Secure Federated Analytics for Set Intersection

Ahmed Elkordy , Yahya H. Ezzeldin , Salman Avestimehr
University of Southern California
{aelkordy, yessa}@usc.edu, avestimehr@ee.usc.edu

Abstract

Private set intersection (PSI) is a popular protocol that allows multiple parties to evaluate the intersection of their sets without revealing them to each other. PSI has numerous practical applications, including privacy-preserving data mining and location-based services. In this work, we develop a new approach for the PSI problem within the federated analytics framework. In particular, we consider a setting where a server wants to determine (query) which among its local set of data identifiers appears coupled with the same value in at least K of N parties. Applications for this framework include: double-filing insurance verification and credit scoring. The proposed setting does not lend itself directly to state-of-the-art PSI approaches based on Oblivious Transfer, since the server does not have a complete representation of a datapoint (only the identifier, but no value). To address the proposed setting, we propose a new protocol `Fed-K-PSI` that allows the server to answer this query while being oblivious to the data of identifiers not satisfying the distributed query at the parties or the parties that contain these identifiers. We show that `Fed-K-PSI` achieves a strong information-theoretic privacy guarantee and is resilient to collusion scenarios among honest-but-curious parties. We also evaluate `Fed-K-PSI` via extensive experiments to study the effect of different system parameters.

1 Introduction

In today's world, data privacy has become a precious and critical commodity that individuals and/or enterprises are suspicious to give up, even when the greater-good goal is the target. However, there is an increasing number of applications where an entity needs to evaluate its available information on data owned by suspicious or non-trusting parties. For instance, consider the following scenarios:

1) A federal agency wants to verify that an individual is not double-filing insurance claims with multiple insurance companies, but the companies are not willing to share their data;

2) A federal tax authority wants to learn whether suspected tax evaders have accounts exceeding a particular threshold at a number of foreign banks. However, the bank's domicile prevents explicit disclosure of clients' account information;

3) An online marketplace (e.g., Amazon) wants to verify that some sellers are not harassing buyers with off-site spam email communications, but the buyers are not willing to share their emails and their bodies with the marketplace.

These examples highlight scenarios where protocols for private-set-operations are needed. In particular, these examples are closely related to the problem of Private set intersection (PSI). PSI is a class of cryptographic protocols that allows for evaluating the intersection of the input sets of two or more parties privately without revealing anything about the sets beyond the intersection. PSI has seen several real-world applications in both the two-party setting and the multi-party setting. Two-party PSI has been used in measuring the effectiveness of online advertising [Ion et al.(2017)], contact discovery [Kales et al.(2019)], and Apple's new system for detecting child sexual abuse material [Bhowmick et al.(2021)]. PSI in the multi-party setting has been used in applications such as COVID-19 contact tracing [Duong et al.(2020)].

In this paper, we consider a generalized multi-party PSI setting called K-PSI where a server is interested in checking whether a datapoint identifier (e.g., email address, ID, etc) appears associated with a repeated value (unknown to the server) in K out of the N parties. For instance in example #1 described at the opening of the section, the server (federal agency) wants to evaluate whether a set of individuals (each identified with their IDs) have filed duplicate claims (represented by the value for the ID in our setting) in at least K of the N potential insurance companies. If the query response is negative for an individual in the server's query list, it should not learn any information about the values associated with this particular individual in the local datasets.

State-of-the-art multi-party PSI protocols essentially rely on two main techniques: (1) Pseudo-random permutation on the data space [Kamara et al.(2014)]; (2) Oblivious Transfer (OT) extension [Pinkas et al.(2014)]. Both techniques provide protection for data values against a polynomial-time honest but curious adversary. In our described K-PSI setting, the use of permutation approaches (where essentially the parties send their datapoints after random permutation in space) can allow the server to learn statistical information about the

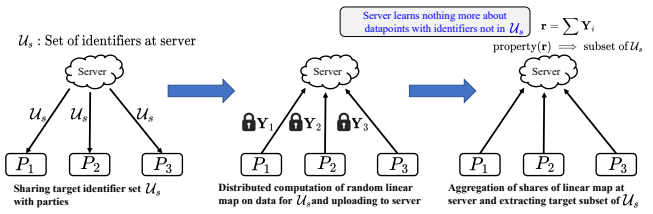


Figure 1: Illustration of our Fed-K-PSI protocol. (1) Server shares its target identifier set; (2) Programming distributed linear map: parties agree on a programmed distributed random linear map and encode their local data using their share of the distributed map; (3) Aggregation at the server and determining target identifier subset.

