SoK: Collusion-resistant Multi-party Private Set Intersections in the Semi-honest Model

Jelle Vos
Cyber Security Group
Delft University of Technology
Delft, Netherlands
J.V.Vos@tudelft.nl

Mauro Conti
SPRITZ
University of Padua
Padua, Italy
conti@math.unipd.it
Cyber Security Group
Delft University of Technology
Delft, Netherlands
M.Conti@tudelft.nl

Zekeriya Erkin
Cyber Security Group
Delft University of Technology
Delft, Netherlands
Z.Erkin@tudelft.nl

***Abstract*—Private set intersection protocols allow two parties with private sets of data to compute the intersection between them without leaking other information about their sets. These protocols have been studied for almost 20 years, and have been significantly improved over time, reducing both their computation and communication costs. However, when more than two parties want to compute a private set intersection, these protocols are no longer applicable. While extensions exist to the multi-party case, these protocols are significantly less efficient than the two-party case. It remains an open question to design collusion-resistant multi-party private set intersection (MPSI) protocols that come close to the efficiency of two-party protocols. This work is made more difficult by the immense variety in the proposed schemes and the lack of systematization. Moreover, each new work only considers a small subset of previously proposed protocols, leaving out important developments from older works. Finally, MPSI protocols rely on many possible constructions and building blocks that have not been summarized. This work aims to point protocol designers to gaps in research and promising directions, pointing out common security flaws and sketching a frame of reference. To this end, we focus on the semi-honest model. We conclude that current MPSI protocols are not a one-size-fits-all solution, and instead there exist many protocols that each prevail in their own application setting.**

***Index Terms*—Private Set Intersections, Systematization of Knowledge, Privacy-Enhancing Technologies**

1. Introduction

In 2004, Freedman et al. [1] proposed the first custom protocol for privately computing the intersection between multiple sets of data. Since then, private set intersection protocols have received a great deal of attention from the research community. Most notably, many concretely efficient two-party protocols have been proposed, which currently perform intersections over sets of 1 million elements in less than 6 seconds over a 100 MBit/s communication channel [2]. However, when protocols must scale to an arbitrary number of parties, performance degrades rapidly [3]. In this work, we systemize and summarize the current body of literature on such multi-party private set intersections (MPSI), and identify opportunities for future work.

More formally, an MPSI protocol solves the problem where n parties $\mathcal{P}_1, \dots, \mathcal{P}_n$ with respective private sets $X_1, \dots, X_n$, want to confidentially compute the intersection $X_1 \cap \dots \cap X_n$. They do so in the presence of at most t adversaries that may collude with one another. In this work, as for many previous works [3]–[5], one party $\mathcal{P}_1$ learns the intersection. We refer to this party as the leader, and the other parties $\mathcal{P}_2, \dots, \mathcal{P}_n$ as assistants.

Motivating applications. One motivating application of *multi-party* private set intersections is that of finding a set of suitable meeting dates between multiple private calendars. Here, X_i would be a set of dates at which party $\mathcal{P}_i$ is available. At the end of the protocol, the leader $\mathcal{P}_1$ receives the set of meeting dates at which all parties are available. Several other applications have been mentioned in previous works. We highlight several below.

Miyaji & Nishida [6] and Kolesnikov et al. [3] mention an application where multiple shop owners or digital services want to launch a collective promotion campaign. To do so, these shops must identify their mutual set of customers, without violating the privacy of the other customers.

Kissner & Song [7], Inbar et al. [8] and Kolesnikov et al. [3] mention cyber security applications. They consider a problem between several organizations who want to catch an intruder in a common network. The idea is that each organization keeps a list of suspicious IP addresses, and by computing the intersection, they narrow them down. Since IP addresses reveal personal information, it is important that the other IP addresses remain private. Similarly, Ghosh et al. [9] mention that MPSI can be applied to detect botnets.

Wang et al. [10] discuss an application where an investigative agency needs to narrow down a list of suspects by cooperating with multiple other agencies. Since none of the agencies can share plain data with each other, they engage in an MPSI protocol to only consider relevant suspects.

Freedman et al. [1] and Li & Wu [11] also mention that MPSI protocols may be used in online recommendation services, dating services, medical databases, and data mining in general. Kissner & Song discuss further applications, including detecting fraudulent sales from pharmacies, enforcing no-fly lists privately, combining the results of multiple surveys, and governments checking if their ill citizens are actually receiving aid [7]. Finally, Poddar et al. [12] discuss the application of MPSI for private database join operations.

We note that while PSI protocols are called *private* set intersections, this does not necessary imply that no personal data is being revealed. Instead, such protocols only ensure that the intersection is computed confidentially. If the input sets contain data that may never be revealed, the final intersection can still violate data privacy requirements. In other words, MPSI protocols do not necessarily alleviate every service that requires an intersection from possible privacy violations.

Focus of this paper. This paper restricts itself to protocols that do not require any external parties to collaborate. Moreover, we focus on protocols in the semi-honest model, and only mention whether extensions exist to the malicious model. The reasons are as follows:

A trivial way to tackle the MPSI problem is by selecting some trusted third party, who receives all private sets, computes the intersection, and sends the result to the leader. While such a protocol is extremely efficient, the third party sees all the private input sets. To prevent this, some works provide a slightly stronger notion of security assuming that the third party does not collude with any of the other parties, i.e. $t = 1$. As long as this assumption holds, the private information stays confidential, even from this third party. These protocols are often referred to as server-aided protocols. Note however, that this non-colluding assumption, like the trusted third party assumption, severely restricts the capabilities of an adversary. Moreover, such a protocol may not be trivially altered to allow for $t > 1$. It is for this reason that we consider only MPSI protocols with no external parties, and that support colluding parties.

When it comes to the security model, we only consider semi-honest adversaries. Not only is the semi-honest model often a crucial intermediate step to the malicious model, but typically the main insight behind a custom protocol does not change in the malicious model. We see this in the underlying set encodings used in MPSI protocols, such as polynomial roots, bitsets, garbled Bloom filters, and oblivious key-value stores. These encodings shape the protocol more so than the choice of cryptographic primitives.

State of MPSI. At present, there exist tens of works on MPSI protocols using a variety of underlying encodings, cryptographic primitives, and network topologies. However, many of these works only consider a small amount previous works in their comparison. Many older receive fewer attention in recent works as they are considered irrelevant due to performance comparisons, while this does not consider all possible points of comparison.

At the moment, the fastest works when the number a of elements grows large are based on oblivious key-value stores and vector oblivious linear evaluations. However, these works require several interactions between each pair of parties and a large bandwidth cost. It remains an open question to analyze how these protocols behave for different network conditions. Some alternatives only require assistants to communicate with the leader and require a lower total bandwidth. They are typically based on partially homomorphic encryption, at the cost of more computation. b The cheapest protocols computation-wise are based on secret sharing, but they suffer from a large bandwidth cost. c No current works are comparable in performance and practicality when $n > 2$, compared to the setting with $n = 2$: there is still a demand for more efficient protocols with regards to communication, computation, and the degree of interaction.

Contributions and outline of this paper. The remainder of this paper is structured as follows. In Section 2, we formalize the requirements for semi-honest MPSI protocols. In Section 3, we go over some preliminaries that form the foundations of these protocols. Then, in Section 4 we provide three high-level constructions that fit all protocols, and discuss possible set representations for each of them. After that, in Section 5, we provide a comprehensive overview of currently unbroken MPSI protocols based on these set representations. In Section 6, we highlight the problems in works that have been broken and analyze common security issues. Finally, in Section 7 we present a theoretical performance comparison of the most recent MPSI protocols in each category, and in Sections 8 and 9 we discuss the results, identify remaining research directions, and conclude this work. Appendices A and B contain derivations of the performance aspects of the considered MPSI protocols.

2. Formal problem description

An MPSI protocol π is a multi-party protocol that computes the intersection between private sets $X_1, \ldots, X_n$ with overwhelming probability:

$$X_1 \cap \cdots \cap X_n \approx \pi(X_1, \ldots, X_n)\,. \tag{1}$$

These sets are all subsets of the universe of possible elements $\mathcal{U}$. Moreover, we establish an upper bound k, so $|X_i| \leq k$ for all $i = 1, \ldots, n$. Each set X_i is held by a party $\mathcal{P}_i$ that is semi-honest. In other words, the party faithfully follows the protocol description, but tries to learn as much as possible in the process. For the protocols studied in this work, we consider it sufficient for only the leader $\mathcal{P}_1$ to receive the computed intersection. We present the ideal functionality of such a protocol below.

Definition 1 (MPSI protocol)**.** An MPSI protocol correctly and privately computes the following ideal functionality:

$$f(X_1, \ldots, X_n) = (X_1 \cap \cdots \cap X_n, \Lambda, \ldots, \Lambda)\,,$$

where Λ is the empty string. The upper bound k may be provided as auxiliary information.

Remark 1. Some works consider a case where all parties receive the result. In the semi-honest model it is trivial to transform a protocol where only the leader learns the result to one where all parties do: the leader simply forwards its result to all parties. It is not straightforward to do so in the malicious model [13].

Note that MPSI protocols typically require the elements in the input sets to be mapped to some cryptographic object. For example, the protocol by Kissner & Song [7] maps elements to some group to be encrypted as Paillier ciphertexts. As a result, one must select cryptographic parameters that ensure the group is large enough to contain all distinct elements from $\mathcal{U}$.

Remark 2. One way to circumvent choosing larger parameters when $\mathcal{U}$ is too large is to hash elements of the input sets onto the cryptographic object, thereby supporting a larger universe at the risk of collisions. The chance of collisions is negligible for a sufficiently large cryptographic object.

2.1. Correctness requirements

In (1) and in the remainder of this paper we write $\approx$ to denote that this result could be an approximation. In fact, due to the properties of cryptographic protocols, all schemes considered in this paper are in some way approximations, albeit that they can be arbitrarily accurate. For example, the scheme by Kolesnikov et al. [3] achieves a failure probability of $2^{-\lambda}$, where λ is a customizable statistical security parameter. Typically, $\lambda = 40$, and in the remainder of this paper we will consider any probability lower than 2^{-40} to be negligible:

Definition 2 (Exact MPSI protocol)**.** The probability that the output of an exact MPSI protocol does not equal the actual intersection does not exceed 2^{-40}:

$$\Pr\left[X_1 \cap \cdots \cap X_n \neq \pi(X_1, \ldots, X_n)\right] \leq 2^{-40} .$$

We say that an MPSI protocol is *approximate* if the chosen parameters cause it to suffer from a higher probability of being wrong than 2^{-40}. We measure the accuracy of approximate protocols more precisely by determining the probability ε that a random element x is wrongly claimed to be included or excluded in the computed intersection. Note that in some protocols, x can only be falsely included and never be falsely excluded. We refer to those protocols as satisfying *superset correctness*. We refer to protocols that only false exclude elements of the intersection as satisfying *subset correctness*.

2.2. Privacy in the semi-honest model

Since this work considers protocols in the semi-honest model and since the output of an MPSI protocol is deterministic, one can prove that a protocol is privacy-preserving if it is simulatable [14]. The strongest notion of privacy for semi-honest MPSI protocols is called *size-hiding*, which means that the size of each party's set stays secret:

Definition 3 (Size-hiding MPSI)**.** For each party $\mathcal{P}_i$ in MPSI protocol π, there exists a simulator $\mathcal{S}_i$ that generates an indistinguishable view for all possible combinations of inputs. The simulator receives its input X_i and the intersection $X_1 \cap \cdots \cap X_n$. It must hold that:

$$\{\mathcal{S}_i(1^\lambda, X_i, X_1 \cap \cdots \cap X_n)\}_{X_1,\ldots,X_n \subseteq \mathcal{U}} \overset{c}{\equiv} \{\mathsf{view}_i^\pi(X_1, \ldots, X_n, \lambda)\}_{X_1,\ldots,X_n \subseteq \mathcal{U}} . \quad (2)$$

Note that here we provide the assistants with the intersection as well, because even though they are not required to output it, they are allowed to learn this information.

