Efficient, Secure, and Privacy-preserving Distributed Hot Item Identification

Lea Kissner **Hyang-Ah Kim**
leak@cs.cmu.edu **hakim@cs.cmu.edu**
Dawn Song **Oren Dobzinski**
dawnsong@cmu.edu **orend@cmu.edu**
Anat Talmy
atalmy@andrew.cmu.edu

August 2005 - last modified June 2006
CMU-CS-05-159

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Millions of people every day identify many types of popular or widespread items, including interesting entertainment content and malicious network attacks. Unfortunately, much of the information required for these popularity contests has privacy concerns attached to its use. Unless their privacy is preserved, participants may refuse to participate honestly, reducing the quality of the results. Provably ensuring the privacy of participants, while retaining the level of efficiency and robustness necessary for a practical protocol, is a serious challenge to privacy-preserving popular item identification. Previous efforts have not obtained the necessary efficiency [23] or provide only unproven, heuristic levels of privacy [24, 16, 39, 19]. In this paper, we propose and experimentally evaluate protocols that provably protect the privacy of participants, while adding only a small amount of overhead.

Keywords: privacy, distributed computation, hot item identification, hot item publication, distributed network monitoring, distributed statistics-gathering

1 Introduction

Recently, many researchers have studied network security applications where privacy is important; many of these applications bear a large degree of similarity. For example, a widespread network attack can be identified by a large number of distributed network monitoring devices. Once the monitors have individually identified a possible attack or offender, they must obtain confirmation; anomaly detection is imperfect, and false positives will occur. Many organizations automatically publish the network attacks they detect, and a false positive can result in the leakage of private or proprietary data. Each network monitor can gain confidence in the maliciousness of the attacks it has identified through comparison of its private set with the private sets of other monitors. An attack signature, event, or IP address that is flagged as being possibly malicious by many monitors is more likely to be truly malicious. This extra degree of confidence can prevent false positives from being published or used to establish misguided network filtering.

There are many more examples of statistics-gathering problems in which privacy is important. In computer troubleshooting common configurations among unbroken computers can be used to identify configuration errors and suggest fixes for troubled computers [16, 39, 19]. Privacy is important in this scenario, not only as an end in itself, but to prevent targeted attacks against misconfigured computers. If information was leaked during distributed troubleshooting that revealed a security hole, an attacker might take advantage. In a distributed content delivery network (CDN), distributed identification of popular pages (web pages that are commonly requested at different sites) is important for making effective caching decisions; often-requested pages should have higher priority when caching [7]. A peer-to-peer version of such a network may wish to preserve the privacy of participants; the web browsing habits of many people would reveal a large amount of data about their health, finances, and personal life.

Each of these problems is an application of a pair of more general problems: *privacy-preserving distributed hot item identification* and *privacy-preserving distributed hot item publication*. In these scenarios, players are connected through a network; each holds a private input set. A *hot item* is one that appears in many players' private input sets. The process of distributed hot item identification allows each participant to determine which elements in his local set are hot. Once they have been identified, distributed hot item publication allows the players to publish the hot items. In the case of distributed network monitoring, the monitors wish to determine which tentatively identified attacks are hot, and thus have been found by many other monitors. Distributed computer troubleshooting is simply a problem of finding and publishing hot computer configurations, with the idea that a common configuration is likely to be correct. To improve the performance of distributed caching infrastructure, it must be tuned to cache hot items over unpopular ones.

Care must be taken, however, in the process of hot-item identification and publication. Data about network monitoring can be used to aid in network attacks [24]. If certain types of information are leaked in the process of identifying hot attacks, it can negate the benefits of improved network security. We thus require two forms of security for privacy-preserving distributed hot item identification protocols: owner privacy and data privacy. *Owner privacy* requires that, with high probability, the elements in each players' private input set cannot be associated with that player, even when they are published. In the network monitoring scenario, this prevents an attacker from 'fishing' for information about a specific network; he cannot determine which honest network produced a report. *Data privacy* requires that the union of the data of all players is protected. Each element that appears in any player's private input set is hidden in a crowd of indistinguishable elements; an adversary cannot determine which element of the indistinguishable set was really a

part of some player's input. In the network monitoring situation, this adds a great deal of difficulty to the efforts of adversaries to learn information about honest players' networks; false positives will generally be rare, and thus hidden in a large crowd.

Current efforts to provide security for efficient and flexible hot-item identification and publication protocols are haphazard or heuristic, without concrete definitions of security [24, 39, 19]. Without such a definition and proof of security, it is difficult to evaluate the security of a scheme. Comparison of privacy becomes nearly impossible. Some protocols require central, trusted servers [24]. Others rely on trusting the friends of your friends of your friends (and so on) [39, 19]. We believe these assumption are untenable in many situations. Optionally, a trusted party (or mutually mistrustful group of parties) may ease the operation of our HotItem-ID protocol, but no such trust is necessary.

Many applications of privacy-preserving distributed hot item identification and publication are highly demanding, and so we require several additional properties of such protocols. Previous work includes a cryptographically secure protocol for hot item identification, as well as one for both hot item identification and publication [23]. This protocol, however, requires that all participants share a decryption key, participate in distributed decryption, and continue to participate in the protocol from beginning to end. Turnover in protocol participants will prevent the players from completing the execution of protocols in [23]. In many scenarios, these requirements are untenable. Especially if the network is unreliable, it is unrealistic to assume that *all* of the players will participate in the protocol from start to finish. Instead, we allow players to join or leave at will; they are not required to share a cryptographic key. We must, in addition, prevent malicious players from affecting the results of the protocol; they can lie about their private input set, but further attacks are prevented in our protocols.

The contributions of this work are as follows:

- We design a HotItem-ID protocol that is efficient enough to enable a wide range of applications, while preserving essential privacy. Let k be the threshold number of players who must hold an item before it is considered hot. If hot items appear many more times than non-hot items (skewed distribution), our protocols achieve a multiplicitive factor performance increase (in bandwidth) over a naïve, non-privacy-preserving protocol in the same network that is at least *linear* in k: for $k = 250$, we expect our protocol to use $\frac{1}{10}$th as much bandwidth as a naïve protocol; for $k = 2500$, we expect our protocol to use $\frac{1}{100}$ as much bandwidth. We also achieve linear gains (though with a smaller constant) with non-skewed distributions.
- We design a HotItem-Pub protocol that is efficient, while preserving privacy.
- We experimentally evaluate the efficiency of our protocols, and compare it to naïve, non-privacy-preserving protocols; our HotItem-ID and HotItem-Pub protocols are significantly more efficient, while preserving privacy.
- We achieve a high level of efficiency through the use of several approximation algorithms, which we have adapted for use in a distributed setting.
- We provide formal definitions of privacy for the privacy-preserving distributed hot item identification problem: data privacy and owner privacy. Our protocol achieves both of these properties.
- Our protocols are robust against malicious players: they cannot disproportionally affect the result of the protocol, nor can they prevent the honest players from executing the complete protocol.
- Our protocols are flexible, and allow players to join and leave at any time, because they

need not share cryptographic keys. Instead, we utilize a group signature scheme in a novel way [3, 6].

- We also design an efficient privacy-preserving protocol for the distributed hot item publication protocol. This owner and data private protocol allows players to securely publish the hot items they identified through use of the HotItem-ID protocol. (See Appendix 4.)

Outline. In the remainder of this paper, we first describe our model and problem definitions in Section 2. We describe our approach, tools, and HotItem-ID protocol in Section 3, before describing the HotItem-Pub protocol in Section 4. We then analyze the correctness, security, privacy, and performance of our protocols in Section 5. In Section 6 we briefly describe several extensions to our protocols. We present experimental evaluation in Section 7, discuss related work in Section 8, and conclude in Section 9. We summarize our notation in Appendix A, and give detailed analysis in Appendix B, before giving cryptographic constructions in Appendix C.

2 Problem Definition and Desired Properties

In this section, we give a formal definition of the hot item identification problem, its security and privacy properties, and the adversary model.

Problem Definition. In our problem setting, each player i ($1 \leq i \leq n$) in the system holds a private dataset S_i of m elements from a domain denoted M[1]. Let a *k-threshold hot item* (referred to in this paper as a *hot item*) be any element a that appears in at least k distinct players' private input datasets. H_k is the set of all k-threshold hot items in the system. All items not in H_k are called *cold items*. We define the hot item identification problem as follows: each player i ($1 \leq i \leq n$) learns $S_i \cap H_k$, denoted P_i.

Adversary Model. In this paper, we consider a strong adversary model. First, we assume that the adversary can eavesdrop on all communication. Second, we assume that a fraction λ of the players may maliciously and arbitrarily misbehave, while the rest of the players are honest-but-curious. A *honest-but-curious* player will (1) follow the protocol as specified, and (2) not collude with other players to gain information, though he may try to distill information from the messages he receives.

On the other hand, a malicious player may not follow the protocol; instead he may behave arbitrarily. For example, malicious players could collude such that if λn exceeds k, they can make an arbitrary cold item hot by each reporting it as in their private input set[2]. However, this problem is out of the scope of our problem. Because any player has the freedom to choose its private input set, any protocol in this setting is vulnerable to this manipulation.

Another attack a malicious player could mount is to claim to have many copies of one item to try to boost the frequency of a cold item to be high enough to become a hot item; we call this the *inflation attack*. Thus, our protocols must ensure that during hot item identification, each player can only contribute to the frequency count of an item at most once. Note that this is a challenging task, as we will need to preserve the players' privacy as well as security. We have designed a

[1] Note that here for simplicity, we assume each player has the same number of items in its private set; our protocols can be easily extended to scenarios where this is not the case.

[2] In fact, even when λn is less than k, they could make a cold item that appears at least $k - \lambda n$ times in non-malicious players to appear as a hot item.

novel cryptographic method, *one-show tags* (Section 3.2), both to prevent the inflation attack and preserve the players' privacy. This novel construction could be of independent interest.

Moreover, malicious players could attempt to forge cryptographic signatures, send bogus messages, try to learn honest players' private data, or fool other players. Note that our protocol defends against all of these attacks, by provably achieving the properties of owner and data privacy as defined below.

Definition of Correctness. Given the false positive rate δ_+ and the false negative rate

- $\forall a \in S_i \cap H_k$, player i learns that $a \in P_i$ with probability at least $1 - \delta_-$,
- $\forall a \in S_i \cap \bar{H}_k$, player i learns that $a \notin P_i$ with probability at least $1 - \delta_+$.