parties' datasets even for identifiers that do not satisfy the K-PSI query; for example, how many points exist in the intersection of Party 1 and Party 2, even though this wasn't the intended goal of the query. On the other hand, the use of OT based approaches in our setting is also limited by the fact that the server only has access to partial data about the datapoint. Specifically, if we view the datapoint as (ID, value) pair, the server only has access to the ID portion of the datapoint.

1.1 Our Contributions

In this paper, we explore a new perspective of the PSI problem by considering a federated analytics scenario where a server only knows the identifier of the datapoint (but not its value). The goal is to query N parties for points that appear in at least K of the parties. We call this query scenario K-PSI.

For this setting, we develop a novel private protocol named Fed-K-PSI that can allow the server to verify if each identifier, out of its set of identifiers, appears associated with a particular value in the K-set intersection of N parties. Our studied and proposed Fed-K-PSI solution shifts focus from *private pairwise equality-testing* which is at the core of state-of-the-art PSI protocols towards *programming a random distributed linear map* across the parties. Specifically, a share of a linear map is applied on each party's local dataset, then masked and uploaded to the server for aggregation. Then, the server aggregates the linear map in a one-shot and analyzes the output properties to retrieve the target identifiers set.

As illustrated in Figure 1, the main idea of Fed-K-PSI is that each party uses the set of target identifiers shared by the server in order to compute a vector $\mathbf{Y}_i$ which represents a share in a distributed linear map of the parties' datasets. The parties then mask these vectors and transmit them to the server for aggregation. Finally, the server checks the aggregated vector for a predetermined property to decide which subset of identifiers appeared associated with the same value at least K times across the parties.

We show that our proposed Fed-K-PSI protocol is correct and satisfies a strong information-theoretic privacy guarantee in the honest-but-curious server setting. In particular, Fed-K-PSI fully protects any information about the data related to the identifiers that do not give an affirmative answer to the server's query. We discuss this guarantee formally in Section 2.2. We also compare Fed-K-PSI with other approaches in the literature (e.g., [Pinkas et al.(2014); Pinkas et al.(2015); Rindal and Rosulek(2017); Pinkas et al.(2019); Kolesnikov et al.(2017)], in a specialized version of the K-PSI problem where identifiers can only take a common singular value. We use this specialized version of the K-PSI, since these approaches can not be applied to our general K-PSI setting as we discuss next in Section 1.2. In this comparison, we highlight that for the best achieving PSI protocol (in the privacy sense), our protocol achieves better privacy guarantees for a factor N increase in complexity, which makes it well-suited for applications where the parties are data silos. We also analyze the theoretical complexity of Fed-K-PSI and run extensive experiments to empirically demonstrate its running time as well as the effect of the different protocol components on the end-to-end wall clock time.

1.2 Related work

Two-party and Multi-party PSI. PSI is a well-studied problem and has gained attention over the last two decades [Pinkas et al.(2018)] due to its large number of practical use cases. Based on the number of participating parties, the protocols in PSI problem can be broadly divided into: *two-party PSI* and *multi-party PSI*. In the two-party setting, there are many existing approaches in the literature, including works based on homomorphic encryption [Huberman et al.(1999); Ion et al.(2017); Freedman et al.(2016); Chen et al.(2017)], Oblivious Polynomial Evaluation [Freedman et al.(2004); Dachman-Soled et al.(2009)], Oblivious Transfer [Pinkas et al.(2015); Rindal and Rosulek(2017); Pinkas et al.(2019)], and works based on garbled circuit [Huang et al.(2012); Dong et al.(2013)]. A number of these techniques extend to the multi-party setting. In particular, the closest to our setting is the server-aided PSI problem discussed in [Kamara et al.(2014); Abadi et al.(2020)], where the goal is to compute the PSI between the parties while off-loading the computation to a third party (for example, a cloud server) that has no inputs to the computation and receives no output, but makes its computational resources available to the parties. Our studied setting differentiates from classical PSI settings in two aspects: (a) Although, we can consider the server as one of the parties that wishes to compute the intersection, the server in our K-PSI setting has missing data portions, since it only has access to the identifiers associated with each datapoint (but not its value). Therefore, the server can not perform private pairwise equality testing or membership testing which is a central concept in PSI protocols; (b) In K-PSI, the server is interested in knowing which of its points can be part of at least K sets of the different parties, whereas current protocols target datapoints in the intersection of all N parties.

