# Practical Over-Threshold Multi-Party Private Set Intersection

Rasoul Akhavan Mahdavi
University of Waterloo
Canada
r5akhava@uwaterloo.ca

Thomas Humphries
University of Waterloo
Canada
thomas.humphries@uwaterloo.ca

Bailey Kacsmar
University of Waterloo
Canada
bkacsmar@uwaterloo.ca

Simeon Krastnikov
University of Waterloo
Canada
skrastnikov@uwaterloo.ca

Nils Lukas
University of Waterloo
Canada
nlukas@uwaterloo.ca

John A. Premkumar
University of Waterloo
Canada
jpremkumar@uwaterloo.ca

Masoumeh Shafieinejad
University of Waterloo
Canada
masoumeh@uwaterloo.ca

Simon Oya
University of Waterloo
Canada
simon.oya@uwaterloo.ca

Florian Kerschbaum
University of Waterloo
Canada
florian.kerschbaum@uwaterloo.ca

Erik-Oliver Blass
Airbus
Germany
erik-oliver.blass@airbus.com

## ABSTRACT

Over-Threshold Multi-Party Private Set Intersection (OT-MP-PSI) is the problem where several parties, each holding a set of elements, want to know which elements appear in at least $t$ sets, for a certain threshold $t$, without revealing any information about elements that do not meet this threshold. This problem has many practical applications, but current solutions require a number of expensive operations exponential in $t$ and thus are impractical.

In this work we introduce two new OT-MP-PSI constructions using more efficient techniques. Our more refined scheme, which we call `t-PSI`, runs in three communication rounds. `t-PSI` achieves communication complexity that is linear in the number of parties, the number of elements they hold, and the intersection threshold. The computational cost of `t-PSI` is still exponential in $t$, but it relies on cheap linear operations and thus it is still practical. We implement our new constructions to validate their practicality for varying thresholds, number of parties, and dataset size.

## CCS CONCEPTS

• **Security and privacy → Management and querying of encrypted data**;

## KEYWORDS

private set intersection, homomorphic encryption, oblivious pseudorandom functions

## 1 INTRODUCTION

Two-party private set intersection (PSI) is a widely studied problem of computing which elements two participants have in common without revealing anything outside of the intersection [7–9, 12, 17, 21, 23, 24]. PSI solutions have been practically deployed for different use cases, such as ad conversions [29]. Multi-party private set intersection schemes, where there are more than two participants' sets to compute over, also exist, but without practical deployments [5, 8, 10, 16, 17].

| | Comm. Rounds | Comm. Complexity |
|---|---|---|
| Kissner & Song [15] | $O(m)$ | $O(nm^3)$ |
| Secret-Shared MPC | $O(\log^2 nm)$ | $O(nm^3 \log^2 nm + nm^3 t)$ |
| Our work (`t-PSI`) | $O(1)$ | $O(nmtk)$ |

**Table 1: Communication costs for different OT-MP-PSI protocols where $m$ is the number of participants, $n$ is the size of each database, $t$ is the threshold for subset size, and $k$ is the collusion threshold (number of key holders).**

We consider the following use case for the multi-party set intersection, where network operation centers collaborate to identify common threats. These centers collect attack information as indicators of compromise [4, 26]. These indicators are comparable [13, 14] and the centers may want to determine whether they are subject to a common attack. For example, Wagner et al. [28] report sharing about 4,000 indicators over a period of half a year using a central platform. However, exchanging indicators of compromise without probable cause is often not an option, since these indicators contain privacy-protected information, such as IP addresses [18]. The situation is often complicated by different legislations governing those operation centers. Multi-party PSI allows the data centers to find which indicators they have in common without revealing those indicators not in the intersection. However, computing which indicators of compromise are owned by *all* data centers might be too restrictive, since an attack is worth investigating, even if it affects some but not all of the centers.

In such cases, a related problem, Over-Threshold Multi-Party Private Set Intersection (OT-MP-PSI) [15] applies. For $m$ participants, each with their own dataset of at most size $n$, the goal of OT-MP-PSI is to determine the elements (and their owners) that occur in at least $t$ datasets, but reveal nothing else as long as no more than a defined maximum number of participants collude. Kissner and Song previously proposed a protocol for this problem, however, its high communication and computation complexity make it impractical. Specifically, their protocol requires $O(m)$ rounds and has a communication complexity of $O(nm^3)$. Furthermore, although existing multi-party private set intersection schemes can be leveraged to compute an OT-MP-PSI, by performing the intersections of all subgroups of size $t$, this approach is extremely expensive, since for $m > t$ participants there are $\binom{m}{t}$ possible subgroups of size $t$. For example, we could design a OT-MP-PSI protocol using a secret-shared multi-party protocol such as SCALE-MAMBA [1]. However, to do so for multi-party computation using a circuit similar to the one used in PSI based on two-party computation [11], the optimal circuit size is $O(nm \log^2 nm)$ for $O(n + \log^2 nm)$ rounds or $O(nm \log^2 nm + nmt)$ for $O(\log^2 nm)$ rounds.

In this paper, we present two different constructions of OT-MP-PSI that are more efficient than previous proposals (see Table 1 for an overview of round and communication complexities of our scheme compared to related schemes). Our protocol uses oblivious pseudo-random functions (OPRF) and hashing to achieve efficiency for the OT-MP-PSI problem. These techniques are commonly used in efficient two-party PSI protocols [9, 17, 23], but have not yet been applied to OT-MP-PSI. While the $O(n(m \log n/t)^{2t})$ computational complexity of our reconstruction phase is exponential in the threshold $t$, we take care to keep the constants low in order to scale to practical sizes for the envisioned use case of indicators of compromise exchange. Our protocol is very flexible with tunable parameters for the threshold $t$ and the collusion threshold $k$, where $k$ is the number of key holders.

We first consider the case where there is a single non-colluding key holder and present a strawman scheme ($\texttt{t-PSI}_0$) with communication complexity $O(nm)$ that runs in two communication rounds. However, the constants in reconstruction prevent scaling to practically relevant problem sizes. Then, we present an improved scheme ($\texttt{t-PSI}$) that runs in three communication rounds with a communication complexity $O(nmt)$, but that is based on significantly cheaper operations and thus is faster than our strawman variant. We extend this scheme to the case where there are $k$ key holders with the assumption that the keyholders do not collude with one another. Extending to multiple key holders results in increases in communication complexity to $O(nmk)$ and $O(nmtk)$ for $\texttt{t-PSI}_0$ and $\texttt{t-PSI}$, respectively. We implement and evaluate our protocol, showing that the reconstruction phase is feasible for $m = 10$ participants and a threshold of $t = 6$. Our construction features a new primitive called Oblivious Pseudo-Random Secret Sharing (OPR-SS), which leverages oblivious pseudo-random functions to obliviously generate pseudo-random shares of a secret. This primitive may be of independent interest in applications beyond OT-MP-PSI.

In summary, the contributions of this paper are:

(1) a new OT-MP-PSI protocol with communication complexity $O(nmk)$ and round complexity $O(1)$.
(2) a new OT-MP-PSI protocol with communication complexity $O(nmtk)$, round complexity $O(1)$, and low constants during the reconstruction phase.
(3) a new primitive, OPR-SS, which allows a participant to generate pseudo-random shares of a secret with the help of a key holder, that remains oblivious to the generated shares.
(4) a practical implementation and evaluation of our new OT-MP-PSI protocol.

The remainder of the paper is structured as follows: Section 2 covers relevant background information required for the protocols we introduce in Section 3. Section 4 contains our security proofs. We present our evaluation in Section 5 followed by a review of related work in Section 6. Finally, Section 7 concludes our work.

## 2 BACKGROUND

We briefly summarize the foundations of our over-threshold intersection schemes, secret sharing schemes and oblivious pseudo-random functions, as well as the Paillier cryptosystem.

### 2.1 Shamir's Secret Sharing

The goal of secret sharing is to distribute $m$ shares of a secret $S$ such that an appropriate subset of the shares can recover the secret. A $(t, m)$-*threshold scheme* ensures that any subset of $t$ parties can together *reconstruct* the secret $S$ from their shares, but no subset of fewer than $t$ parties can collude to infer any information about $S$.

In Shamir's secret sharing [27], the distributing party generates $t - 1$ values $\{c_i\}_{i \in [t-1]}$ chosen at random from some finite field $\mathbb{F}_p$ of prime order $p$, and forms the polynomial

$$f(x) = c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \dots c_1 x + S\,. \tag{1}$$

The distributing party generates $m$ shares by evaluating $f$ at $m$ publicly-known distinct values. For instance, the share for party $i$ (with $i \in \mathbb{F}_p$) is $s_i = (i, f(i))$. Since $t$ points uniquely determine a polynomial of degree $t - 1$, anyone possessing $t$ shares $s_i$ can recover the secret $S$ by means of Lagrange interpolation, i.e.,

$$S = \sum_{i \in [t]} \left[ s_i \cdot \prod_{\substack{j \in [t] \\ j \neq i}} \frac{-j}{i - j} \right]. \tag{2}$$

However, fewer than $t$ shares reveal no information about $S$.