To allow more flexible and efficient protocols, while preserving a reasonable level of privacy for participants, we define two new concepts of security: owner privacy and data privacy.

Definition of Owner Privacy. Intuitively, a protocol that is *owner private* prevents an adversary from determining that any element a is part of a particular honest player's private input set. Even an element that is known to be hot cannot be shown to be held by any player; the elements of a player's private input set are anonymous.

Formally, we say a k-hot item identification protocol satisfies owner privacy if: no coalition of at most λn malicious 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 simply are given the set of hot items P and their frequencies.

Definition of Data Privacy. *Data privacy* concerns protection of the union of the players' inputs, especially cold items. Ideally, a truly privacy-preserving hot-item identification protocol should not reveal any information about the cold items; such a solution is given in [23]. However, as shown in [23], such a strong privacy definition entails inefficient solutions, as the cryptographic techniques add too much overhead in computation and communication for many situations. Thus, in this paper, we study the tradeoff between privacy and efficiency; in particular, we rigorously define a relaxed notion of privacy, show that it provides sufficient protection for many applications, and design efficient solutions to achieve it.

In our definition of data privacy, the degree of the privacy of an item describes the size of the 'crowd' in which the element is hidden; no adversary can determine which element of the large crowd was included in the players' private input sets. Formally, we say an element a which appears in f_a players' private input sets has $\Phi(f_a)$-degree of data privacy if: no coalition of at most λ malicious probabilistic polynomial-time (PPT) adversaries [18] can distinguish the element a from an expected indistinguishable set of $\Phi(f_a)$ elements. Thus, for a cold item a, the larger $\Phi(f_a)$ is, the better protected it is in general.

Efficiency. We also want the protocol to be highly efficient. In particular, to identify hot items that appear in at least a certain fraction of the total number of players' private input sets, we would like the protocol to have constant per player communication overhead. We will show that our protocol achieves this property by using a combination of various approximate counting methods.

3 HotItem-ID Protocol

In Figure 1, we show an overview of the components of our efficient privacy-preserving hot item identification protocol HotItem-ID and their purposes. We first introduce the intuition behind

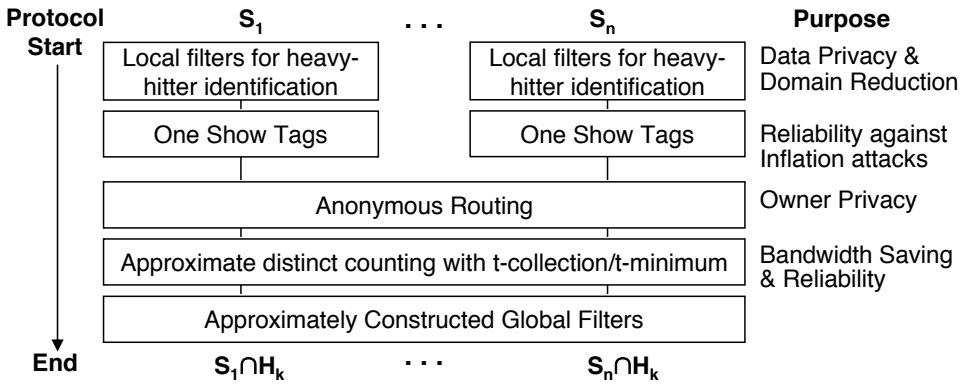


Figure 1: Components of HotItem-ID protocol: HotItem-ID defines how to efficiently compute an approximate representation of H_k in distributed fashion. Each player i ($1 \leq i \leq n$) constructs local filters to approximately represent his private input set S_i, generates one-show tags for marked bits in filters, and sends a subset of those one-show tags to the network using anonymous routing. A distributed approximate distinct element counting protocol aggregates those tags in the network. At the end of the protocol, all players learn the global filters that approximate H_k. At the right side of the figure we list the purpose of each component.

each component. Then, in Section 3.6, we describe the full construction of HotItem-ID. Once all players learn the hot items in their private datasets, they can run our HotItem-Pub protocol (Section 4) to securely publish the identified hot items.

3.1 Approximate Heavy-Hitter Detection

A naïve approach to hot item identification is to count the number of players holding each possible element, then determine whether that element is hot. However, performing this task is extremely inefficient for large domains; many applications require use of strings of 1024 bits or more. In most scenarios with a large number of players, it is prohibitively expensive to count each of these 2^{1024} (or more) distinct elements separately. Even if the players only count the frequency of those elements that appear in their private input sets, the bandwidth required will often be prohibitive. In order to avoid this inefficient naïve approach, we utilize an *approximate heavy-hitter identification scheme* [13]. This approximation scheme allows us to efficiently combine the process of counting the elements held by all players.

In this approximate heavy-hitter identification scheme, each player constructs a local filter. The players then combine their local filters to construct a global filter; this global filter approximately identifies hot items. We illustrate this process of local and global filter construction in Figure 2.

First, each player constructs a set of T *local filters*, which approximately represent his private input set. Let $h_1, \ldots, h_T : \{0,1\}^\omega \to \{1, \ldots, b\}$ be polynomial-wise independent hash functions with uniformly distributed output, such as cryptographic hash functions [26]. Each filter q ($1 \leq q \leq T$) for player i ($1 \leq i \leq n$) is an array of b bits, represented by the bit array $\{w_{i,q,j}\}_{j \in \{1,\ldots,b\}}$. A local filter bit set to 1 indicates that at least one element in the player's private input set hashes to that bit; we refer to such bits 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*, using methods described below; global filters approximately represent the players' combined private input sets. We will represent each global filter as the bit array $\{x_{q,j}\}_{j \in \{1,\ldots,b\}}$ ($1 \leq q \leq T$).

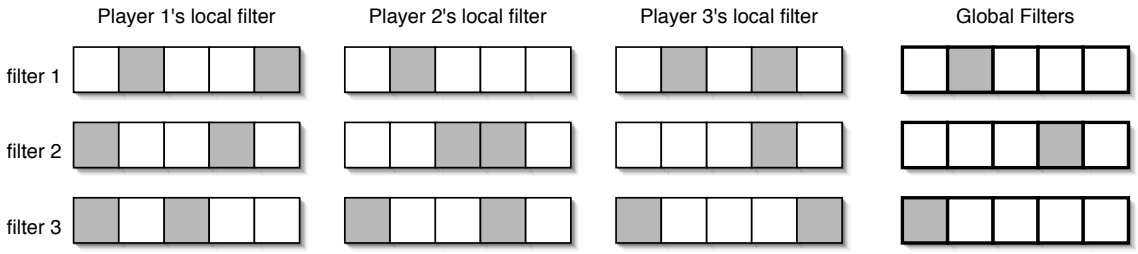


Figure 2: In our HotItem-ID protocol, each player i constructs a set of local filters from his private input set S_i (dark bits are 'hit'). The players then construct global filters using an approximate counting scheme; if a bit was hit by at least k players, then it is 'hit' (dark) as well. If an element hashes to a dark bit in each of the global filters, then it is classified as hot.

If at least k players marked bits 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 $x_{q,j} := 0$. Given this global filter, a is hot with high probability if $x_{1,h_1(a)} = 1, \ldots, x_{T,h_T(a)} = 1$. For statistical analysis of this approximation scheme, see Section 5.1.

3.2 One-Show Tags

In order to construct the global filters, we must count how many players hit each bit in their local filters. Malicious players may attempt to affect this count by 'voting' multiple times for a bit; this *inflation attack* could lead to elements being erroneously marked as hot. If players later publish their hot items, the attacker would learn private information through this attack. Most techniques that ensure players 'vote' at most once would reveal an unacceptable level of information about each players' private filters. The voting process must therefore be anonymous to prevent the adversaries from learning information about any particular players' private input. We ensure that each player can 'vote' only once, without compromising their privacy, with *anonymous one-show tags*. If a player set a bit, he constructs a tag for that bit; the players then count the number of valid tags for each bit to construct the global filters. We require that one-show tags posses the following properties: (a) no PPT adversary can construct more than one valid tag for any bit with non-negligible probability; (b) any PPT player can detect tags that are invalid, or not associated with a particular bit, with overwhelming probabilty; (c) for every bit, the tags constructed by any two players are distinct, with overwhelming probability; (d) no PPT 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.

Let all players share a group public key for tag verification. Each player holds an individual private key, used to construct tags. The message and one-show parameters vary by bit, to ensure that tags constructed for one bit are not confused with those for another bit. We denote the algorithm for construction of a one-show tag `OST`(group public key, private signing key, message, one-show parameter). We provide a construction of such tags in Appendix C. Note that this cryptographic tool is communication-efficient; our construction requires only $1,368$ bits.

3.3 Approximate Distinct Element Counting

We may now securely identify hot items by utilizing global filters and anonymous one-show tags. However, exactly counting the number of valid, distinct one-show tags is inefficient; the players would have to collect every single tag. To ensure that no tag is counted twice, the players would also have to store information about every tag.

We can perform the task of approximate counting of distinct tags through an efficient algorithm for *approximate distinct element counting* [4]. In our protocol, however, we gain even more efficiency by estimating directly, through modified use of this approximation, whether there are more or fewer than k tags for a bit; this is the task that must be performed to construct the global filters.

To approximate the number of distinct tags, we need t specific tags from the set. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\omega$ be a collision-resistant cryptographic hash function [26]. Let $v_1, \ldots, v_t$ represent valid one-show tags that, when hashed with $\mathcal{H}$, have the smallest values of any tags in the set. We estimate the total number of distinct tags as $\frac{t2^\omega}{\mathcal{H}(v_t)}$ [4]. To increase the accuracy of this approximation, one may choose α 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.

To perform the task of determining whether there are at least k distinct tags corresponding to a bit, the players can perform a full approximation. However, we have designed a more efficient variant for this particular problem. Note that, if there exist t tags such that the values of their hashes is at most $\frac{t2^\omega}{k}$, the approximation scheme we describe above will conclude that there are at least k total distinct tags. The players thus can instead perform this more efficient collection task, often without even examining ever tag. For example, any tag with value greater than $\frac{t2^\omega}{k}$ may be immediately discarded without affecting the approximation.