Threshold-PSI. Another variant of PSI problem is the threshold-PSI, where the intersection is computed between parties if the intersection cardinality exceeds a threshold. Protocols have been developed for two-party [Ghosh and Simkin(2019)] and the multi-party [Branco et al.(2021)] setting with the most famous implementation being by Apple [Bhowmick et al.(2021)] to counter child sexual abuse material in the iCloud. The key difference from our K-PSI problem is that in Threshold-PSI, the threshold is on the number of elements that are shared across ALL the parties, while in K-PSI, the threshold is on the number of parties that share a particular element, not the number of such elements.

Approach	rounds	Parties per round	Communication Complexity/party	Leakage for $\mathcal{U}_S^{K^c}$?	Extends to non-singular values $v(u)$?
Homomorphic encryption and pseudo-random permutations [Kamara et al.(2014); Ion et al.(2017)], [Chen et al.(2017)]	1	N	$O(n_s)$	✓ (Server can compute a permuted histogram of the parties' sets)	✓
Oblivious Transfer [Pinkas et al.(2014)] [Pinkas et al.(2015); Pinkas et al.(2019)], [Rindal and Rosulek(2017)]	N	1	$O(n_s)$	✓ (Server knows $\mathcal{U}_s \cap \mathcal{P}_i, \forall i \in [N]$)	✗
Oblivious Programmable PRFs [Kolesnikov et al.(2017)]	$\binom{N}{K}$	K	$O\left(n_s\binom{N-1}{K-1}\right)$	✓ ($\forall u \in \mathcal{U}_s^K$, Server additionally knows which K parties have u)	✗
Ours Approach (`Fed-K-PSI`)	1	N	$O\left(n_s\binom{N}{K}(K-1)\right)$	✗	✓

Table 1: Comparison between our proposed approach and different classes of PSI protocols in the literature when adapted to the singular-value K-PSI setting (where $v(u) = v, \forall u$). The table highlights number of rounds needed as well as leaked information to the server.

2 Problem Formulation (K-PSI)

2.1 Problem setting

We consider a federated PSI problem with a server and N parties $\{P_1, P_2, \cdots, P_N\}$. Each party P_i has a set of n_i datapoints defined as an identifier-value pair $(u_j^i, v_i(u_j^i))$, where $v_i(u_j^i)$ is the value for identifier u_j^i at party P_i. An identifier u is unique across the data points at P_i, but the value $v_i(u)$ might be the same for multiple identifiers. We denote the set of data points at P_i with $\mathcal{P}_i = \left\{d_j^i \middle| d_j^i = (u_j^i, v_j^i),\ \forall j \in [n_i]\right\}$. For a set of identifiers $\widehat{U}$, we denote with $\mathcal{P}_i(\widehat{U})$ the subset of $\mathcal{P}_i$ with only identifiers in $\widehat{U}$.

The server has a set of identifiers $\mathcal{U}_s$ and targets to know whether for each identifier $u \in \mathcal{U}_s$, it appears with an associated value v at least K times across the parties. In formal terms, the server would like to compute the set

$$\mathcal{U}_s^K = \mathcal{U}_s \bigcap \mathcal{U}_P^K,$$

$$\text{where}: \mathcal{U}_P^K = \left\{ u \;\middle|\; \exists v',\ \text{s.t.,}\ (u, v') \in \bigcup_{\substack{\Lambda \subseteq [N],\ j \in \Lambda \\ |\Lambda| = K}} \bigcap \mathcal{P}_j \right\}. \quad (1)$$

The set $\mathcal{U}_P^K$ represents the set of identifiers u that with the same value v' in K of the N parties. The server is interested to learn which such identifiers appear in its local set $\mathcal{U}_s$.

