USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Over-Threshold Multiparty Private Set Intersection for Collaborative Network Intrusion Detection

Onur Eren Arpaci, Raouf Boutaba, and Florian Kerschbaum,
University of Waterloo

https://www.usenix.org/conference/nsdi26/presentation/arpaci

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Over-Threshold Multiparty Private Set Intersection for Collaborative Network Intrusion Detection

Onur Eren Arpaci
University of Waterloo

Raouf Boutaba
University of Waterloo

Florian Kerschbaum
University of Waterloo

Abstract

An important function of collaborative network intrusion detection is to analyze the network logs of the collaborators for joint IP addresses. However, sharing IP addresses in plain is sensitive and may be even subject to privacy legislation as it is personally identifiable information. In this paper, we present the privacy-preserving collection of IP addresses. We propose a single collector, over-threshold private set intersection protocol. In this protocol N participants identify the IP addresses that appear in at least t participant's sets without revealing any information about other IP addresses. Using a novel hashing scheme, we reduce the computational complexity of the previous state-of-the-art solution from $O(M(N\log M/t)^{2t})$ to $O(t^2M\binom{N}{t})$, where M denotes the dataset size. This reduction makes it practically feasible to apply our protocol to real network logs. We test our protocol using joint networks logs of multiple institutions. Additionally, we present two deployment options: a collusion-safe deployment, which provides stronger security guarantees at the cost of increased communication overhead, and a non-interactive deployment, which assumes a non-colluding collector but offers significantly lower communication costs and applicable to many use cases of collaborative network intrusion detection similar to ours.

1 Introduction

Many cyberattacks are coordinated and target multiple victims. An analysis from the security company Risk Analytics shows that 75% of attacks on institutions spread to a second institution within one day, and over 40% spread within one hour [23]. It is difficult for standalone intrusion detection systems to identify large scale threats due to lack of contextual information [28]. This underscores the need for collaborative network security, where institutions share and compare security data (logs, alerts, indicators) to quickly identify common threats.

A considerable body of work exists on collaborative threat detection [28, 52, 53] as well as privacy preserving collaborative threat detection [29, 32, 45]. However, until now the research focused on sharing alerts, anomalies, and indicators with other institutions. This approach assumes that the individual intrusion detection systems locally filter the raw network data such as IP logs, connections etc., and only report the "out-of-ordinary" events to other institutions. Zabarah et al. [50] demonstrated that analyzing raw network data across multiple institutions can reveal attacks that would go undetected by individual institutions alone. However, the biggest challenge for this approach is privacy. The problem is twofold. First, the raw data is significantly more sensitive than security alerts because it contains personally identifiable information. Second, the raw data is orders of magnitude larger than security alerts, making any existing solution computationally unfeasible. We propose a protocol that addresses both of these problems.

Zabarah et al. [50] found that a large portion of cyberattacks are typically initiated from a small number of IP addresses that are external to the institutions. Moreover, an external IP address initiating a connection to the institution's internal nodes is an atypical behavior if this connection is not for a publicly advertised service such as a website. Following these characteristics, Zabarah et al. propose a simple criterion with 95% recall rate for detecting multi-institution attacks. If an external IP connects to at least t institutions within a specific time window, it is classified as malicious, where t is an empirically decided threshold. This solution is simple and effective. However, it requires institutions to share their network logs, which creates the aforementioned privacy concern.

We address the privacy concern with an efficient Over-Threshold Multiparty Private Set Intersection (OT-MP-PSI) protocol. OT-MP-PSI is a cryptographic problem where N participants each hold a set that contains at most M elements. These participants want to know which elements appear in at least threshold t number of sets without revealing any information about elements that do not meet this threshold. Figure 1 shows a visualization of the problem. This problem is a direct abstraction of the Zabarah et al.'s [50] approach. The institutions do not want to share possibly benign IP addresses, but they are willing to share the IP addresses appearing in

multiple institutions, and are likely malicious.

While OT-MP-PSI has other applications such as heavy hitter identification within the network [11,24,31], private file deduplication on cloud [48] or identifying high-risk individuals in the spread of disease [6], this paper uses the problem of multi-institution attacks as the main focus and subject of evaluation.

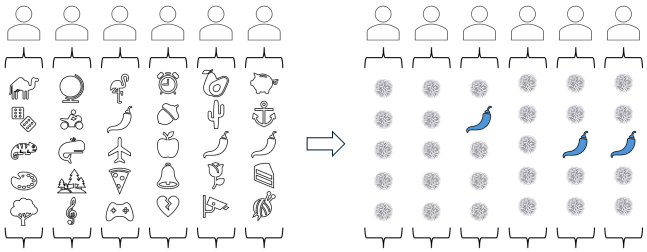

Figure 1: The OT-MP-PSI protocol visualization for parameters $N = 6, M = 5, t = 3$

Kissner and Song proposed the first solution to the OT-MP-PSI problem [26]. However, their solution requires a high number of communication rounds and it is computationally expensive. Specifically, their protocol requires $O(N)$ communication rounds, has $O(N^3M)$ communication complexity, and $O(N^3M^3)$ computation complexity. This makes it impractical for real-life applications with large data sets because it requires each participant to have significant computational resources and to be online for the duration of the protocol. Additionally, the sequential nature of the protocol hinders the scalability as adding more compute resources does not directly result in faster execution times.

Mahdavi et al. [34] introduce a solution with a constant number of rounds and better communication complexity. However, their scheme still imposes impractically high computational costs for our use case.

In this work, we propose an OT-MP-PSI protocol that outperforms the previous state-of-the-art solution [34] by factors ranging from $33\times$ to $23,066\times$ in terms of computational overhead. This efficiency improvement enables the practical usage of the protocol in the collaborative network intrusion detection problem because the nature of the problem requires the processing of large IP address sets as each institution receives connections from hundreds of thousands of IP addresses every hour. In a setting with 33 participating institutions and at most 144,045 IPs connecting to any institution, the system can detect malicious IPs in 170 seconds, as we show in our experiments. The previous state-of-the-art solution [34] requires multiple days to complete the same computation.

We use Shamir's secret sharing [47] to reveal identical items that are over the threshold. Shamir's secret sharing allows a secret data item to be split into many shares such that any size-t subset of the shares can be used to reconstruct the original secret. However, subsets smaller than t do not reveal anything about the original secret. We make every participant create a single secret share for each element they own, and then combine these secret shares among participants to find the elements that exist in t or more sets. The main challenge with this approach is that all t combinations of secret shares must go through the reconstruction process to find the shares corresponding to the same set element because secret shares do not reveal anything about the underlying set element. A naive approach requires $\binom{N}{t}M^t$ different combinations. Mahdavi et al. [34] uses a binning technique that reduces the necessary combinations to $O(M(N\log M/t)^{2t})$. As our main contribution, we introduce a new hashing scheme that reduces this number to $O(tM\binom{N}{t})$ linear in M and hence scalable to real problem sizes.

Moreover, we show that our protocol efficiently handles the case where the threshold is equal to the number of participants $(t = N)$ with a computational complexity of $O(N^2M)$. This problem is of independent interest [6, 37].

We prove the protocol's security under the semi-honest multiparty computation model. The semi-honest model prevents many practical attacks, such as eavesdropping by participants, but can be extended to the malicious model to handle active adversaries. However, even under the malicious model, private set intersection protocols remain vulnerable to input substitution attacks, which can reveal targeted information.

For network efficiency and ease of deployment, the protocol uses a dedicated Aggregator that combines the secret shares with some negligible leakage about the contents of the sets. We will explain precisely what this leakage is in Section 3.

We offer two separate deployment options for our protocol: collusion-safe option and non-interactive option. Collusion safe option requires constant rounds of interactions and has more communication overhead but it resists collusions. The non-interactive option has a much simpler structure and less communication overhead but it assumes the aggregator is non-colluding.

We implement our protocol and evaluate its performance on real-world network connection logs from CANARIE IDS program [10] to show its practicality.

The remainder of this paper is structured as follows. We define the security model and describe some preliminary cryptographic concepts in Section 2. We explain the use case of the protocol in detail and provide the formal functionality of the protocol in Section 3. We present the main challenges, our approach, and the formal description of our protocol in Section 4. We give a formal analysis of the failure probability of our new hashing scheme in Section 5. We prove our protocol's security and show its efficiency with both theoretical analysis and experimental results in Section 6. We review related work in Section 7 before we conclude our findings and discuss future work in Section 8.