In many situations, there is a large gap between the frequency of hot items and cold items. For example, worm attacks will be detected a large number of times by network monitors; normal traffic is far more varied. Thus, to gain greater efficiency in detecting bits with at least k one-show tags, we may adjust the t-collection protocol to follow the algorithm of [38]: each player attempts to collect t tags with hash values of at most $(1+\gamma)\frac{t2^\omega}{k}$. ($\gamma$ is a constant based on the size of the 'gap' between the frequency of hot and cold items.) We gain efficiency in such situations by reducing t, while retaining accuracy. In practice, we have found $\gamma := .2, t := 5$ sufficient. This scheme is identical to the above scheme if $\gamma := 0$.

3.4 Anonymous Communication

We must now apply our tools to a network structure to complete the protocol. In order to disassociate the anonymous one-show tags each player constructs from their location in the network, each player anonymously routes his tags to ρ random players. The constant ρ can be varied to achieve greater or lesser degrees of robustness; at least one participating player should receive each player's tags. We require an anonymous routing scheme allows any player to send a message to a uniformly randomly selected player, without revealing the source to any intermediate routing node. Simple and lightweight schemes can be used, as we require only that each player send anonymous messages to a uniformly selected destination node, without revealing who sent the message. Some previously proposed anonymous networking schemes include [10, 8, 31, 32, 28].

Protocol: t-Collection
Input: All n players know $\mathcal{H}, t, k, \gamma$, 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 bit.
Output: Each player outputs `success` if they collected t valid one-show tags with one show value v such that $\mathcal{H}(v) \leq (1+\gamma)\frac{t2^\omega}{k}$; otherwise, they output `failure`.

1. For $\ell = 1, \ldots, D$, each player i $(1 \leq i \leq n)$:
   (a) Sends the set of new small tags $U_{i,\ell-1} \cap U'_{i,\ell-1}$ to all of his neighbors.
   (b) If $|U_{i,\ell-1}| = t$ then player i **stops participating** in the protocol and goes to Step 2.
   (c) Receives a set of tags as a result of Step 1a (a) – let the set $U_{i,\ell}$ be those which are valid tags for the correct bit, with one show value v such that $\mathcal{H}(v) \leq (1+\gamma)\frac{t2^\omega}{k}$.
   (d) Calculates $U'_{i,\ell} = U'_{i,\ell-1} \cup U_{i,\ell}$.
2. Each player i $(1 \leq i \leq n)$ outputs `success` if $|U_{i,\ell}| = t$, and otherwise outputs `failure`.

Figure 3: t-collection protocol

3.5 Distributed One-Show Tag Collection

The players, once they have anonymously received their initial set of tags, then count the number of tags associated with each bit of the filters. There are many different ways in which they can accomplish this task, such as through the use of gossip networks or centralized servers; the ideal method depends on the particular network in which it is employed. We suggest two distributed, secure, and efficient protocols that require only minimal network capabilities: each player must be able to send messages to each of its neighbors, who form a connected graph. As we require only minimal network capabilities, even more efficient schemes than appear in previous work may be employed [30, 33, 35]. For clarity of presentation, our protocols assume synchronous communication, but can be easily adapted to an asynchronous model.

Our two protocols for distributed one-show tag collection function as follows: (a) collecting t one-show tags (with sufficiently small hash value) to approximate whether there are at least k valid, distinct tags for a bit or, (b) finding the t one-show tags with the smallest hash values to approximate how many valid, distinct tags there are for a bit. Note that these protocols can be executed in parallel for each bit of the global filters.

We give an efficient, robust protocol for (a), the *t-collection* task in Figure 3. Each player maintains a set of at most t valid tags which have hash values at most $(1+\gamma)\frac{t2^\omega}{k}$. Upon learning a new tag that will be added to his set, a player sends the new tag to all of his neighbors. When a player has collected t tags with hash values of at most $(1+\gamma)\frac{t2^\omega}{k}$, and sent each of these tags to his neighbors, he ends his participation in the t-collection protocol. If there do not exist t such small-hash-value tags, the protocol must continue until it converges. The players must ensure that information travels from one side of the network to the other; this requires D rounds, where D is the diameter of the network.

We give an efficient, robust protocol for (b), the *t-minimum* task in Figure 4. Like the *t-collection* protocol, each player passes on small-valued valid tags to his neighbors, retaining the t tags with the smallest hash values. This process continues until it converges (at D rounds) and all players have learned the t tags with the smallest hash values. This protocol is based on that of [29].

Protocol: t-Minimum Aggregation
Input: All n players know $\mathcal{H}, t$, 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 bit.
Output: Each player learns the set U', those t tags with the smallest hashes of their one-show values.

1. For $\ell = 1, \ldots, D$, each player i ($1 \leq i \leq n$):
   (a) Sends the set of new small tags $U_{i,\ell-1} \cap U'_{i,\ell-1}$ to all of his neighbors.
   (b) Receives a set of tags as a result of Step 1a – let those which are valid one-show tags for the correct bit form the set $U_{i,\ell}$.
   (c) Calculates $U'_{i,\ell}$ to be those t tags with one-show values $v_1, \ldots, v_t$ such that $\mathcal{H}(v_1) < \cdots < \mathcal{H}(v_t)$ and $\forall_{w \in (U_{i,\ell} \cup U'_{i,\ell-1}) \cap \overline{U'_{i,\ell}}} \mathcal{H}(w) > \mathcal{H}(v_t)$.
2. Each player i ($1 \leq i \leq n$) outputs the set $U' = U'_{i,D}$.

Figure 4: t-minimum value aggregation protocol

Protocol: HotItem-ID
Input: There are n players, λ maliciously colluding, each with a private input set S_i. Each player $i \in \{1, \ldots, n\}$ holds a secret key sk_i allowing him to construct one-show tags, as well as a common public key pk for tag verification. These keys can be chosen by a trusted administrator or by a distrustful collective [3]. $h_1, \ldots, h_T$ are independently chosen cryptographic hash functions with range $\{1, \ldots, b\}$. $\mathcal{H}$ is a cryptographic hash function with range $\{0,1\}^\omega$. $\rho, b, T, t, \alpha, \gamma$ are parameters known to all participants.
Output: Each player i ($1 \leq i \leq n$) 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 $\{x_{q,j}\}_{j \in \{1,\ldots,b\}}$ ($1 \leq q \leq T$).

1. Each player i ($1 \leq i \leq n$) constructs T local approximate heavy-hitter identification filters $\{w_{i,q,j}\}_{j \in \{1,\ldots,b\}}$ ($1 \leq q \leq T$) from their private input set S_i: $\exists_{a \in S_i} h_q(a) = j \Rightarrow w_{i,q,j} = 1$. $w_{i,q,j} = 0$, otherwise.
2. Each player i ($1 \leq i \leq n$), for each bit j of filter q ($1 \leq j \leq b$, $1 \leq q \leq T$) s.t. $w_{i,q,j} = 1$:
   (a) constructs a one-show tag $\texttt{OST}(pk, sk_i, q||j, \text{one-show-value})$
   (b) player i anonymously routes the tag to ρ randomly chosen players (if the players will be utilizing t-collection in Step 3, do not send the tag unless $\mathcal{H}$ of the one-show value is $\leq (1+\gamma)\frac{t2^\omega}{k}$)
3. For each bit j of filter q ($1 \leq j \leq b$, $1 \leq q \leq T$) , all players $1, \ldots, n$, perform exactly one of the following tasks:
   - perform t-collection, with α independent hash functions, on the tags received in Step 2, attempting to collect, for each hash function $\mathcal{H}$, t valid one-show tags such that each one-show value $v \leq (1+\gamma)\frac{t2^\omega}{k}$. If t-collection was successfully performed for the majority of hash functions, set $x_{q,j} = 1$. Otherwise, $x_{q,j} = 0$.
   - perform t-minimum value aggregation, with α independent hash functions, on the tags received in Step 2. If v is the tth minimum hash value, approximate that there are $\frac{t2^\omega}{v}$ total tags; the median of each such approximation forms the final approximation. If this process determines that there were at least k tags, 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 $\forall_{q \in \{1,\ldots,T\}} x_{q,h_q(a)} = 1$, then $a \in P_i$.

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

3.6 Putting HotItem-ID to Work

We can now outline our HotItem-ID protocol, given in Figure 5. Let sk_i be player i's ($1 \leq i \leq n$) private key, allowing him to construct one-show tags. These keys can be distributed by a trusted 'group manager', by a distrustful group of players acting as such, or by all players acting in concert (see Appendix C).

In Step 1, each player i ($1 \leq i \leq n$) constructs T local heavy-hitter identification filters $\{w_{i,q,j}\}_{j\in\{1,\ldots,b\}}$ ($1 \leq q \leq T$) from their private input sets. They then construct a one-show tag for each bit in the filters marked as hit ($w_{i,q,j} = 1$); for a specific construction of the one-show value, see Appendix C. In Step 2, player i anonymously sends the tags for counting to ρ randomly chosen players. (If the players will be utilizing t-collection for counting, they may save bandwidth by only sending those tags where the hash ($\mathcal{H}$) of the one-show value is at most $(1+\gamma)\frac{t2^{\omega}}{k}$.)

To construct the global filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ ($1 \leq q \leq T$), the players must determine, for each bit in the global filters, whether at least k players hit that bit in their local filters. In Step 3, to efficiently and securely perform this task, the players utilize either the t-collection protocol (Figure 3) or t-minimum aggregation protocol (Figure 4). Using one of these protocols, the players approximate whether there were at least k valid, distinct tags constructed for each bit (for greater accuracy, the players perform this approximation α times. taking the median of the approximations). If the players conclude that there were at least k valid, distinct tags constructed for bit j of filter q ($1 \leq j \leq b$, $1 \leq q \leq T$), then they set $x_{q,j} := 1$; else, set $x_{q,j} := 0$.

Once they have constructed the global filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ ($1 \leq q \leq T$), each player can utilize these filters, as shown in Step 4, to determine whether each element of his input is a hot item. If, for an element a, $\forall_{q\in\{1,\ldots T\}}\ x_{q,\mathcal{H}(a)} = 1$, then a is a hot item with high probability.

4 Hot Item Publication Protocol

In this section, we present our protocol for HotItem-Pub. This protocol utilizes the HotItem-ID protocol, so that each player may identify his hot items, but also ensures that the players may securely end efficiently publish their common hot items.

4.1 Our Approach

When players publish hot items, we must prevent two attacks by malicious players: (1) *suppressing* hot items, so players do not learn them; (2) *fooling* honest players into accepting cold items as hot. As we have shown, our HotItem-ID protocol (see Section 3) effectively allows all players to identify the hot items in their private input sets, even in the presence of malicious players. By using the HotItem-ID protocol to identify hot items, we must only address attacks on the publication process itself.