We assume an arbitrary identifier space $\mathbb{U}$, but consider all operations on the values $v(u)$ to be over a finite field $\mathbb{F}_q$ for some prime field size q, such that: (i) $v_i(u) \neq 0, \forall u$; (ii) the maximum value v_{max} satisfies that $q > v_{max} + N$[1].

Example Application. As a motivation for the above problem setting, we encourage the reader to think of an insurance fraud detection application, where the server (a federal auditor) has a set of potential suspects. The server would like to check a set of suspects with insurance companies (parties) without a company having to reveal its internal, potentially private, information about its clients. The server declares an individual on its potential list as a fraud if the person submitted the same claim to at least K (typically equal 2 or 3) out of the insurance companies, i.e., the same (ID, claim) pair was found in the internal dataset of K of the N parties.

2.2 Threat model and privacy guarantee

We consider a threat model where the server is honest but curious. We assume no-collusion between the server and the parties. There are many settings in practice where there is no collusion between the server and the parties, e.g., due to legal constraints and/or economic incentives. In our setting, the server could be a government agency or an enterprise, and it is reasonable - given the consequences of legal action and bad publicity — to assume that the server will not collude with the parties. This assumption was also adopted in other PSI works (e.g., [Kamara et al.(2014)]).

We consider privacy guarantee for the non-target set $\mathcal{U}_s^{K^c}$ in the strong information-theoretic sense, where $\mathcal{U}_s^{K^c} = \mathbb{U} \backslash \mathcal{U}_s^K$ is the compliment of $\mathcal{U}_s^K$. This requires that at the server, we must have this mutual information guarantee

$$I\left(\{\mathcal{P}_i(\mathcal{U}_s^{K^c})\}_{i \in [N]}; \mathbf{Y}_s | \mathcal{U}_s^K\right) = 0, \quad (2)$$

where $\mathbf{Y}_s$ is the collection of information received at the server and $\mathcal{U}_s^K$ is the target set as defined in (1).

Objective. We aim to design a protocol to solve the K-PSI problem setting described in Section 2.1 that is simultaneously *Correct* and *Private*, as defined formally below:

Definition 1 (Correctness)**.** *We say that a protocol $\mathbb{A}$ is correct if $\mathbb{A}(\mathcal{U}_s, \mathcal{P}_{i \in [N]}, K)$ gives $\mathcal{U}_s^K$ error-free at the server.*

Definition 2 (Privacy)**.** *A federated `K-PSI` protocol $\mathbb{A}$ is said to be private if it satisfies the condition* (2)*.*

2.3 Summary and placement of paper results

After introducing the K-PSI problem and the target theoretical privacy guarantees, we can now summarize the key results in this paper and show how they compare with solutions built using state-of-the-art PSI protocols.

Table 1 compares the communication complexity[2] and privacy leakage (violations to the privacy definition in (2)) for

[1] The assumptions that the values do not take zero and that $q > v_{max} + N$ can be satisfied by mapping the value space into a large enough field-size and shifting the value space away from zero.

[2] Computation loads are order-wise similar to communication costs.

prototype approaches using the state-of-the-art algorithms and our Fed-K-PSI protocol (introduced in the next section). Although Fed-K-PSI pays the highest complexity, it provides the strongest privacy guarantees with no leakage beyond $\mathcal{U}_s^K$. In fact, the complexity of Fed-K-PSI is only a factor of $N\frac{K-1}{K}$ away from the second-best approach (in the sense of privacy), which use oblivious programmable pseudo random functions [Kolesnikov et al.(2017)][3]. Approaches that rely on pseudo-random permutation and Oblivious transfer rely on either sending encrypted data to the server, or creating 1-to-1 PSI sessions between the server and every party, which leads to the leakage highlighted in Table 1.

3 Proposed protocol (Fed-K-PSI)

In this section, we formally present our Fed-K-PSI proposed protocol for the k-PSI problem. The central idea of the protocol is as follows: For each identifier $u \in \mathcal{U}_s$, the N parties will collaboratively *program* a random linear mapping of the values associated with the identifier u in their local datasets; we denote the set of these values $\mathcal{V}_u$. The parties then communicate messages $\{\mathbf{Y}_i(u)\}_{i\in[N]}$ to the server, which ensure that the server can compute a set of programmed linear mappings of the values but nothing else. The server declares an identifier u to be in $\mathcal{U}_s^K$ only if at least one of the computed maps is equal to zero. Our proposed approach is composed of three phases. We start by giving an overview of the different phases of the protocol. Afterwards, we discuss in detail how to construct the different elements that satisfy the probabilistic assumptions described in the protocol. An illustration of the different phases of the protocol is shown through a running example given by Fig. 2.