If the size of each party's set is not private, we have the following definition:

Definition 4 (Size-revealing MPSI)**.** For each party $\mathcal{P}_i$ in MPSI protocol π, there exists a simulator $\mathcal{S}_i$ that generates an indistinguishable view for all possible combinations of inputs. The simulator receives its input X_i, set sizes $|X_1|, \ldots, |X_n|$, and the intersection $X_1 \cap \cdots \cap X_n$. It must hold that:

$$\{\mathcal{S}_i(1^\lambda, X_i, |X_1|, \ldots, |X_n|, X_1 \cap \cdots \cap X_n)\}_{X_1,\ldots,X_n \subseteq \mathcal{U}} \overset{c}{\equiv} \{\mathsf{view}_i^\pi(X_1, \ldots, X_n, \lambda)\}_{X_1,\ldots,X_n \subseteq \mathcal{U}} . \quad (3)$$

Remark 3. Size-revealing MPSI protocols can typically be turned into size-hiding MPSI protocols by letting users submit dummy elements in their input sets to always pad their set to size k. These can either be repetitions of other elements in the set or random elements which have a negligible probability of appearing in the resulting intersection.

We consider the privacy definitions above in the presence of $1 < t < n$ colluding parties. These colluding parties are still semi-honest but they share their views with one another to learn as much private information as possible. When we say that a protocol has a collusion resistance of t, this means that even t colluding adversaries cannot break the claimed notion of privacy.

There are protocols that achieve statistical ($\overset{s}{\equiv}$) rather than computational ($\overset{c}{\equiv}$) indistinguishability in (2) or (3). These protocols are information-theoretically secure; they do not rely on any computational hardness assumptions. Note that such protocols can only achieve a collusion threshold $t < \frac{n}{2}$ [15]. In the best case, computationally-secure protocols can withstand $t = n - 1$ colluding parties.

2.3. Privacy in the *augmented* semi-honest model

Some MPSI protocols consider the *augmented* semi-honest model, which allows an adversary to choose a different input than its actual input immediately before running the protocol [16]. While this may seem like a stronger form of security, it is not necessarily compatible with the semi-honest model. That is, a protocol proven to be secure in the augmented semi-honest model may not be secure in the semi-honest model. The reason is that the augmented semi-honest model empowers simulators to choose a convenient input for themselves, while a semi-honest simulator must work for every input. Counter-ituitively, this also implies that protocols secure in the malicious model are not necessarily secure in the semi-honest model. On the contrary, protocols in the malicious model *are always* secure in the augmented semi-honest model. For this reason, we include protocols that adopt this model in this work.

Concretely, the benefit of the augmented model to the design of MPSI protocols is that certain precomputations can be done before the start of the protocol. For example, Inbar et al. [8] perform secret sharing ahead of time, after which parties only have to change their local shares during the actual protocol execution.

3. Preliminaries

We briefly recall the background for MPSI protocols. In the remainder of this paper, we work over sets of integers. Where an ordering is necessary, we use $1, 2, 3, \dots$ implicitly. Table 1 summarizes the symbols in this work.

3.1. Network topologies

The network topology defines the structure of communication channels between the parties involved in a protocol.

In a star topology, each assistant only shares one bidirectional communication channel with the leader, and none with other assistants. Such a topology resembles the ideal world scenario, and thereby provides the minimal number of communication channels necessary to realize an MPSI protocol. The drawback of this topology is that if a malicious leader refuses to communicate, no communication is possible. The number of channels grows as $O(n)$.

In a mesh topology, each party has a bidirectional communication channel with all other parties. As a result, each party has the ability to broadcast. Such a topology has maximum redundancy when malicious parties refuse to communicate. The number of channels grows as $O(n^2)$.

All other topologies sit in between star and mesh topologies. One topology used in earlier MPSI protocols [7] combines a star and ring structure where each assistant shares a communication channel with the leader as well as one other assistant in a circular fashion. We refer to this as a 'wheel' topology. Another topology, which is used by Qiu et al. [17], resembles a binary tree.

3.2. Building blocks for secure MPC

Partially-homomorphic encryption. A partially-homomorphic encryption (PHE) scheme is a collection of methods to securely encrypt and decrypt data, which still allows parties to manipulate the underlying data while it is encrypted. For example, the Paillier cryptosystem [18] allows one to encrypt some integers $x, y \in \mathbb{Z}_{pq}$, and then homomorphically combine the two resulting ciphertexts to compute a ciphertext that encrypts the sum $x+y$. This makes the Paillier cryptosystem additively homomorphic. Another cryptosystem comes in the form of the ElGamal [19] cryptosystem, which is multiplicatively homomorphic in its standard form. It works over any cyclic group $\mathbb{G}$ in which the decisional Diffie-Hellman assumption holds. By encoding values in the exponent, the cryptosystem becomes additive in some sense. Moreover, by using an elliptic curve group for $\mathbb{G}$, all operations become significantly faster than over the integers [20]. MPSI protocols use threshold versions of homomorphic cryptosystems, which require $t+1$ parties to work together to decrypt a ciphertext.

Secret sharing. Secret sharing is an information-theoretically secure method of storing data among multiple parties, without individual parties being able to access it. To do so, a secret is split up into multiple shares that combine to retrieve the original secret. One example is `XOR`-sharing, which splits a secret x into shares $s_1, \dots, s_n$ so that $x = s_1 \oplus \cdots \oplus s_n$. Other secret sharing schemes allow for arithmetic operations to be performed over the underlying secrets by manipulating the shares. For example, a simple additive secret sharing scheme with $x = s_1 + \cdots + s_n \pmod q$ allows parties to compute the sum of two shared secrets, by adding their respective shares. By interacting with the other parties, it is also possible to privately multiply additive secret shares [15].

Oblivious transfers. An oblivious transfer (OT) is a two-party protocol between a sender and a receiver. In its simplest form, the receiver chooses one of two messages s_0 and s_1 to receive using bit $b \in \mathbb{Z}_2$. The sender sends message s_b as requested. Crucially, the sender may not learn b and the receiver may not learn the other message s_{1-b}. Oblivious transfers are computationally cheap to evaluate in large quantities due to the existence of OT extensions. OT extensions execute a small amount of full-fledged oblivious transfers in the opposite direction, after which the received data can be used to perform future transfers with minimal computational effort.

Oblivious linear evaluation. An oblivious transfer can be seen as a protocol that privately computes $s_b = s_0 + (s_1 - s_0)b$ for $b \in \mathbb{Z}_2$. In other words, it computes a linear function over b, which is binary. Oblivious linear evaluations (OLEs) extend oblivious transfers to compute linear functions over elements from larger groups $\mathbb{Z}_q$. More precisely, the sender holds values u and v, and the receiver learns $w = ux+v$ for some x of its choosing [21]. In some cases, a receiver might want to perform a large amount of OLEs with the same input x. There exist custom protocols for this special case called vector oblivious linear evaluations (VOLEs), which are significantly cheaper to evaluate.

Oblivious pseudorandom functions. Another primitive for evaluating functions obliviously is an oblivious pseudorandom function (OPRF). Such a protocol allows a receiver to compute a pseudorandom function over an input x of its choosing, while the sender chooses which key K is used to compute the output [22]. Importantly, the receiver does not learn K and the sender does not learn x, nor the output of the PRF. A multi-point OPRF evaluates multiple OPRFs at the same time on the same key K for different inputs [22]. It does so more efficiently than it is to execute multiple single point OPRFs. The most efficient schemes currently rely on oblivious transfers.

3.3. Common methods in cryptographic protocols

Finally, we explain three common techniques at the core of several MPSI protocols.

3.3.1. Secure `AND` operations. In many MPSI protocols there is a need to compute an `AND` operation between multiple bits $x_1, \ldots, x_n$. While a simple approach is to compute the product, the multiplications are expensive to execute privately. Instead, a typical alternative is to compute some aggregate based on the input bits that is 0 when all bits were 0, and random otherwise. Kolesnikov et al. [3], for example, achieve this using `XOR`-based secret sharing. Other works [5], [20] privately compute a randomized sum $\mathbf{r}(x_1 + \cdots + x_n)$, where $\mathbf{r}$ is some random number unknown to any set of colluding parties. In both cases, the result of the `AND` operation is 1 when the aggregate equals 0.

3.3.2. Arithmetic on encrypted polynomials. Many MPSI protocols (e.g. [1], [4], [7]) rely on arithmetic over private polynomials. One can do so using any method that privately performs arithmetic over integers, such as secret sharing or additively homomorphic encryption. By working over the coefficients of the polynomial, adding polynomials only requires one to add the coefficients of the two polynomials in element-wise fashion. Multiplying one private and one plaintext polynomial can be done using school-book multiplication [7]. Evaluating the polynomial at a plaintext value x requires computing the plaintext powers $x^2, \ldots, x^d$ up to degree d and performing a dot product. Instead of working over the coefficients, Cheon et al. [23] translate the polynomials to a point-value representation, which makes polynomial multiplication computationally cheaper. We discuss this in more detail in Section 5.1.1.

3.3.3. Binning techniques. If an MPSI protocol does not scale linearly with the number of elements k, binning techniques can improve the over-all efficiency by splitting the problem into several small MPSI problems; one for each bin. There are two kinds of binning techniques: those where a bin can have at most one element, and those where bins can have any number of elements, but preferably as small as possible. One popular technique for assigning a bin at most one element is that of Cuckoo hashing [24], which populates a set of m bins by repeatedly inserting an element into a bin selected by a hash function. If that bin was occupied, the element is placed into another bin, evicting the element that was previously there and must now be re-inserted. A suitable set of parameters makes this process highly likely to terminate and succeed. If multiple elements may occupy the same bin, one does not have to consider eviction. One technique called balanced allocations [25] distributes k elements over $m = \frac{k}{\ln \ln k}$ bins by repeatedly selecting two random bins and assigning the element to the most empty bin. The number of elements in each bin is then bounded by $O(\ln \ln k)$ with overwhelming probability.

TABLE 1. SUMMARY OF THE NOTATION IN THIS WORK

Symbol	Description
n	Number of parties
t	Maximum number of colluding parties
$\mathcal{P}_i, X_i$	The ith party and its input set
k	Maximum set size
$\mathcal{U}$	Universe
$\mathcal{S}$	Simulator
view_i	View of party $\mathcal{P}_i$
$\overset{s}{\equiv}, \overset{c}{\equiv}$	Statistical and computational indistinguishability
λ	Statistical security parameter
$\{\ldots\}$	Some unordered (multi)set
$[\ldots]$	Some ordered vector
$\uplus$	Multiset sum
$\hat{X}, m$	Set representation of set X and its number of bins
$\mathsf{Enc}, \mathsf{Dec}$	Encodes or decodes a set in a set representation
$\odot$	Homomorphism of the set representation

4. Common constructions

All MPSI schemes thus far can be expressed as a series of membership checks. This is possible because the result of a set intersection is always a subset of each party's input. Since privacy-preserving set intersections may not leak any of the intermediate computations, not all constructions are possible. We identify three remaining general constructions that are used to construct MPSI protocols.

In the first construction, the parties represent their sets so that they can be combined, and the resulting representation can be revealed to the leader. In this case, the membership queries can be done in plain text. In the second construction, the resulting representation is queried by the leader, because revealing it would leak information about the input sets. In the third construction, the leader queries each other set separately, then combines the outcomes to only reveal that an element is in the intersection if it was in *all* input sets.

Since these methods rely on ways of encoding sets into new, convenient representations, we introduce the following notation: $\hat{X} \leftarrow \mathsf{Enc}(X)$ is a representation of set X by some encoding Enc. The encoded set can be extracted using $X \leftarrow \mathsf{Dec}(\hat{X})$, but note that this function sometimes requires knowledge of a compact superset of X. It is always possible to use the universe $\mathcal{U}$ for this purpose, but this is not efficient when it contains many elements. In the remainder of this section we present the currently used set representations for each general construction in the same order as above.

4.1. Private homomorphic set representations

Private homomorphic set representations are set representations that can be homomorphically combined using some operation $\odot$ to compute the set intersection:

$$X_1 \cap \cdots \cap X_n \approx \mathsf{Dec}(\mathsf{Enc}(X_1) \odot \cdots \odot \mathsf{Enc}(X_n)) \, . \quad (4)$$

Crucially, the resulting representation does not reveal anything about the original inputs. In other words, the result is computationally indistinguishable from the actual encoding of the intersection.

$$\mathsf{Enc}(X_1) \odot \cdots \odot \mathsf{Enc}(X_n) \overset{c}{\equiv} \mathsf{Enc}(X_1 \cap \cdots \cap X_n)\ . \quad (5)$$

Given such a private homomorphic set representation, one can design an MPSI protocol given that $\odot$ can be efficiently computed in private. We discuss four such representations.