Common Protocol Outline. Each of the variants of our HotItem-Pub protocol prevents both item suppression and fooling attacks; however, they achieve different levels of owner privacy. For clarity, we will first describe the basic outline for the protocol, which is common between all variants. In the HotItem-Pub protocol, the players follow four steps:

1. *Commitment.* Each player constructs a computationally-binding and computationally-hiding *commitment* to their private input set [26]; the exact form of the commitment varies according to the level of owner-privacy desired. This is later used to prove (by *opening* the commitment) that a published hot item was held by some player before the filters were constructed, without revealing any information at this stage of the protocol. If owner-privacy is desired, the players then send each of these commitments to ρ random players. This commitment is then distributed to all players.

2. HotItem-ID. The players then execute the HotItem-ID protocol, described in Section 3. Each player learns, with high probability, which of the elements of his private input set are hot, even in the presence of malicious players. In addition, each player also obtains the filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ $(1 \leq q \leq T)$ used to identify their hot items.
3. *Publication.* Each player, for each hot item in his input, constructs an opening for the commitment to that element (see the Commitment step). If owner-privacy is desired, the players then send each of these element/opened commitment pairs to ρ random players. All players the use a *redundant element distribution* protocol, ElementDist, to efficiently and robustly publish the element/opened commitment pairs. Such a protocol follows from the following rule: when a player receives a previously-unseen hot item, he: (1) using the opened commitment attached to the hot item, checks that he received a valid commitment to that item during the commitment phase of the protocol; (2) if it passes the check, sends the hot item (and associated commitment opening) to each of his neighbors and retains it himself in the set P. By following this simple protocol, duplicate publications of hot items are efficiently suppressed, while ensuring that all players receive each hot item in their *provisional set* P.
4. *Verification.* As a result of publication, each player obtains a provisional set of elements P, each of which is possibly hot. For each such item $a \in P$, the player checks that it passes the hot item identification filters; if $\forall_{q\in\{1,\ldots,T\}}\ x_{q,h_q(a)} = 1$, then it is truly a hot item.

In the remainder of this section, we will describe the details of each variant of owner privacy, as well as the commitments and openings used to meet each definition. Note that to increase the level of owner privacy provided, the players must utilize more bandwidth. We formally describe the HotItem-Pub protocol in Figure 6.

No Owner Privacy. If the players are not concerned about owner privacy, they may simply commit to their private input sets using 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 commitment opening: a verifiable proof (given the committment) that an element $a \in S$ [27].

Correlated Owner Privacy. In correlated owner privacy, the players wish to ensure that the elements of their private input sets cannot be traced to them, but do not care if an adversary can determine that some pair of elements came from the same player. To achieve this level of privacy, the players may also use Merkle hash tree commitments [27]. The protocol only differs from the non-owner private version in that the players anonymously send their commitments and hot items to random players before they are distributed in the commitment and publication phases of the protocol. This ensures that the elements can not be associated with any player.

Uncorrelated Owner Privacy. In uncorrelated owner privacy, the players wish to ensure that the elements of their private input sets cannot be traced to them, and *also* care if an adversary can determine that some pair of elements came from the same player. When the players desire this level of privacy, we cannot utilize the efficient Merkle commmitments, but instead must commit to each element separately. As the commitments, as well as the hot items, are sent anonymously to random players before publication, they cannot be linked with the player who constructed them. In addition, as elements have independent commitments, no player can gain advantage in determining whether one player held both elements.

Protocol: HotItem-Pub

Input: There are n players, λ 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. ρ, 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, each player constructs a hash tree from his randomly-permuted private input set S_i, with root hash-value y_i; then sends y_i to all players
   - If the players wish to ensure *Correlated Owner-Privacy*, each player constructs a hash tree from his randomly-permuted private input set S_i, with root hash-value y_i; then 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 $\{x_{q,j}\}_{j \in \{1,\ldots,b\}\{1,\ldots,b\}}$ $(1 \leq q \leq T)$
3. All players $1, \ldots, n$ publish the hot items in their private input sets:
   - If the players are not concerned with Owner-Privacy:
     (a) 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
     (b) 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*:
     (a) 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 ρ randomly chosen players.
     (b) the players distribute all the elements and proofs received in Step 3a 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*:
     (a) for each element $a \in P_i$, opens his commitment to a, and anonymously routes it to ρ randomly chosen players.
     (b) the players distribute all the elements and proofs received in Step 3a 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
   - passes the filter – $\forall_{q \in [T]}\ x_{q,h_q(a)} = 1$

Figure 6: HotItem-Pub protocol, for publishing the hot items in each players' private input.

5 Analysis

Now we proceed to analyze the security and performance of our HotItem-ID protocol.

5.1 Correctness

In Theorem 1 we prove that our HotItem-ID protocol functions correctly: hot items are identified as hot and cold items are identified as cold with high probability. Note that the filter sizes b, T must be chosen as described in the theorem for our guarantees to hold, though they may be smaller in practice.

Theorem 1. *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{m \lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda} \rfloor}{b} + \frac{\delta_-}{T} \right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1-\delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least $1-\delta_-$, every element a that appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets is identified as a k-threshold hot item.*

We defer the proof of Theorem 1 to Appendix B.1.

5.2 Privacy in HotItem-ID

In these theorems, we prove the owner and data privacy of our HotItem-ID protocol.

Theorem 2. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the* HotItem-ID *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$).*

When considering data privacy, we wish to prove that the cold elements in $S_1 \cup \cdots \cup S_n$ remain hidden from adversaries. For example, in the HotItem-ID protocol, no player or coalition of at most λ malicious players may gain more than an expected $I = \frac{tnm}{k} \lg b$ bits of information about all players' inputs $\bigcup_{i=1}^n S_i$. We prove a tighter bound on the degree to which each element is hidden in a crowd of *indistinguishable elements*. Two elements are indistinguishable if an attacker cannot distinguish which one was in players' private input sets based on the information gained in HotItem-ID. For an element a, its indistinguishable set consists of all elements indistinguishable from it by an adversary who has no prior knowledge. To provide data privacy, we wish for cold items to have large indistinguishable sets; we compromise perfect (semantic) security, in which all items are indistinguishable, to achieve much greater efficiency.

Theorem 3. *In the* HotItem-ID *protocol, each element a, which appears in f_a distinct players' private input sets, has an indistinguishable set of expected size $\Phi(f_a) = \sum_{\ell=1}^{T} \binom{T}{\ell} \left(1-\frac{t}{k}\right)^{f_a(T-\ell)} \left(1-\left(1-\frac{t}{k}\right)^{f_a}\right)^{\ell} \frac{|M|}{b^\ell}$.*

We defer the proofs of Theorems 2 and 3 to Appendices B.2 and B.3, respectively.

We graph the fraction of the indistinguishable set size to the domain size, $\frac{\Phi(x)}{|M|}$ in Figure 7. When an item a appears in f_a players' private datasets, the higher $\frac{\Phi(f_a)}{|M|}$ indicates that it is harder

for adversaries to distinguish a from other items in the domain $|M|$. Note that if a appears in only a few players' private input sets, a very large proportion of the domain is indistinguishable from a. As f_a approaches $\frac{k}{t}$, the size of the indistinguishable set decreases; this character ensures that truly rare elements are highly protected. In many applications, there is a big gap between the frequency of hot items and cold items. In this case, our protocol guarantees that the cold items will be extremely well protected, as their fequency will be much smaller than $\frac{k}{t}$.

5.2.1 Privacy in HotItem-Pub

We now consider the degree of owner privacy conferred by each version of our HotItem-Pub protocol. Correlated owner privacy allows adversaries to link the items published by each player, while uncorrelated owner privacy prevents this.

Theorem 4. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Correlated Owner-Private* HotItem-Pub *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Theorem 5. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Uncorrelated Owner-Private* HotItem-Pub *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$ for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Additionally, given two elements $a, a' \in P$, no coalition of at most λ malicious players can gain more than a negligable advantage in determining if there exists a honest player i ($1 \leq i \leq n$) such that $a, a' \in S_i$, assuming that the adversary is given the set of hot items P, and the frequency of each hot item.

We prove these theorems is given in Appendix B.3.

5.3 Performance

By utilizing a novel combination of approximation counting algorithms, our protocol represents a clear gain in efficiency over exact counting. In particular, aside from the cost of anonymous routing messages, which vary according to the scheme employed, our protocol achieves a multiplicitive factor of Tt/k more efficiency than the exact-counting based non-privacy-preserving protocol using the same message propagation abilities. (A network with improved capability for routing messages will increase the performance of both protocols.) Note that this improvement in performance is significant, especially considering that the baseline protocol provides no privacy at all, where our protocols enforce principled guarantees of privacy while retaining efficiency. In particular, if k is a fraction of n, our protocol achieves constant per player communication overhead. We present experimental results to validate the efficiency of our protocols in distributed networks in Section 7.

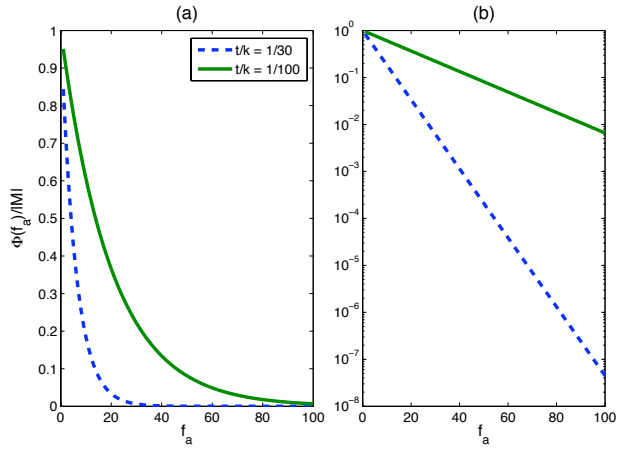


Figure 7: The degree of data privacy for an element with frequency f_a is $\frac{\Phi(f_a)}{|M|}$. Ideally, the degree of data privacy would be 1 for all frequencies, but by compromising this strong definition of security, we obtain more efficient and robust protocols. We graph the degree of data privacy **(a)**, showing the increase in protection for rare elements. The same function is graphed in **(b)** on a logarithmic scale, for increased detail.