### 2.2 Oblivious Pseudo-Random Functions

Let $F$ be a Pseudo-Random Function (PRF), which takes as inputs an element $\ell$ and a secret key $\mathsf{sk}_{PRF}$ and outputs $y = F(\ell, \mathsf{sk}_{PRF})$. Formally, we define the security of a PRF using a game $\mathrm{Game}^{\mathsf{PRF}}$ between an adversary and a challenger (see Figure 1). In this game, the challenger first generates a secret key $\mathsf{sk}_{PRF}$ and public key $\mathsf{pk}_{PRF}$ according to the security parameter $\lambda$. The adversary $\mathcal{A}$ is given access to $\mathsf{pk}_{PRF}$, oracle access to the PRF $F(\cdot, \mathsf{sk}_{PRF})$ and is free to choose any input (to $F$) $\ell' \neq \ell$. The adversary outputs $\ell$ to the challenger. The challenger computes $y_0 = F(\ell, \mathsf{sk}_{PRF})$, chooses $y_1$ randomly, and then with equal probability provides the adversary with either $y_0$ or $y_1$. The adversary succeeds if they correctly distinguish one of these two cases. We say that $F$ is secure if for any probabilistic polynomially bounded adversary $\mathcal{A}^F$, the

```
$\mathsf{Game}^{\mathsf{PRF}}$
$(\mathsf{sk}_{PRF}, \mathsf{pk}_{PRF}) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathsf{st}, \ell) \leftarrow \mathcal{A}^{F(\ell' \neq \ell, \mathsf{sk}_{PRF})}(\mathsf{pk})$
$y_0 \leftarrow F(\ell, \mathsf{sk}_{PRF})$
$y_1 \leftarrow\$ \mathbb{G}$
$b \leftarrow\$ 0, 1$
$b' \leftarrow \mathcal{A}^{F(\ell' \neq \ell, \mathsf{sk}_{PRF})}(y_b, \mathsf{st})$
return $b = b'$
```

**Figure 1: Game$^{\mathsf{PRF}}$ between an adversary and a challenger.**

probability that $\mathcal{A}^F$ succeeds in Game$^{\mathsf{PRF}}$ is at most $1/2 + \mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda) \in 2^{-\Omega(\lambda)}$.

An Oblivious Pseudo-Random Function (OPRF) is a protocol between a key holder that holds a secret key $\mathsf{sk}_{PRF}$ and a participant that holds an input $\ell$, where the participant learns $F(\ell, \mathsf{sk}_{PRF})$ without learning anything about $\mathsf{sk}_{PRF}$ or the value of $F(\ell', \mathsf{sk}_{PRF})$ for other inputs $\ell'$, and the key holder does not learn anything about $\ell$ nor $F(\ell, \mathsf{sk}_{PRF})$. OPRF's can be established through generic methods for secure multiparty computation (on top of circuits that implement ordinary pseudo-random functions), or by means of the Diffie-Hellman assumption (see Section 4).

## 2.3 Paillier Cryptosystem

The Paillier cryptosystem [22] is an additively homomorphic scheme for public key encryption based on the intractability of the Composite Residuosity Class Problem. The ciphertexts are elements over the multiplicative field $\mathbb{F}_{N^2} = \mathbb{Z}^*_{N^2}$, where $N = p'q'$ for primes $p'$ and $q'$. Letting $\mu$ be the least common multiple of $p' - 1$ and $q' - 1$, and choosing $g$ to be a random base such that its order is divisible by $N$, the public key is $(N, g)$ and the private key is $(p', q')$. Encrypting a plaintext message $0 \leq \ell < N$ is done by choosing a random $r < N$ and computing:

$$Enc(\ell) = g^\ell r^N \mod N^2 \,. \tag{3}$$

A given ciphertext $C < N^2$ is decrypted using $\mu$:

$$Dec(C) = \frac{L(C^\mu \bmod N^2)}{L(g^\mu \bmod N^2)} \mod N \,, \tag{4}$$

where

$$L(\ell) = \frac{\ell - 1}{N} \,. \tag{5}$$

Such a scheme satisfies the following homomorphic properties (the moduli are implicit):

$$Dec(Enc(\ell_1)Enc(\ell_2)) = \ell_1 + \ell_2 \,,$$
$$Dec(Enc(\ell_1)^{\ell_2}) = \ell_1 \ell_2 \,.$$

The public key $\mathsf{pk}_{HE}$ is $N$ and $g$ and the secret key $\mathsf{sk}_{HE}$ is $\mu$.

## 3 PROTOCOL DESCRIPTION

In this section, we present two schemes to compute an over-threshold multi-party private set intersection (OT-MP-PSI). First, we introduce our notation and assumptions. Second, we present a new protocol that we use in our schemes that we call Oblivious Pseudo Random Secret Sharing (OPR-SS), and third we present our schemes `t-PSI`$_0$ and `t-PSI`, which differ in how they implement the OPR-SS. We initially explain our schemes for a simplified scenario, and later explain how to generalize them. We summarize all relevant notation for our protocol in Table 2.

### 3.1 Protocol Overview

In our protocol, there are three entity types. First, there are $m$ participants, each denoted by an index $i \in [m]$. Each participant $i$ holds a set of elements $\mathbb{L}_i$, and we use $n$ to denote the maximum set size $n \doteq \max_{i \in [m]} |\mathbb{L}_i|$. The set of all elements across all participants is $\mathbb{L}$. Additionally, there are $k$ key holders and $r$ reconstructors. A participant can simultaneously be a key holder, a reconstructor, both or neither. The goal of the OT-MP-PSI protocol is to return which elements appear at least $t$ times among the group and who the owners of those elements are, where $t$ is the intersection *threshold.*

We assume that all participants are semi-honest, i.e., they follow the protocol specifications, but might try to infer private information from other participants based on passive observation. We also assume that, when there is a single key holder, the key holder is not in the set of reconstructors and no reconstructor colludes with the key holder. Note that we do not require this assumption in the case of multiple keyholders. That is, when using multiple keyholders, it is possible for a keyholder to be one of the reconstructors.

In the case where $k > 1$ and $r > 1$, we require that at least one key holder does not collude with any reconstructor, and that no set of $t - 1$ colluding parties includes a reconstructor.

The three entities we define for our protocol can all be understood as roles that participants undertake. There are rules about non-collusion among the participants, and additional non-collusion rules for a single keyholder case, however there are no "trusted third parties". Our protocol is configurable through the roles participants take in terms of whether there are multiple or single reconstructors and keyholders. However, for simplicity, we first explain our schemes in the case where there is one key holder and one reconstructor. Later, in Sections 3.5 and 3.6 we explain how to extend them to the multi-reconstructor and multi-key holder cases, respectively. Both our schemes rely on a new protocol that we call Oblivious Pseudo-Random Secret Sharing (OPR-SS) (defined in Section 3.2). We use OPR-SS to implement OT-MP-PSI protocols, as follows:

(a) Each participant $i \in [m]$ engages with the key holder in an OPR-SS protocol to generate element dependent shares of a fixed secret $S = 0$ for each of their elements $\ell \in \mathbb{L}_i$ while keeping the key holder oblivious to the elements and the shares.
(b) After this, each participant stores the shares of each of their elements $\ell \in \mathbb{L}_i$ in a hash table with $b$ bins. The bins in each table are denoted $b_j$, with $j \in [b]$.
(c) The participants pad all of their bins to a pre-determined size $|b|_{\max}$ with dummy elements, and send them to the reconstructor.
(d) For each bin, the reconstructor builds $t$-sized subsets of secrets from different users, and tries to recover a secret. If a certain subset yields the pre-determined secret, $S = 0$, this

**Table 2: Notation**

| | |
|---|---|
| **General Parameters** | |
| $\lambda$ | Security parameter |
| $m$ | Number of participants, indexed by $i \in [m]$. |
| $k$ | Number of key holders. |
| $r$ | Number of reconstructors. |
| $n$ | Maximum elements owned by a participant. |
| $t$ | Threshold for the intersection. |
| $\mathbb{L}_i$ | Set of elements owned by participant $i$; $\lvert\mathbb{L}_i\rvert \leq n$. |
| $\mathbb{L}$ | Set of all elements: $\mathbb{L} = \bigcup_{i\in[m]} \mathbb{L}_i$. |
| $b$ | Total number of bins. |
| $\lvert b\rvert_{\mathsf{max}}$ | Maximum bin size allowed. |
| $H_B(\cdot)$ | Hash function used in hashing-to-bins. |
| **OPR Secret Sharing** | |
| $H(\cdot)$ | Hash function used in the OPR-SS scheme. |
| $\mathbb{G}_q$ | Group of the PRF. |
| $\mathbb{F}_p$ | Base prime field in the OPR-SS scheme. |
| $s_i^\ell$ | Share of the element $\ell$ for participant $i$. |
| $S$ | Secret (we use the secret $S = 0$ for validation). |
| **Homomorphic Encryption Scheme** | |
| $\mathbb{F}_N$ | Plaintext field. |
| $\mathbb{F}_{N^2}$ | Ciphertext field. |
| $Enc_{pk_{HE}}$ | Encryption with public key $pk$. |
| $Dec_{sk_{HE}}$ | Decryption with private key $sk$. |

means that all of the elements in this subset correspond to the same element.