4.1.1. Bitsets. Bitsets are binary vectors that represent a set X by indicating for each element of the universe $x \in \mathcal{U}$ whether $x \in X$. To do so, there must exist some ordering over all elements in $\mathcal{U}$. Two bitsets can be homomorphically combined when their orderings agree. The homomorphism $\odot$ is then simply an element-wise `AND` operation.

Example 1. Given $\mathcal{U} = \{1, 2, 3, 4\}$ with ordering $[1, 2, 3, 4]$:

$$\begin{array}{rcr} \{1,3,4\} & \xrightarrow{\mathsf{Enc}} & [1,0,1,1] \\ \cap \quad \{1,2,3\} & \xrightarrow{\mathsf{Enc}} & \wedge \quad [1,1,1,0] \\ \hline \{1,3\} & \xleftarrow{\mathsf{Dec}} & [1,0,1,0] \end{array}$$

There is a clear drawback to using bitsets for MPSI: the representation requires $|\mathcal{U}|$ bits to encode a set, even if it is empty. On the other hand, bitsets always exactly represent the original set, even when combined. In other words, the condition in (5) actually holds under equality.

4.1.2. Hash sets. Another example of private homomorphic set representations comes in the form of hash sets. Like a bitset, a hash set is a binary vector. It has m bits, referred to as bins, initially set to 0, and a hash function $H : \mathcal{U} \mapsto \{1, \ldots, m\}$. To encode set X, one computes the index $H(x)$ for each element $x \in X$ and sets the bin at that index to 1. If two hash sets agree on the hash function, they can also be combined homomorphically using an `AND` operation. Decoding is not straightforward since it requires computing the inverse of a hash function H^{-1}. However, given a superset S of the elements possibly contained in the hash set, one can instead check for each element $s \in S$ whether it is in the hash set.

Example 2. For some common H:

$$\begin{array}{rcr} \{1,3,4\} & \xrightarrow{\mathsf{Enc}} & [1,1,1,0] \\ \cap \quad \{1,2,3\} & \xrightarrow{\mathsf{Enc}} & \wedge \quad [0,1,1,1] \\ \hline \{1,3\} & \xleftarrow{\mathsf{Dec}} & [0,1,1,0] \end{array}$$

Note that the encoded 1s are not in the same position as in a bitset; the positions are selected by hash function H.

Hash sets can be seen as Bloom filters with only one hash function; see Section 4.2.1. Note, however, that by increasing the number of hash functions, the representation opens up a mechanism for information leakage. Another way of looking at hash sets is as a bitset with Remark 2 applied to it. As such, (5) holds under equality.

The hash function H allows hash sets to require only m bits to represent a set, rather than $|\mathcal{U}|$. However, as a consequence, they do not exactly represent the original set. This occurs when H maps multiple elements to the same bin by H. If one of these elements is encoded in the hash set, all those other elements will falsely appear to be in the set. Using (7) with $h = 1$, such false positives can happen for each query with probability $\varepsilon \approx 1 - e^{\frac{k}{m}}$, where k is the number of elements encoded in the hash set, assuming that the output of H is statistically uniform. False negatives never occur in hash sets.

4.1.3. Sorted multisets. Blanton & Aguiar [26] propose sorted multisets to combine sets and compute their sorted intersection. The key idea is to combine multiple sets $X_1, \ldots, X_n$ into multiset $X_1 \uplus \cdots \uplus X_n$, and then isolate those elements with multiplicity n. By sorting the resulting multiset, there is a straightforward way of identifying those elements that appear n times. By also sorting the input sets, the sorted multiset can be computed without performing a full sort operation. Instead, sets can be combined using a simpler merging operation. To achieve this, Enc simply sorts the set. The homomorphism $\odot$ can then be computed by merging the sets, checking for elements that appear n times, and removing the other elements. This last step is dubbed 'monotonizing' by Poddar et al. [12]. Note that for a set to be sorted, a partial ordering must exist within $\mathcal{U}$.

A general way of merging two sorted sets is a bitonic sorter [27]. Such a circuit merges two sets of total length $k = 2^p$ in p steps. After that, selecting elements that appear n times can be done by checking equality for all elements that are n places apart. In currently known protocols [12], [26] the result then contains the elements of the intersection as well as 0s. To go back to a sorted set, those 0s can be removed using a monotonization circuit [12]. If the output must be revealed but does not have to be a sorted set without 0s, the output can be sorted [26] or shuffled [12] instead.

Example 3. Computing the homomorphism between two $(n = 2)$ sorted multisets $[1, 3, 4] \leftarrow \mathsf{Enc}(\{1, 3, 4\})$ and $[1, 2, 3] \leftarrow \mathsf{Enc}(\{1, 2, 3\})$ goes as follows:

$$\begin{aligned} [1,3,4] \odot [1,2,3] &= \mathsf{Mono}(\mathsf{Check}(\mathsf{Merge}([1,3,4],[1,2,3]))), \\ &= \mathsf{Mono}(\mathsf{Check}([1,1,2,3,3,4]))\ , \\ &= \mathsf{Mono}([1,0,0,3,0])\ , \\ &= [1,3]\ . \end{aligned}$$

A drawback of this representation is that the homomorphism is more complex to compute than the `AND` operations required to combine two bitsets or hash sets. However, some complexity is reduced by sorting the sets in advance. The benefit of sorted multisets is that they are both compact and exact: the representation's size is similar to the original set.

4.1.4. Polynomial roots. The roots of a polynomial form a multiset, so they can be used to compute intersections. To encode a set S in the roots of a polynomial in $\mathbb{Z}_q$, a mapping must exist from $\mathcal{U}$ to $\mathbb{Z}_q$. The polynomial encoding of S is:

$$\mathsf{Enc}(S) = P(x) = \prod_{s \in S} (x - s)\ , \quad (6)$$

which is a polynomial of degree $|S|$. Checking membership of a single element s is possible by evaluating the polynomial and checking if $P(s) = 0$.

Computing the polynomial that encodes the intersection of two polynomial set encodings requires computing the greatest common divisor, after which the roots of the resulting polynomial represent the multiset intersection of the original roots. The resulting polynomial does not contain any information other than the roots in the intersection, so this is a private homomorphic set representation [7].

One problem with applying the previous approach to MPSI protocols is that the homomorphism requires the parties to privately compute the greatest common divisor of two polynomials. Instead, Kissner & Song [7] show that one can still satisfy (5) when computing $\hat{X}_1 \odot \hat{X}_2$ as $r_1\hat{X}_1 + r_2\hat{X}_2 = \mathbf{r}\gcd(\hat{X}_1, \hat{X}_2)$, where r_1 and r_2 are random polynomials. To achieve this, we adapt the earlier encoding function, multiplying by some random polynomial r. This randomization also helps reduce false positives, which are now spread uniformly over the range of the coefficients.

In the following example, we demonstrate that addition of two polynomials indeed retains the common roots, even when they are not randomized.

Example 4. Consider the following without randomization:

$$\begin{aligned}&\mathsf{Enc}(\{1,3,4\}) \odot \mathsf{Enc}(\{1,2,3\})\\ &= (x-1)(x-3)(x-4) + (x-1)(x-2)(x-3)\\ &= 2(x-1)(x-3)^2 \ .\end{aligned}$$

Indeed, the roots are 1 and 3. Note that the root 3 appears twice, which is technically a false positive.

Polynomials are convenient set representations because they grow linearly with the size of the encoded set k, and the false positives are spread uniformly randomly over the space. A drawback is that polynomial multiplications by default scale quadratically with k.

4.2. Leaky homomorphic set representations

Leaky homomorphic set representations are set representations that satisfy (1), but for which (5) does not necessarily hold. In other words, the result of the homomorphism may leak information about the original sets. As such, these representations cannot be revealed to any of the parties, and instead, need to be privately queried to ensure that the parties only learn the intersection. Consequently, the representation must support an efficient way to perform private membership queries.

4.2.1. Bloom filters. A Bloom filter is a hash set with multiple hash functions $H_1, \ldots, H_h$, see Section 4.1.2. The benefit of using $h > 1$ is that the number of bins m can be lower for the same false positive rate ε. As a consequence however, computing the intersection between two or more Bloom filters through an AND operation may leak more information about the original sets than just their intersection. In other words, (5) does not hold.

The probability ε that a query falsely returns a positive result is approximately given by [28]:

$$\varepsilon \approx \left(1 - \left(1 - \frac{1}{m}\right)^{hk}\right)^h \approx \left(1 - e^{\frac{-hk}{m}}\right)^h \ . \tag{7}$$

So the minimal number of bins m for a Bloom filter with at most k elements and a false positive rate of ε is:

$$m = -\frac{k \ln \varepsilon}{\ln^2 2} \ , \quad h = -\log_2 \varepsilon \ . \tag{8}$$

The false positive rate after computing $\odot$ is bound by the false positive rate of the original Bloom filters [29]. Instead of using the approximation (7), one can also use the upper bound by Goel & Gupta [28] to choose parameters as described by Vos et al. [20].

The reason that Bloom filter intersections leak information is that a bin can be set to 1 even if it does not contribute to the intersection. The probability of this happening grows with the number of hash functions h. In the following example we show a situation where the filter of the actual intersection does not match the filter computed using $\odot$.

Example 5. Consider a Bloom filter with $m = 6$ and $h = 2$. The bins are indexed using $0, \ldots, m-1$. The first hash function behaves as $H_1(2) = 2$, $H_1(3) = 1$, $H_1(4) = 1$. The second as $H_2(2) = 3$, $H_2(3) = 2$, $H_2(4) = 5$.

Now, given sets $X_1 = \{2, 3\}$ and $X_2 = \{4\}$, we have:

$$\begin{aligned}\mathsf{Enc}(X_1 \cap X_2) &\neq \mathsf{Enc}(X_1) \wedge \mathsf{Enc}(X_2) \ ,\\ \mathsf{Enc}(\emptyset) &\neq [0,1,1,1,0,0] \wedge [0,1,0,0,0,1] \ ,\\ [0,0,0,0,0,0] &\neq [0,1,0,0,0,0] \ .\end{aligned}$$

The benefit of Bloom filters is that they scale linearly with the size of the set k. However, the drawback is that the constant factor of $O(k)$ can be large if ε must be small. For example, if $\varepsilon \approx 2^{-40}$ then $m \approx 57.7k$. In other words, it takes almost 58 bins to represent each element with negligible failure rate. A bitset requires $|\mathcal{U}|$ bins and has zero failure rate, so it stands to reason that one would then better choose a bitset representation if $|\mathcal{U}| \leq 57.7k$.

4.2.2. Garbled Bloom filters. In a *garbled* Bloom filter, the bins selected by the hash functions $H_1, \ldots, H_h$ are not set to 1, but to an XOR-sharing of some value. This makes a garbled Bloom filter a key-value store (see Section 4.3.3). The bins in such a filter are bitstrings of length λ. Garbled Bloom filters were first proposed by Dong et al. [30].

Let b_i denote the ith bin of a garbled Bloom filter. Kolesnikov et al. [3] provide the following algorithm to encode a set of key-value pairs in such a filter:

1) Initialize all bins b_i for $i = 1, \ldots, m$ to $\perp$.
2) For each key-value pair (x, y), select indices $I_j = H_j(x)$ for $j = 1, \ldots, h$. For empty bins ($b_{I_j} = \perp$), choose random λ-bit strings so that $b_{I_1} \oplus \cdots \oplus b_{I_h} = y$.
3) Replace empty bins ($b_i = \perp$) with a random λ-bit string.

A simple way of using garbled Bloom filters to encode sets for performing set intersections is to encode the set elements as the keys of the filter, and set the values to $0 \ldots 0$, or some other well-formed value. The filter representing the intersection can be computed using an element-wise XOR operation. Only the bins pertaining to elements in the intersection then XOR to $0 \dots 0$. Garbled Bloom filters inherit their *false negative* probability from the *false positive* probability Bloom filters. They also incur a chance of false positives, which occur when randomly chosen bins accidentally XOR to $0 \dots 0$. This happens with probability $2^{-\lambda}$.

In the following example we show why these filters leak information through the homomorphism.