6 Extensions

In this section, we briefly describe several extensions to our HotItem-ID and HotItem-Pub protocols.

Private Input Multisets. Instead of identifying hot items as those that appear in at least k players' private input sets, we may identify those items that appear at least k times in the players' private input multisets. In the modified protocol, we associate γ one-show values with a single (filter,bucket) pair so that up to γ hits to a single bucket by a single player can be counted as distinct elements. The constants b, T may be chosen by adjusting the analysis given in the proof of Theorem 1.

Using Bloom Filters. Bloom filters provide a compact probabilistic representation of set membership [5]. Instead of using T filters, we can use a combined Bloom filter. This achieves the asymptotic communication complexity, however, in practice the Bloom filter approach can have smaller constants. We describe our approach using T filters in this paper in the interests of clarity. Our scheme can be easily adapted to use Bloom filters. The choices of the constants b, T can be adjusted to give an acceptable false-positive rate, given the domain M; the analysis may be performed similarly to that of [5] and the analysis of the HotItem-ID protocol given in this paper.

Theorem 6. *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{mT \lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda} \rfloor}{b} + \frac{\delta_-}{T} \right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least* $1 - \delta_-$*, every element* a *that appears in* $f_a \geq \frac{k}{1-\epsilon}$ *players' private input sets is identified as a* k*-threshold hot item.*

We defer the proof of Theorem 6 is given in Appendix B.4. As the properties of owner and data privacy are unchanged by this variant, we do not perform a separate analysis.

Top-k. The Top-k problem is identifying those k elements that appear most often in the players' private input sets. An efficient non-private solution to this problem is given in [7]. In their protocol, a trusted authority calculates an estimated threshold value τ; every element in the top k appears at least τ times.

We now briefly describe how to calculate the top k values without use of a central authority and while preserving each players privately. Like in HotItem-ID, the players calculate global filters, but by using t-minimum value aggregation instead of t-collection. In this way, each player may estimate the number of players who hit each (filter,bucket) pair. Using these estimates, the players may approximate τ, marking each (filter,bucket) pair hit by at least τ players as 1. The global filters may then be used to identify the top-k.

7 Experimental Results

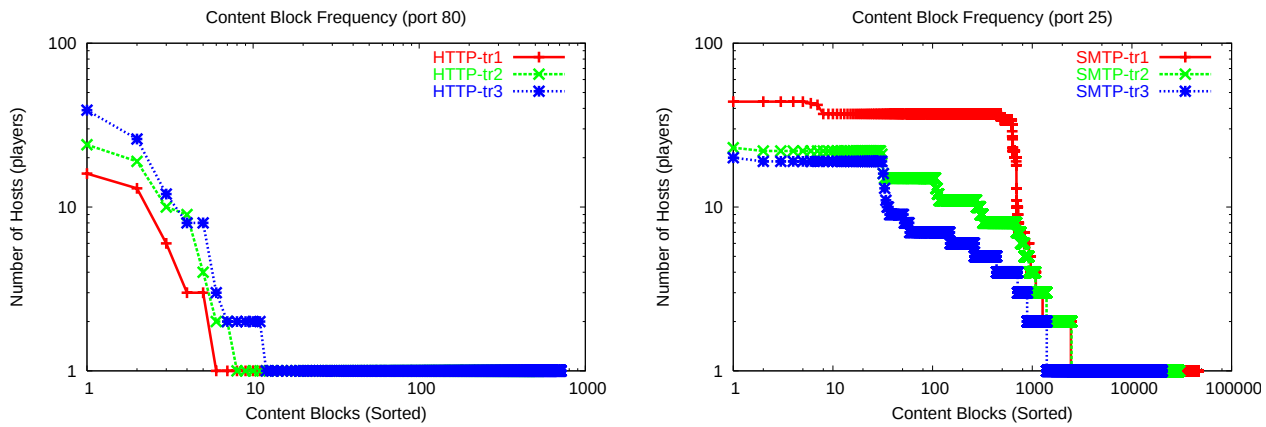
In order to experimentally evaluate the efficiency and accuracy of our protocol, we implemented our HotItem-ID protocol with t-collection and applied it to distributed worm signature detection. As we describe in Section 7.3, our experimental results support the efficiency and accuracy of our protocol.

7.1 Distributed Worm Signature Detection

A widely utilized tactic for detecting worms is to search for byte strings that appear with unusually high frequency in network traffic [22, 34]. By distributing the execution of this strategy over a large number of hosts, players can increase the accuracy of their results [22]. However, as network traffic often contains sensitive information such as email or information that could aid a network attack, it is imperative to protect the privacy of participants.

In a distributed network monitoring scenario, each monitor attempts to detect malicious traffic on his own network. As anomaly detection is imperfect, this process will often identify innocuous traffic as malicious. Thus, monitors must take further steps to improve the accuracy of their results; comparing the results with other monitors. If the malicious traffic belongs to a worm, there will be other monitors that observed similar traffic; an effective heuristic for identifying possibly malicious traffic is that worms often send a large volume of repetitive data as they attempt to infect other computers. We use our HotItem-ID protocol to perform this comparison while protecting the privacy of non-worm network traffic.

Because polymorphic worms often change their form, players must compare many small pieces of a possible worm payload instead of the entire payload. We use a content-based payload partitioning technique (proposed in [22]) to split a payload into many small segments. In this section, we call such segments *content blocks*. The payload partitioning technique is robust against small payload byte changes and generates the same content blocks for different forms of a worm. Once monitors have identified possible worms and split them into content blocks, they then perform hot item identification over the sets of content blocks. Content blocks generated from innocuous, private



(1) HTTP (port 80) **(2) SMTP (port 25)**

Figure 8: Number of hosts (players) that have generated each content block (item) from observed suspicious flows.

network traffic appear only in the sets from a few monitors. Hot content blocks indicate that the traffic has been seen at an unusually large number of hosts; it is therefore almost certainly part of a worm attacking those hosts and can be used as a signature of the worm.

7.2 Real-world Data and Experiment Method

We performed simulated distributed network monitoring on traces captured on a campus network that uses one third of a class B IP address space. `HTTP-tr1`, `HTTP-tr2`, and `HTTP-tr3` are one-hour long traces containing all HTTP (tcp port 80) packets and payloads addressed to hosts in the monitored network. Similarly, `SMTP-tr1`, `SMTP-tr2`, and `SMTP-tr3` contain SMTP (tcp port 25) packets captured for an one-hour time period. During the trace collection periods, a total of 5246 IP addresses received at least one packet. Our trace does not contain any known worm traffic and does not contain hot items (threshold $k = 100$). We graph the number of hosts who generate each innocuous content block (by identification of possible worm traffic) in Figure 8. Only at most 2.2% (HTTP) or 1.4% (SMTP) of content blocks appear at more than one host. Through manual examination, we determined that content blocks that appeared at more than one host indicated web crawler (HTTP) or spamming (SMTP) activity, and never appeared at more than 45 hosts.

We injected simulated worm traffic into the input of 200 of 1024 monitors in the network. Each monitor generates a set of content blocks from captured anomalous traffic. Note that all the sets generated from those 200 attacked monitors include the content blocks from the worm traffic. We ran our HotItem-ID protocol in the overlay network of 1024 monitors and measured the number of messages and the required bandwidth per monitor, while varying the t-collection parameters t and γ, and the average number of neighbors ψ in the overlay network. We also computed the false positive and false negative rates at the end of the HotItem-ID execution by counting the innocuous content blocks identifed as to be hot (false positives), and the worm content blocks that are not identified (false negatives). In our experiment, we utilized the parameters $b := 606, T := 5$ for HTTP traces, and $b := 4545, T := 5$ for SMTP traces, chosen according to the guidelines in Section 5.1. We compared our protocol to a non-private naïve protocol, in which all content blocks

		HTTP-tr3						SMTP-tr1					
		# of messages			bandwidth			# of messages			bandwidth		
		t=3	t=5	t=10	t=3	t=5	t=10	t=3	t=5	t=10	t=3	t=5	t=10
ψ=5	naïve	1 (11,399msgs)			1 (228KB)			1 (300,757msgs)			1 (6015KB)		
	γ=0	0.044	0.075	0.151	0.412	0.708	1.422	0.026	0.039	0.073	0.248	0.364	0.687
	γ=0.2	**0.048**	0.081	0.162	**0.455**	0.764	1.524	0.032	0.048	0.088	0.306	0.448	0.825
	γ=0.5	0.052	0.088	0.177	0.493	0.831	1.664	0.042	**0.062**	0.117	0.395	**0.583**	1.100
ψ=10	naïve	1 (25,643msgs)			1 (513KB)			1 (676,612msgs)			1 (13,532KB)		
	γ=0	0.038	0.068	0.141	0.363	0.638	1.331	0.017	0.021	0.065	0.162	0.199	0.611
	γ=0.2	**0.042**	0.071	0.146	**0.394**	0.665	1.372	0.022	0.028	0.068	0.208	0.261	0.638
	γ=0.5	0.044	0.074	0.149	0.414	0.695	1.402	0.036	**0.062**	0.124	0.341	**0.584**	1.169

Figure 9: Normalized bandwidth consumption per player in performing hot item identification ($k = 100$). Underline indicate that there were false positives or false negatives.

are forwarded to all participating 1024 monitors, using the same network topology and the same communication model for both the naïve protocol and the HotItem-ID protocol.

7.3 Bandwidth Consumption and Accuracy

Our HotItem-ID protocol efficiently identified every simulated worm injected into the network traces, while generating no false positives; no innocuous data was mistakenly categorized as malicious except when we use `SMTP-tr1` with $t \leq 3$ and $\gamma \geq 0.5$. We present our comparison of the required bandwidth and messages in Figure 9. `HTTP-tr3` contains 724 unique content blocks, while `SMTP-tr1` contains 46,120 unique ones. Gray boxes in the figure indicate there were false negatives (failed to identify worm content blocks). Our HotItem-ID protocol scales better than the naïve protocol (Section 5.2); as problems increase in size, our protocol becomes more attractive. Even at small problem sizes, the overhead required to protect the privacy of participants is not high. Our experiments show that to ensure correctness while retaining efficiency, we may set $\gamma := 0.2, t := 3$ (HTTP) and $\gamma := 0.5, t := 5$ (SMTP). Our HotItem-ID implementation, based on an efficient group signature scheme [6], requires 193 bytes per message (Appendix C) while the naïve protocol utilizes only 20 bytes per message. [3] However, our protocol requires only a small number of message transmissions; as a result, HotItem-ID used only 39% and 58% of the bandwidth used by the naïve protocol in the `HTTP-tr3` and `SMTP-tr1` experiments, respectively. Note that the performance gain from our HotItem-ID protocol increases as the more monitors participate. For example, when 10240 monitors participate in worm signature generation and we need only 10% of them to catch worm traffic ($k = 1000$), our HotItem-ID protocol uses less than 6% of the bandwidth used by naïve protocol.