2 Preliminaries

2.1 Security Model

We follow Goldreich's definition where a semi-honest party adheres to the established protocol but records all its intermediate calculations and any messages it receives from other parties [19]. We say a protocol is secure under the semi-honest model if each party cannot gain any other information except the intended output of the protocol. More formally, let $f:(\{0,1\}^*)^n \longmapsto (\{0,1\}^*)^n$ be an n-ary functionality, where $f_i(x_1,...,x_n)$, denotes the i^{th} element of $f(x_1,...,x_n)$. Let Π be an n-party protocol for computing f. The view of the i^{th} party during an execution of Π on $\bar{x}=(x_1,...,x_n)$ is defined as $VIEW_i^{\Pi}=(x_i,m_1,...,m_t)$ where m_i represents the i^{th} message it has received. We say that Π privately computes f with respect to the semi-honest model if there exists a polynomial-time algorithm, denoted SIM, such that for every $i \in [n]$

$$\{SIM(x_i,f_i(\bar{x}))\}_{\bar{x}\in(\{0,1\}^*)^n} \stackrel{c}{\equiv} \{VIEW_i^{\Pi}(\bar{x})\}_{\bar{x}\in(\{0,1\}^*)^n} \quad (1)$$

where $\stackrel{c}{\equiv}$ denotes computationally indistinguishable.

2.2 Shamir's Secret Sharing

The objective of secret sharing is to divide a secret value V into n shares so that it can be reconstructed when a specific subset of these shares is combined. In a (t,n)-threshold scheme, any group of t parties can collaborate to retrieve the secret value V using their shares, while any group with fewer than t members cannot gather any information about the secret, regardless of their efforts to collude.

In Shamir's secret sharing [47], the distributing party generates $t-1$ values $\{c_i\}_{i\in[t-1]}$ chosen at random from some finite field F_q of prime order q and forms the polynomial

$$\mathbb{P}(x) = c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + ... + c_1x + V \quad (2)$$

The distributing party generates n secret shares by evaluating $\mathbb{P}$ at n publicly-known distinct values. For instance, the secret share for party P_i (with $i \in F_q$) is $V_i=(i,f(i))$. Since t points uniquely determine a polynomial of degree $t-1$, anyone possessing t shares V_i can recover the secret V using Lagrange interpolation:

$$V = \sum_{i=1}^{t}\left(V_i \cdot \prod_{j=1,j\neq i}^{t}\frac{-j}{i-j}\right) \quad (3)$$

In our protocol, we leverage Shamir's secret sharing to indicate the existence of a set element only if at least t sets include that element.

2.3 Oblivious Pseudo Random Function (OPRF)

An Oblivious Pseudo-Random Function (OPRF) is a protocol between a key holder that holds a secret key K and a participant that holds an input x, where the participant learns the result of the Pseudo-Random Function $\mathbb{H}_K(x)$ without learning anything about the secret K, and the key holder does not learn anything about the input x or the value of $\mathbb{H}_K(x)$. In our protocol, we use the following 2HashDH OPRF protocol introduced by Jarecki et al. [22]. Let H and H' be two cryptographic hash functions.

Participant		**Key Holder**
$r \leftarrow_R \mathbb{Z}_q$		
$a \leftarrow H(x)^r$	$\xrightarrow{\quad a \quad}$	$b \leftarrow a^K$
Output: $H'(x,b^{1/r})$	$\xleftarrow{\quad b \quad}$	

This protocol obliviously evaluates $H'(x,H(x)^K)$. We can extend this protocol to multiple key holders with the following: the participant runs the protocol with the same input for all k key holders and combines the output by multiplying them.

$$H'\left(x,H(x)^{K_1}\cdot H(x)^{K_2}\cdots H(x)^{K_k}\right)$$
$$= H'\left(x,H(x)^{K_1+K_2+\cdots+K_k}\right) = \mathbb{H}_{K_1\ldots K_k}(x)$$

2.4 Oblivious Pseudo Random Secret Sharing (OPR-SS)

Oblivious Pseudo-Random Secret Sharing (OPR-SS) [34] is a protocol that combines share generation and reconstruction properties of secret sharing and the security properties of the OPRFs. It allows participants to get a secret share from key holders, unique for their input and their identity, without key holders learning anything about the input or the secret share, and without participants learning anything about the secret keys of key holders. Later, any t secret shares from t different participants with the same input can be used to reconstruct the original value. The functionality of the OPR-SS is given in Figure 2.

The functionality is parameterized by a threshold t, the number of key holders k and a value V.
Input of P_i: The participant P_i provides the input s
Input of each KH_j: Each key holder KH_j provides t secrets $\{K_{j,1},\ldots,K_{j,t}\}$.
Functionality: The functionality computes the polynomial

$$\mathbb{P}_s^{K_1,\ldots K_k}(i) = V + \sum_{m=1}^{t-1} i^m \cdot H(s)^{K_{1,m}+K_{2,m}+\cdots+K_{k,m}}$$

Output to P_i: The secret share $\mathbb{P}_s^{K_1,\ldots K_k}(i)$.
Output to each KH_j: Nothing.

Figure 2: Oblivious Pseudo-Randon Secret Sharing (OPR-SS) Functionality $\mathcal{F}_{OPR-SS}$

In our use case V is a public value that is set to 0. Reconstructing 0 from t different secret shares will indicate that they correspond to the same input.

3 Use Case

Detection of multi-institution cyberattacks is a critical issue faced by organizations today. Institutions that share a common context, such as universities within the same country or hospitals in the same city, often become simultaneous targets of malicious actors. Attackers typically leverage vulnerabilities across multiple organizations simultaneously, aiming to maximize their gains before these vulnerabilities are patched. Leveraging this intuition, Zabarah et al. [50] introduced an effective indicator with a 95% recall rate, which counts the number of distinct institutions contacted by an external IP address within a predefined time window. If this count exceeds an empirically determined threshold t, the IP address is classified as malicious. This approach is used in practice, due to its simplicity and general applicability, as it requires only monitoring external IP addresses without relying on specific attack signatures.

CANARIE [10] is an institutional internet provider that powers the network of 700+ Canadian research and educational institutions. The CANARIE IDS Program [10] is a collaborative initiative between CANARIE, and 106 Canadian research and education institutions. The program collects and analyzes participating institutions' network logs for intrusion detection purposes. It has deployed the indicator proposed by Zabarah et al. [50] and uses it to identify potential threats. However, participating institutions currently send their logs to CANARIE in plaintext so that IP addresses connecting to multiple institutions can be identified. This centralized, plaintext log-sharing model is also common in commercial outsourced Security Operations Centers, so the situation is not unique to our setting. Although straightforward, this method poses significant privacy risks. Institutions are hesitant to share complete network logs because these logs contain personally identifiable information and sensitive operational data unrelated to security threats. The aggregator, learning the entire dataset, obtains far more information than necessary. This also creates a compliance problem with privacy laws around the world. Many jurisdictions, including the European Union's GDPR [15], Canada's PIPEDA [14], and California's CCPA [27], classify IP addresses as personally identifiable information, limiting their sharing and processing without explicit consent. While these regulations allow exceptions for cybersecurity purposes, such exceptions are permitted only when strictly necessary.

Our protocol addresses these privacy and compliance challenges by enabling institutions to collaboratively identify IP addresses appearing in at least t institutional datasets, indicating potential malicious activity, without disclosing any IP addresses below this threshold. Consequently, institutions can privately and securely detect multi-institution attacks without entrusting a single aggregator with unrestricted access to sensitive data.

We assume that all parties are semi-honest, which means participants adhere to the protocol but may try to learn additional information beyond their intended output. Our protocol offers two deployment options: collusion-safe and non-interactive.

Collusion-safe deployment includes participants, key holders, and the aggregator. Some participants may also serve as key holders. In this topology, participants connect directly to the aggregator in a star configuration, key holders connect to all participants, and at least one key holder connects to all other key holders. This deployment assumes at least one key holder does not collude with the aggregator and requires a constant number of interactions. In practice, this deployment option is most useful when no neutral third party exists to serve as the Aggregator because it allows one of the participants to act as the Aggregator.

Non-interactive deployment eliminates the key holders and assumes a non-colluding aggregator. Participants may still collude among themselves. This deployment option is useful when there is a neutral, semi-trusted party that can act as the Aggregator, such as our use case with the CANARIE IDS Program [10].

The functionality is parameterized by a threshold t, the number of participants N, and number of key holders k. Let $P = \{P_1, \ldots, P_N\}$ be the set of participants, and A be the aggregator. If using the collusion-safe deployment, let $KH = \{KH_1, \ldots, KH_k\}$ be the set of key holders.

Input of each P_i: Each participant P_i provides a set $S_i \subseteq S$, where S is the universe of possible elements (e.g., IPv4/IPv6 addresses).