(e) The reconstructor communicates to each participant which of their elements were also owned by at least $t - 1$ other participants.

## 3.2 Oblivious Pseudo-Random Secret Sharing

Both of our OT-MP-PSI schemes rely on a new protocol that we call Oblivious Pseudo-Random Secret Sharing (OPR-SS). This protocol combines the share generation and reconstruction properties of Secret Sharing (SS), and the security properties of OPRFs. Formally, an OPR-SS is a protocol between two parties: a key holder, that holds a secret key $\mathsf{sk}_{PRF}$ and a value $S$, and a set of participants that each hold inputs from $\mathbb{L}$. The protocol allows each participant $i$ to generate shares $s_i^\ell(S)$ of $S$ given an element $\ell \in \mathbb{L}$, such that the key holder remains oblivious to the input $\ell$ and participants remain oblivious about $\mathsf{sk}_{PRF}$. Additionally, the generated shares meet certain reconstruction properties for a threshold $t$ even though the key holder chooses the $\mathsf{sk}_{PRF}$, and value $S$. More precisely, a $t$-OPR-SS should fulfill the following security definition with respect to the obliviousness of the share generation process for the keyholder and any participant $i$.

*Definition 3.1.* We say a $t$-OPR-SS protocol is secure in the semi-honest model if there exist two simulators $\mathsf{Sim}_{Keyholder}(\mathsf{sk}_{PRF})$ and $\mathsf{Sim}_{Participant_i}(\ell, s_i^\ell(S))$, such that their outputs are computationally indistinguishable (denoted $\overset{c}{\equiv}$) from the respective views $\mathsf{VIEW}^{OPR-SS}_{Keyholder}(\mathsf{sk}_{PRF}, \ell)$ and $\mathsf{VIEW}^{OPR-SS}_{Participant_i}(\mathsf{sk}_{PRF}, \ell)$ of their

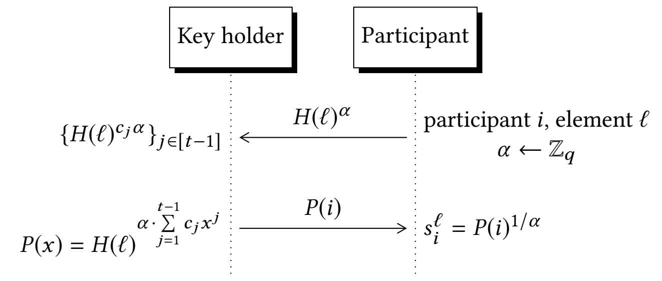

Key holder
Participant
$\{H(\ell)^{c_j\alpha}\}_{j\in[t-1]}$
$H(\ell)^\alpha$
participant $i$, element $\ell$
$\alpha \leftarrow \mathbb{Z}_q$
$P(x) = H(\ell)^{\alpha\cdot\sum_{j=1}^{t-1} c_j x^j}$
$P(i)$
$s_i^\ell = P(i)^{1/\alpha}$

**Figure 2: Generation of share $s_i^\ell$ of element $\ell$ owned by participant $i$ in $\mathtt{t\text{-}PSI}_0$.**

parties during a real OPR-SS protocol, i.e.

$$\mathsf{Sim}_{Keyholder}(\mathsf{sk}_{PRF}) \overset{c}{\equiv} \mathsf{VIEW}^{OPR-SS}_{Keyholder}(\mathsf{sk}_{PRF}, \ell)$$
$$\mathsf{Sim}_{Participant_i}(\ell, s_i^\ell(S)) \overset{c}{\equiv} \mathsf{VIEW}^{OPR-SS}_{Participant_i}(\mathsf{sk}_{PRF}, \ell)$$

In addition to the above security definition, an OPR-SS must satisfy a correctness property for reconstruction. Let $\mathsf{Recon}(\cdot)$ be the reconstruction function for a set of secret shares. The correctness of the OPR-SS protocol depends on the threshold parameter $t$. Formally, the shares $s_i^\ell(S)$ generated by the $t$-OPR-SS protocol fulfill the following correctness properties:

- $\mathsf{Recon}(s_{i_1}^\ell(S), \ldots, s_{i_t}^\ell(S)) = S$
- For all sets $\{\ell_1, \ldots, \ell_t\}$, such that there exist at least two different elements $\ell_j \neq \ell_{j'}$:

$$Pr[\mathsf{Recon}(s_{i_1}^{\ell_1}(S), \ldots, s_{i_t}^{\ell_t}(S)) = S] \leq \mathsf{negl}(\lambda)$$

In our OT-MP-PSI protocols, we set $S$ as a (non-secret) constant ($S = 0$). This allows us to use the reconstruction property to validate whether or not a set of shares were generated with some identical element $\ell$ (i.e., by validating whether a set of shares recover $S = 0$). In the remainder of this paper, for simplicity, we use $s_i^\ell$ to denote $s_i^\ell(0)$. However, in other applications, it may be appropriate to use OPR-SS with $S$ as an actual secret.

## 3.3 Strawman Scheme: $\mathtt{t\text{-}PSI}_0$

We now formally explain our first OT-MP-PSI scheme, that we denote by $\mathtt{t\text{-}PSI}_0$. We use $\mathbb{G}_q$ to denote a group of size $q$ and $\mathbb{F}_p$ to denote a field of size $p$, where $p = 2q + 1$ and $\mathbb{G}_q$ is a subgroup of $\mathbb{F}_p$. We use $H(\cdot)$ to denote a hash function that maps values from a binary string to $\mathbb{G}_q$.

*3.3.1 Share generation.* To initialize the protocol, the key holder performs a one time generation of a set of $t - 1$ random numbers $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$ to use across all participants and elements.

Participant $i$ generates a random number $\alpha$ to obliviously send an element's hash, $H(\ell)$, to the key holder. The key holder forms a polynomial, as in Shamir's secret sharing, with coefficients $\{c_j\}_{j\in[t-1]}$ and whose constant term is $S = 0$. Then, it evaluates this polynomial in the participant's identifier ($i$) and appends this value in the exponent of the masked hash $H(\ell)^\alpha$. The key holder returns this value to the participant, who unmasks it (by raising to $1/\alpha$) and the resulting value is the share $s_i^\ell$. Figure 2 illustrates this process. This is repeated to generate $s_i^\ell$, for all $i \in [m]$ and $\ell \in \mathbb{L}_i$.

Note that we are using Shamir's secret sharing scheme in a slightly different style than its conventional use. In our scheme, the goal is not to reconstruct the constant term $P(0)$ as the secret, but to use $P(0) = 0$ as a confirmation that different shares used to reconstruct a polynomial actually belong to the same element $\ell$. This allows the reconstructor to verify integrity without actually recovering the element.

*3.3.2 Hashing-to-bins.* Each participant $i$ stores their shares in a hash table containing $b$ bins. Each share $s_i^\ell$ (for $\ell \in \mathbb{L}_i$) is placed in the bin indexed by $j = H_B(\ell)$, where $H_B(\cdot)$ is the binning hash function. We choose $b = \beta \cdot n/\log n$, which guarantees that after storing $n$ elements in the hash table, the size of a bin never exceeds $|b|_{\mathsf{max}} = \beta_{\mathsf{m}} \cdot \log n$ with overwhelming probability. The constants $\beta$ and $\beta_{\mathsf{m}}$ are a system-wide fixed parameter (see Section 5.4 for the performance trade-offs of tuning these constants). After all elements are stored into the appropriate bins, participants add dummy elements until each bin reaches the maximum size $|b|_{\mathsf{max}}$. Padding hides the number of real elements each participant stored in their table.

*3.3.3 Reconstruction.* Every participant sends their hash tables to the reconstructor. The reconstructor picks a bin location $j \in [b]$. Then, it chooses $t$ of the $m$ participants and forms sets of $t$ shares by taking exactly one share from bin $b_j$ from each of the chosen participants. The reconstructor verifies if the polynomial defined by these shares passes through the point $(0, 0)$ by performing Lagrange interpolation in the exponent. For example, let $\mathcal{S}$ be the set of $t$ participants for which the reconstructor wishes to perform the verification, and let $\{s_i\}_{i\in\mathcal{S}}$ be the chosen shares from each of these participants. The reconstructor computes:

$$1 = \prod_{i\in\mathcal{S}} s_i^{\left(\prod_{j\in\mathcal{S}\setminus i} \frac{-j}{i-j}\right)} . \quad (6)$$

If the above equality holds, then the selected shares were generated from the same element. Note that the reconstructor still does not learn the element $\ell$ (unless the reconstructor is a participant and has $\ell$ within their own set).