8 Related Work

In a non-distributed context, [4, 17, 21] examine the identification of elements that appear often in a data stream, through the use of approximate counting. We generalize this task to a distributed setting, as well as enforcing important privacy-preserving properties.

Previous work also includes many protocols cryptographically secure against semi-honest players for related problems, including set-intersection [2, 1, 15, 40], equijoin [2, 1], association rule

[3] To save bandwidth and give a trivial measure of privacy against casual attacks [24], we hash each content block with SHA-1 in the naïve protocol.

mining [36, 20], classification [12, 25], statistical analysis [11], and top-k queries [37]. Several of these protocols are also secure against malicious players [15, 40].In our work, we achieve more efficient and flexible results, secure against malicious players, by designing our protocols to achieve more restricted notions of privacy.

Previous work in cryptographically secure protocols includes techniques on privacy-preserving set operations, which can be applied to distributed hot item identification and publication [23]; this work is secure under a very strict notion of privacy [18], but is insufficient for many applications. For example, it requires the players to share a decryption key and perform joint decryption. We do not believe that such an assumption is tenable in all situations. In our work, we achieve more efficient and flexible results by designing our protocols to achieve more restricted notions of privacy that we believe are sufficient for many applications.

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 [24]. They did not, however, give a concrete definition of security for their attempts to defeat such attacks, and their techniques 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 [39, 19]. Like the work in [24], 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. It is nearly impossible to evaluate the claims of privacy of these works, without a formal definition of security. We also believe some of the assumptions made in these works are untenable in many scenarios.

Our work avoids the strong impossibility results of the Byzantine agreement problem by utilizing one-show tags, a cryptographic construction related to group signatures.

9 Conclusion

In this paper, we design protocols for efficient, secure, and robust privacy-preserving distributed hot item identification. In this paper, we define the problem of privacy-preserving hot item identification (HotItem-ID), using novel definitions of privacy specific to this scenario. We design an efficient owner- and data-private protocol for HotItem-ID, robust against malicious players and unreliable networks. We also experimentally validate the efficiency and accuracy of our protocol on real-world data; we efficiently perform worm signature detection on real-world data and successfully identify worm payloads without false positives.

Acknowledgments. We would like to thank David Molnar, Christopher Colohan, Lujo Bauer, Nikita Borisov, and Alina Oprea for their invaluable comments.

References

[1] R. Agrawal, D. Asonov, and R. Srikant. Enabling sovereign information sharing using web services. In *Proceedings of the 23rd ACM SIGMOD international conference on Management of data*, pages 873–877, 2004.

[2] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the 22nd ACM SIGMOD international conference on Management of data*, pages 86–97, 2003.

[3] 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.

[4] 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.

[5] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[6] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of Computer and Communications Security (CCS)*, pages 168–177, 2004.

[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.

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

[9] D. Chaum, J.-H. Evertse, J. Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In *Advances in Cryptology—CRYPTO '86*, pages 200– 212. Springer-Verlag, 1987.

[10] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[11] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and claasification. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 222–233, 2004.

[12] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE International Conference on Privace, Security and Data Mining*, pages 1–8, 2002.

[13] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, August 2003.

[14] 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.

[15] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptography: Proceedings of Eurocrypt 2004*, 2004.

[16] 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.

[17] 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.

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

[19] 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.

[20] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions of Knowledge and Data Engineering*, 16:1026–1037, 2004.

[21] 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.

[22] Hyang-Ah Kim and Brad Karp. Autograph: Toward Automated, Distributed Worm Signature Generation. In *Proceedings of the 13th Usenix Security Symposium (Security 2004)*, San Diego, CA, August 2004.

[23] Lea Kissner and Dawn Song. Private and threshold set-intersection. In *Proceedings of CRYPTO '05*, August 2005.

[24] 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.

[25] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 36–54, 2000.

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

[27] Ralph C. Merkle. A certified digital signature. In *Proceedings on Advances in cryptology*, pages 218–238. Springer-Verlag New York, Inc., 1989.

[28] 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.

[29] 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.

[30] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[31] 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.

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

[33] 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.

[34] Sumeet Singh, Cristian Estan, George Varghese, and Stegan Savage. Automated worm fingerprinting. In *Proceedings of 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*, San Francisco, CA USA, December 2004.

[35] 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.

[36] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644, 2002.

[37] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *Proceedings of the 21st International Conference on Data Engineering*, 2005.

[38] Shobha Venkataraman, Dawn Song, Phil Gibbons, and Avrim Blum. New streaming algorithms for superspreader detection. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, February 2005.

[39] 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.

[40] Nan Zhang and Wei Zhao. Distributed privacy preserving information sharing. In *Proceedings of the 31st VLDB Conference*, pages 889–900, 2005.

A Notation

Notation.

- $a||a'$ - a and a' concatenated
- $a := a'$ - the value of a' is assigned to a

Variable List.

- a - an item, member of a private input set

- b - number of buckets per filter
- d - $d(x) = \frac{\Phi(x)}{|M|}$, a measure of data privacy which varies by the number of players holding any particular item
- e_i - group certificate for player i ($1 \leq i \leq n$) in group signature for one-show tags scheme
- f_a - number of players' private input sets in which an element a appears
- g_1, g_2 - group elements in one-show tag scheme
- $g_{q,j}$ - group element for producing one-show value ($1 \leq q \leq T$, $1 \leq j \leq b$)
- $\mathcal{H}(\cdot)$ - a cryptographic hash function with range $\{0,1\}^\kappa$
- $h_1(\cdot), \ldots, h_T(\cdot)$ - independent cryptographic hash functions with range $[b]$
- i - index over players ($1 \leq i \leq n$)
- I - expected number of bits of information that any coalition can learn about players' private input sets
- j - index over buckets ($1 \leq j \leq b$)
- k - minimum number of players that must hold an alert for it to be published
- ℓ - index
- m - maximum size of S_i
- M - domain of elements ($\bigcup_{i=1}^{n} S_i \subseteq M$)
- n - the number of players
- o - a secret value used in constructing one-show tags
- p_e - probability of edge existence in a random network graph
- p_t - probability that an anonymously routed message is traced to its source
- P_i - each player's set of alerts for publication
- $P = \bigcup_{i=1}^{n} P_i$
- q - index over filters ($1 \leq q \leq T$)
- r_1, r_2, r_1', r_2' - prime numbers used in one-show tags
- s - group manager's secret element in group signature scheme
- s_i - each player's secret element in group signature scheme ($1 \leq i \leq n$)
- S_i - each player's private input set ($1 \leq i \leq n$)
- $S = \bigcup_{i=1}^{n} S_i$
- t - number of values collected to estimate the number of distinct elements in a set
- T - number of filters
- u - estimated set size
- U_i - the tags collected during a t-collection or t-minimum value aggregation protocol by player i ($1 \leq i \leq n$)
- v - a one-show value
- $[w_{i,q,j}]_{j \in [b]}$ - local filter q ($1 \leq q \leq T$) for player i ($1 \leq i \leq n$)
- $[x_{q,j}]_{j \in [b]}$ - global filter q ($1 \leq q \leq T$)
- y_i - the root of a Merkle hash tree, a commitment to the input set S_i of player i ($1 \leq i \leq n$)
- z_0, z_1, z_2 - elements of the group signature public key for one-show tags
- Z_n - the ring of integers modulo n
- α - number of separate approximate distinct element counting estimations that must be combined to obtain the desired confidence probability δ_1
- β - secret value for ZK proof in one-show tags scheme
- γ - "gap" factor in approximate distinct element counting
- δ_1 - confidence probability for the approximate distinct element counting algorithm

- δ_+ - maximum probability of false positive
- δ_- - maximum probability of false negative
- ϵ - allowed error bound for the approximate distinct element counting algorithm is between $(1-\epsilon)$ times the actual value and $(1+\epsilon)$ times the actual value
- κ - security parameter for modified unpredictable-value group signature scheme
- λ - number of malicious players
- μ - secret value for ZK proof in one-show tags scheme
- η - part of public key for one-show scheme $\eta = r_1 r_2$
- ϕ - secret value for ZK proof in one-show tags scheme
- φ - maximum number of duplicates of each element allowed in a private input multiset (see Section 6)
- ρ - number of randomly chosen players chosen to receive anonymous messages in HotItem-ID and HotItem-Pub, so that, with high probability, at least one message reaches an honest player
- τ - threshold value for top-k protocol extension to HotItem-ID (see Section 6)
- σ - seed for pseudo-random number generator
- σ_ℓ - the ℓth member of Z_ν output by a pseudo-random number generator on input σ
- $\Phi(f_a)$ - expected size of indistinguishable set, if element a appears in f_a players' private inputs
- ψ - average number of neighbors per node
- χ - part of group signature public key for one-show tags scheme
- ζ - secret value for ZK proof in one-show tags scheme

B Detailed Analysis

B.1 Correctness

In this section, we analyze our HotItem-ID protocol to ensure that, given certain choices of the parameters b, T, our protocol correctly identifies hot items with high probability. In analyzing this protocol, we must consider both false positives (in which cold items are identified as hot), and false negatives. Errors may be caused by inaccurate approximate distinct element counting, a badly constructed filter, or a combination of the two.

Recall that $h_1, \ldots, h_T : \{0,1\}^\kappa \to \{1, \ldots, b\}$ are polynomial-wise independent hash function with uniformly distributed output, such as cryptographic hash functions, and that $\mathcal{H}$ is a collision-resistant cryptographic hash function [26].

Error from Approximate Distinct-Element Counting. To begin, we state a simple lemma that shows the t-collection and t-minimum value aggregation protocols (Figures 3 and 4) obtain the information needed for approximating the number of 'hits' to any particular bucket.

Lemma 1. *Each participant in the t-collection protocol of Figure 3 collects t 'small-value' one-show tags created by distinct players, if such tags exist.*

Each participant in the t-minimum value aggregation protocol of Figure 4 obtains the t one-show tags created by distinct players with the smallest hash values.