Phase 1 (Initialization) The parties start by jointly agreeing on a secret random seed a_N (for e.g., using Diffie-Hellman agreement [Diffie and Hellman(1976)]). The server shares its identifier set $\mathcal{U}_s$ with the N parties. The following two phases are repeated identically for each identifier $u \in \mathcal{U}_s$.

Phase 2A (Programming of random linear map)
Using the seed a_N, the random constructions in this phase will be shared across the parties. Each party starts by sampling a uniformly random subset $I^\ell \subset [N]$ of size K (without repetition). For each such subset I^ℓ, we uniformly sample an independent matrix $\widehat{\mathbf{Z}}_u^\ell \in \mathbb{F}_q^{K\times K-1}$, $\forall \ell \in [L]$ such that: (1) the matrix has full column rank; (2) $\mathbf{1}^T\widehat{\mathbf{Z}}_u^\ell = \mathbf{0}$. Thus, in total we sample $L = \binom{N}{K}$ matrices with the properties aforementioned. Next, we expand each matrix $\widehat{\mathbf{Z}}_u^\ell$ into a sparse matrix $\mathbf{Z}_u^\ell \in \mathbb{F}_q^{N\times K-1}$, such that the I_ℓ rows of $\mathbf{Z}_u^\ell$ are populated with $\widehat{\mathbf{Z}}_u^\ell$ and all other rows are zeros, i.e.,

$$\mathbf{Z}_u^\ell[I_\ell,:] = \widehat{\mathbf{Z}}_u^\ell, \qquad \mathbf{Z}_u^\ell[I_\ell^c,:] = \mathbf{0}_{(N-K)\times(K-1)}. \quad (3)$$

The key intuition here is that, each matrix $\widehat{\mathbf{Z}}_u^\ell$ is designed to test whether a subset of K parties I^ℓ have the same value for identifier u. In particular, since the matrix $\widehat{\mathbf{Z}}_u^\ell$ is uniformly distributed over matrices $\mathbb{F}_q^{K\times(K-1)}$ with columns representing a basis for the null space of $\mathbf{1}_K$, then each sampled matrix $\mathbf{Z}_u^\ell$ represents a random mapping $\mathbb{F}_q^N \to \mathbb{F}_q^{K-1}$ which maps the input $\mathbf{x} \in \mathbb{F}_q^N$ uniformly to a vector $\mathbf{x}^T\mathbf{Z}_u^\ell \in \mathbb{F}_q^{K-1}\backslash\{\mathbf{0}\}$ except if $\mathbf{x}[I_\ell] = v'\mathbf{1}$ for some v' in $\mathbb{F}_q$; in which case, it is always mapped to $\mathbf{0}_{K-1}$ (recall the property that $\mathbf{1}^T\widehat{\mathbf{Z}}_u^\ell = \mathbf{0}$).

Phase 2B (Encoding K-intersection)
In order to facilitate readability of this phase of the protocol, we will view the values associated with an identifier u, $\mathcal{V}_u$, as structured in a vector form $\mathbf{v}_u \in \mathbb{F}_q^{N\times 1}$, such that

$$\mathbf{v}_u[i] = \begin{cases} v_i(u) & \text{if } (u, v_i(u)) \in \mathcal{P}_i \\ i + v_{max} & \text{otherwise.} \end{cases} \quad (4)$$

Note that if identifier u does not exist at party P_i, its value is substituted with a unique value $(i + v_{max})$ that is not shared by other parties. We observe that from the construction of $\mathbf{Z}_u^\ell$, we have that $\mathbf{v}_u^T\mathbf{Z}_u^\ell = 0$ if and only if $\mathbf{v}_u[\mathcal{I}_\ell] = v'\mathbf{1}_K$ for some $v' \in \mathbb{F}_q$.