The reconstructor repeats this verification for all possible $t$ sized subsets of the $m$ participants $\binom{m}{t}$ and all possible subsets of shares formed by these members $(|b|_{\mathsf{max}})^t$. Finally, the reconstructor informs each participant as to which of their shares were part of a combination that met condition (6). Each participant knows which of their elements was used to create each of their shares, and thus learns which of their elements belong to the over-threshold intersection. The reconstruction process is summarized in Algorithm 1.

**Algorithm 1** Reconstruction

```
1: for each bin $b_j$ with $j = 1, \ldots, b$ do
2:     for each $t$-subset of users $\mathcal{S}$ do
3:         for each share combination $\{s_i\}_{i\in\mathcal{S}}$ do
4:             Reconstruct polynomial evaluating $\{s_i\}_{i\in\mathcal{S}}$ in
               (6) (for t-PSI$_0$) or (8) (for t-PSI).
5:             if reconstruction succeeds then
6:                 Reveal to each $i \in \mathcal{S}$ that their shares $s_i$ are
                   in the over-threshold intersection.
```

*3.3.4 Complexity.* We analyse the complexity of $\texttt{t-PSI}_0$ with a single key holder and discuss multiple key holders in Section 3.6.

Theorem 3.2. *$\texttt{t-PSI}_0$ runs in two communication rounds with a communication complexity of $O(nm)$ where $m$ is the number of participants and $n$ is the maximum number of elements owned by a participant.*

Proof. As shown in Figure 2, the communication between the key holder and each participant takes places in one round. This communication can happen in parallel for all participants and all of their elements. Sending the shares to the reconstructor and receiving the notification about which elements are in the intersection requires a second communication round. The key holder exchanges two messages with each participant for each of their elements, which requires $2mn = O(nm)$ messages. The participants $(m)$ each send a table of size $\beta\beta_{\mathsf{m}}n = O(n)$ to the reconstructor. Therefore, the asymptotic communication complexity is $O(nm)$. □

Theorem 3.3. *The computation complexity of $\texttt{t-PSI}_0$ is $O(n(m\log n/t)^{2t})$ where $m$ is the number of participants, each participant holds at most $n$ elements, and $t$ is the intersection threshold.*

Proof. The reconstructor forms all possible combinations of $t$ shares from distinct participants in each bin. There are $b = \beta \cdot n/\log n$ bins, and $\binom{m}{t}$ combinations of $t$ participants out of $m$ of them. Moreover, each of these participants have $|b|_{\mathsf{max}} = \beta_{\mathsf{m}} \cdot \log n$ shares in each bin. Hence, each *for* loop in Algorithm 1 repeats $b$, $\binom{m}{t}$, and $(|b|_{\mathsf{max}})^t$ times, respectively. Applying Lagrange interpolation to each combination takes $t$ computations. (It also requires an $O(t^2)$ setup phase to compute the terms in the exponents in (6), which we can disregard in this asymptotic analysis since they only need to be computed once.) Then, using $\binom{m}{t} < (m \cdot e/t)^t$ where $e$ is Euler's constant, we get a cost of

$$b \cdot \binom{m}{t} \cdot (|b|_{\mathsf{max}})^t \cdot t \leq \frac{\beta \cdot n}{\log n} \left(\frac{m \cdot e}{t}\right)^t \cdot (\beta_{\mathsf{m}} \cdot \log n)^t \cdot t \, . \quad (7)$$

We can safely assume $t < \log n$ and $\beta_m \cdot e \leq m \cdot \log n/t$. Thus, in asymptotic notation we have $O(n\,(m\log n/t)^{2t})$. □

## 3.4 Faster Scheme: t-PSI

The reconstruction in $\texttt{t-PSI}_0$ is slow (it has very high constants), since Lagrange interpolation is executed in the exponent and modular exponentiations are expensive. To improve the reconstruction time, we construct an OPR-SS for t-PSI with Lagrange interpolation in the base. This is challenging, since the PRF requires exponentiation, but secret sharing requires addition. No cryptographic scheme supports both secure operations on its plaintext and does so efficiently. Hence, we use a conversion between two cryptographic primitives: PRFs using discrete logs and additively homomorphic encryption.

We use $\mathsf{sk}_{HE}$ and $\mathsf{pk}_{HE}$ to denote the secret and public keys of Paillier Cryptosystem, respectively. *Enc* and *Dec* denote the homomorphic encryption and decryption operations respectively. The plaintext field is $\mathbb{F}_N$ and the ciphertext field is $\mathbb{F}_{N^2}$, where $N > 2^\lambda p^2$ (see Section 3.4.1 for more details). We use $\mathbb{G}_q$, $\mathbb{F}_p$, and $H(\cdot)$ as defined for $\texttt{t-PSI}_0$ in Section 3.3.

*3.4.1 Share Generation.* The key holder generates $t-1$ random numbers $c_1, \ldots, c_{t-1} \leftarrow\!\$\, \mathbb{Z}_q$ that are used for share generations across all participants during this run of the protocol.

The generation of a share for participant $i$ using element $\ell$ proceeds as follows. First, participant $i$ generates a random number $\alpha$ to obliviously send its element's hash value, $H(\ell)$, to the key holder. The participant sends $H(\ell)^\alpha$, as well as $g^\alpha$, to the key holder. The key holder generates random numbers $r_1, \ldots, r_{t-1} \leftarrow \mathbb{Z}_p$ for this communication session (in addition to the $c_1, \ldots, c_{t-1}$ it generated initially). The key holder sends back $g^{r_j \alpha} \cdot H(\ell)^{c_j \alpha}$ for each $j \in [t-1]$. The goal of the $r_j$ values is to prevent the participant from learning $H(\ell)^{c_j}$. The participant removes $\alpha$ from the exponents, encrypts the resulting values using the Paillier cryptosystem, and sends these ciphertexts to the key holder. Since the Paillier cryptosystem allows multiplication by a plaintext, and the key holder knows the values $r_j$ and the generator $g$, it multiplies each received ciphertext $Enc[g^{r_j} \cdot H(\ell)^{c_j}]$ by $i^j / g^{r_j}$ (for all $j \in [t-1]$). This results in ciphertexts $\{Enc[i^j \cdot H(\ell)^{c_j}]\}_{j \in [t-1]}$, which are encryptions of the polynomial coefficients already multiplied by the evaluation of the polynomial in $x = i$. Since the scheme supports homomorphic addition, the key holder just builds this polynomial, evaluated at $x = i$, by adding these ciphertexts, and sends this addition back to the participant. The participant decrypts the message and recovers its share $P(i) = \sum_{j=1}^{t-1} i^j \cdot H(\ell)^{c_j}$.

Figure 3 shows this protocol. The key holder and the participant repeat this process for every element $\ell \in \mathbb{L}_i$. This process can run in parallel among elements and among participants.

Note that this share generation algorithm requires switching between the base prime field $\mathbb{F}_p$ and the plaintext field of the homomorphic encryption $\mathbb{F}_N$. We use the statistical secret sharing scheme introduced by Damgård and Thorbek [6] to convert an encrypted message to a secret shared message over a field smaller than the plaintext field. This conversion requires using a field $\mathbb{F}_N$ where $N > 2^\lambda p^2$.

*3.4.2 Hashing-to-bins.* This step is identical to the hashing-to-bins described in Section 3.3.2.

*3.4.3 Reconstruction.* Reconstruction proceeds as in Section 3.3.3 (i.e., Algorithm 1), with the exception that the polynomial recovery does not happen in the exponent. Given a set $\mathcal{S}$ of $t$ users and one share $s_i$ from each of them, the reconstructor determines that the shares were generated with the same element, and thus the element is in the intersection, if the following equality holds:

$$0 = \sum_{i \in \mathcal{S}} \left[ s_i \cdot \prod_{\substack{j \in \mathcal{S} \\ j \neq i}} \frac{-j}{i-j} \right]. \tag{8}$$

*3.4.4 Complexity.* We analyse the complexity of `t-PSI` with a single key holder (see Section 3.6 for the multi key-holder case).