Input of A: The aggregator A provides no private input.

Input of each KH_j: The key holders provide no private input.

Functionality: The functionality computes the sets

$$I = \{s \in S \mid s \text{ appears in at least } t \text{ of the sets } \{S_1, \ldots, S_N\}\}.$$

$$B = \{(b_1, \ldots, b_N) \mid \begin{cases} b_i = 1 \text{ if } s \in S_i \\ b_i = 0 \text{ otherwise} \end{cases} \forall s \in I\}$$

Output to each P_i: Each participant P_i receives the set $I \cap S_i$.

Output to A: The aggregator A receives the set B.

Output to each KH_j: Nothing.

Figure 3: OT-MP-PSI Functionality $\mathcal{F}_{OT\text{–}MP\text{–}PSI}$

As shown in Figure 3, only the over-threshold elements are revealed to the participants and the aggregator only learns the existence of the over-threshold elements among participants. With this functionality, institutions can safely collaborate to detect multi-institution attacks without compromising the privacy of their complete network logs. In our use case of collaborative intrusion detection, the participants identified to be involved in an attack would share the identified poten-

Table 1: Notations

P_i	:=	i^{th} Participant
S	:=	Domain of elements
S_i	:=	Set of elements held by the participant P_i
$s_{i,j}$	:=	j^{th} element in the set S_i
A	:=	Aggregator
$H(\cdot)$	:=	Cryptographic hash function
$H_K(\cdot)$	:=	Hash-based message authentication code (HMAC) with key K
$h_K()$	:=	The keyed hash function used to map elements into bins
$\mathbb{P}_s(x)$	:=	Polynomial used for secret share creation
K	:=	Symmetric secret key used by participants
		Protocol Parameters
N	:=	Number of participants
t	:=	Threshold
M	:=	Maximum number of elements in each set
k	:=	Number of key holders

tially malicious IP addresses with other participants and the aggregator through a threat sharing platform such as MISP[1], identify the significant threats with severity estimation and take precautions using next-threat prediction as sugested by Zabarah et. al. [50].

4 Protocol

We first present a high-level overview of our OT-MP-PSI protocol. The principal idea is that all participants create a secret share for each element in their set and send these shares to the Aggregator. The Aggregator can only reconstruct secrets that have t or more shares. If a successful reconstruction happens, the Aggregator then informs the participants who sent the secret shares about the reconstruction to indicate that the underlying element is in the over-threshold intersection. This approach has a couple of challenges.

4.1 Using the same polynomial

We need to coordinate the use of Shamir's secret sharing, but the security of our protocol would be compromised if there was a central party that straightforwardly creates the shares, because the participants would have to disclose their set elements to this party to get a secret share. The OPR-SS protocol described in Section 2.4 is a good solution for this problem. This approach provides resistance to collusion but it is interactive. We also use a simpler approach that is non-interactive but less collusion resistant. In the non-interactive deployment, participants communicate a secret key K among themselves, which is not disclosed to the Aggregator, and they produce the coefficients of the polynomial themselves using a keyed hash function, as described in Equation 4. Here we secret share the value 0, similar to the OPR-SS protocol. This allows the aggregator to identify the successful reconstructions. Because if the shares correspond to the same element, they reconstruct the number 0, and if not, they reconstruct a random element of F_q.

$$\mathbb{P}_s^K(i) = 0 + \sum_{j=1}^{t-1} H_K^j(s) i^j \tag{4}$$

Where s is the set element, i is the identifier of the participant P_i, H is the hash-based message authentication code (HMAC) function, K is the secret key, and the superscripts of H_K indicate iteration, i.e., $H_K^i(s) = H_K(H_K^{i-1}(s))$. In our use case, the protocol uses IPv4/IPv6 addresses directly as the domain of elements without any preprocessing or mapping. With this approach, the Aggregator does not learn more than the intended output of the protocol as long as it is non-colluding.

4.2 Reconstruction Complexity

If we naively send all secret shares to the Aggregator, the Aggregator would need to try $\binom{N}{t}M^t$, an exponential number in t, different combinations to find all matching shares in all cases. We cannot use polynomial-time decoding algorithms for error-correcting codes, such as the list decoding [46] or Berlekamp-Welch' algorithm [49], because they may fail, if the number of elements exceeds the threshold t, but does not reach the decoding threshold for the error-correcting code. Comparing exponentially many combinations is infeasible if the set sizes M are large, even if the exponent t is rather low. We need to give the Aggregator some hint on how to combine the shares, but if we just put an identifier (e.g. keyed hash of the element) alongside the secret share, then the protocol would leak the similarity distribution of the sets and the Aggregator could see which sets have how many overlapping elements, even when those elements are under the threshold. This is more information than the intended output of our protocol.

A common approach to this problem is to put the shares into bins using a hash table. In this way, the Aggregator has to only try t-sized combinations of shares that are from the same bins. However, in order to not leak any distribution information, participants have to pad all of the bins with dummy shares so that all of the bins are equal in size. This approach reduces the complexity to $O(M(N \log M/t)^{2t})$, as shown by Mahdavi et al. [34]. Empirically, the maximum bin size, corresponding to the $\log M$ term in the complexity, carries large constants. This prevents the system from scaling to our problem size.

As the main contribution of this paper, we introduce a new hashing algorithm. We use hash tables (hereafter, simply tables) with bins of size 1 to map the secret shares. Instead of

[1] https://github.com/MISP/MISP

accommodating hash collisions with bigger bin sizes, we select one of the secret shares that map into the same bin using a pseudo-random ordering and only put that secret share into the table. Consequently, a single table will be missing some of the secret shares. Each participant creates multiple such tables. The motivation is that if participants create enough tables, missing values in one table will appear in others, and the probability of failure can be bounded to an acceptably low value with minimal cost. The main advantage of this approach is that the Aggregator does not need to try combinations of shares. It only needs to try combinations of participants. After selecting t participants, the Aggregator applies Lagrange interpolation to the shares corresponding to the same table and bins (e.g., the first participant's first table's first bin and the second participant's first table's first bin). An illustration of the new hashing scheme is shown in Figure 4. We show that if the size of each table is set to $t \times M$ and each participant produces 20 tables (with some additional optimizations), the probability of getting an incorrect result is smaller than 2^{-40}. We explain this hashing scheme in detail and show the failure probability in Section 5.

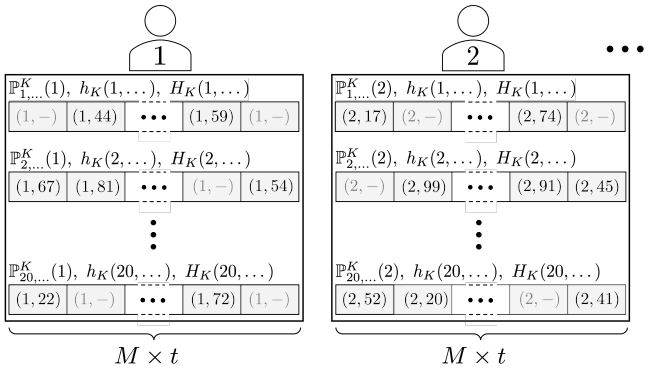

Figure 4: The new hashing scheme. Every participant creates multiple tables, $\mathbb{P}$ is the polynomial function, h_K is the keyed hash function used for mapping, and H_K is the keyed hash function used for pseudo-random ordering. The tuples in boxes represent the points on the polynomial, i.e. secret shares, and light gray numbers represent the dummy shares.

4.3 Protocol Description

Combining the solutions of these challenges, we describe our protocol as follows:

4.3.1 Non-Interactive Deployment

Let K be the symmetric key held by all participants $\boldsymbol{P_i}$ $\forall i \in \{1,\dots,N\}$. Let $h_K : \{0,1\}^* \longmapsto [Mt]$ be the keyed hash function used for mapping the elements to the tables. Let H_K be the hash-based message authentication code (HMAC) function with key K, and let r be the id of the current execution of the protocol.

1. Each participant $\boldsymbol{P_i}$ creates a table called *Shares*. It has 20 sub-tables and each sub-table has Mt bins. We denote the x^{th} bin of the y^{th} sub-table as $Shares[y][x]$. Participants fill the tables according to Equation 5 for each element $s_{i,j}$ in their set and for each $\alpha \in \{1,\dots,20\}$.

$$\textbf{if} \quad \begin{aligned} & H_K(\alpha, s_{i,j}, r) \leq H_K(\alpha, s, r) \\ & \forall s \in \{s \in S_i \colon h_K(\alpha, s, r) = h_K(\alpha, s_{i,j}, r)\} \end{aligned} \qquad (5)$$
$$\textbf{then} \quad Shares[\alpha][h_K(\alpha, s_{i,j}, r)] = \mathbb{P}^K_{\alpha, s_{i,j}, r}(i)$$

2. All $\boldsymbol{P_i}$ fill up the empty bins in *Shares* with random bits, then send the tables to the Aggregator $\boldsymbol{A}$.