Example 6. Consider a garbled Bloom filter with $m = 5$ bins, $h = 3$ hash functions and $\lambda = 3$. The hash functions behave like $H_1(1) = 0, H_2(1) = 1, H_3(1) = 2$, and $H_1(4) = 2, H_2(4) = 4, H_3(4) = 3$. Then, the set intersection between $\{1, 4\}$ and $\{1\}$ can be computed like:

$$\begin{array}{rlcrl}
 & \{1,4\} & \xrightarrow{\textsf{Enc}} & & [010, 011, 001, 010, 011] \\
\cap & \{1\} & \xrightarrow{\textsf{Enc}} & \oplus & [001, 100, 101, 110, 110] \\
\hline
 & \{1\} & \xleftarrow{\textsf{Dec}} & & [011, 111, 100, 100, 101]
\end{array}$$

Notice that the party holding the second set can infer that the other party has 4 in its set with high probability. It can do so because the sum of the bins pertaining to 4 sum to the same value in its own garbled Bloom filter as well as the result: $101 \oplus 110 \oplus 110 = 101 = 100 \oplus 100 \oplus 101$.

The most significant benefit of garbled Bloom filters over regular Bloom filters is their ability to be used as oblivious key-value stores. That is, if the values they encode are statistically random, then it is unknown which keys are encoded into them. Moreover, the number of bins m decides the probability of false negatives rather than false positives.

4.3. Aggregatable membership queries

A third construction for MPSI protocols performs membership queries on the encoded sets and combines the results of these queries. As such, no homomorphism is required over the set representation. Instead, we require an aggregation method $\odot$ for which it holds that:

$$\bigwedge_{i=1}^{n} x \in X_n = \mathsf{Reveal}(\mathsf{Query}_x(\hat{X}_1) \odot \cdots \odot \mathsf{Query}_x(\hat{X}_n)) \,, \tag{9}$$

where $\hat{X}_i \leftarrow \mathsf{Enc}(X_i)$ conveniently encodes set X_i.

Essentially, these approaches execute two-party protocols between the leader and the assistants, and combine the results. Since this high-level construction requires that the output of the two-party protocols remains private, all three approaches discussed here output a value linked to the queried element rather than a Boolean. For example, MPSI protocols could return a secret share of 0 for each of the elements in the queried party's set, and randomness otherwise. If so, $\odot$ is simply the reconstruction operation of the secret sharing scheme.

4.3.1. Embedding payloads in polynomial roots. In Section 4.1.4, we saw that polynomials encoding a set in their roots can be homomorphically combined. The same encoding can also be used to return aggregatable query results from a polynomial $p(x)$ by privately computing $rp(x) + P$, where P is the payload and x is the queried element. The result is P only when evaluated over any of the roots of p with overwhelming probability, or randomness otherwise. This method is particularly useful in protocols based on homomorphic encryption, because it allows the leader to locally evaluate the assistant's encrypted polynomial. Consequently, the leader can decide on the payload P. The leader can generate secret shares of zeroes to embedded as payloads without communicating with the assistants.

4.3.2. Oblivious programmable PRFs. Proposed by Kolesnikov et al. [3], an oblivious programmable PRF (OPPRF) is an oblivious PRF that contains hardcoded values in the form of key-value pairs. In other words, given a secret input x', if x' matches one of the key-value pairs (x, y) so that $x = x'$, then the output is y. Otherwise, the output is randomness. The approach in Section 4.3.1 already satisfies a form of this, where the keys are given by the roots of the polynomial and the values are decided on by the evaluator of the polynomial. OPPRFs, however, are more specific: the sender decides on the hardcoded key-value pairs. Importantly, the receiver is only allowed to perform a limited number of queries.

Kolesnikov et al. [3] propose three instantiations of OPPRFs. The first design interpolates a polynomial over the hardcoded key-value pairs, the second uses a garbled Bloom filter (see Section 4.2.2), and the third returns a key generated by a pseudo-random function that can be used to decrypt one of a set of encryptions if it pertains to one of the hardcoded keys. The authors also show how to extend the OPPRFs to be queried multiple times using cuckoo filters. The polynomial roots approach in Section 4.3.1 requires the polynomial to be evaluated in private, which is expensive, while these OPPRFs only rely on symmetric primitives.

4.3.3. Oblivious key-value stores. Garimella et al. [31] describe a primitive similar to OPPRFs called oblivious key-value stores (OKVS), which perform the same functionality but they can be transmitted in plain text. The polynomial-based OPPRF described above is an example of such an OKVS. In fact, this is the most compact OKVS possible, as it takes c polynomial coefficients to hard-code c key-value pairs. Crucially, the hardcoded values must be randomly distributed to hide which keys are encoded in the data structure. While it is compact, a drawback of this polynomial-based OKVS is that encoding it is computationally expensive.

Garimella et al. [31] propose an OKVS that is significantly cheaper to encode based on cuckoo hashing. While it is not optimally compact, it is approximately 1.5 to 2.5 times larger than the polynomial-based OKVS [32]. Moreover, by the way this OKVS is constructed, one can guarantee that encoding succeeds with overwhelming probability $(1-2^{-40})$. One of the most significant advantages of OKVSs over OPPRFs is that they reduce the communication needed for MPSI protocols in the malicious model.

This primitive originates from the garbled Bloom filters by Dong et al. [30], after which they were applied to two-party protocols under the name of 'PaXoS' [33]. In parallel, the field of structured linear functions studies similar data structures. One example is the frayed ribbon filter [34].

5. Proposed protocols

In this section, we provide a comprehensive overview of collusion-resistant MPSI protocols in the semi-honest model. We established this body of literature by exploring the references and citations of an initial set of works on MPSI protocols. In this process, we disregard works that rely on differential privacy as they do not satisfy the privacy requirements established in Section 2. We also do not consider server-aided protocols, which rely on a non-colluding assumption. Note that we only consider set intersection protocols, so we omit variants such as threshold-intersections [35] from the overview.

We summarize all protocols that do not have any known security flaws in Table 2. This table sorts the works by their high-level construction as discussed in the previous section and the set representation. Within these categories, the protocols are sorted chronologically. For each work, we state their communication and computation complexity, as well as the number of rounds of interaction. We provide derivations of these complexities in Appendix A. We also state the network topologies, the maximum collusion resistance, and whether there also exists a maliciously-secure version of the protocol (yes ● or no ○). We do not consider whether a protocol is size-hiding, see Remark 3 in Section 2.2. Note that complexities using $\tilde{O}$-notation omit logarithmic terms.

Remark 4. Note that in the semi-honest model, any protocol with a mesh topology can be converted to a star topology by encrypting all messages for the intended receiver and routing them through the leader at the cost of more interactions.

5.1. Private homomorphic set representations

5.1.1. Using polynomial roots. Kissner & Song [7] propose an MPSI protocol based on the polynomial roots representation. After encoding their set in the roots of a polynomial, each party uses an additively homomorphic cryptosystem (the Paillier cryptosystem) to encrypt the coefficients and sends them to t other parties. Each party locally randomizes the received polynomials, after which the parties sum up the result in a circular fashion, and finally decrypt. Li & Wu [11] propose a similar approach, except they use secret sharing. This lowers the maximum collusion threshold from $n-1$ to $\lfloor \frac{n-1}{2} \rfloor$, but the computation is significantly cheaper. Patra et al. [43], [44] show that the malicious version of Li & Wu's protocol requires more operations than claimed in the original paper, and they provide alternatives with a higher maximum collusion threshold and better efficiency.

After this, Sang et al. [36] discuss several steps in optimizing the randomized aggregation operation necessary to compute the intersection in the semi-honest model. In [37], they propose a malicious extension of this protocol. The protocols use degree 1 polynomials when randomizing to save computations, and provide proof that this is indeed still resistant against collusion attacks given that the coefficients are multiplied by a non-singular matrix. In later works, Sang & Shen [38], [45] describe a second protocol based on bilinear pairings. However, the pairings are only necessary in the malicious model. We note that it suffices to use elliptic curve-based ElGamal in the semi-honest model, for example. The idea behind this protocol is that it suffices to randomize each polynomial once using a single scalar instead of $t+1$ times by different parties if we only reveal the evaluation of the resulting polynomial. Importantly, this is only secure for cryptosystems where parties cannot determine the discrete log of the final decryption. For this reason, we mark PHE with an asterisk in Table 2.

Instead of eliminating polynomial multiplications, some other works focus on lowering their computational cost. Cheon et al. [23], for example, propose a protocol based on that of Kissner & Song [7] where polynomials are in point-value representation: instead of encrypting a polynomial by its coefficients, the parties first evaluate the polynomial locally on a set of public points, and encrypt the resulting values. Multiplication of polynomials in this representation scales linearly with the degree rather than quadratically. However, one must ensure there are at least as many encrypted values as the degree of the resulting polynomial. Focusing on the two-party setting, Kim et al. [46] propose several other techniques to perform computationally cheaper polynomial multiplications and evaluations.

Ghosh & Nilges [9] use the same point-value representation as above, but they show how to compute the randomized sum without the need for homomorphic encryption. Instead, they propose to have every assistant perform a series of OLEs with the leader. In the setting with two parties $\mathcal{P}_1$ and $\mathcal{P}_2$ where only $\mathcal{P}_1$ receives the result, the parties compute $r\hat{X}_1 + \hat{X}_2$ using an OLE, where r is chosen by $\mathcal{P}_2$. In the multi-party setting, the authors propose to use an OLE between each assistant and the leader in both directions, to achieve a secret sharing of the resulting polynomial. By clever use of masking and a non-interactive secret sharing scheme, the authors make sure that the leader must aggregate all randomized polynomials before being able to reveal the intersection. Note that recently, Abadi et al. [47] identified multiple attacks against the version of the protocol that was claimed to withstand malicious adversaries. The authors compare the bandwidth cost of their protocol with that by Kolesnikov et al. [3]. For $k = 2^{20}$ and $t = n-1$, the bandwidth per party is ≈ 1.25 GB for the protocol by Ghosh & Nilges, and $(n-1) \cdot 467$ MB for the previous work.

The latest work in this category comes from Gordon et al. [13], who propose an MPSI protocol secure in the malicious model. They also propose a version where all parties receive the result. The protocols use OLEs to compute the

TABLE 2. Overview of semi-honest MPSI protocols with no known attacks, resisting multiple colluders

Work		**Technique**		**Communication**				**Computation**		**Security**	
First author	Year	Encoding	Primitive	Topology	Leader	Assistant	Rounds	Leader	Assistant	Collusion	Malicious
Private homomorphic set representations											
Kissner [7]	2005	Polynomial roots	PHE	Mesh	$O(nk)$	$O(tk)$	3	$O(tk^2)$	$O(tk^2)$	$n-1$	●
Sang [36]	2006		PHE	Mesh	$O(nk)$	$O(nk)$	3	$O(n^2k)$	$O(n^2k)$	$n-1$	[37]
Li [11]	2007		-	Mesh	$O(ntk^2)$	$O(ntk^2)$	3	$O(ntk^2)$	$O(ntk^2)$	$\lfloor \frac{n-1}{2} \rfloor$	●
Sang [38]	2009		PHE*	Mesh	$O(nk)$	$O(nk)$	2	$O(nk^2)$	$O(nk^2)$	$n-1$	●
Cheon [23]	2012		PHE	Mesh	$O(nk)$	$O(nk)$	3	$O(nk)$	$O(nk)$	$n-1$	●
Ghosh [9]	2019		OLE	Star	$O(nk)$	$O(k)$	6	$\tilde{O}(nk)$	$\tilde{O}(k)$	$n-1$	●
Gordon [13]	2022		OLE	Star	$\tilde{O}(nk+nt)$	—	5	—	—	$n-1$	●
Blanton [26]	2016	Sorted multiset	-	Mesh	$\tilde{O}(tk)$	$\tilde{O}(tk)$	$O(\log k)$	$\tilde{O}(tk)$	$\tilde{O}(tk)$	$\lfloor \frac{n-1}{2} \rfloor$	●
Poddar [12]	2021		GC	Mesh	—	—	—	—	—	$n-1$	●
Bay [39]	2021	Bitset	PHE	Star	$O(tk)$	$O(\vert\mathcal{U}\vert)$	3	$O(nk)$	$O(\vert\mathcal{U}\vert)$	$n-1$	○
Vos [20]	2022										
Leaky homomorphic set representations											
Inbar [8]	2018	Garbled Bloom filter	OT	Star	$O(nkh)$	$O(nkh)$	3	$O(nkh)$	$O(nkh)$	$n-1$	[40]
Kavousi [41]	2021		OT	Wheel	$O(nk)$	$O(kh)$	4	$O(nk)$	$O(kh)$	$n-1$	○
Bay [5]	2022	Bloom filter	PHE	Star	$O(tk)$	$O(k)$	3	$O(nkh)$	$O(k)$	$n-1$	○
Vos [20]	2022										
Aggregatable membership queries											
Freedman [1]	2004	Polynomial payloads	-	Mesh	$O(n^2k^2)$	$O(n^2k^2)$	4	$O(n^2k^2)$	$O(n^2k^2)$	$n-1$	○
Hazay [4]	2017		PHE	Star	$O(nk)$	$O(k)$	4	$\tilde{O}(nk)$	$O(k)$	$n-1$	●
Kolesnikov [3]	2017	OPPRF	OT	Mesh	$O(nk)$	$O(tk)$	4	$O(n)$	$O(tk)$	$n-1$	[31]
Chandran [42]	2021		OT	Star	$\tilde{O}(nk)$	$\tilde{O}(k)$	8	$O(nk)$	$O(k)$	$\lfloor \frac{n-1}{2} \rfloor$	○
Garimella [31]	2021	OKVS	OT	Star	$O(nk)$	$O(k)$	4	$O(k)$	$O(nk)$	$n-1$	●
Nevo [32]	2021		OT	Mesh	$O(tk)$*	$O(k)$	4	$O(nk-tk)$	$O(tk)$	$n-1$	●