Theorem 3.4. *`t-PSI` runs in three communication rounds with a communication complexity of $O(nmt)$ where $m$ is the number of participants, each holding at most $n$ elements, and $t$ is the intersection threshold.*

Proof. Figure 3 shows that the key holder and a participant communicate for two rounds. Additionally, the participants communicate with the reconstructor for another round. This communication can be done in parallel for all participants. The number of messages exchanged to generate a share $s_i^\ell$ is $O(t)$, as shown in the figure. Since this process is performed for each participant $i \in [m]$ and each of their elements $\ell \in \mathbb{L}_i$, and $|\mathbb{L}_i| \leq n$, the communication complexity is $O(nmt)$. The cost of sending the hash tables to the reconstructor, $O(nm)$, does not affect the asymptotic cost. □

Theorem 3.5. *The computation complexity of `t-PSI` is $O(n(m \log n/t)^{2t})$ where $m$ is the number of participants, each participant holds as most $n$ elements and $t$ is the intersection threshold.*

Proof. The proof is analogous to the proof of Theorem 3.3, as the only difference in the reconstruction between `t-PSI`$_0$ and `t-PSI` is the fact that the reconstruction in `t-PSI` does not happen in the exponent. This operation is cheaper than the reconstruction in the exponent of `t-PSI`$_0$, but both require $O(t)$ computations in asymptotic notation, and thus the proof follows accordingly. □

## 3.5 Multiple Reconstructors

Thus far we have described the schemes with a single reconstructor ($r = 1$) that checks whether or not a subset of shares were generated with the same element. We now emphasize that the reconstruction stage is not restricted to a single participant and can be performed by multiple participants in parallel. Since the reconstruction in each bin $b_j$ ($j \in [b]$) can be performed independently, in the multiple reconstructor case ($r > 1$) we can assign different subsets of the bins to each reconstructor. The workload can even be distributed according the computational power of each reconstructor. In this case, each reconstructor would be responsible for informing the participants for the bins they manage as to which of the participants' elements are in the over-threshold intersection. Since the participants would send each bin to only one of the reconstructors, the communication complexity of the protocol is not affected.

## 3.6 Multiple Key Holders

It is possible to extend `t-PSI` for a multi key holder scenario ($k > 1$); while such a setting trivially applies to `t-PSI`$_0$. For the multiple key holder setting, the key holders receive the initial message from the participant and run a secure computation protocol among themselves using a group key and group random numbers. The group keys $c_j = c_{j,1} + c_{j,2} + \ldots c_{j,k}$ are shared additively and the group random number is $R_j = g^{\alpha r_j}$. Key holder with identifier $i$ holds shares $c_{j,i}$ and $R_{j,i}$. The $k$ key holders need to multiply the product of the exponentiations of the participant's messages to their key shares and their random group shares using a secure computation. Instead of performing this secure computation using secret-share based secure computation with $O(k^2)$ communication in $O(\log k)$ rounds, the participants can use precomputed shares $R_j = R_{j,1} \cdot R_{j,2} \cdots R_{j,k}$, since all inputs are random and independent of the elements in $\mathbb{L}$. Each key holder sends the product $R_{j,i} H(x)^{\alpha c_{j,i}}$ to the first key holder, who multiplies them.

Since we assume semi-honest key holders, we must ensure they do not learn additional information. The first key holder learns no information, since all products are indistinguishable from random

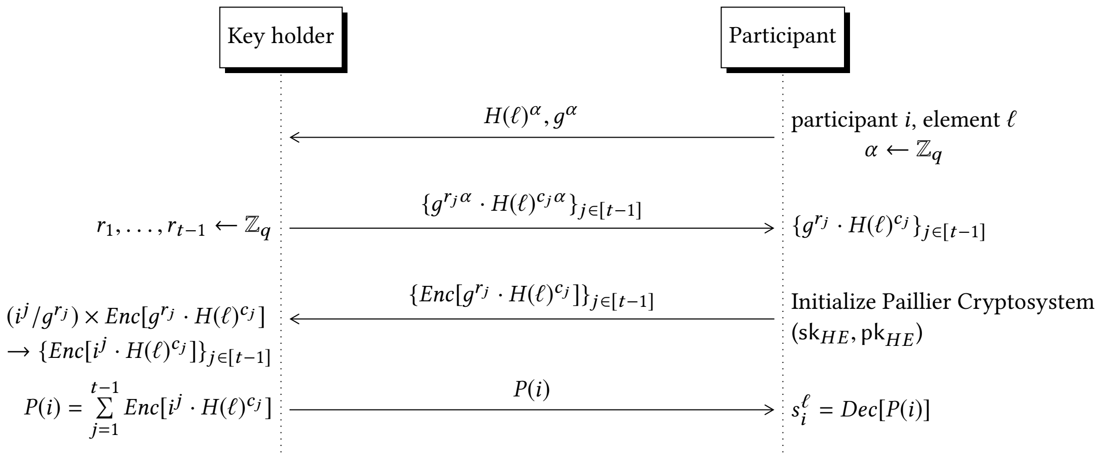

Key holder
Participant
$H(\ell)^{\alpha}, g^{\alpha}$
participant $i$, element $\ell$
$\alpha \leftarrow \mathbb{Z}_q$
$r_1, \ldots, r_{t-1} \leftarrow \mathbb{Z}_q$
$\{g^{r_j \alpha} \cdot H(\ell)^{c_j \alpha}\}_{j \in [t-1]}$
$\{g^{r_j} \cdot H(\ell)^{c_j}\}_{j \in [t-1]}$
$(i^j / g^{r_j}) \times Enc[g^{r_j} \cdot H(\ell)^{c_j}]$
$\rightarrow \{Enc[i^j \cdot H(\ell)^{c_j}]\}_{j \in [t-1]}$
$\{Enc[g^{r_j} \cdot H(\ell)^{c_j}]\}_{j \in [t-1]}$
Initialize Paillier Cryptosystem
$(\mathsf{sk}_{HE}, \mathsf{pk}_{HE})$
$P(i) = \sum_{j=1}^{t-1} Enc[i^j \cdot H(\ell)^{c_j}]$
$P(i)$
$s_i^{\ell} = Dec[P(i)]$

**Figure 3: Share generation for an element $\ell \in \mathbb{L}_i$, owned by $id_i$, in `t-PSI`.**

numbers. In the second communication round of `t-PSI`, the key holders use precomputed additive shares of the inverse $R_j^{-1/\alpha} = R'_{j,1} + R'_{j,2} + \ldots R'_{j,k}$ of the group random number. Then, the product (of the ciphertexts, or sum of the plaintexts) can again be performed by sending it to the first key holder, since they are semantically secure ciphertexts. The entire protocol has $O(k)$ communication complexity per coefficient, element and participant. It runs in $O(1)$ rounds with an offline pre-computation phase independent of the input. The multi key holder scheme otherwise follows the exact same procedure as in `t-PSI`. Hence, the total communication cost of `t-PSI` with $k$ key holders is $O(nmtk)$.

## 4 SECURITY

In this section, we perform the security analysis of our second scheme, `t-PSI`. We do not derive the security proofs for `t-PSI`$_0$ due to space restrictions, and because for practical use cases `t-PSI` outperforms `t-PSI`$_0$.

*Definition 4.1.* We say the Decisional Diffie-Hellman (DDH) Assumption holds iff, for any probabilistic polynomial-time adversary $\mathcal{A}$,

$$\rho, \sigma, \tau \leftarrow\!\!\$\, \mathbb{Z}_q$$
$$\Pr[\mathcal{A}(g^{\rho}, g^{\sigma}, g^{\tau}) = 1] - \Pr[\mathcal{A}(g^{\rho}, g^{\sigma}, g^{\rho\sigma}) = 1] < \mathsf{negl}(\lambda)$$

Theorem 4.2. *If the DDH Assumption holds and $H(\cdot)$ is a random oracle, each coefficient $H(\ell)^{c_j}$ of a share's polynomial in `t-PSI` is a pseudo-random function on $\ell$.*

Proof. We prove Theorem 4.2 by reducing an adversary $\mathcal{A}_{\mathsf{PRF}}$ in Game$^{\mathsf{PRF}}$ to an adversary $\mathcal{A}_{\mathsf{DDH}}$ against the DDH Assumption using a programmable random oracle. Let $g^{\rho}, g^{\sigma}, g^{\tau}$ be a DDH instance. We program $H(\cdot)$ to output $g^{\rho}$ on input $\ell$. We set the public key $\mathsf{pk}_{PRF}$ to $g^{\sigma}$. We can still answer queries for $F(\ell', \mathsf{sk}_{PRF}) = H(\ell')^{\mathsf{sk}_{PRF}}$ where $\ell' \neq \ell$ by programming $H(\cdot)$ to choose and store $r \leftarrow\!\!\$\, \mathbb{Z}_q$ and output $R = g^r$. An answer to a PRF query for $\ell'$ is then $g^{\sigma r} = H(\ell')^{\sigma}$. We set $y_b$ to $g^{\tau}$. We set the output of adversary $\mathcal{A}_{\mathsf{DDH}}$ to the output of adversary $\mathcal{A}_{\mathsf{PRF}}$. All outputs are indistinguishable from the real scheme and any advantage between the two adversaries translates directly. □

Theorem 4.3. *The share generation of* `t-PSI` *is an OPR-SS scheme that is secure in the semi-honest model.*

Proof. By the protocol construction, each $s_i^{\ell}$ returned to the participant is computed using the same coefficients in Shamir's secret sharing scheme and hence $t' \geq t$ participants can reconstruct 0. However, since each coefficient is the output of a PRF on $\ell$ (Theorem 4.2), $t$ shares, where at least one share is for a different $\ell'$, reconstruct to $S = 0$ only with a negligibly small probability.