Proof. The proof of this theorem relies on the proof of information distribution in [29], which uses a closely related information distribution mechanism. We can thus observe that all information held

by connected honest players is distributed to *all* connected players unless it is specifically filtered by an honest player. In the t-minimum value aggregation protocol, the only tags filtered are those which do not appear in the final result; that is, they are not one of the t one-show tags with smallest hash value. Note also that the one-show value of a tag is unique per player, and that each player cannot construct tags with different, valid one-show values with overwhelming probability. Thus, in the t-minimum value aggregation protocol, all players obtain the t one-show tags with smallest values created by distinct players.

We may reason similarly about our t-collection protocol. Each player only filters tags (by terminating his involvement in the protocol) if he has collected t tags and sent them on to all neighbors. Thus, the player has ensured that if t small hash-value tags exist, each of his neighbors collect them as well. By induction, all connected honest players thus collect t small-value tags created by distinct players, if such tags exist. □

As detailed in [4], if $\mathcal{H}(v)$ is the tth smallest hash value in a set S, where $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\kappa$, then the estimate of $|S|$ is $\frac{t2^\kappa}{\mathcal{H}(v)}$. By computing the median of $O\left(\lg \frac{1}{\delta_1}\right)$ such estimates, with computationally independent hash functions, we obtain $\overline{|S|}$, an (ϵ, δ_1)-approximation of $|S|$:

Lemma 2. *This algorithm, for any $\epsilon, \delta_1 > 0$, (ϵ, δ_1)-approximates $|S|$. That is,*

$$Pr\left[\overline{|S|} > (1+\epsilon)|S| \vee \overline{|S|} < (1-\epsilon)|S|\right] \leq \delta_1$$

Proof. Proof of this theorem is given in [4]. □

As this is an (ϵ, δ_1)-approximation algorithm, we may note that we are concerned with elements that appear fewer than $\frac{k}{1+\epsilon}$ times, when computing false positives, and elements that appear at least $\frac{k}{1-\epsilon}$ times, when computing false negatives. Based on this algorithm, we may conclude that, if there exist at least t elements with hash values at most $\frac{t2^\kappa}{k}$, our estimate of $|S|$ is at least k:

Corollary 7. *If there exist t values $v_1, \ldots, v_t \in S$ such that $\forall_{\ell \in [t]}\, \mathcal{H}(v_\ell) \leq \frac{t2^\kappa}{k}$, then the approximate distinct element counting algorithm of [4] will estimate that $|S| \geq k$.*

Proof. Let $w = \max_{\ell \in [t]}\{\mathcal{H}(v_\ell)\}$. The approximation algorithm specifies that $\overline{|S|} = \frac{t2^\kappa}{w}$. As $w \leq \frac{t2^\kappa}{k}$, then $\overline{|S|} \geq k$. □

Error from Filters. We now calculate the probability that an element a, which appears in $f_a < k'$ players' private input sets, is identified as a k'-threshold hot item. We set $k' := \frac{k}{1+\epsilon}$, so as to account for the allowed inaccuracy in the approximate counting algorithm.

As illustrated in Figure 10, we may bound the probability that a is erroneously identified as k'-hot by **one** filter j' $(1 \leq j' \leq T)$ by determining the maximum number of filter buckets that were hit by $k' - f_a$ distinct players[4]. If a bucket j $(1 \leq j \leq b)$ was hit by $k' - f_a$ players who do not hold a, then if $h_{j'}(a) = j$, then a will be identified by filter j' as a k'-threshold hot item. As malicious players may claim to hit all buckets, a minimum of $k' - f_a - \lambda$ honest players must hit any given bucket for it to cause any possibility of error.

[4]Note that the f_a signatures related to the element a will raise the total number of signatures to k'; if there are at least k' signatures, error in the approximate counting algorithm may cause a to be identified as a k-hot item.

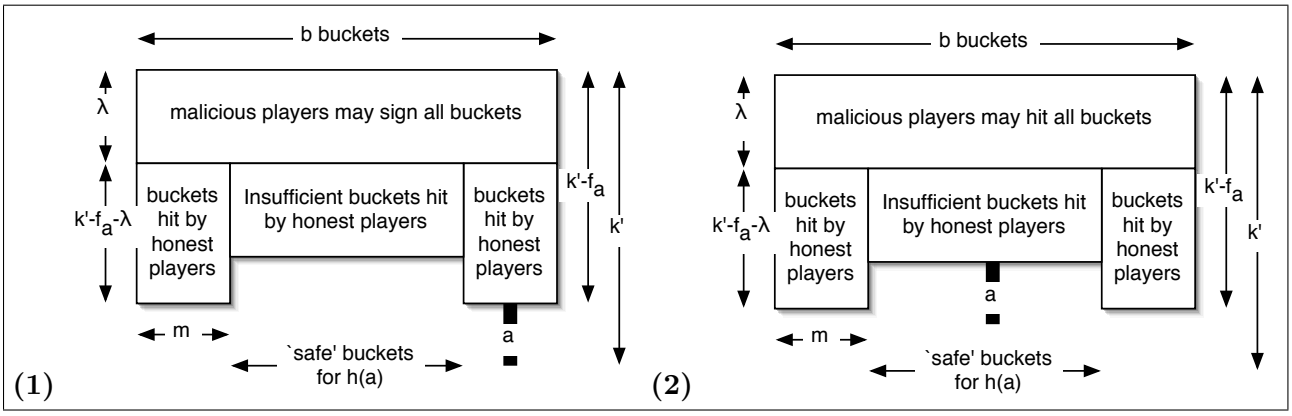


Figure 10: Buckets that are unsafe for $h_{j'}(a)$ $(1 \leq j' \leq T)$ are those that have been marked as hit by a sufficient number of players to allow the possibility that a might be erroneously identified as a k'-threshold hot item. In **(1)**, a is mapped to a sufficiently full bucket, causing a to be erroneously identified as a k'-hot item. In **(2)**, a is mapped to a safe bucket.

Each honest player has a maximum of m items in their private input set. Thus, using every element of each players' private input set, each group of $k' - f_a - \lambda$ honest players may hit at most m buckets a sufficient number of times to introduce any danger of error[5]. There are $n - f_a - \lambda$ honest players who do not hold a as an element of their private input set, and thus $\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda} \rfloor$ groups of players that can hit m buckets per group enough times to allow danger of an error. Note that any group of fewer than $k' - f_a - \lambda$ players cannot hit any particular bucket a sufficient number of times to cause a total of $k' - f_a$ hits; each player may only hit any particular bucket at most once.

Thus, at most $m\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda} \rfloor$ buckets of each filter can be 'unsafe'; if a is not mapped to one of those buckets, then there is no possibility of error from the malfunctioning of the filter. As $h_{j'}(a)$ is distributed computationally indistinguishably from uniformly over $[b]$, we may thus bound the probability that bucket $h_{j'}(a)$ is erroneously designated as 'hot':

$$\Pr\left[a \text{ is identified as } k'\text{-hot by one filter}\right] \leq \frac{m\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda} \rfloor}{b}$$

Combined Error. We may now consider the two sources of error together. There are two possible error types: false positives (in which cold items are identified as hot), and false negatives (in which hot items are not identified as hot).

Theorem 1. *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{m\lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda} \rfloor}{b} + \frac{\delta_-}{T} \right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

[5]Note that, with high probability, more that one element of an honest players' private input set will hash to the same bucket, and thus this is a conservative analysis.

In the HotItem-ID *protocol, with probability at least* $1 - \delta_-$*, every element* a *that appears in* $f_a \geq \frac{k}{1-\epsilon}$ *players' private input sets is identified as a* k*-threshold hot item.*

Proof. The probability that an element a, which appears in $f_a < \frac{k}{1-\epsilon}$ players' private input sets, is not identified as a k-threshold hot item can be bounded as follows (recall that we have set $k' = \frac{k}{1+\epsilon}$, to account for the allowed tolerance in approximate counting):

$$\begin{aligned}
\Pr[a \text{ is identified as } k\text{-hot}] &\leq \Pr[\text{in all filters, element } a \text{ is identified as } k'\text{-hot} \vee \\
&\quad \text{set of size} < k' \text{ approximated as } \geq k] \\
&= \prod_{j'=1}^{T} \Pr[\text{in filter } j', \text{ element } a \text{ is identified as } k'\text{-hot} \vee \\
&\quad \text{set of size} < k' = \frac{k}{1+\epsilon} \text{ approximated as } \geq k] \\
&\leq \prod_{j'=1}^{T} \left(\frac{m \lfloor \frac{n - f_a - \lambda}{k' - f_a - \lambda} \rfloor}{b} + \delta_1 \right) \\
&= \left(\frac{m \lfloor \frac{n - f_a - \lambda}{k' - f_a - \lambda} \rfloor}{b} + \delta_1 \right)^T \\
&= \left(\frac{m \lfloor \frac{n - f_a - \lambda}{\frac{k}{1+\epsilon} - f_a - \lambda} \rfloor}{b} + \delta_1 \right)^T
\end{aligned}$$

If an element a, which appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets, is not identified as a k-threshold hot item, it is due to error in the set-counting approximation. When the number of hits for every bucket in a filter is counted exactly, there can be no false negatives. Thus, we may bound the probability of a false negative as follows:

$$\begin{aligned}
\Pr[a \text{ is not identified as } k\text{-hot}] &= \Pr\left[\text{in at least one filter, a set of size } \geq \frac{k}{1-\epsilon} \text{ approximated as } < k\right] \\
&\leq \sum_{j'=1}^{T} \Pr\left[\text{in filter } j', \text{ a set of size } \geq \frac{k}{1-\epsilon} \text{ approximated as } < k\right] \\
&\leq \sum_{j'=1}^{T} \delta_1 \\
&= \delta_1 T
\end{aligned}$$

□

Choice of Constants. Given our analysis, we may outline how to choose the constants δ_1, t, b, T based on the parameters $\epsilon, \delta_-, \lambda, n, m, \delta_d, \beta$.