randomized polynomial sum:

$$\hat{X}_\cap = \left(\sum_{i=1}^{n} r_i\right)\hat{X}_1 + \sum_{i=2}^{n}(s_1^i + s_i)\hat{X}_i \,. \tag{10}$$

Here, r_i and s_i are random polynomials selected by each party, and s_1^i are $n-1$ polynomials sampled by the leader. For each set element, each assistant performs four OLEs with the leader. For $k = 2^{20}$ and $t = n-1$, the malicious protocol by Gordon et al. consistently outperforms Kolesnikov et al. in the semi-honest model. Care must be taken to interpret these numbers as the experiments were run on different hardware (3.60GHz CPUs versus 2.30GHz).

5.1.2. Using sorted multisets. Huang et al. [48] use sorted multisets to perform a private set intersection between two parties, and Poddar et al. [12] provide an extension to any number of parties. Both works use garbled circuits (GC) to privately compute the intersection between sorted multisets. While garbled circuits by default offer a way to privately perform a two-party computation, Poddar et al. use the method by Wang et al. [49], which first uses an expensive offline phase to build a garbled circuit that can be used for secure multi-party computation. Since the MPSI operation is part of a large compiler pipeline, it is hard to state any asymptotic complexities for the resulting protocol.

Before the work by Poddar et al., Blanton et al. [26] used the same representation to compute intersections and other set and multiset operations using secret sharing. The benefit of this method is that computation is extremely efficient, but the protocol does not run in a constant number of rounds due to the merging operations: it scales logarithmically with the size of the sets k. For $n = 3$, $t = 1$, $k = 2048$ and with a 1Gb Ethernet connection, the protocol takes 25 seconds.

5.1.3. Using bitsets. Ruan et al. [50] proposed the first bitset-based private set intersection protocol. Their protocol is restricted to two parties, so Bay et al. [39] provide an extension to the multi-party setting, achieving up to $n-1$ collusion resistance. The protocol uses additively homomorphic encryption to compute an element-wise `AND` operation between all bitsets using the method described in Section 3.3.1. Vos et al. [20] show how to perform this operation in a way that is significantly cheaper. By using ElGamal over an elliptic curve group, operations are less computationally demanding, and the ciphertext size is restricted to 64 bytes. Another improvement comes from reordering the randomization, so that the leader has to perform fewer plaintext multiplications. Without considering communication, these optimizations reduce run time from 100 seconds to 3 seconds for $k = 2^{12}$, $n = 5$ and $t = 4$.

5.2. Leaky homomorphic set representations

5.2.1. Using garbled Bloom filters. The first protocol based on garbled Bloom filters was a two-party private set intersection by Dong et al. [30]. Inbar et al. [8] provided three protocols that extend this protocol to the multi-party setting. The first protocol is in the server-aided model, relying on a non-colluding third party, while the next two protocols are collusion-resistant and secure in the semi-honest and augmented semi-honest model, respectively. The latter two protocols rely on oblivious transfers so each party only learns the contents of the garbled Bloom filter bins that they should learn. The augmented semi-honest model allows the parties to perform fewer OT extensions while realizing the same functionality. While OTs are cheap to compute, the protocols require all pairs of parties to perform them, causing the communication to scale quadratically with the number of parties. Recently, Ben-Efraim et al. [40] provided an extension of this work to the malicious model. They also show how to reduce the number of OTs by 25% by choosing other parameters for the garbled Bloom filter. Their results for $t = n - 1$ show that regardless of k and n, Kolesnikov et al. and Ben-Efraim et al. consistently outperform the work by Inbar et al. Moverover, for $n \geq 10$ and $k \geq 2^{16}$ the malicious protocol by Ben-Efraim et al. consistently outperforms the semi-honest protocol by Kolesnikov et al.

A protocol by Kavousi et al. [41] uses garbled Bloom filters differently, extending the two-party work of Chase & Miao [22], who propose a highly efficient OPRF that can be queried in multiple points at once using OTs. Bay & Kayan [51] also propose a multi-party extension, but without a garbled Bloom filter and therefore require the asisstants to send the PRF of the hash of each element to the leader. We argue that this renders the protocol insecure, as the leader can enumerate the universe $\mathcal{U}$ to identify preimages.

5.2.2. Using Bloom filters. Bay et al. [5] propose a collusion-resistant version of the protocol by Miyaji & Nishida [6]. Their protocol uses the Paillier cryptosystem to compute the `AND` operation between each party's encrypted Bloom filter following the technique described in Section 3.3.1. Vos et al. [20] provide a similar improvement as described in Section 5.1.3 to significantly reduce the computation and communication cost of the protocol, although it does not impact the asymptotic complexities. These protocols satisfy superset correctness: they only incur false positives with non-negligible probability. Vos et al. show that for $\varepsilon \geq 0.01\%$, $t = n - 1$ and $n \geq 3$, the protocol consistently outperforms Kolesnikov et al. [3].

5.3. Aggregatable membership queries

5.3.1. Embedding payloads in polynomial roots. Freedman [1] proposed the first MPSI protocol. This secret sharing-based protocol uses the polynomial set encoding to embed payloads in shares containing the evaluation of the polynomial in the elements of the leader's set. This payload is a secret share, so when all polynomials contain a root at this element, the secret can be reconstructed. Hazay & Venkitasubramaniam propose another protocol using PHE, which allows the leader to evaluate the polynomials, instead of the assistants. The leader only inserts payloads of 0. After randomizing and decrypting (reminiscent to Section 3.3.1), the leader receives a result of 0 when the element is in the intersection with overwhelming probability.

5.3.2. Using oblivious programmable PRFs. Kolesnikov et al. [3] introduce OPPRFs and show how they can be used to instantiate an MPSI protocol. The first part of the protocol is expensive, establishing an `XOR`-sharing of zero for each element that could be in the intersection. After that, the parties hardcode those values in their OPPRF for each element in their set. The leader then queries all assistants' OPPRFs and reconstructs the secret. Chandran et al. [42] create three modifications of the protocol by Kolesnikov et al., removing the expensive construction of secret shares that `XOR` to zero, replacing it with Shamir's secret sharing scheme. As a consequence, collusion resistance is halved. In their results, this leads to a 1.2 to 6.2 times speedup over Kolesnikov et al. in different settings. For $n = 15$, $t = 7$, and $k = 2^{20}$, protocol C took 244.8 seconds to complete over WAN, while Kolesnikov et al. took 1524.5 seconds.

5.3.3. Using oblivious key-value stores. Garimella et al. [31] propose OKVSs as a way of reducing the communication cost required in OPPRF-based protocols. The authors show how to use VOLEs to optimize previous protocols, and they instantiate them with oblivious transfers. They show that in slow networks, this approach leads to faster protocol executions. Nevo et al. [32] lower both the computational and communication cost compared to previous protocols. They do so by cleverly using a set of 'pivot' parties that hold the OKVSs, reducing the overall cost. At the same time these pivot parties directly decide the protocols resistance to collusions. For this reason, when $t = n - 1$, the protocol converges with that of Garimella et al. [31]. In Table 2 we mark the leader's communication complexity with an asterisk: When $t \leq \frac{n}{2}$, the leader's communication is instead given by $O(nk - tk)$. Qiu et al. [17] consider the protocol by Garimella et al. in a tree-shaped network topology and provide a maliciously-secure protocol. This lowers the computational cost while changing the number of rounds to $2\lfloor \log_2(n + 1) \rfloor$. Nevo et al. consistently outperform the works by Kolesnikov et al., Ben-Efraim et al., and Chandran et al. in terms of required bandwidth and run time. An exception is when $t = n - 1$, the run time is then the same as that of Kolesnikov et al. The authors report that when

$n = 15$, $t = 7$, and $k = 2^{20}$, the protocol takes 3 minutes to execute and 1416.9 MB of bandwidth.

6. Common pitfalls

In this section, we put forward several common pitfalls in designing MPSI protocols and analyze the ways that these vulnerabilities present themselves in these protocols.

6.1. Leakage from set representations

As we established in Section 4, there are two classes of homomorphic set representations: those that remain private when revealed and those that leak information about the inputs. While it is not always obvious how leaky set representations expose private data, the leakage is not negligible. A common mistake is that a Bloom filter representing the intersection is revealed to some of the parties. This happens for example in the work by Many et al. [52]. Another work by Lai et al. [53] even uses Bloom filters in plain text, but this allows an attacker to brute force all possible elements encoded within it. In other cases, the protocol involuntarily leaks information about the Bloom filter or its queries. For example, the protocols by Debnath et al. [54], [55] leak the sum of the queried bins, which reveals information about how many parties have a given element in their sets.

6.2. Unsafe randomness during aggregation

As mentioned in Section 3.3.1, many MPSI protocols rely on privately computing an `AND` operation. For bits $x_1, \ldots, x_n$, one way is to compute $\mathbf{r}(x_1 + \cdots + x_n)$, where $\mathbf{r}$ is some randomness unknown to any set of colluding parties.

One might think that the requirement above is met when we let the parties compute $r_1(1 - x_1) + \cdots + r_n(1 - x_n)$, where r_i is some randomness only known to party $\mathcal{P}_i$. However, whether parties know $\mathbf{r}$ now depends on the inputs. Notice, for example, that when $x_1 = 0$ and $x_2, \ldots, x_n = 1$, $\mathbf{r} = r_1$, which is known to party $\mathcal{P}_1$ and anyone they collude with. This allows a set of colluding parties to tell if they are the only parties with an input bit of 1.

A related error was made in the protocol by Wei et al. [56], which privately computes:

$$y = (s_1 x_1 + r'_1(1 - x_1)) + \cdots + (s_n x_n + r'_n(1 - x_n)) \ , \quad (11)$$

where s_i is a secret share of 0 encoded in the exponent of a generator, so it holds that $s_1 + \cdots + s_n = 0$. A share is privately constructed as follows: $s_i = \sum_{j=1}^{i-1} r_j - \sum_{j=i+1}^{n} r_j$, where $r_1, \ldots, r_n$ is randomness generated by parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, respectively. Importantly, a party does not know its share until (11) is computed. At first sight, this seems to solve the earlier issue. However, since party $\mathcal{P}_j$ knows the randomness r_j that they contributed, they can remove their contribution from other parties shares $s_{j'}$ for $j' \neq j$. This allows a party to tell that it is the only party with an input bit of 1.

Another mistake is to only let one party contribute randomness. Doing so reduces the maximum collusion resistance to 1. This mistake was made in the protocol by Miyaji & Nishida [6], and was patched by Bay et al. [5]. In this protocol, only the leader $\mathcal{P}_1$ randomized the sum. In other words, the parties compute $r_1(x_1 + \cdots + x_n)$. As a result, the leader can invert this randomization at the end of the protocol to extract the number of input bits set to 1.