We want to highlight a subtlety of the protocol that stems from the known "secret" 0. A set of $t-1$ secret shares $s_i^{\ell_i}$ are computationally indistinguishable from a set of $t-1$ uniformly chosen random numbers, since they leave one degree of freedom for choosing $\ell_t$, but any set of $t$ secret shares allows testing whether they are from the same element $\ell$. To explain further, consider an adversary that controls $t-2$ parties. If this adversary obtains a share $s_i^{\ell'}$ where party $i$ is not controlled by the adversary, e.g., during reconstruction, then this adversary cannot determine whether $\ell' = \ell$ for any $\ell$ chosen by the adversary, since any $t-1$ shares may reconstruct to 0. Now, consider an adversary that controls $t-1$ parties. This adversary can choose $\ell$, obtain secret shares and even reconstruct the coefficients of the secret share polynomial using $t-1$ shares, since they know the "secret" 0. Hence, they can test whether for another share $s_i^{\ell'}$ it holds that $\ell' = \ell$. However, the output of this test is included in the output of the protocol and the attack would be feasible for any adversary admissible to the protocol. Our initial claim then follows: $t-1$ secret shares $s_i^{\ell_i}$ are computationally indistinguishable from a set of $t-1$ uniformly chosen random numbers, but any set of $t$ secret shares allows testing whether they are from the same element $\ell$. It remains to show that the protocol is oblivious and the simulators exist.

We construct the simulator $\mathsf{Sim}_{Participant}(\ell, s_i^{\ell})$ as follows. The simulator outputs $t-1$ random elements $r \leftarrow\!\!\$\, \mathbb{F}_p$. This is perfectly indistinguishable, since the key holder chooses a uniform random blinding element per message. Let $r \leftarrow_\$ [0, 2^\lambda p]$. The simulator outputs $\mathsf{Enc}(rp + s_i^\ell)$. This is statistically indistinguishable, since the key holder uses share conversion to hide the multiplications in $\mathbb{F}_N$.

We construct the simulator $\mathsf{Sim}_{Keyholder}(\mathsf{sk}_{PRF})$ as follows. The simulator chooses $\rho, \tau \leftarrow_\$ \mathbb{Z}_q$, outputs $g^\rho$ and $g^\tau$, and programs the random oracle $H(\cdot)$ for $\ell \leftarrow \mathbb{L}$ to $g^{\tau/\rho}$. This is perfectly indistinguishable, since all values are uniform and the random oracle is consistent. The simulator outputs $t-1$ random elements $r \leftarrow_\$ \mathbb{F}_{N^2}$. This is computationally indistinguishable, since Paillier ciphertexts are semantically secure [22].

□

## 5 EVALUATION

In this section we provide performance benchmarks for t-PSI$_0$ and t-PSI. For the share generation, we measure the communication and computation overhead per participant with different deployment settings. For the reconstruction, we evaluate the scalability of our proposed schemes with varying input size $n$, number of parties $m$, and intersection threshold $t$.

### 5.1 Setup

We evaluate online share generation and offline reconstruction in two separate experiments. We implement t-PSI$_0$ and t-PSI in C++ using NTL and GMP for large arithmetic operations and libhcs[1] for the Pailler homomorphic encryption. Reconstruction is parallelized using OpenMP.[2] For share generation, we execute both the participant and key holder roles using virtual machine instances (t2.micro) from the Amazon Elastic Compute Cloud (AWS EC2).[3] These machines are connected via a network with a maximal bandwidth of 25 Gbps. Each virtual machine only has access to a single virtual CPU, to demonstrate that neither participants, nor the key holder, require access to significant computing capacity. We also use three connectivity settings in our evaluation of share generation.

- **Local**: Two servers deployed in the same data center.
- **Remote**: Two servers in different data centers located on the same continent (the U.S. west and east coast).
- **Distant**: Two servers in different data centers separated by continents (the U.S. east coast and central Europe).

Note that as we use a real network for our experiments, i.e., between two AWS instances that are actually deployed on the internet, we do not use any active network delay and throughput mechanism in reporting the results of our experiments.

Finally, we execute reconstruction on a server running Ubuntu 18.04 in 64-bit mode using up to 128 cores of an IBM POWER8 CPU (2.4GHz) with up to 1TB of accessible RAM. In all experiments, for the field $\mathbb{F}_p$ we choose a prime $p$ with at least 2048 bits as recommended by NIST [3].

### 5.2 Share Generation

We micro-benchmark the generation of a share for a single element between one participant and the key holder to evaluate how latency and computational time contribute to the total share generation runtime. Latency measures round-trip time and computational time measures the joint local execution time for a participant and the key holder. By distinguishing local, remote, and distant connectivity settings, we evaluate the impact of latency on the share generation time. In practice, latency is important as a lower bound to the total running time of the protocol. The share generation is also impacted by the network throughput, i.e. the maximum amount of data that can be transmitted per second. We do not measure the delay caused by the throughput separately, because the amount of data sent in a micro-benchmark does not saturate the network bandwidth; making its total runtime impact negligible. Our share-generation implementation does not support transmitting and processing multiple elements within one round, which would be necessary to separately measure delay caused by insufficient network bandwidth.

We execute the share generation protocol ten times in each connectivity setting and report the average and standard deviation of the runtime (in milliseconds) and the network traffic (in kilobits) for schemes t-PSI$_0$ and t-PSI in Table 3. With respect to communication, t-PSI$_0$, sends approximately an order of magnitude less data than t-PSI and is also up to two orders of magnitude faster. However, even when factoring in the higher latency of t-PSI, the largest contributor to runtime for t-PSI is computation. This can be attributed to t-PSI requiring rather slow homomorphic encryption and decryption whereas t-PSI$_0$ relies only on exponentiation. Thus, in t-PSI$_0$ the runtime and communication is constant for arbitrary $t$, whereas in t-PSI we observe the runtime and communication to grow linearly with $t$, where the correlation with $t$ can be explained by t-PSI transmitting and processing $t-1$ elements in each round.

### 5.3 Reconstruction

We measure the time it takes to reconstruct elements from all participants within a single bin for set sizes of up to $n = 10^6$. Reconstruction over multiple bins is inherently paralellizable and the total reconstruction time can be extrapolated by multiplying the time of for a single bin by the total number of bins. We set the number of bins $b = \lceil \beta \cdot n / \log n \rceil$, and the maximum bin size as $|b|_{\mathsf{max}} = \lceil \beta_{\mathsf{m}} \cdot \log n \rceil$, where $\beta$ and $\beta_{\mathsf{m}}$ are constants.

Note that although the value of $\beta_{\mathsf{m}}$ can be obtained from $n$ and $|b|_{\mathsf{max}}$ using the formula $\frac{\beta_{\mathsf{m}}}{\log n}$, we use $\beta_{\mathsf{m}}$ to determine $|b|_{\mathsf{max}}$ empirically. We are able to optimize and find tighter bounds through our empirical measurments. Thus, we use $\beta \in \{1, 4, 64\}$ and experimentally find the values of $\beta_{\mathsf{m}}$ for each case by simulating the hashing process 10 000 times. The resulting values for $|b|_{\mathsf{max}}$ are shown in Figure 4a. We observe a slightly higher than expected value for $|b|_{\mathsf{max}}$ for $n = 512$, which explains elevated runtimes for $n = 512$ in our benchmarks. Figure 4a also shows the communication overhead, i.e., the number of shares that each participant sends to the reconstructor (which is the same for t-PSI$_0$ and t-PSI). This overhead increases with the number of bins (i.e., with $\beta$) but is linear with $n$, as predicted by our analysis.

In our benchmarks, we vary the number of parties $m$, threshold $t$, number of inputs $n$ and the constant $\beta$. We report the average execution time over three consecutive runs and show the standard deviation as vertical error bars. Most of these bars are too small to be seen at the scale at which we display the plots. Figure 4b shows

[1]https://github.com/tiehuis/libhcs
[2]https://www.openmp.org/
[3]https://console.aws.amazon.com/ec2/v2/home

| | | Local | | Remote | | Distant | | |
|---|---|---|---|---|---|---|---|---|
| **Scheme** | **t** | **Time (ms)** | **$\sigma$ (ms)** | **Time (ms)** | **$\sigma$ (ms)** | **Time (ms)** | **$\sigma$ (ms)** | **Comm (Kb)** |
| t-PSI$_0$ | * | 20 | 1 | 140 | 1 | 190 | 1 | 1.24 |
| t-PSI | 2 | 425 | 1 | 670 | 2 | 760 | 1 | 10.13 |
| | 3 | 635 | 2 | 870 | 2 | 970 | 2 | 13.26 |
| | 4 | 847 | 2 | 1090 | 3 | 1180 | 1 | 16.39 |
| | 5 | 1057 | 3 | 1300 | 2 | 1400 | 8 | 19.53 |
| | 6 | 1267 | 4 | 1500 | 4 | 1610 | 2 | 22.66 |
| | 7 | 1479 | 2 | 1720 | 4 | 1820 | 2 | 25.80 |
| | 8 | 1687 | 2 | 1920 | 3 | 2020 | 4 | 28.92 |
| | 9 | 1901 | 6 | 2140 | 3 | 2230 | 4 | 32.06 |

**Table 3: Share generation time and communication cost per participant of t-PSI$_0$ and t-PSI averaged over 10 runs. t-PSI$_0$ has constant communication and computation costs for arbitrary $t$.**

the reconstruction time per bin of t-PSI$_0$ and t-PSI versus the number of parties $m$ for different threshold values $t$ and a fixed input set size of $n = 1\,024$ and constant $\beta = 1$. We can see the reconstruction time of t-PSI is almost three orders of magnitude faster than t-PSI$_0$. The reconstruction time scales polynomially with $m^t$ where $t$ is fixed in all schemes, as predicted by Theorems 3.3 and 3.5. We did not evaluate t-PSI$_0$ for $t > 5$ and t-PSI for $t > 7$ since the running times were impractical.