3. After receiving N *Shares* tables, $\boldsymbol{A}$ takes every t combination of $\boldsymbol{P_i}$ and checks if the shares in the same bins produce a valid secret.

4. For each $\boldsymbol{P_i}$, $\boldsymbol{A}$ sends the indexes of valid reconstructions in the *Shares* table that $\boldsymbol{P_i}$ has sent.

5. Each participant $\boldsymbol{P_i}$ matches the indexes with the underlying elements in their set S_i and get their output $S_i \cap I$.

4.3.2 Collusion-safe Deployment

For the collusion-safe deployment, participants do not hold a symmetric key. All the secret shares $\mathbb{P}^K_{\alpha, s_{i,j}, r}(i)$ are computed with the OPR-SS protocol described in Section 2.4 and instead of the keyed hash functions H_K and h_K, the multi key OPRF protocol described in Section 2.3 is used. Since H_K and h_K uses the same inputs for the same elements, a single OPRF call is used to produce both values. All the interactive rounds of the OPR-SS and OPRF protocols for each element is batched together to achieve constant number of interactions. The rest of the protocol is identical with the non-interactive deployment.

4.4 Privacy of Set Sizes

Our core protocol do not consider participant's set sizes as private information. So by default, participants communicate their set sizes in plaintext and find the max set size M before running the protocol. If set sizes are private for a particular use case, then M could be decided with a differentially private process. However, this process would need to add positive noise to M since underestimating M will break the core protocol, and this positive noise will adversely affect the performance of the core protocol because the runtime complexity is dependent on M.

5 Analysis of Our Hashing Scheme

In this section, we analyze our new hashing scheme described in Section 4.3 designed to efficiently address the complexity of the matching problem. The challenge lies in the fact that the secret shares, by definition, do not reveal the underlying element, making it difficult to select a matching combination. In our hashing scheme, each participant places their secret shares into bins of size 1 using a hash function. Consequently, the Aggregator needs to process only one share per participant that fall into the same bin instead of selecting all subsets among a larger set.

In cases of hash collisions, where multiple elements map to the same bin, we must select a single element from those that collide. Rather than choosing this element randomly, we employ an additional hash function to establish an order among the shares and then select the smallest one according to this order. This method increases the likelihood that different participants will choose the same element for a given bin.

Participants repeat this process to create multiple tables using a distinct mapping hash function and ordering hash function for each table. Our goal is to limit the probability of missing an intersection to an acceptably low level by re-randomizing the mapping and ordering for each table. We set our target failure probability to 2^{-40} as 40 is a common statistical security parameter [7, 13, 30].

We first show that if the size of the table is set to $M \times t$, the probability of missing an intersection is a constant number smaller than 1 for a single table. Let $h : \{0,1\}^* \longmapsto [Mt]$ denote the mapping hash function, H denote the cryptographic ordering hash function, and $shares_i$ denote a single sub-table of the participant P_i.

Given a set element $s_{i,j} \in S_i$ of the participant P_i, let p denote the probability of $H(s_{i,k}) < H(s_{i,j})$ for a random $s_{i,k} \in S_i$. Since the output distribution of H is uniform, p can be calculated with the following equation:

$$p := \frac{H(s_{i,j})}{max(H)}$$

The probability that a random set element $s_{i,k}$ maps to the same bin as $s_{i,j}$ is given by

$$P(h(s_{i,j}) = h(s_{i,k}) \mid s_{i,j}) = \frac{1}{Mt}$$

The probability that a random set element $s_{i,k}$ blocks $s_{i,j}$, that is, $s_{i,k}$ maps to the same bin and is smaller than $s_{i,j}$ in the ordering:

$$P\Big((h(s_{i,j}) = h(s_{i,k}) \text{ and } H(s_{i,k}) < H(s_{i,j}) \mid s_{i,j}\Big) = \frac{p}{Mt}$$

To calculate a lower bound for the probability that $s_{i,j}$ will be placed in the table, we raise the probability that $s_{i,j}$ is not blocked by a random element to the M^{th} power because $|S_i| \leq M$.

$$P\Big(H(s_{i,j}) < H(s)\ \forall s \in \{s \in S_i\ :\ h(s_{i,j}) = h(s)\} \mid s_{i,j}\Big)$$
$$= P(s_{i,j} \in shares_i) = \Big(1 - \frac{p}{Mt}\Big)^{|S_i|-1} \geq \Big(1 - \frac{p}{Mt}\Big)^{M}$$

With large M, the expression $\left(1 - \frac{p}{Mt}\right)^M$ approximates $e^{-p/t}$ and for $M > 50$ this approximation does not affect any of our results. We now calculate the probability that t different participants will put $s_{i,j}$ in the table given that $s_{i,j}$ is in their sets. This is because, for a successful reconstruction, all participants that have the set element have to put the element into the same table. Since all participants use the same ordering hash function for the same table, the p values will be the same, so we can raise the $e^{-p/t}$ to the t^{th} power to calculate this probability.

$$P(s_{i,j} \in shares_k\ \forall_{k\in[t]} \mid s_{i,j} \in S_k\ \forall_{k\in[t]}) \geq \Big(e^{-p/t}\Big)^t = e^{-p}$$

From this equation, it follows that the probability of failure for a particular set element is less than or equal to $1 - e^{-p}$, given that this element exists in t different sets.

To calculate the probability of failure for any set element, we treat p as a continuous random variable with a uniform distribution between 0 and 1.

$$P(\text{fail} \mid p) \leq 1 - e^{-p}$$
$$P(\text{fail}) \leq \int_0^1 (1 - e^{-p})dp = e^{-1} \approx 0.3678$$

This shows that the probability of failure, more explicitly, the probability of missing any given over-threshold intersection with a single table is at most e^{-1}. Every participant needs to create 28 different tables so that the probability of failure is at most $(e^{-1})^{28} \approx 2^{-40.4}$

We implement two additional optimizations to the hashing scheme that brings the required number of tables to 20. First, instead of using a unique ordering hash function for each table, we use the same ordering hash function for every two consecutive tables and reverse the ordering for even-numbered tables. Second, We do a second insertion after the first insertion for every table. We use a different mapping hash function h' for the second insertion. The second insertion utilize the unused bins from the first insertion. The theoretical analyses of these optimizations are described in Appendix A.

6 Evaluation

We evaluate the security and performance of our OT-MP-PSI protocol. First, we provide a security analysis using the semi-honest multi-party computation model. We then provide a theoretical complexity analysis for computation and communication costs. Finally, we present experimental performance benchmarks for our protocol, including on the real-world data set from CANARIE IDS Program [10].

6.1 Security Analysis

In this section, we analyze the security of our protocol under the semi-honest model and show that it is secure by constructing the simulators for the participants and the Aggregator.

Theorem 1. *The non-interactive protocol described in Section 4.3 is secure in the semi-honest model given that the Aggregator A is non-colluding.*

Proof. We will prove security by constructing the simulator functions, defined in Section 2.1, for the participants and the Aggregator. The construction of the simulator for participants $SIM_{P_i}((S_i,K,r),I\cap S_i)$ is as follows: The simulator needs to produce the message sent by the Aggregator, which is the indexes of the successful reconstructions as stated in the step 4 of the protocol in Section 4.3. To do this, simulator creates the *Shares* table of P_i by replicating the first step of the protocol. Then using the protocol output $I\cap S_i$, it identifies the indexes in the *Shares* table that contains elements in the intersection. These indexes will be indistinguishable with the $VIEW$ of P_i as they will be identical.