6.3. Adapting to the malicious model

While this paper focuses on the semi-honest model, one should note that translating protocols from this model to the malicious model comes with its own set of challenges. For example, the polynomial set representation as described in Section 4.1.4 is susceptible to manipulation by malicious adversaries [47]. While the first two attacks discussed by Abadi et al. are specific to the protocol of Gordon et al. [9], the third attack highlights a problem inherent to the point-value representation. Specifically, it allows an adversary to omit elements from the intersection by multiplying each element of the point-value representation by a scalar.

Qiu et al. [17] propose a different type of attack against the protocol by Garimella et al. [31] for returning the output of the MPSI to all parties. The attack allows an adversary to learn information about the elements held by honest parties. The attack does not affect the security of the OKVS.

7. Analytical evaluation of computational costs

The performance of MPSI protocols is affected by many parameters, including: its hardness assumptions, the security parameter, the network topology, collusion threshold t, the number of interactions, the number of parties n, the number of elements k, the error probability ε, latency, throughput, the distribution of computer power over different parties, and the number of available threads. As a result, any concrete comparison that makes assumptions about any of these parameters unfairly puts certain schemes at an advantage (in fact, almost any scheme can win in a specific setting), and without established public uses of MPSI protocols, it remains unclear what realistic sets of parameters look like.

Instead, we study the theoretical computational cost of MPSI protocols. We consider PHE-based protocols separately from protocols based on aggregatable membership queries, which rely on two-party computations (2PCs). For brevity, we denote $E = \ln(\varepsilon^{-1})$. At the end of this section we nevertheless try to describe what reasonable parameter sets might look like, and discuss which protocols suit them.

7.1. Efficiency of PHE-based protocols

We instantiate the latest version of each PHE-based protocol discussed in Section 5 using elliptic curve-based ElGamal, as used by Vos et al. [20]. Next, we analyze the computational cost of these protocols by counting the number of elliptic curve multiplications required to evaluate

TABLE 3. NUMBER OF PRIMITIVE OPERATIONS IN THE WORST CASE

	Leader	Assistant	Total
Elliptic curve multiplications per element			
Sang	$1.5 + 2n + 2k$	1.5	$4n + 2k + t + 1$
Cheon	$4n + 3$	$4n + 3$	$5n + 4nt + 2t + 2$
Hazay	$10n - 7$	3.5	$12n - 6 + 3t$
Vos	3	$1.04E + 3$	$1.04(n - 1)E + 3t + 3$
Oblivious transfers			
Inbar	$2.08(n - 1)Ek$	$2.08Ek$	$2.08(n - 1)Ek$
Kolesnikov	$7(n - 1)k$	$3.5(n - 1)k$	$3.5(n + 1)(n - 1)k$
Chandran	$8.96(n - 1)k$	$8.96k$	$8.96(n - 1)k$
Kavousi	$588(n - 1)$	588	$588(n - 1)\lambda$
Garimella	—	—	—
Nevo	$3.5tk$	$3.5k$	$3.5tk$
Oblivious linear evaluations			
Ghosh	$2(n - 1)k$	$2k$	$2(n - 1)k$
Gordon	$(n - 1)k$	k	$(n - 1)k$

them. Since it is possible for one curve point to precompute a basepoint table, we distinguish between regular and precomputed multiplications: precomputed multiplications are approximately four times cheaper. We report the results in Table 3. Note that these equations differ from the asymptotic complexities, because they do not consider homomorphic addition, which is negligible compared to multiplications.

Based on this, it depends on the use case which protocol is more computationally efficient. For example, if the assistants have low computational power, the protocol by Hazay et al. [4] is faster. Sang et al. [38] is also cheap for an assistant, but scales quadratically with k for the leader. If there is no difference in computational power and t is small, Cheon et al. [23] is faster. If t is close to n and ε can be high, then Vos et al. [20] may be cheaper.

7.2. Efficiency of 2PC-based protocols

In Table 3, we state the number of calls to the sub-protocols of the 2PC-based protocols from Section 5. Where possible we analyzed the protocols in the augmented semi-honest model. We ignore setup operations, such as the initial OT phase. It is important to note that we expressed these in terms of the sub-protocols used in the respective papers. However, one might also instantiate the protocol by Kolesnikov et al. [3], for example, with a VOLE-based OPPRF. Also note that Kavousi et al. [41] requires the parties to take part in a constant number of OTs, regardless of k. Other parts of the protocol do scale with k.

7.3. Analysis towards potential use cases

Use case	Setting	n	t	k	ε
(1) Mutual customers	Mesh	≤ 5	$n - 1$	$\leq 2^{20}$	5%
(2) Common IP addresses	Star	≤ 50	$\frac{n}{2}$	$\sim 2^{10}$	1%
(3) Consensus voting	Star	≥ 100	$\frac{n}{10}$	$\leq 2^{8}$	2^{-40}

We briefly analyze which protocols would be suitable for three highlighted applications. We present potential parameters in the table above. Application (1) involves some companies that want to identify mutual customers for an ad campaign, (2) involves cyber security organizations that want to identify suspicious IP addresses for investigation, and (3) involves a consensus vote between many parties. The companies in (1) run few multi-threaded servers that are always online. A good fit seems to be the work by Nevo et al. [32], which outperforms other 2PC approaches, supports $t = n - 1$, and is fast for large sets with few parties. With $n \leq 50$, (2) requires a lowly-interactive protocol in the star topology like Ghosh, Gordon, Chandran, Vos, Inbar, Hazay, or Garimella. For $\varepsilon = 1\%$ and $n = 50$, Vos only takes 313 multiplications compared to Hazay's 669. The same applies to (3), but it does not permit approximations. Depending on the network, one might choose Ghosh, Gordon, Chandran, or Hazay, as assistants' computation does not scale with n.

8. Discussion

Based on Sections 5 and 7 we observe that older works are still relevant today. Specifically, when instantiating PHE-based protocols with modern cryptosystems such as elliptic curve-based ElGamal, their performance becomes competitive with the latest proposals. Moreover, while it may be convenient to designate one protocol as the state of the art, this is not possible for MPSI protocols: it is possible for each protocol to find application settings where they require fewer calls to sub-protocols or fewer EC operations than other works. This also begs the question what real-life application settings look like: what are common parameters for n, t, and k, and what are the network settings like? Until then, direct comparisons cannot be considered conclusive.

When looking at how MPSI protocols come to be, the process sometimes starts with a two-party protocol, which is extended to a semi-honest protocol, and finally to a version that is maliciously secure. We see a trend where more recent works only propose a multi-party maliciously-secure protocol, skipping the semi-honest model where it might be explained more plainly and nuances are better understood.

When it comes to technical advances, we identify improvements in OKVSs and homomorphic cryptosystems as the most impactful. OKVSs can be optimized in isolation of the protocols that use them so they do not require significant knowledge of cryptography, and more efficient versions would immediately increase performance of the latest 2PC-based protocols. So far, we are not aware of any collusion-resistant MPSI protocols that use lattice-based homomorphic encryption, or which use leveled/fully-homomorphic encryption to outperform PHE-based protocols.

Finally, when comparing PHE-based MPSI protocols to 2PC-based protocols, we note that the first are easier to analyze for different application settings. This complexity in analyzing 2PC-based protocols comes from having to choose cryptographic parameters, which do not correlate in a straightforward manner with k, for example. Moreover, multiple sets of cryptographic parameters are suitable for each instance. At the same time, it is this flexibility that also allows the user of a protocol to tune between computation

Projects: 1) Improvements based on FHE, possibly w. packing? 2) Parameters choice legend. 3) Survey works beyond this survey: malicious security, server-aided, more? 4) New results on server-aided with t>1, ideally, t<all but one user, 5) New results: better OKVS? 6) Applications - implement (lab)

and communication, a feature that is not typical of PHE-based protocols. Future research could elaborate on how to choose these parameters, easing the analysis and deployment of 2PC-based protocols.

9. Conclusion

This work provides a systematization of collusion-resistant MPSI protocols, focusing on the semi-honest model. We describe the formal requirements that MPSI protocols must satisfy, and present high-level constructions that describe all published MPSI protocols. Next to that, we provide a comprehensive overview of collusion-resistant MPSI protocols and broken protocols, as well as an analytical evaluation of their computational cost. This evaluation shows that there is no such thing as a single state of the art, but rather that each protocol outperforms the others depending on the application setting. We highlight several future research directions, intending to bring the performance of MPSI protocols closer to that of two-party PSIs.

References

[1] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 1–19. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_1

[2] F. Kerschbaum, E. Blass, and R. A. Mahdavi, "Faster secure comparisons with offline phase for efficient private set intersection," *CoRR*, vol. abs/2209.13913, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.13913

[3] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 1257–1272. [Online]. Available: https://doi.org/10.1145/3133956.3134065

[4] C. Hazay and M. Venkitasubramaniam, "Scalable multi-party private set-intersection," in *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, ser. Lecture Notes in Computer Science, S. Fehr, Ed., vol. 10174. Springer, 2017, pp. 175–203. [Online]. Available: https://doi.org/10.1007/978-3-662-54365-8_8

[5] A. Bay, Z. Erkin, J. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1–15, 2022. [Online]. Available: https://doi.org/10.1109/TIFS.2021.3118879

[6] A. Miyaji and S. Nishida, "A scalable multiparty private set intersection," in *Network and System Security - 9th International Conference, NSS 2015, New York, NY, USA, November 3-5, 2015, Proceedings*, ser. Lecture Notes in Computer Science, M. Qiu, S. Xu, M. Yung, and H. Zhang, Eds., vol. 9408. Springer, 2015, pp. 376–385. [Online]. Available: https://doi.org/10.1007/978-3-319-25645-0_26

[7] L. Kissner and D. X. Song, "Privacy-preserving set operations," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 241–257. [Online]. Available: https://doi.org/10.1007/11535218_15

[8] R. Inbar, E. Omri, and B. Pinkas, "Efficient scalable multiparty private set-intersection via garbled bloom filters," in *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, ser. Lecture Notes in Computer Science, D. Catalano and R. D. Prisco, Eds., vol. 11035. Springer, 2018, pp. 235–252. [Online]. Available: https://doi.org/10.1007/978-3-319-98113-0_13

[9] S. Ghosh and T. Nilges, "An algebraic approach to maliciously secure private set intersection," in *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, ser. Lecture Notes in Computer Science, Y. Ishai and V. Rijmen, Eds., vol. 11478. Springer, 2019, pp. 154–185. [Online]. Available: https://doi.org/10.1007/978-3-030-17659-4_6

[10] Z. Wang, K. Banawan, and S. Ulukus, "Multi-party private set intersection: An information-theoretic approach," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 366–379, 2021. [Online]. Available: https://doi.org/10.1109/JSAIT.2021.3057597

[11] R. Li and C. Wu, "An unconditionally secure protocol for multi-party set intersection," in *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds., vol. 4521. Springer, 2007, pp. 226–236. [Online]. Available: https://doi.org/10.1007/978-3-540-72738-5_15

[12] R. Poddar, S. Kalra, A. Yanai, R. Deng, R. A. Popa, and J. M. Hellerstein, "Senate: A maliciously-secure MPC platform for collaborative analytics," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 2129–2146. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/poddar

[13] S. D. Gordon, C. Hazay, and P. H. Le, "Fully secure PSI via mpc-in-the-head," *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 3, pp. 291–313, 2022. [Online]. Available: https://doi.org/10.56553/popets-2022-0073

[14] Y. Lindell, "How to simulate it - A tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography*, Y. Lindell, Ed. Springer International Publishing, 2017, pp. 277–346. [Online]. Available: https://doi.org/10.1007/978-3-319-57048-8_6

[15] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, J. Simon, Ed. ACM, 1988, pp. 1–10. [Online]. Available: https://doi.org/10.1145/62212.62213

[16] C. Hazay and Y. Lindell, "A note on the relation between the definitions of security for semi-honest and malicious adversaries," *IACR Cryptol. ePrint Arch.*, p. 551, 2010. [Online]. Available: http://eprint.iacr.org/2010/551