Figure 4c shows the reconstruction times when varying the input size $n$ and using $m = 10$ and $\beta = 1$. We observe a logarithmic increase in runtime with $n$ because we only reconstruct over a single bin. When reconstructing over all $b$ bins, we would observe a linear increase in runtime.

Figure 4d compares the reconstruction times for varying threshold values, using $m = \{2t, 10\}$ and $\beta = \{1, 64\}$, while $n = 1\,024$ is fixed. The figure confirms that the total runtime is dominated by the threshold $t$, as predicted by Theorems 3.3 and 3.5. We observe that the choice of the constant $\beta$ has a significant impact on the runtime, i.e., in t-PSI for $t = 4$ the runtime is decreased by two orders of magnitude between $\beta = 1$ and $\beta = 64$. Note that the runtime improvement is at the cost of higher reconstruction communication as more bins have to be sent to the reconstructor, albeit each bin contains fewer elements. The final decision on the choice of $\beta$ depends on the available network and computation capacity. In our experiments, the reconstruction is the bottleneck which explains why t-PSI with $\beta = 64$ performs best.

## 5.4 Discussion

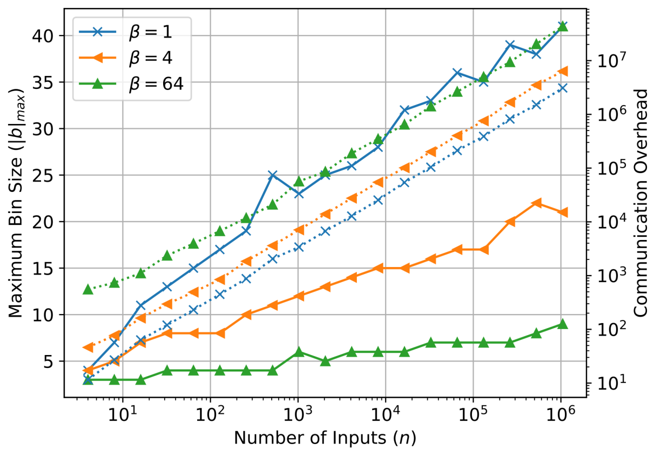
There is a trade-off in the share generation time and the reconstruction time between t-PSI$_0$ and t-PSI. Although t-PSI$_0$ has constant communication overhead in $t$ and t-PSI does not, our benchmarks in Section 5.3 show that reconstruction in t-PSI is orders of magnitude faster than reconstruction in t-PSI$_0$. We also observe that the reconstruction time is greater than the share generation time in Section 5.2, meaning that in practice t-PSI has a lower overall runtime than t-PSI$_0$ when taking both share generation and reconstruction time into consideration.

We observe that tuning constants $\beta$ and $\beta_{\mathsf{m}}$ adjusts the computation and bandwidth costs. Increasing the number of bins $b$ (i.e., $\beta$), allows decreasing the maximum bin size that the parties pad each bin to $|b|_{\mathsf{max}}$ (i.e., decrease $\beta_{\mathsf{m}}$) which increases communication overhead (larger hash tables sent) while decreasing the computational cost of reconstruction. Recall that the reconstructor selects a subset of $t$ users and performs Lagrange interpolation with combinations of one share from each user, iterating over all $(|b|_{\mathsf{max}})^t$ such combinations. Thus, increasing $b$ reduces $|b|_{\mathsf{max}}$, and therefore reduces the number of combinations that the reconstructor must try. Experimentally (see Figure 4d), the impact of the constant $\beta$ on the computation is significant for $\beta = 1$ versus $\beta = 64$.

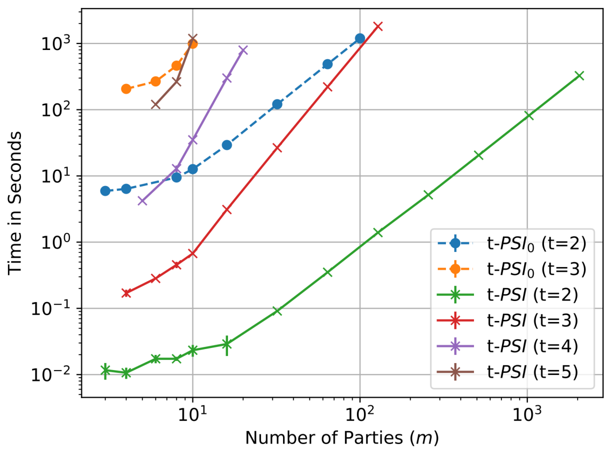
Our benchmarks show that the bottleneck of our protocol is the offline reconstruction, which can be improved up to the lower bound of a single reconstruction through parallelizing across more cores. The experiments already show results with a parallelization across up to 128 cores. Our benchmarks also show that participants do not require large computation capacity to participate in the protocol. Nonetheless, specifically for t-PSI, the total runtime can be improved by adding higher computation capacity at the key holder and participants. In practice, generating shares for multiple elements can be optimised by batching multiple elements per round and parallelizing the computation, which we have not implemented.

We confirm with our benchmarks sublinear scalability per bin in the number of inputs. When reconstructing over all $b$ bins, this turns into a subquadratic reconstruction complexity, as predicted by our theorems. We also confirm polynomial scalability by $m^t$ where $t$ is fixed and the number of parties $m$ is variable. For variable thresholds $t$, we show an exponential increase in runtime which can be traded-off for higher communication by choosing the constant $\beta$ for the number of bins accordingly.

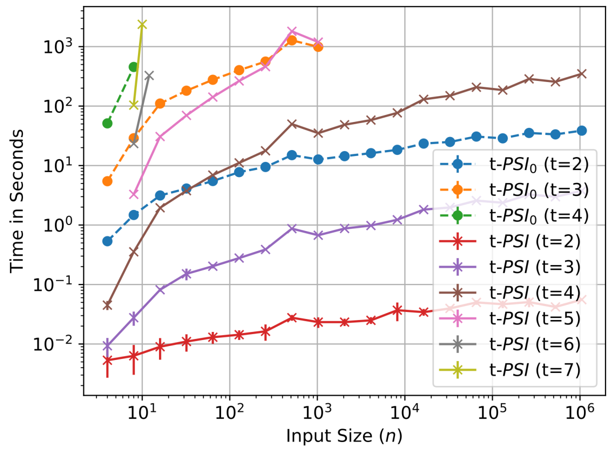
We now highlight the following results in terms of practicality. (For reference, Wagner et al. [28] report sharing about $n = 4,000$ indicators over six months using a central platform.) For $m = 10$ participants, $n = 1\,024$ inputs per participant and threshold $t = 4$, each participant in scheme t-PSI has a communication overhead of 2.10 MByte in the share generation. In the reconstruction, with $\beta = 1$, each party sends 3 404 elements from the field $\mathbb{F}_p$, which results in a communication overhead of 0.87 MByte per participant. If we assume each participant is a reconstructor, the reconstruction over all bins ($b = 148$) is in total about 9 minutes. For $\beta = 64$, the communication overhead in the share-generation stays the same, but it increases to about 21.78 MByte in the reconstruction. The reconstruction time over all bins ($b = 9\,455$) decreases to 3 minutes per participant. At $n = 10^6$ inputs, $m = 7$ participants and



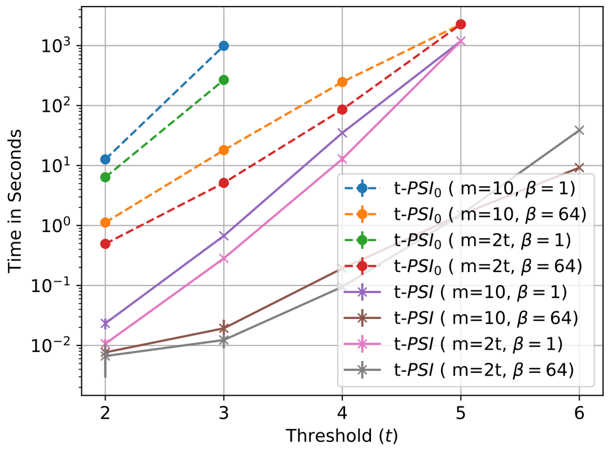
**(a) Maximum bin sizes (solid lines) and communication overhead (dotted lines) derived experimentally.**



**(b) Reconstruction time vs. number of parties $m$ ($n = 1\,024$, $\beta = 1$).**



**(c) Reconstruction time vs. input set sizes $n$ ($m = 10$, $\beta = 1$).**



**(d) Reconstruction time vs. thresholds $t$ ($n = 1\,024$).**

**Figure 4: Reconstruction time of `t-PSI`$_0$ and `t-PSI` for a single bin with different parameter configurations.**

a threshold $t = 3$, we have a communication overhead in the share-generation of 1.66 GByte per participant. In the reconstruction, with $\beta = 1$, each participant sends 0.79 GByte and the reconstruction time over all bins ($b = 75\,639$) would take about 5h and 4 minutes to complete. The reconstruction time can be reduced by increasing $\beta$. For $\beta = 8$, the total reconstruction time over all bins ($b = 605\,111$) decreases to 4h and 23 minutes, while the communication overhead increases to 2.32 GByte per participant.