We can construct the simulator $SIM_A(r,B)$ for the Aggregator as follows: The simulator creates N different sets $\{S_1^{'},\ldots,S_N^{'}\}$ that satisfy the protocol output B of the Aggregator. For each $j\in[|B|]$, let tuple $(b_1,\ldots,b_N)$ be the j^{th} tuple in B, the simulator puts the same random element x_j to all $S_i^{'}$ where $b_i=1$. Then it fills each set with independent uniform random elements until their size is M. The simulator then creates the secret shares and the *Shares* tables for each simulated set, following the protocol description with hash functions parameterized on a random key K'. The simulated *Shares* tables have the same probability distribution of successful reconstructions as the real ones; secret shares do not reveal any information without a successful reconstruction and the indexes of the tables with successful reconstructions are following the same random distribution. Thus, the simulated *Shares* tables are indistinguishable from the real ones. □

Theorem 2. *The collusion-safe protocol described in Section 4.3 is secure in the semi-honest model given that the at least one key holder is not colluding with the Aggregator.*

Proof. The security proofs of the OPR-SS protocol and the OPRF protocol are shown by Mahdavi et al. [34] and Jarecki et al. [22] respectively. Thus, we can assume that no participant or the Aggregator learns anything about the additively shared key given at least one key holder is non-colluding. Hence, the simulator's output distributions remain independent given an honest party with secret key information.

Therefore, we can construct a simulator for a subset of participants colluding with the Aggregator using a simple combination of the simulators described in the previous proof using Goldreich's composition theorem for the semi-honest model [19]. □

6.2 Complexity Analysis

In this section, we provide a theoretical complexity analysis for our OT-MP-PSI protocol. We analyze the computation and communication costs for participants and the Aggregator.

6.2.1 Computational Complexity

Theorem 3. *The computational complexity of the Aggregator for the protocols described in Section 4.3 is* $O(t^2M\binom{N}{t})$.

Proof. The Aggregator has to try $\binom{N}{t}$ different combinations of participants. For each combination, the Aggregator has to do $O(tM)$ Lagrange interpolations. For threshold t, a single Lagrange interpolation has a complexity of $O(t)$. Thus, the total complexity is $O(t^2M\binom{N}{t})$. □

As a corollary of Theorem 3, we can say that the computational complexity for the set intersection problem with two participants $(N=t=2)$ is $O(M)$, and for the case of a threshold equal to the number of participants $(N=t)$ is $O(N^2M)$.

Theorem 4. *The computational complexity of the participants for the non-interactive protocol described in section 4.3 is* $O(tM)$.

Proof. The participants have to create $2M$ secret shares for each subtable because of the second insertion optimization described in Appendix A.2 and $20\cdot2\cdot M$ secret shares in total. The cost of creating a secret share is $O(t)$. Thus, the total complexity is $O(tM)$. □

6.2.2 Communication Complexity

Theorem 5. *The communication complexity of the non-interactive protocol described in Section 4.3 is* $O(tMN)$.

Proof. All the participants have to send their *Shares* tables to the Aggregator. The size of each *Shares* table is $O(tM)$. Thus, with N participants, the total communication complexity is $O(tMN)$. □

Theorem 6. *The communication complexity of the collusion-safe protocol described in Section 4.3 is* $O(tkMN)$ *and it runs in 5 communication rounds.*

Proof. OPR-SS protocol runs in three communication rounds and it has $O(tk)$ communication complexity where k is the number of key holders as shown by Mahdavi et al. [34]. OPRF protocol runs in one communication round and it has constant communication cost. There will be $20\cdot2\cdot MN$ OPR-SS and OPRF invocations in the entire protocol. Therefore, the total communication cost of the protocol will be $O(tkMN)$ All of the OPR-SS invocations can be batched together and after getting the secret shares, all of the OPRF invocations can also be batched. Thus, adding the one additional communication round between the participants and the aggregator, the protocol will run in 5 communication rounds in total. □

6.3 Correctness Evaluation

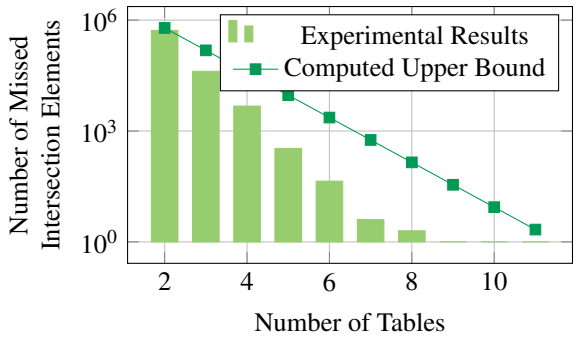

Figure 5: Number of missed Intersection Elements in 10^7 trials ($M = 200, t = 4$)

We experimentally evaluate the failure rate of our hashing scheme to validate our theoretical failure probability analysis. Figure 5 shows the number of missed over-threshold intersection elements out of 10 million trials for different number of tables. For odd number of tables the last table does not have a pair, so the failure probability upper bound is calculated as $(2 \text{ table fail probability})^{(i-1)/2} \times (1 \text{ table fail probability})$ where i is the number of tables. It can be observed that the experimental results are well below the computed upper bounds.

It is not computationally feasible to experimentally show the 2^{-40} lower bound that we claim in Section 5. However, these results are still helpful to show the soundness of our calculations.

6.4 Performance Evaluation

In this section, we provide performance benchmarks for our OT-MP-PSI protocol. The protocol has two distinct phases that occur in sequential order: share creation in the participants and reconstruction in the Aggregator. Thus, we measure their runtime separately using synthetic data. We compare our results with the implementation of Mahdavi et al. [34] as the only other solution with a public code repository at the time of writing. We also evaluate our system using real-world network data obtained from CANARIE IDS Program [10].

6.4.1 Setup

We implement our protocol using Julia language with 430 lines of code. The cryptographic libraries that we use are SHA.jl and Nettle.jl. We use Julia's threads library for parallelization. We used the 61-bit Mersenne prime for the finite field. This is so that we can take advantage of fast modulo arithmetic with Mersenne primes and use 128-bit integers instead of arbitrary precision integers. Our implementation and the benchmark scripts used to generate the graphs in Section 6 are available at https://github.com/onurerenarpaci/Rand-Hashing-OT-MP-PSI. The data collected from the CANARIE IDS Program is not disclosed due to the privacy agreements between institutions.

We use a server with 8 x Intel Xeon E7-8870 processors for all of our experiments. In total, we have 80 physical cores, each running at 2.4 GHz. The server has 1 TB of memory. However, none of our experiments require more than 14 GB of memory. Most of the memory is consumed by parallel Lagrange interpolations, and the degree of parallelism depends on the number of available CPU cores.

6.4.2 Reconstruction

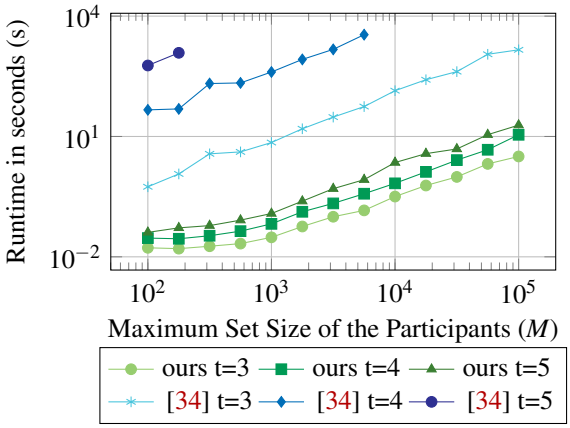

Figure 6: Reconstruction time comparison with Mahdavi et al. [34] ($N = 10$)

We compare the reconstruction time of our protocol with the reconstruction time of Mahdavi et al. [34] in Figure 6. We terminated some of the experiments using Mahdavi et al.'s work [34] because they ran for more than an hour. Our protocol is at least two orders of magnitude faster than Mahdavi et al. [34] for all parameters, and the difference increases exponentially with bigger threshold values.

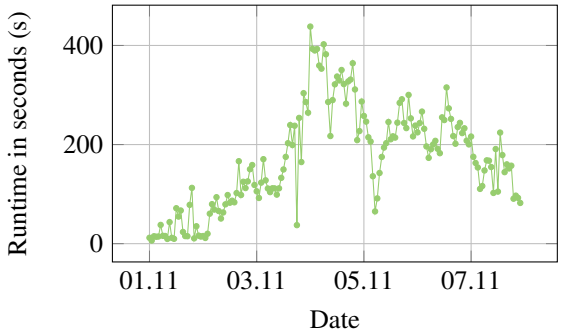

Figure 7: Reconstruction time on CANARIE IDS data [10]

To demonstrate the practicality of our protocol in a real-world setting, we collected network connection logs from 54 institutions via the CANARIE IDS Program for a one-week period (November 1–8, 2023). During this time, the

CANARIE IDS Program ingested approximately 8 billion logs per day, totaling about 700 GB (gzip-compressed) daily. We filtered the logs to only include records where the source was an external IP address and the destination was an internal IP address. Following Zabarah et al. [50], we ran the OT-MP-PSI protocol on hourly batches. For each institution and each hour, we extracted the unique external IP addresses connecting to that institution during that hour to form the participant sets for the protocol. If an institution has no external IPs that start connections in that hour, the institution is not included in the protocol. We set the threshold value to 3 which is the suggested value by Zabarah et al. [50]. The reconstruction time during this week is shown in Figure 7. The maximum reconstruction time during the week is 438 seconds, where 40 institutions participate in the protocol, and the maximum set size is 220,011. The mean and median reconstruction times are 170 and 168 seconds, the mean and median participating institution counts are 33 and 32, and the mean and median maximum set sizes are 144,045 and 162,113 respectively. Since the protocol runs once every hour, we consider the mean/median reconstruction time acceptable for a real-world application.