The goal of the N parties in our Fed-K-PSI protocol is to allow the server to compute $\mathbf{r}_{u_{(1\times K-1)}}^\ell = \mathbf{v}_u^T\mathbf{Z}_u^\ell$, $\forall \ell \in [L]$. We observe that from the construction of $\mathbf{Z}_u^\ell$, we have $\mathbf{r}_u^\ell = \mathbf{0}$ if and only if $\mathbf{v}_u[\mathcal{I}_\ell] = v'\mathbf{1}_K$ for some $v' \in \mathbb{F}_q$. Thus, if $\exists \ell$, such that $\mathbf{r}_u^\ell = \mathbf{0}$, then the current identifier u in question belongs to $\mathcal{U}_s^K$. Otherwise, $u \notin \mathcal{U}_s^K$.

To allow the server to compute $\mathbf{r}_u^\ell$, each party P_i can simply compute its $1 \times K-1$ vector

$$\widehat{\mathbf{y}}_{u,i}^\ell = \mathbf{v}_u[i] \cdot \mathbf{Z}_u^\ell[i,:] \quad (5)$$

and transmits it to the server. The server adds the received $\widehat{\mathbf{y}}_{u,i}^\ell$ vectors from the N parties to verify that they add to $\mathbf{0}$ or not. However, note that for any $\ell \in [L]$, the vector $\widehat{\mathbf{y}}_{u,i}^\ell$ is zero for some parties, particularly when $i \notin I_\ell$. As a result, the server (who is fully knowledgeable about the protocol procedure) can also learn which K parties have the same shared value $v(u)$. In order to combat this, the fourth phase masks which K parties are part of I_ℓ.

Phase 2C (Masking and uploading linear components)
Each party uses the shared seed a_N to generate a matrix $\mathbf{M}_u^\ell \in \mathbb{F}_q^{N\times(K-1)}$ uniformly at random such that: (1) $\mathbf{1}^T\mathbf{M}_u^\ell = \mathbf{0}$; (2) For any submatrix of $\widehat{\mathbf{M}}_u^\ell$ of size $N-1 \times K-1$, the elements are iid and distributed uniformly over $\mathbb{F}_q$.

After each party P_i locally computes its component $\widehat{\mathbf{y}}_{i,u}^\ell$ of $\mathbf{r}_u^\ell$, $\forall \ell \in [L]$ using (5), P_i masks it by adding its associated row of $\mathbf{M}_u^\ell$, to get

$$\mathbf{y}_{u,i}^\ell = \widehat{\mathbf{y}}_{u,i}^\ell + \mathbf{M}_u^\ell[i,:]. \quad (6)$$

Each party then transmits $\mathbf{y}_{u,i}^\ell$ to the server. Note that after masking with $\mathbf{M}_u^\ell[i,:]$, the server always receives a random vector from P_i, regardless of whether $i \in I_\ell$ or not.

Phase 3 (Decoding)
The server computes $\mathbf{r}_u^\ell$ by adding the received vectors to get

$$\mathbf{r}_u^\ell = \sum_{i=1}^N \mathbf{y}_{u,i}^\ell = \mathbf{v}_u^T\mathbf{Z}_u^\ell + \mathbf{1}^T\mathbf{M}_u^\ell \overset{(a)}{=} \mathbf{v}_u^T\mathbf{Z}_u^\ell, \quad (7)$$

where (a) follows due to the properties of $\mathbf{M}_u^\ell$. The server declares $u \in \mathcal{U}_s^K$ iff $\exists\ \ell$, s.t., $\mathbf{r}_u^\ell = \mathbf{0}$. Otherwise $u \notin \mathcal{U}_s^K$.

[3] This follows since $\binom{N}{K} = \frac{N}{K}\binom{N-1}{K-1}$.