## 6 RELATED WORK

In this section we present an overview of related aproaches used in other PSI problems (see Pinkas et al. for a more comprehensive overview of other PSI problems [23]) before we discuss various potential and existing approaches to over-threshold set operations. Note that OT-MP-PSI is a distinct problem from the recent, similarly named, multi-party threshold private set intersection problem, where the participants wish to compute the intersection of their sets only if the intersection set size meets some threshold [2].

### 6.1 Private Set Intersection

*6.1.1 Public-Key PSI Protocols.* Many approaches to PSI, including this work, rely on public-key cryptography. For example, there are protocols that rely on Diffie-Hellman key exchange [12, 21] and protocols that rely on RSA [7]. The protocols by Freedman et al. [8, 9] share a similar reliance to our work in that they make use of polynomial interpolation and the Paillier cryptosystem (or alternatively, ElGamal), ultimately relying on the decisional Diffie-Hellman assumption to obtain security guarantees. Public-key approaches to PSI tend to have better communication complexity than other methods, but higher overall computational costs [23].

*6.1.2 PSI Protocols Based on Oblivious Transfer.* Oblivious Transfer (OT) [25] is a two-party protocol where the sender holds several pieces of data, one of which will be sent to the receiver. The receiver can freely choose which of the messages to learn while the sender does not obtain any information about the receiver's choice. A common way to perform a large number of OT executions with low amortized cost is through OT extension [23], which is often used to efficiently construct OPRF's for use in PSI [17, 23, 24].

*6.1.3 Multi-Party PSI Protocols.* Freedman et al. [8] first considered PSI in the multi-party setting, achieving a communication complexity of $O(nm)$ and a computational complexity of $O(nm^2)$, in the semi-honest case. Cheon et al. [5] reduced the computational complexity to $O(nm)$ through the use of more efficient approaches for polynomial evaluation. More recently Hazay et al. [10] and Kolesnikov et al. [17] achieve $O(nm)$ communication complexity through schemes where a designated party participates in a protocol with all other parties (who perform very little local computation) and then combine the results. Although these protocols achieve good results for conventional private set intersection, at this time none of the protocols described above support practical over-threshold set operations.

### 6.2 Over-Threshold Set Operations

Kissner and Song proposed protocols for private set intersection and various related problems, including over-threshold set union [16] and over-threshold set intersection [15]. Their approach to over-threshold intersection requires local computation to factor a polynomial of degree $mn$ over a field or to evaluate the polynomial on $nm$ points to reconstruct the intersection. Unfortunatley, their protocol has a total communication complexity of $O(nm^3)$ and requires $O(m)$ rounds of communication, which makes it less suitable than our $O(nmt)$ scheme, which requires $O(1)$ rounds.

Although there are other potential designs for OT-MP-PSI protocols, as well as the protocol from Kissner and Song [15], they ultimately suffer in terms of practical efficiency. For example, one potential approach to perform OT-MP-PSI is to leverage generic multi-party computation techniques. However, communication-efficient, constant-round multi-party computation protocols [19, 20] are not yet practical. Alternatively, recall from the introduction that an OT-MP-PSI protocol could be constructed using a secret-shared multi-party protocol such as SCALE-MAMBA [1], but the optimal circuit size is $O(nm \log^2 nm)$ for $O(n + \log^2 nm)$ rounds or $O(nm \log^2 nm + nmt)$ for $O(\log^2 nm)$ rounds.

## 7 CONCLUSION

In this work, we introduce two over-threshold multi-party private set intersection schemes (`t-PSI`$_0$ and `t-PSI`) that use our new oblivious pseudo-random secret sharing protocol to achieve higher efficiency than previous proposals. Our protocols offer flexibility through tunable parameters that allow for tradeoffs between computation and communication cost. We additionally support spreading the security requirements among multiple key holders as well as distributing the computational cost among multiple reconstructors. The asymptotic communication of our schemes improves over previous schemes. In our most efficient scheme, `t-PSI`, our computational cost is exponential in terms of our intersection threshold, however, we leverage homomorphic encryption to reduce the overall cost of the scheme. With these optimizations, `t-PSI` for seven participants each holding one million elements and a threshold $t = 3$ takes around 5 hours. Furthermore, we show that when accounting for the overall cost of both reconstruction and share generation, in practice, `t-PSI` is most suitable for deployment.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful comments and feedback. We gratefully acknowledge the support of NSERC for grants RGPIN-05849, CRDPJ-531191, IRC-537591, and the Royal Bank of Canada for funding this research.

## REFERENCES

[1] Abdelrahaman Aly, Marcel Keller, Dragos Rotaru, Peter Scholl, Nigel P.Smart, and Tim Wood. SCALE–MAMBA software. https://homes.esat.kuleuven.be/~nsmart/SCALE/, 2020.
[2] Saikrishna Badrinarayanan, Peihan Miao, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. Cryptology ePrint Archive, Report 2020/600, 2020. https://eprint.iacr.org/2020/600.
[3] Elaine Barker, William Barker, William Burr, William Polk, Miles Smid, et al. *Recommendation for key management: Part 1: General.* National Institute of Standards and Technology, Technology Administration, 2006.
[4] Eric W. Burger, Michael D. Goodman, Panos Kampanakis, and Kevin A. Zhu. Taxonomy model for cyber threat intelligence information exchange technologies. In *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*, pages 51–60, 2014.
[5] Jung Hee Cheon, Stanislaw Jarecki, and Jae Hong Seo. Multi-party privacy-preserving set intersection with quasi-linear complexity. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 95(8): 1366–1378, 2012.
[6] Ivan Damgård and Rune Thorbek. Efficient conversion of secret-shared values between different fields. *IACR Cryptology ePrint Archive*, 2008:221, 01 2008.
[7] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *International Conference on Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.
[8] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.
[9] 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.
[10] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In *IACR International Workshop on Public Key Cryptography*, pages 175–203. Springer, 2017.
[11] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, 2012.
[12] Bernardo A Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 78–86, 1999.
[13] Christopher Johnson, Mark Badger, David Waltermire, Julie Snyder, and Clem Skorupka. Guide to cyber threat information sharing. Technical Report SP 800-150, National Institute of Standards and Technology, 2016.
[14] Panos Kampanakis. Security automation and threat information-sharing options. *IEEE Security & Privacy*, 12(5):42–51, 2014.
[15] Lea Kissner and Dawn Song. Private and threshold set-intersection. Technical report, Carnegie-Mellon University, 2004.
[16] Lea Kissner and Dawn Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO*, pages 241–257, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31870-5.
[17] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829, 2016.
[18] Frederick Lah. Are IP addresses "personally identifiable information"? *I/S: A Journal of Law and Policy for the Information Society*, 4:681–707, 2008.

[19] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. More Efficient Constant-Round Multi-party Computation from BMR and SHE. In *Proceedings of the 14th International Conference on Theory of Cryptography*, pages 554–581, 2016.

[20] Yehuda Lindell, Benny Pinkas, Nigal P. Smart, and Avishay Yanai. Efficient Constant-Round Multi-party Computation Combining BMR and SPDZ. *Journal of Cryptology*, 32(3):1026–1069, 2019.

[21] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134. IEEE, 1986.

[22] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48910-8.

[23] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[24] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse OT extension. In *Annual International Cryptology Conference*, pages 401–431. Springer, 2019.

[25] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[26] David Ross, Jason Shiffer, Tony Dell, William Gibb, and Doug Wilson. Openioc. https://www.openioc.org/, 2020.

[27] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. ISSN 0001-0782. doi: 10.1145/359168.359176. URL https://doi.org/10.1145/359168.359176.

[28] Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the ACM on Workshop on Information Sharing and Collaborative Security*, pages 49–56, 2016.

[29] Moti Yung. From mental poker to core business: Why and how to deploy secure computation protocols? In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 1–2, 2015.