δ_1. Recall that the probability of a false negative, for an element a which appears in at least $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets, is required to be at most δ_-. We may then choose δ_1 as

follows:

$$\begin{aligned} \Pr\left[a \text{ is not identified as } k\text{-hot}\right] &= \delta_- \\ &\leq \delta_1 T \\ \delta_1 &:= \frac{\delta_-}{T} \end{aligned}$$

b, T. Given this assignment of δ_1, we may simplify our bound on the probability of a false positive on an element a, which appears in $\beta k < \frac{k}{1+\epsilon}$ players' private input sets, as follows:

$$\begin{aligned} \Pr\left[a \text{ is identified as } k\text{-hot}\right] &\leq \left(\frac{m \lfloor \frac{n - f_a - \lambda}{\frac{k}{1+\epsilon} - f_a - \lambda} \rfloor}{b} + \delta_1 \right)^T \\ &\leq \left(\frac{m \lfloor \frac{n - \beta k - \lambda}{\frac{k}{1+\epsilon} - \beta k - \lambda} \rfloor}{b} + \frac{\delta_-}{T} \right)^T \end{aligned}$$

We choose b, T so as to minimize $b \times T$, while satisfying the above constraint.

t, α. In the approximate distinct element counting algorithm in [4], $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$. In practice, one may safely choose to retain $t < \lceil \frac{96}{\epsilon^2} \rceil$ smallest values per parallel execution, and run only one parallel execution, while retaining a confidence bound of δ_1. We found that $t := 25$ was sufficient.

Note that, when running very small examples, or with very high accuracy requirements, one may obtain an assignment for t that is $\geq k$. In this case, simply set $t := k$, and note that $\epsilon, \delta_1 = 0$; each player is now collecting a sufficient number of signatures so as to determine, without error, whether any particular bucket was hit by at least k distinct players.

ρ. The choice of ρ must be based on the choice of anonymous message system. We provide here an analysis based upon the scheme of [28], in which each node, upon receipt of a message to be anonymously routed, sends it to a random neighbor with probability $p_f > .5$, and to its intended destination with probability $1 - p_f$. Note that the probability that a message will, somewhere along its anonymously routed path, encounter a malicious node is at most $\frac{\lambda}{n} + \frac{\lambda}{n} p_f + \frac{\lambda}{n} p_f^2 + \cdots \leq \frac{\lambda}{n}$. As we cannot ensure that a message is delivered if it encounters a malicious node, we wish to choose ρ such that at least one message will be delivered to its destination, with probability at least δ_d. Thus, $\rho := O\left(\lg \frac{1}{\delta_d}\right)$.

B.2 Owner Privacy

Theorem 2. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the* HotItem-ID *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$).*

Proof. The proof of this theorem is straightforward. □

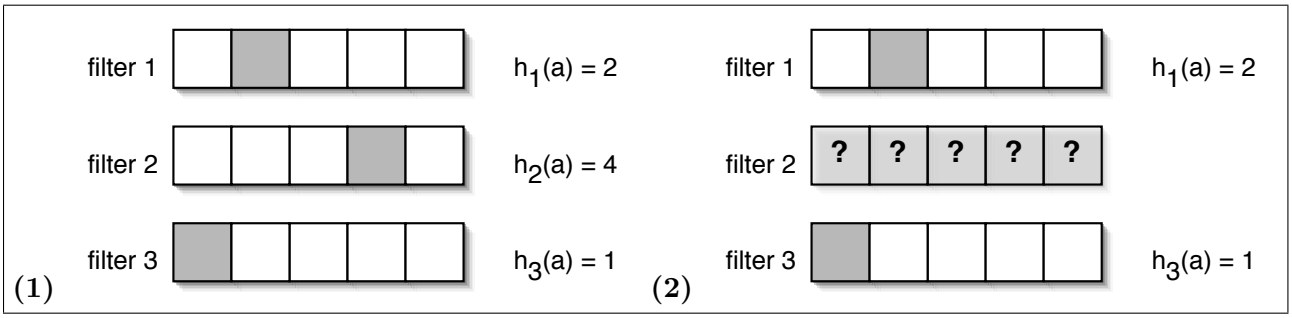


Figure 11: Players collect one-show tags for each filter bucket during the HOTITEM-ID protocol. Given a complete set of filters for some element, one may still not determine which element produced the filters. However, each tag has a probability of $1 - \frac{t}{k} - \frac{1}{2^\kappa}$ of having too high a value for the t-collection phase, and thus of being hidden. When all tags for an element in a specific filter are removed, as in **(2)**, an even larger number of elements could have produced the observed filters.

Theorem 4. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Correlated Owner-Private* HOTITEM-PUB *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Proof. The proof of this theorem is straightforward. □

Theorem 5. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Uncorrelated Owner-Private* HOTITEM-PUB *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$ for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Additionally, given two elements $a, a' \in P$, no coalition of at most λ malicious players can gain more than a negligable advantage in determining if there exists a honest player i ($1 \leq i \leq n$) such that $a, a' \in S_i$, assuming that the adversary is given the set of hot items P, and the frequency of each hot item.

Proof. The proof of this theorem is straightforward. □

B.3 Data Privacy

Theorem 3. *In the* HOTITEM-ID *protocol, each element a, which appears in f_a distinct players' private input sets, has an indistinguishable set of expected size* $\Phi(f_a) = \sum_{\ell=1}^{T} \binom{T}{\ell} \left(1 - \frac{t}{k}\right)^{f_a(T-\ell)} \left(1 - \left(1 - \frac{t}{k}\right)^{f_a}\right)^{\ell} \frac{|M|}{b^\ell}$.

Proof. Let M be the domain of possible private input set elements, such that $\forall_{i\in[n]}\ S_i \subseteq M$. Given knowledge of $h_1(a), \ldots, h_T(a)$, we may infer that approximately $\frac{|M|}{b^T}$ possible elements $a \in M$ could have produced that filter pattern, as there are b^T total possible filter patterns caused by one element and $h_1, \ldots, h_T$ are cryptographically secure hash functions. As illustrated in Figure 11, if information about one or more filters has been elided, a correspondingly larger number of elements could have produced that filter pattern. We may use Bernoulli trials to calculate the expected size of the indistinguishable set for element a, which appears in f_a players' private input sets:

$$\begin{aligned}
\mathrm{E}\left[\text{size of indis. set}\right] &= \sum_{\ell=1}^{T} \Pr\left[\ell \text{ filters are elided}\right] \frac{|M|}{b^\ell} \\
&= \sum_{\ell=1}^{T} \binom{T}{\ell} \left(1-\frac{t}{k}\right)^{f_a(T-\ell)} \left(1-\left(1-\frac{t}{k}\right)^{f_a}\right)^{\ell} \frac{|M|}{b^\ell}
\end{aligned}$$

We graph in Figure 7 the expected proportion of the total domain M of the indistinguishable set, as f_a increases. Note that if a appears in only a few players' private input sets, a very large proportion of the domain is indistinguishable from a. As f_a approaches $\frac{k}{t}$, less and less of the domain is indistinguishable; this character ensures that truly rare elements are highly protected. As t is a constant, independent of n, while k will often grow with the size of the network, we see that protection for rare elements generally increases as the network increases in size.

□

B.4 Analysis for Bloom Filters

Theorem 6: *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{mT\lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda}\rfloor}{b} + \frac{\delta_-}{T}\right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1-\delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least $1-\delta_-$, every element a that appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets is identified as a k-threshold hot item.*

Proof. The proof of this theorem very closely follows the proof of Theorem 1. □

C Details of One-Show Tags

In this section, we describe a one-show tag scheme obtained through the modification of the group signature scheme of Boneh and Shacham [6]. One one-show tag is lightweight, requiring only 1539 bits. Using the same techniques, similar constructions can be obtained from other group signature schemes.

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.

In the Boneh/Shacham group signature scheme, the group public key is $pk = \{g_1, g_2, w\}$, where $g_1 \in G_1$, $g_2 \in G_2$, and $w = g_2^{\gamma}$ for $\gamma \leftarrow Z_p^*$. (p can be taken to be a 170-bit prime, and the elements of G_1, G_2 can be represented in 171 bits [6].) The trusted group manager holds the group secret key γ. Each user i ($1 \leq i \leq n$) has a private key $s_i = \{A_i, x_i\}$, where $x_i \in Z_p^*$, $A_i \in G_1$. Using his private key, each player may sign a message, using a variant of the Fiat-Shamir heuristic [14], by proving knowledge of a pair $\{A_i, x_i\}$ such that $A_i^{x_i+\gamma} = g_1$.

We may modify this group signature scheme to include provably correct one-show values, making each signature a one-show tag. 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 $q||j$ essentially as in the original signature scheme; (2) computing two additional values to enable the recipient to compute the one-show value.

Each user generates $g_{q,j} \in G_2$ by an agreed-on scheme; we discuss this element later in the section. We utilize the same bilinear mapping e as in the computation of the main signature, as well as the intermediate values v, α, r_α computed as intermediate values utilized in the main signature. User i computes these additional elements for a one-show tag:

$$\begin{aligned} T_3 &= e(v, g_{q,j})^{r_\alpha} \\ T_4 &= e(v, g_{q,j})^{\alpha} \end{aligned}$$

The sole change to the original signature scheme is that the challenge value c is computed as $c = H(pk, q||j, r, T_1, T_2, T_3, T_4, R_1, R_2, R_3)$.

The recipient conducts all the validity checks specified in the Boneh/Shacham signature scheme, as well as the following additional check, derived from a proof of discrete logarithm equality [9]:

$$e(v, g_{q,j})^{s_\alpha} = T_4^c T_3$$

We define the one-show value as $e(A_i, g_{q,j})$; note that this value cannot be constructed by any player other than i and that player i can construct exactly one such value. To compute this value, the signature recipient computes:

$$\left(e(T_2, g_{q,j})^c e(v, g_{q,j})^{-s_\alpha} T_3\right)^{\frac{1}{c}}$$

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.

The parameter $g_{q,j}$, used to construct a one-show value associated with bucket j of filter q ($1 \leq q \leq T$, $1 \leq j \leq b$), can be efficiently constructed in a variety of ways such that no player knows its discrete logarithm. In this section, we briefly describe one such approach.

Let `PRNG` be a pseudo-random number generator with range G_2 [26]. Let H_{PRNG} be a hash function that takes bit strings as input and outputs data suitable for input into `PRNG`. Let $\sigma = H_{PRNG}(g||q)$. Let the ℓth element output from this `PRNG` on seed σ be denoted σ_ℓ. Each player calculates the element $\sigma_{\ell'}$, the first generator of G_2 in the sequence $\sigma_1, \sigma_2, \ldots$. Set $g_{q,j} = \sigma_{\ell'}$.