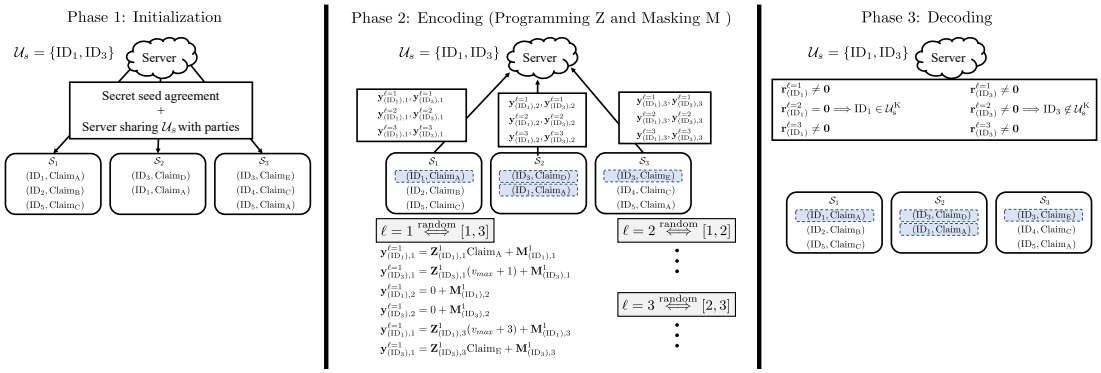


Figure 2: An illustrative example of `Fed-K-PSI` protocol with three parties. Following the insurance fraud detection application, the identifier represent the ID of an individual in the party's (company's) dataset, while the value for each identifier represents the claim submitted (if any). The parties first agree on a secret seed a_N and the server shares its identifier set $\mathcal{U}_s$ with the parties. Every subset of $K = 2$ parties encodes their data with a linear map $\mathbf{Z}$ that returns $\mathbf{0}$ iff their values for identifier u are equal; the N parties mask the participation of the selected K parties using a mask $\mathbf{M}$ that is removed by aggregation at the server. The server declares that $u \in \mathcal{U}_s^K$ iff $\exists \ell \in [\binom{N}{K}], \mathbf{r}^\ell(u) = \mathbf{0}$.

Remark 1 (Masking using SecAgg)**.** The masking procedure in Phase 2C is equivalent to the secure aggregation protocol [Segal et al.(2017); So et al.(2022)] used in federated learning. In this context, the masks are generated by exchanging pairwise secrets between parties during learning. Since in our protocol, the parties share a secret a_N to construct the linear map $\mathbf{Z}_u^\ell$, we use the same seed in our construction of $\mathbf{M}_u^\ell$. As discussed, the key benefit of this masking step is to prevent the server from knowing which parties participated in constructing $\mathbf{Z}_u^\ell$ based on their transmitted messages.

Construction of matrices $\mathbf{M}_u^\ell$. To efficiently sample a matrix $\mathbf{M}_u^\ell \in \mathbb{F}_q^{N\times(K-1)}$, such that $\mathbf{1}^T\mathbf{M}_u^\ell = \mathbf{0}$, each party samples a matrix $\widehat{\mathbf{M}}_u^\ell$ of size $N-1\times K-1$ with iid elements uniformly sampled from $\mathbb{F}_q$. We finally add an additional row to $\widehat{\mathbf{M}}_u^\ell$ equal to $q-\mathbf{1}^T\widehat{\mathbf{M}}_u^\ell$, which gives us the intended property. Note that for this construction, any submatrix of $\widehat{\mathbf{M}}_u^\ell$ with less than N rows is composed of iid element distributed uniformly over the field. As a result the server cannot learn anything if it only listens to $N-1$ parties.

Construction of matrices $\widehat{\mathbf{Z}}_u^\ell$. We would like to sample a matrix $\widehat{\mathbf{Z}}_u^\ell$ of size $K\times(K-1)$ with full column rank and $\mathbf{1}^T\widehat{\mathbf{Z}}_u^\ell = \mathbf{0}$. We can restate this as uniformly sampling a basis for the null space of $\mathbf{1}_K$ in $\mathbb{F}_q^K$. For this, we give the following explicit construction

$$\widehat{\mathbf{Z}}_u^\ell = \begin{bmatrix} \mathbf{I}_{K-1} \\ (q-1)\mathbf{1}^T \end{bmatrix} \mathbf{A}_u^\ell, \tag{8}$$

where $\mathbf{A}_u^\ell \in \mathbb{F}_q^{(K-1)\times(K-1)}$ is sampled uniformly from the set of full-rank square matrices. The first matrix in the RHS in (8) has a full column rank and ensures that $\mathbf{1}^T\widehat{\mathbf{Z}}_u^\ell = \mathbf{0}$ as it represents a basis for the null space of $\mathbf{1}_K$. We now have the following proposition which gives us our target distribution.