[17] Z. Qiu, K. Yang, Y. Yu, and L. Zhou, "Maliciously secure multi-party PSI with lower bandwidth and faster computation," in *Information and Communications Security - 24th International Conference, ICICS 2022, Canterbury, UK, September 5-8, 2022, Proceedings*, ser. Lecture Notes in Computer Science, C. Alcaraz, L. Chen, S. Li, and P. Samarati, Eds., vol. 13407. Springer, 2022, pp. 69–88. [Online]. Available: https://doi.org/10.1007/978-3-031-15777-6_5

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Springer, 1999, pp. 223–238. [Online]. Available: https://doi.org/10.1007/3-540-48910-X_16

[19] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 10–18. [Online]. Available: https://doi.org/10.1007/3-540-39568-7_2

[20] J. Vos, M. Conti, and Z. Erkin, "Fast multi-party private set operations in the star topology from secure ands and ors," *IACR Cryptol. ePrint Arch.*, p. 721, 2022. [Online]. Available: https://eprint.iacr.org/2022/721

[21] P. Schoppmann, A. Gascón, L. Reichert, and M. Raykova, "Distributed vector-ole: Improved constructions and implementation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 1055–1072. [Online]. Available: https://doi.org/10.1145/3319535.3363228

[22] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious PRF," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, ser. Lecture Notes in Computer Science, D. Micciancio and T. Ristenpart, Eds., vol. 12172. Springer, 2020, pp. 34–63. [Online]. Available: https://doi.org/10.1007/978-3-030-56877-1_2

[23] J. H. Cheon, S. Jarecki, and J. H. Seo, "Multi-party privacy-preserving set intersection with quasi-linear complexity," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 95-A, no. 8, pp. 1366–1378, 2012. [Online]. Available: https://doi.org/10.1587/transfun.E95.A.1366

[24] R. Pagh and F. F. Rodler, "Cuckoo hashing," in *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, ser. Lecture Notes in Computer Science, F. M. auf der Heide, Ed., vol. 2161. Springer, 2001, pp. 121–133. [Online]. Available: https://doi.org/10.1007/3-540-44676-1_10

[25] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations (extended abstract)," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, F. T. Leighton and M. T. Goodrich, Eds. ACM, 1994, pp. 593–602. [Online]. Available: https://doi.org/10.1145/195058.195412

[26] M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations," *Int. J. Inf. Sec.*, vol. 15, no. 5, pp. 493–518, 2016. [Online]. Available: https://doi.org/10.1007/s10207-015-0301-1

[27] K. E. Batcher, "Sorting networks and their applications," in *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968*, ser. AFIPS Conference Proceedings, vol. 32. Thomson Book Company, Washington D.C., 1968, pp. 307–314. [Online]. Available: https://doi.org/10.1145/1468075.1468121

[28] A. Goel and P. Gupta, "Small subset queries and bloom filters using ternary associative memories, with applications," in *SIGMETRICS 2010, Proceedings of the 2010 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, New York, USA, 14-18 June 2010*, V. Misra, P. Barford, and M. S. Squillante, Eds. ACM, 2010, pp. 143–154.

[29] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for bloom filters," *Distributed Parallel Databases*, vol. 28, no. 2-3, pp. 119–156, 2010. [Online]. Available: https://doi.org/10.1007/s10619-010-7067-2

[30] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, A. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM, 2013, pp. 789–800. [Online]. Available: https://doi.org/10.1145/2508859.2516701

[31] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. Malkin and C. Peikert, Eds., vol. 12826. Springer, 2021, pp. 395–425. [Online]. Available: https://doi.org/10.1007/978-3-030-84245-1_14

[32] O. Nevo, N. Trieu, and A. Yanai, "Simple, fast malicious multiparty private set intersection," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1151–1165. [Online]. Available: https://doi.org/10.1145/3460120.3484772

[33] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "PSI from paxos: Fast, malicious private set intersection," in *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Canteaut and Y. Ishai, Eds., vol. 12106. Springer, 2020, pp. 739–767. [Online]. Available: https://doi.org/10.1007/978-3-030-45724-2_25

[34] M. Hamburg, "Compressed maps without the keys, based on frayed ribbon cascades," 2020. [Online]. Available: https://github.com/bitwiseshiftleft/compressed_map

[35] S. Ghosh and M. Simkin, "The communication complexity of threshold private set intersection," in *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Boldyreva and D. Micciancio, Eds., vol. 11693. Springer, 2019, pp. 3–29. [Online]. Available: https://doi.org/10.1007/978-3-030-26951-7_1

[36] Y. Sang, H. Shen, Y. Tan, and N. Xiong, "Efficient protocols for privacy preserving matching against distributed datasets," in *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, ser. Lecture Notes in Computer Science, P. Ning, S. Qing, and N. Li, Eds., vol. 4307. Springer, 2006, pp. 210–227. [Online]. Available: https://doi.org/10.1007/11935308_15

[37] Y. Sang and H. Shen, "Privacy preserving set intersection protocol secure against malicious behaviors," in *Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007), 3-6 December 2007, Adelaide, Australia*, D. S. Munro, H. Shen, Q. Z. Sheng, H. Detmold, K. E. Falkner, C. Izu, P. D. Coddington, B. Alexander, and S. Zheng, Eds. IEEE Computer Society, 2007, pp. 461–468. [Online]. Available: https://doi.org/10.1109/PDCAT.2007.59

[38] ——, "Efficient and secure protocols for privacy-preserving set operations," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, pp. 9:1–9:35, 2009. [Online]. Available: https://doi.org/10.1145/1609956.1609965

[39] A. Bay, Z. Erkin, M. Alishahi, and J. Vos, "Multi-party private set intersection protocols for practical applications," in *Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021, July 6-8, 2021*, S. D. C. di Vimercati and P. Samarati, Eds. SCITEPRESS, 2021, pp. 515–522. [Online]. Available: https://doi.org/10.5220/0010547605150522

[40] A. Ben-Efraim, O. Nissenbaum, E. Omri, and A. Paskin-Cherniavsky, "Psimple: Practical multiparty maliciously-secure private set intersection," in *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, Y. Suga, K. Sakurai, X. Ding, and K. Sako, Eds. ACM, 2022, pp. 1098–1112. [Online]. Available: https://doi.org/10.1145/3488932.3523254

[41] A. Kavousi, J. Mohajeri, and M. Salmasizadeh, "Efficient scalable multi-party private set intersection using oblivious PRF," in *Security and Trust Management - 17th International Workshop, STM 2021, Darmstadt, Germany, October 8, 2021, Proceedings*, ser.

Lecture Notes in Computer Science, R. Roman and J. Zhou, Eds., vol. 13075. Springer, 2021, pp. 81–99. [Online]. Available: https://doi.org/10.1007/978-3-030-91859-0_5

[42] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty PSI and extensions to circuit/quorum PSI," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1182–1204. [Online]. Available: https://doi.org/10.1145/3460120.3484591

[43] A. Patra, A. Choudhary, and C. P. Rangan, "Information theoretically secure multi party set intersection re-visited," in *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, ser. Lecture Notes in Computer Science, M. J. J. Jr., V. Rijmen, and R. Safavi-Naini, Eds., vol. 5867. Springer, 2009, pp. 71–91. [Online]. Available: https://doi.org/10.1007/978-3-642-05445-7_5

[44] ——, "Round efficient unconditionally secure MPC and multiparty set intersection with optimal resilience," in *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*, ser. Lecture Notes in Computer Science, B. K. Roy and N. Sendrier, Eds., vol. 5922. Springer, 2009, pp. 398–417. [Online]. Available: https://doi.org/10.1007/978-3-642-10628-6_26

[45] Y. Sang and H. Shen, "Privacy preserving set intersection based on bilinear groups," in *Computer Science 2008, Thirty-First Australasian Computer Science Conference (ACSC2008), Wollongong, NSW, Australia, January 22-25, 2008*, ser. CRPIT, G. Dobbie and B. Mans, Eds., vol. 74. Australian Computer Society, 2008, pp. 47–54. [Online]. Available: https://dl.acm.org/citation.cfm?id=1378290

[46] M. Kim and B. Z. Kim, "An experimental study of encrypted polynomial arithmetics for private set operations," *J. Commun. Networks*, vol. 19, no. 5, pp. 431–441, 2017. [Online]. Available: https://doi.org/10.1109/JCN.2017.000075

[47] A. Abadi, S. J. Murdoch, and T. Zacharias, "Polynomial representation is tricky: Maliciously secure private set intersection revisited," in *Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, E. Bertino, H. Shulman, and M. Waidner, Eds., vol. 12973. Springer, 2021, pp. 721–742. [Online]. Available: https://doi.org/10.1007/978-3-030-88428-4_35

[48] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. [Online]. Available: https://www.ndss-symposium.org/ndss2012/private-set-intersection-are-garbled-circuits-better-custom-protocols

[49] X. Wang, S. Ranellucci, and J. Katz, "Global-scale secure multiparty computation," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 39–56. [Online]. Available: https://doi.org/10.1145/3133956.3133979

[50] O. Ruan, Z. Wang, J. Mi, and M. Zhang, "New approach to set representation and practical private set-intersection protocols," *IEEE Access*, vol. 7, pp. 64 897–64 906, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2917057

[51] A. Bay and A. Kayan, "A new multi-party private set intersection protocol based on OPRFs," *Mugla Journal of Science and Technology*, vol. 8, no. 1, pp. 69–75, 2022.

[52] D. Many, M. Burkhart, and X. Dimitropoulos, "Fast private set operations with sepia," *ETZ G93*, 2012.

[53] P. K. Y. Lai, S. Yiu, K. Chow, C. F. Chong, and L. C. K. Hui, "An efficient bloom filter based solution for multiparty private matching," in *Proceedings of the 2006 International Conference on Security & Management, SAM 2006, Las Vegas, Nevada, USA, June 26-29, 2006*, H. R. Arabnia and S. Aissi, Eds. CSREA Press, 2006, pp. 286–292.

[54] S. K. Debnath, P. Stanica, N. Kundu, and T. Choudhury, "Secure and efficient multiparty private set intersection cardinality," *Adv. Math. Commun.*, vol. 15, no. 2, pp. 365–386, 2021. [Online]. Available: https://doi.org/10.3934/amc.2020071

[55] S. K. Debnath, T. Choudhury, N. Kundu, and K. Dey, "Post-quantum secure multi-party private set-intersection in star network topology," *J. Inf. Secur. Appl.*, vol. 58, p. 102731, 2021. [Online]. Available: https://doi.org/10.1016/j.jisa.2020.102731

[56] L. Wei, J. Liu, L. Zhang, and W. Zhang, "Efficient and collusion resistant multi-party private set intersection protocols for large participants and small sets setting," in *Cyberspace Safety and Security - 14th International Symposium, CSS 2022, Xi'an, China, October 16-18, 2022, Proceedings*, ser. Lecture Notes in Computer Science, X. Chen, J. Shen, and W. Susilo, Eds., vol. 13547. Springer, 2022, pp. 118–132. [Online]. Available: https://doi.org/10.1007/978-3-031-18067-5_9

[57] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious PRF with applications to private set intersection," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 818–829. [Online]. Available: https://doi.org/10.1145/2976749.2978381

Appendix A. Derived complexities

Using polynomial roots. The protocol by Kissner & Song first requires each party to send encrypted polynomials to t other parties. The leader must also send the final encrypted polynomial to all $n-1$ assistants. The polynomials grow with a constant factor so the communication complexity for an assistant is $O(tk)$ bits, and $O(nk)$ for the leader. Computation-wise, the most expensive part of the protocol is when each party computes the dot product of $t+1$ encrypted polynomials with random polynomials. This takes $O(tk^2)$ cryptographic operations. The protocol takes 3 rounds: encryption, randomization and addition, and decryption.

The most expensive part of the protocol by Li & Wu is the computation phase. Here, each party sends $O(n(k+1)(k+2))$ secret values to t other parties, which takes $O(ntk^2)$ bits. The computation required is at least equal to this complexity. Note that in this protocol, all parties receive the output, so there is no actual notion of a leader. We only consider the semi-honest case here, which takes 3 rounds.

We derive complexities from the analysis by Sang et al. [36]. Each party performs $O(n^2k)$ computations. The total communication is $O(n^2k)$, which we divide by n to get the complexity per party. There are 3 rounds of interaction. The second protocol by Sang & Shen [38] takes $O(nk^2)$ computations in step 2.2 for all parties, and $O(nk)$ communicated bits in step 2.3. This takes 2 rounds of interaction.

