Yi-Min Wang. Friends troubleshooting network: Towards privacy-preserving, automatic troubleshooting. In *Proceedings of IPTPS*, February 2004. 1.1, 2, 2

[51] Andrew C-C Yao. Protocols for secure computations. In *Proc. of FOCS*, 1982. 2, 2