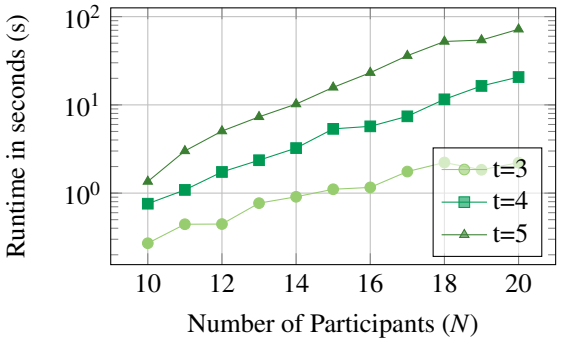

Figure 8: Reconstruction time versus number of participants ($M = 10000$)

We measure the reconstruction time of our protocol for different numbers of participants in Figure 8. The reconstruction time increases polynomially with the number of participants. This is because of the $\binom{N}{t}$ term in our protocol's complexity has an upper bound of $(\frac{eN}{t})^t > \binom{N}{t}$.

We measure the reconstruction time of our protocol for different threshold values in Figure 9. The reconstruction time increases exponentially with the threshold value until $t = N/2$, after which it decreases exponentially. This is because of the $\binom{N}{t}$ term in the complexity of our protocol.

6.4.3 Share Generation

We measure the share generation time of a single participant for different maximum set sizes and different threshold t values with the collusion-safe deployment and the non-interactive deployment in Figure 10. The share generation

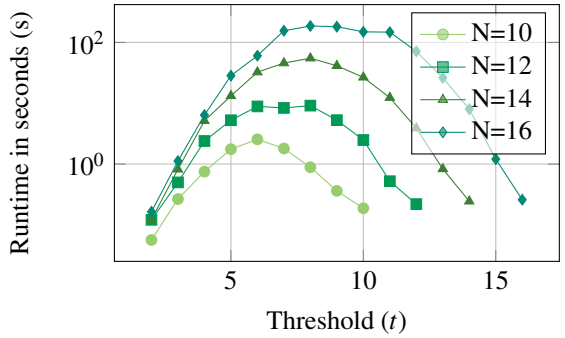

Figure 9: Reconstruction time vs threshold ($M = 10000$)

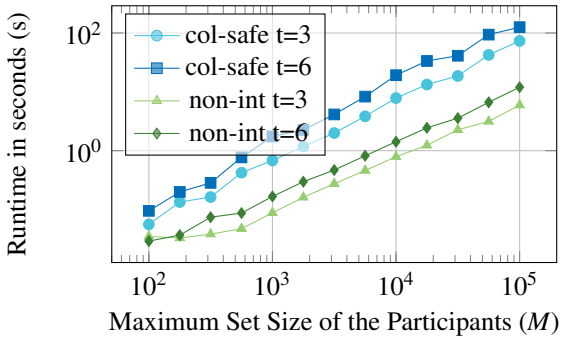

Figure 10: Share generation time of a single participant with the collusion-safe and non-interactive deployment

time increases linearly with the maximum set size, which confirms the $O(tM)$ complexity of the participants. We observe that the collusion-safe deployment is approximately an order of magnitude slower than the non-interactive version.

We also compare the runtimes of share generation process and the reconstruction process in Figure 11. We see that our new-hashing algorithm shifted the bottleneck from reconstruction to share generation.

7 Related Work

We discuss the previous solutions to the OT-MP-PSI problem, some other closely related problems, and related work regarding our new hashing scheme.

7.1 Previous Solutions to OT-MP-PSI

7.1.1 Kissner and Song

Kissner and Song proposed the first solution to the OT-MP-PSI problem [26]. They use polynomial rings to represent multisets. Given a multiset $S = \{S_j\}_{1 \le j \le k}$, they represent it as a polynomial $f(x) = \Pi_{j=1}^{k}(x - S_j)$. With this representation, the union of two sets becomes the product of two polynomials. They use homomorphic encryption to encrypt the polynomials; then, all the participants sequentially multiply their sets

Table 2: Comparison of OT-MP-PSI Solutions

Solution	Comp. Complexity	Comm. Complexity	Comm. Rounds	Collusion Resistance
Kissner and Song [26]	$O(N^3M^3)$	$O(N^3M)$	$O(N)$	up to k collusions
Mahdavi et al. [34]	$O(M(N\log M/t)^{2t})$	$O(tMNk)$	$O(1)$	up to k collusions
Ma et al. [33]	$O(N\|S\|)$	$O(N\|S\|)$	$O(1)$	two non-colluding server
Ours (Non-interactive)	$O(t^2M\binom{N}{t})$	$O(tMN)$	1	non-colluding server
Ours (Collusion-safe)	$O(t^2M\binom{N}{t})$	$O(tMNk)$	$O(1)$	up to k collusions

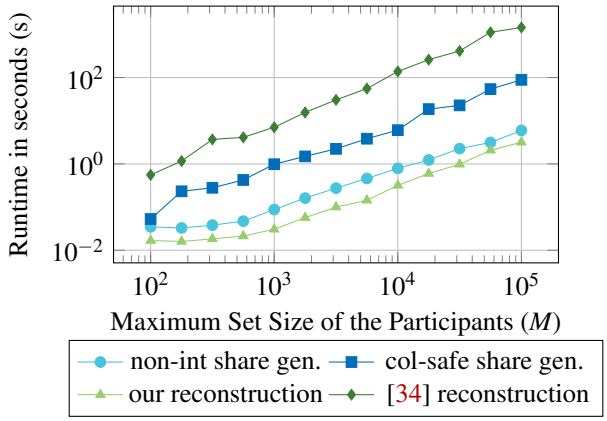

Figure 11: Comparison of reconstruction of the Aggregator and share generation of a single participant (t=3)

to produce a polynomial that represents the union of all sets. Then, they do homomorphic derivative operations on this polynomial to find the elements that exist in at least threshold number of sets. Their solution requires a high number of communication rounds. Specifically, their protocol requires $O(N)$ communication rounds and $O(N^3M)$ communication complexity. Moreover, multiplying the polynomials and evaluating them for each set element has a computational complexity of $O(N^2M^3)$ for each participant, totaling to $O(N^3M^3)$ computation when serialized. There are three reasons why this is worse than our $O(t^2M\binom{N}{t})$ complexity. First, every arithmetic operation is performed with homomorphic encryption, which is computationally expensive. Second, each polynomial multiplication depends on the previous participants' multiplication, making it less parallelizable than our reconstruction process. Finally, in the use cases we are interested, M is much larger than N, i.e, few users with large datasets, resulting in $t^2\binom{N}{t} \ll N^3M^2$. However, if the threshold t is close to $N/2$ and N is large enough, this assumption might not hold.

7.1.2 Mahdavi et al.

Mahdavi et al. [34] introduce a solution with a constant number of rounds and better communication complexity than Kissner and Song [26]. They use Shamir's secret sharing scheme to reveal identical items over the threshold, similar to our solution. They introduce the OPR-SS protocol that we use for our collusion-safe deployment option.

Their scheme has $O(M(N\log M/t)^{2t})$ computational complexity due to the binning technique, which we substantially improve with our new hashing scheme.

7.1.3 Ma et al.

Ma et al. [33] propose a solution designed for use cases involving small input sets and small domain sizes. Their approach employs two mutually non-colluding servers to reduce the OT-MP-PSI problem to a two-party computation (2PC) problem. Uniquely, in this protocol, servers can compute results for multiple thresholds at no extra client cost. However, the main limitation of this solution is its computational and communication complexity, which is $O(N|S|)$, where $|S|$ is the size of the domain of the elements. Therefore, if the inputs come from a large domain, such as the IPv6 addresses in our use case, this solution would not be feasible.

7.1.4 Threshold Private Set Intersection (TPSI)

Although very similarly named, the OT-MP-PSI and TPSI problems are very different. We can observe this similar naming caused some confusion in the existing literature: Mohanty et al. [36] incorrectly classify [6, 26, 34] as TPSI protocols and compare them with other TPSI protocols.