The protocol by Cheon et al. is similar to Kissner & Song's, but with the encrypted polynomial multiplications taking $O(k)$ rather than $O(k^2)$. It is designed so that all parties receive the result, but we change steps 2-4 so only the leader receives the result. In step 2, assistants send their randomized polynomials only to the leader, step 3 is then computed by the leader, and the threshold decryption continues with only t parties. Then, each party performs $O(nk)$ computations in step 2 and each party sends its encrypted polynomial to all others, which takes $O(nk)$ bits. This takes 3 rounds of online communication: encryption, randomization in steps 2 & 3, then threshold decryption.

For the protocol by Ghosh et al. we use the complexities from their paper: the computation complexity for the leader is $O(nk \log k) = \tilde{O}(nk)$ and for the assistant it is identical to the two-party case $\tilde{O}(k)$. The communication complexity is $O(nk)$ for the leader, $O(k)$ for the assistant. The setup takes 1 interaction, share computation takes 4 given that an OPA takes 2 interactions, the output takes 1 interaction.

The protocol by Gordon et al. requires one interaction for the input sharing phase, one interaction for the coin toss, one interaction for the output aggregation, and one interaction for the OLE if it is instantiated using an efficient OT. We copy the leader's communication complexity from the paper: $O(nk+nt)$. The other complexities are non-trivial as they depend on the choice of primitives and parameters.

Sorted multisets. There is no concept of a leader in the protocol by Blanton & Aguiar. We assume the parties pre-sort their sets, which allows the multiset to be sorted using a merge operation requiring $O(k \log k)$ operations (Section 7.1) rather than a full sort. Next, the parties perform $O(k)$ multiplications and secure equality operations. Since each multiplication requires at least t parties to communicate, this requires $O(tk)$ bits for each party. The total communication and computation complexity for one party is at least $O(k \log k + tk)$. For brevity, we denote this by $\tilde{O}(tk)$. As mentioned in Section 8 of their paper, the protocol runs in $O(\log k)$ rounds due to the merge operation at the start.

Using bitsets. The protocol by Bay et al. requires each assistant to send their encrypted bitset to the leader, which takes $O(|\mathcal{U}|)$ bits. After that, communication is restricted to the leader's k bits, which the leader sends to t assistants for randomization and decryption, taking $O(tk)$ bits. For each assistant, computation is dominated by encryption, which takes $O(|\mathcal{U}|)$ operations. The leader's most expensive step is in aggregating the bitsets, but it only has to consider the k elements in its own set, taking $O(nk)$ operations. In total, the protocol requires 3 rounds of interactions: encryption and aggregation, randomization, and decryption. The protocol by Vos et al. is asymptotically equivalent when using the composed logic sub-protocol. Otherwise, the leader incurs a factor $|\mathcal{U}|$ in computation and communication.

Using Garbled Bloom filters. For the protocol by Inbar et al., we use the complexities reported by the original paper in the semi-honest model. Assuming a two-round OT protocol, the protocol requires 3 rounds of interaction. For the protocol by Kavousi et al. we extract the complexities from Table 2 in the original paper. Assuming a two-round OT protocol, the protocol requires 4 rounds of interaction.

Using Bloom filters. The protocol by Bay et al. requires each assistant to send an encrypted Bloom filter to the leader, which takes $O(k)$. The leader only has to consider its own k elements, which it sends for randomization and decryption to t assistants, taking $O(tk)$ bits. Each assistant encrypts their entire Bloom filter, taking $O(k)$ operations. The leader must aggregate $O(nkh)$ bins. In total, this protocol requires 3 rounds of interactions, as in Section A. The protocol by Vos et al. is asymptotically equivalent to this work.

Using polynomial payloads. The work of Freedman et al. [1] uses the trick from Section 5 to transform the protocol to the star topology. For a fair comparison, we consider their scheme without this transformation, requiring private channels between all parties. The complexities are then $O(n^2k^2)$ in each aspect, and the protocol requires 4 rounds.

Using OPPRFs and OKVSs. Nevo et al. [32] provide detailed complexities for the works of Chandran et al. [42], Garimella et al. [31], Kolesnikvo et al. [3], and their own.

Appendix B. Derived operation counts

B.1. Elliptic curve multiplications

We refer to multiplications with precomputations as PMULs, and to regular ones as MULs. We assume MUL $\approx$ 4PMUL and that the leader always takes part in decryption. In elliptic curve ElGamal, a homomorphic multiplication with a plaintext requires two elliptic curve multiplications.

Sang & Shen. We alter the protocol by Sang & Shen [38] to only let the leader receive the result.

1) n parties encrypt their polynomial: $2k$ PMULs.
2) The leader multiplies nk coefficients by a scalar, which takes $2nk$ MULs.
3) The leader evaluates the polynomial k times, each time multiplying k coefficients by a scalar, which takes $2k^2$ MULs in total. Next, $t+1$ parties each decrypt the k resulting encryptions, which takes k MULs in total.

The leader performs $\frac{2}{4}k + 2nk + 2k^2 + k = 1.5k + 2nk + 2k^2$ MULs. In the worst case, an assistant performs $\frac{2}{4}k + k = 1.5k$ MULs. The total cost is $2nk + 2nk + 2k^2 + (t+1)k = 4nk + 2k^2 + tk + k$ MULs.

Cheon et al. We alter the protocol to only let the leader receive the result and to make the collusion resistance variable. The polynomials use the point-value representation, so an encrypted polynomial contains $2k$ ciphertexts.

This protocol has two phases. The input data conversion:

1) n parties each encrypt their polynomial: $4k$ PMULs.
2) No MULs.

The online phase is as follows:

1) No computations (polynomials are sent to $t+1$ parties).
2) $t+1$ parties randomize all polynomials: $4nk$ MULs.

3) The leader sums all randomized polynomials: which takes no MULs.
4) $t+1$ parties each decrypt the resulting polynomial, which takes $2k$ MULs.

Worst-case, the leader performs the same effort as an assistant: $\frac{4}{4}k + 4nk + 2k = 4nk + 3k$ MULs. The total cost is $\frac{4}{4}nk + 4n(t+1)k + 2(t+1)k = 5nk + 4ntk + 2tk + 2k$.

Hazay & Venkitasubramaniam. The protocol has two phases. Note that it uses the coefficient representation, so polynomial multiplication scales quadratically with the degree. The 2PC phase has two steps:

1) $n-1$ assistants encrypt k coefficients: $2k$ PMULs.
2) The leader evaluates $n-1$ polynomials k times, and randomizes (scalars can be multiplied in advance), which takes $(n-1)k(2k) = 2(n-1)k^2$ MULs.

Concluding the intersection goes as follows:

1) $t+1$ parties each randomize the summed ciphertext which takes $2k$ MULs.
2) The leader adds them up, which takes no MULs.
3) $t+1$ parties each decrypt, which takes k MULs.
4) The leader performs additions and zero-checks, which takes no MULs.

An assistant performs $0.5k + 3k = 3.5k$ MULs in the worst case. The leader performs $2(n-1)k^2 + 2k + k = 2(n-1)k^2 + 3k$ MULs. Now consider the balanced allocation optimization. Here, the maximum number of elements in one bin is at most 5 with overwhelming probability. Now, the leader receives $B = \frac{k}{\log \log k}$ bins in the 2PC part, where each set element is assigned to only one bin. So it evaluates $n-1$ polynomials of degree 5 k times. Each homomorphic multiplication also costs two EC multiplications. So, the leader's total is $10(n-1)k + 2k + k = 10nk - 7k$ MULs. This totals $2(n-1)k + 10nk - 7k + 2(t+1)k + (t+1)k = 12nk - 6k + 3tk$.

Vos et al. We go through the protocol step-by-step:

1) $n-1$ assistants each perform $2m$ PMULs.
2) The leader performs $2k$ MULs.
3) t assistants each perform $2k$ MULs.
4) The leader performs no MULs.
5) $t+1$ parties each perform k MULs.
6) The leader performs no MULs.

The leader performs $2k + k = 3k$ MULs. An assistant performs $2m$ PMULs and $2k + k = 3k$ MULs in the worst case. From (8) we have that $m \approx 2.08k \ln(\varepsilon^{-1})$, so an assistant performs approximately $1.04k \ln(\varepsilon^{-1}) + 3k$ MULs. Together, the parties perform $\frac{2}{4}m(n-1) + 2k + 2tk + (t+1)k = 1.04(n-1)k \ln(\varepsilon^{-1}) + 3tk + 3k$ MULs.

B.2. Efficient two-party subprotocols

We use the fact that one OPPRF costs one OPRF [3], and one OPRF costs approximately 3.5 OTs [57]. Since OLEs can be instantiated using OT or based on the learning with errors problem [13], we count OLEs separately.

Inbar et al. In Table 1 of their work, Inbar et al. [8] describe that the leader performs mn OT extensions in the augmented semi-honest model, which we reduce to $m(n-1)$ as the leader does not interact with itself. From (8) we have that $m \approx 2.08k \ln(\varepsilon^{-1})$, so the leader performs $2.08(n-1)Ek$ OTs. An assistant performs $2.08Ek$ OTs. In total, there are $2.08(n-1)Ek$ OTs (the leader is involved in each of those).

Kolesnikov et al. The protocol first requires each party to perform k OPPRFs with $n-1$ other parties. After that, the leader performs k additional OPPRFs with the $n-1$ assistants. So, the leader performs $2(n-1)k$ OPPRFs and each assistant performs $(n-1)k$ OPPRFs. In total, there are $n(n-1)k$ OPPRFs in the first part, and $(n-1)k$ in the second. One OPPRF is 3.5 OTs.

Chandran et al. We derive the number of OTs for the 're-laxed batch OPPRF' described in Appendix B of Chandran et al. [42]. Here, a wPSM between two parties requires two rounds of β OPPRFs, where the authors select $\beta = 1.28k$. The wPSM protocol is executed between every assistant and the leader. So, the leader performs $2.56(n-1)k$ OPPRFs, and an assistant $2.56k$. In total: $2.56(n-1)k$ OPPRFs.

Kavousi et al. In Section 3.3 of their work, Kavousi et al. [41] explain how to choose parameter w. The authors propose to set $m = k$ (where k is the number of set elements). Now, we show that p scales regardless of k:

$$p = \left(1 - \frac{1}{k}\right)^k \approx \frac{1}{e}\ , \tag{12}$$

which holds as k grows to infinity, but the approximation is already accurate for small k. As a result, w can be a constant. The lowest value causing the probability to fall below 2^{-40} is $w = 558$. We note that if m is variable, one might choose a lower w, trading off computation and communication.

Garimella et al. We consider the multi-party MPSI protocol in Section 7.2 of the work by Garimella et al. [31] in the star topology. The authors defer an analysis to a full version of the paper, but at the time of writing this paper is unavailable.

Nevo et al. The OPPRFs are executed in the final step of the protocol as part of the zeroXOR functionality between $t+1$ parties. We consider the leader to be part of this group, acting as the receiver. Here, t assistants perform k OPPRFs with the leader. This comes down to tk OPPRFs for the leader, k for an assistant, and tk in total.

Ghosh & Nilges. The OLEs are performed in the OPA subprotocol, which are performed between each assistant and the leader on k elements. Each OPA requires two calls to an OLE. So, the leader performs $2(n-1)k$ OLEs, an assistant performs $2k$ OLEs. In total: $2(n-1)k$ OLEs.

Gordon et al.. The main cost of this protocol comes from OLEs. For one-sided output, each assistant only performs one OLE with the leader per input item. As a result, the leader performs $(n-1)k$ OLEs, an assistant performs k OLEs, and in total $(n-1)k$ OLEs are performed.

Appendix C. Meta-Review

C.1. Summary

This is a revised submission of an SoK manuscript synthesizing the literature on multi-party PSI. The authors focused on the >2 party setting and semi-honest models, excluding server-aided approaches.

C.2. Scientific Contributions

- Provides a valuable step forward in an established field
- Independent confirmation of important results with limited prior Research

C.3. Reasons for Acceptance

1) Presents a very important research field with a great number of publications.
2) For every presented work, there is sufficient amount of information included to understand where each work stands.
3) A number of examples and applications provide appropriate points for comparison.

C.4. Noteworthy Concerns

1) Evaluation of existing implementations is limited to discussion-based analysis.
2) Limited discussion about future work in the area.