TPSI [51] is a cryptographic problem where two or more participants holding sets of elements want to learn the intersection of their sets if the intersection size is above a certain threshold. Several solutions have been proposed for the TPSI problem [5, 18, 21, 51].

7.1.5 Quorum-PSI

Quorum-PSI, introduced by Chandran et al. [12], is a less general version of the OT-MP-PSI problem. In Quorum-PSI there is a dedicated party P_1 (the only party with output) which has the element in the intersection also in its set, i.e., not all possible subsets of parties that exceed the threshold are considered, but only those that contain P_1. This difference

enables more efficient solutions [12], which are not possible for the OT-MP-PSI problem.

7.2 Related Work Regarding the Hashing Scheme

Although there are no previous works that directly address the OT-MP-PSI problem with a hashing scheme, there are some works that address similar problems in the context of two-party private set intersection (2P-PSI) and multi-party private set intersection (MP-PSI) protocols, which are special cases of OT-MP-PSI where $N = t = 2$ for 2P-PSI and $t = N$ for MP-PSI. Both of these problems have many real-world use cases, such as path discovery in social networks [35], botnet detection [38], web ad campaign performance measurement [44], or cheater detection in games [9]. Our hashing scheme efficiently extends to both problems because our computational complexity $O(t^2M\binom{N}{t})$ becomes $O(M)$ for $N = t = 2$ and $O(N^2M)$ for $t = N$ as shown in Section 6.2.1.

7.2.1 Cuckoo Hashing with Simple Hashing

Cuckoo Hashing. Cuckoo hashing [39] is a technique to avoid hash collisions by assigning each key multiple possible locations using different hash functions. Originally designed for worst-case constant-time lookups, it has since been applied in private protocols such as private set intersection (PSI) [41, 43], private information retrieval (PIR) [1, 2], oblivious RAM (ORAM) [4, 20], and symmetric searchable encryption (SSE) [8, 40]. While many other variants have been proposed [3, 16, 25], the original construction by Pagh and Rodler [39] uses two hash functions h_0, h_1 mapping M elements to two tables T_0, T_1, each with $(1+\varepsilon)M$ bins, storing at most one element per bin. When inserting x into bin $h_b(x)$ in table T_b, if another element y already occupies that bin, y is evicted to $h_{1-b}(y)$ in T_{1-b}, with b toggled each time. The process continues until no collisions remain or a relocation limit is reached.

A naive PSI approach would have both parties, Alice and Bob, use Cuckoo hashing and compare items in corresponding bins. However, an element x might be placed by Alice in $h_0(x)$ but by Bob in $h_1(x)$, and comparing both bins would reveal to Bob that Alice holds an item corresponding to those two bins, breaking privacy. The solution, as used in [17, 42, 44], has Alice use Cuckoo hashing while Bob uses simple hashing, mapping each of his items to both bins $h_0(x)$ and $h_1(x)$. Bob pads each bin with $b = O(\log M / \log\log M)$ elements, resulting in $O(M \log M / \log\log M)$ total comparisons. This construction applies only to two-party protocols.

7.2.2 2D Cuckoo Hashing

Pinkas et al. [43] propose a new variant of Cuckoo hashing called 2D Cuckoo Hashing that reduces the number of comparisons to $O(M)$ for 2P-PSI. The construction involves two tables with two sub-tables in each table. Alice and Bob use different strategies to assign their items to the tables. These strategies are modified versions of the Cuckoo hashing insertion strategy designed in a way to ensure that if Alice and Bob have the same element, there will be a sub-table that has the element in both participants.

Although both our hashing scheme and the 2D Cuckoo hashing achieve $O(M)$ complexity for 2P-PSI, our scheme is more general and can be applied to any number of participants and any threshold t. 2D Cuckoo hashing is specifically designed for 2P-PSI and does not generalize to more than two participants or a threshold t due to Alice and Bob's asymmetric insertion strategy. However, being this specific allows 2D Cuckoo hashing to achieve lower constants in concrete implementations for two parties.

8 Conclusion

In this work, we introduced a practical Over-Threshold Multi-party Private Set Intersection (OT-MP-PSI) protocol designed specifically to support collaborative network intrusion detection. Our protocol efficiently identifies IP addresses appearing across multiple institutions' logs, allowing timely detection of coordinated attacks without compromising privacy.

Previous approaches, such as those by Kissner and Song [26] and Mahdavi et al. [34], either required extensive communication rounds or incurred prohibitive computational costs for large datasets. Our solution addresses these limitations, achieving a computational complexity of $O(t^2M\binom{N}{t})$ through a novel hashing scheme, thus making it practical for real-world network security scenarios.

Our experimental evaluation, using real-world network logs from CANARIE IDS Program [10], demonstrates the practicality and efficiency of our protocol. With an average runtime of just 170 seconds for analyzing data from 33 institutions and datasets containing over 144,000 IP addresses, our solution is well-suited for operational deployment in network security contexts.

In summary, our protocol provides a scalable, privacy-preserving mechanism essential for collaborative intrusion detection. Future work will explore optimizations for efficiently handling participant combinations, further enhancing performance in large-scale deployments.

Acknowledgments

We gratefully acknowledge the support of NSERC for grants RGPIN-2023-03244, RGPIN-06587-2019, IRC-537591, the Government of Ontario, and the Royal Bank of Canada for funding this research, and CANARIE for providing the network logs used in the experiments.

References

[1] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–Computation trade-offs in PIR. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1811–1828. USENIX Association, August 2021.

[2] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 962–979, 2018.

[3] Yuriy Arbitman, Moni Naor, and Gil Segev. De-amortized Cuckoo hashing: Provable worst-case performance and experimental results. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, pages 107–118, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[4] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. OptORAMa: Optimal oblivious RAM. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 403–432, Cham, 2020. Springer International Publishing.

[5] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. In Juan A. Garay, editor, *Public-Key Cryptography – PKC 2021*, pages 349–379, Cham, 2021. Springer International Publishing.

[6] Aslı Bay, Zekeriya Erkin, Jaap-Henk Hoepman, Simona Samardjiska, and Jelle Vos. Practical multi-party private set intersection protocols. *IEEE Transactions on Information Forensics and Security*, 17:1–15, 2022.

[7] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of cryptology*, 14:101–119, 2001.

[8] Angèle Bossuat, Raphael Bost, Pierre-Alain Fouque, Brice Minaud, and Michael Reichle. SSE and SSD: Page-efficient searchable symmetric encryption. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 157–184, Cham, 2021. Springer International Publishing.

[9] Elie Bursztein, Mike Hamburg, Jocelyn Lagarenne, and Dan Boneh. OpenConflict: Preventing real time map hacks in online games. In *2011 IEEE Symposium on Security and Privacy*, pages 506–520, 2011.

[10] Canarie. About us, Jul 2021. https://www.canarie.ca/about/.

[11] Jing Cao, Yu Jin, Aiyou Chen, Tian Bu, and Z-L Zhang. Identifying high cardinality internet hosts. In *IEEE INFOCOM 2009*, pages 810–818. IEEE, 2009.

[12] Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. Efficient linear multiparty PSI and extensions to circuit/quorum PSI. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1182–1204, New York, NY, USA, 2021. Association for Computing Machinery.

[13] Alfonso De Gregorio. Cryptographic key reliable lifetimes: Bounding the risk of key exposure in the presence of faults. In *International Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 144–158. Springer, 2006.

[14] Canada Department of Justice. Personal information protection and electronic documents act. https://laws-lois.justice.gc.ca/eng/acts/P-8.6, 2000.

[15] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016.

[16] Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul Spirakis. Space efficient hash tables with worst case constant access time. *Theory of Computing Systems*, 38(2):229–248, Feb 2005.

[17] Michael J Freedman, Carmit Hazay, Kobbi Nissim, and Benny Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, 2016.

[18] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 3–29, Cham, 2019. Springer International Publishing.

[19] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 1st edition, 2009.

[20] Michael T. Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 576–587, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[21] Per Hallgren, Claudio Orlandi, and Andrei Sabelfeld. PrivatePool: Privacy-preserving ridesharing. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 276–291, 2017.

[22] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 276–291, 2016.

[23] Scott E Jasper. US cyber threat intelligence sharing frameworks. *International Journal of Intelligence and CounterIntelligence*, 30(1):53–65, 2017.

[24] N. Kamiyama, T. Mori, and R. Kawahara. Simple and adaptive identification of superspreaders by flow sampling. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 2481–2485, 2007.

[25] Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. More robust hashing: Cuckoo hashing with a stash. *SIAM Journal on Computing*, 39(4):1543–1561, 2010.

[26] Lea Kissner and Dawn Song. *Private and threshold set-intersection*. School of Computer Science, Carnegie Mellon University, 2004.

[27] California State Legislature. California consumer privacy act of 2018. https://oag.ca.gov/privacy/ccpa, 2018.

[28] Wenjuan Li, Weizhi Meng, and Lam For Kwok. Surveying trust-based collaborative intrusion detection: State-of-the-art, challenges and future directions. *IEEE Communications Surveys & Tutorials*, 24(1):280–305, Firstquarter 2022.

[29] Patrick Lincoln, Phillip Porras, and Vitally Shmatikov. Privacy-preserving sharing and correction of security alerts. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 17, USA, 2004. USENIX Association.

[30] Yehuda Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. *Journal of Cryptology*, 29(2):456–490, 2016.

[31] Yang Liu, Wenji Chen, and Yong Guan. Identifying high-cardinality hosts from network-wide traffic measurements. *IEEE Transactions on Dependable and Secure Computing*, 13(5):547–558, 2016.

[32] M.E. Locasto, J.J. Parekh, A.D. Keromytis, and S.J. Stolfo. Towards collaborative security and P2P intrusion detection. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 333–339, June 2005.

[33] Liju Ma, Hao Wang, Ziyu Niu, Zhi Li, Lei Wu, Xiaochao Wei, and Ye Su. Over-threshold multi-party private set operation protocols for lightweight clients. *Computer Standards & Interfaces*, 88:103781, 2024.

[34] Rasoul Akhavan Mahdavi, Thomas Humphries, Bailey Kacsmar, Simeon Krastnikov, Nils Lukas, John A Premkumar, Masoumeh Shafieinejad, Simon Oya, Florian Kerschbaum, and Erik-Oliver Blass. Practical over-threshold multi-party private set intersection. In *Annual Computer Security Applications Conference*, pages 772–783, 2020.

[35] Ghita Mezzour, Adrian Perrig, Virgil Gligor, and Panos Papadimitratos. Privacy-preserving relationship path discovery in social networks. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security*, pages 189–208, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[36] Tapaswini Mohanty, Vikas Srivastava, Sumit Kumar Debnath, Ashok Kumar Das, and Biplab Sikdar. Quantum secure threshold private set intersection protocol for IoT-enabled privacy-preserving ride-sharing application. *IEEE Internet of Things Journal*, 11(1):1761–1772, 2024.

[37] Daniel Morales, Isaac Agudo, and Javier Lopez. Private set intersection: A systematic literature review. *Computer Science Review*, 49:100567, 2023.

[38] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *19th USENIX Security Symposium (USENIX Security 10)*, Washington, DC, August 2010. USENIX Association.

[39] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

[40] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 79–93, New York, NY, USA, 2019. Association for Computing Machinery.

[41] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 739–767, Cham, 2020. Springer International Publishing.

[42] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, Washington, D.C., August 2015. USENIX Association.

[43] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via Cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 125–157, Cham, 2018. Springer International Publishing.

[44] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 797–812, USA, 2014. USENIX Association.

[45] Davy Preuveneers and Wouter Joosen. Privacy-preserving correlation of cross-organizational cyber threat intelligence with private graph intersections. *Computers & Security*, 135:103505, 2023.

[46] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[47] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979.

[48] Jan Stanek, Alessandro Sorniotti, Elli Androulaki, and Lukas Kencl. A secure data deduplication scheme for cloud storage. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 99–118, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[49] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[50] Saif Zabarah, Omar Naman, Mohammad A Salahuddin, Raouf Boutaba, and Samer Al-Kiswany. An approach for detecting multi-institution attacks. *Annals of Telecommunications*, pages 1–14, 2023.

[51] Yongjun Zhao and Sherman S.M. Chow. Can you find the one for me? In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, WPES'18, page 54–65, New York, NY, USA, 2018. Association for Computing Machinery.

[52] Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. Evaluation of a decentralized architecture for large scale collaborative intrusion detection. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 80–89. IEEE, 2007.

[53] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *computers & security*, 29(1):124–140, 2010.

A Hashing Scheme Additional Optimizations

A.1 Reversing the ordering

Instead of using a unique ordering hash function for each table, using the same ordering hash function for every two consecutive tables and reversing the ordering for even-numbered tables gives better results. The intuition behind this approach is that we know the "unlucky" elements in the odd-numbered tables; these are elements with big p values. By reversing the order and making them "lucky" in the next consecutive table, we get better probabilities than re-randomizing for each table.

With this technique, the value of p for a particular element will be equal to $1-p$ in the next table. Therefore, the probability of missing a particular intersection in two consecutive tables is given by

$$P(\text{fail with two tables} \mid p) \leq (1-e^{-p})(1-e^{-(1-p)})$$

We then calculate the probability of failure for any intersection similarly to the single table calculation.

$$\begin{aligned} P(\text{fail with two tables}) &\leq \int_0^1 (1-e^{-p})(1-e^{-(1-p)})dp \\ &= 3e^{-1}-1 \approx 0.10363 \end{aligned}$$

This technique reduces the required number of tables to 26 to obtain $(3e^{-1}-1)^{13} \approx 2^{-42.5}$ failure probability.

A.2 Utilizing the empty bins

Normally, after inserting their elements into the tables, the participants would fill the empty bins with dummy shares. However, we can utilize this wasted space to reduce the number of required tables. With this optimization, participants will do a second insertion after the first insertion for every table. They will use a different mapping hash function h' for the second insertion and reverse the ordering hash function from the first insertion. While doing the second insertion, elements from the first insertion have priority in the table; in other words, if an element maps to an already occupied position, we do not make any changes to the table.

Therefore, two events must occur for an element to be successfully put into the table in the second insertion: it must map into an empty bin and be the smallest element among the elements that map into the same bin. We already know the probability of the second event occurring in t different sets; it is $e^{-(1-p)}$ due to reverse ordering. To calculate the probability of the first event, we first calculate the probability of any bin being empty after the first insertion. We do this by

calculating the probability of a single element not mapping to that bin repeated M times. Again, the approximation does not affect the results for $M > 50$.

$$P(shares_i[x] \text{ is empty}) = \left(\frac{tM-1}{tM}\right)^{|S_i|} \geq \left(\frac{tM-1}{tM}\right)^{M} \approx e^{-1/t}$$

We then calculate the probability of $s_{i,j}$ being mapped into empty bins in all t sets in the second insertion, given that $s_{i,j}$ exists in t different sets.

$$P\Big(shares_k[h'(s_{i,j})] \text{ is empty } \forall_{k\in[t]} \mid s_{i,j} \in S_k \ \forall_{k\in[t]}\Big) \geq \left(e^{-1/t}\right)^t = e^{-1}$$

Now, we multiply these two probabilities to find the second insertion success probability of a particular intersection.

$$P(\text{success in second insertion} \mid p) \geq e^{-(1-p)} \times e^{-1} = e^{p-2}$$

Lastly, the probability of missing a particular intersection in a single table is given by the failure probability of the first insertion multiplied by the failure probability of the second insertion. Afterward, we calculate the failure probability for any intersection.

$$P(\text{fail} \mid p) \leq (1-e^{-p})(1-e^{p-2})$$

$$P(\text{fail}) \leq \int_0^1 (1-e^{-p})(1-e^{p-2})dp = 2e^{-2} \approx 0.2706$$

This approach reduces the required number of tables to 22 to have $\left(2e^{-2}\right)^{22} = 2^{-41.5}$ failure probability. We also explored the strategy of doing multiple insertions until the table is filled. However, for use cases where t is significantly smaller than M this approach results in insignificant improvements.

Finally, we can get better results by combining the two optimizations; more explicitly, we will do second insertions for every table and order reversing for every two consecutive tables. In this case, the failure probability for a particular intersection in two consecutive tables is given by

$$P(\text{fail with 2 tables} \mid p) \leq \underbrace{(1-e^{-p})(1-e^{p-2})}_{\text{first table}} \ \underbrace{(1-e^{-(1-p)})(1-e^{-p-1})}_{\text{second table}}$$

We then calculate the failure probability for any intersection

$$P(\text{fail with 2 tables}) \leq \int_0^1 (1-e^{-p})(1-e^{p-2})(1-e^{-(1-p)})(1-e^{-p-1})dp = 2e^{-1} + 2e^{-2} + 3e^{-4} - 1 \approx 0.06138$$

The combination of these two approaches reduces the number of required tables to 20 to obtain a failure probability of $(0.06138)^{10} = 2^{-40.3}$.