Proposition 3.1. *By picking a full rank square matrix $\mathbf{A}_u^\ell$ uniformly at random and constructing $\widehat{\mathbf{Z}}_u^\ell$ as in* (8)*, we have that $\widehat{\mathbf{Z}}_u^\ell$ is uniformly distributed over all matrices satisfying: (a) $\mathbf{1}^T\widehat{\mathbf{Z}}_u^\ell = 0$; (b) $\widehat{\mathbf{Z}}_u^\ell$ has full column rank.*

Remark 2 (Uniform sampling of full rank square matrices)**.** A central part of the construction of $\widehat{\mathbf{Z}}_u^\ell$ in (8) is to sample a full-rank square matrix uniformly. In a finite field, there does not exist an efficient approach to construct such a function with uniform probability. Fortunately, for large enough field sizes, full-rank square matrices represent the extreme majority of square matrices. In fact, we can characterize the exact probability of an $n \times n$ matrix sampled with uniformly distributed iid terms in $\mathbb{F}_q$ being full-rank as follows: The total number of different matrices of size $n \times n$ is equal to q^{n^2}. The set of full-rank $n \times n$ square matrices in the finite field $\mathbb{F}_q$ is the famous general linear group $GL(n, q)$ [Borel(2012)], for which the number of group elements is given by $q^{n(n-1)/2}\prod_{i=1}^n(q^i-1)$. Thus, the probability of getting a full-rank matrix is $q^{n(n-1)/2}\prod_{i=1}^n(q^i-1)/q^{n^2}$ which is increasing in q. For instance, for $q = 103$ and $n = 3$, the probability is equal to 0.9901. Thus, for $\mathbf{A}_u^\ell$ in (8), we sample a matrix of size $K-1\times K-1$ with iid uniformly distributed elements and repeat the sampling if the matrix is not full-rank, where each resampling has a probability of success close to unity if q is sufficiently large.

4 Theoretical Analysis

4.1 Theoretical Guarantees

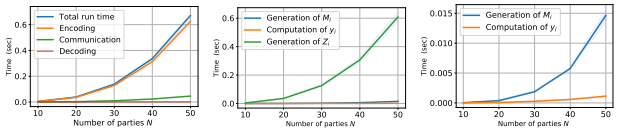
We now state our main theoretical result for the `Fed-K-PSI` protocol which ensures that for $u \notin \mathcal{U}_s^K$, no information is leaked beyond the fact that it is not in $\mathcal{U}_s^K$.

Theorem 4.1. Consider a K-PSI problem with N parties as described in Section 2.1. Then, the proposed `Fed-K-PSI` protocol is simultaneously (a) private, and (b) correct.

4.2 Complexity of `Fed-K-PSI`

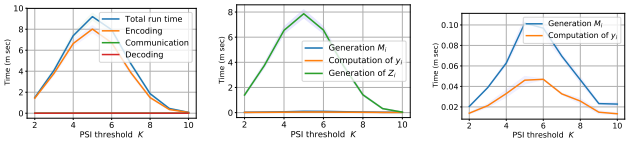
We analyze the theoretical communication load and computational load of `Fed-K-PSI` in terms of number of units of elements communicated or operations computed in $\mathbb{F}_q$.

Computation Load Let $L = \binom{N}{K}$. For each $\ell \in [L]$, each party locally computes two matrices $\mathbf{Z}_\ell \in \mathbb{F}_q^{K\times(K-1)}$ and



(a) (b) (c)

Figure 3: Breakdown of the running time of Fed-K-PSI vs N parties (with $K = 3$ threshold) for one data point: (a) Total run time; (b) Encoding phase components; (c) Designing $\mathbf{M}_i$ and $\mathbf{y}_i$.



(a) (b) (c)

Figure 4: Breakdown of the running time of Fed-K-PSI vs threshold K (with $N = 10$ parties) for one data point: (a) Total run time; (b) Encoding phase components; (c) Designing $\mathbf{M}_i$ and $\mathbf{y}_i$.

$\mathbf{M}_u^\ell \in \mathbb{F}_q^{N\times(K-1)}$. With very high-probability (see Remark 1, 2), the generation of the two matrices follows an iid uniform sampling of elements which takes $O(N(K-1))$ for the larger matrix. The encoding of the local data and masking takes a similar complexity. Thus, the total computational complexity is $O\left(n_s LN(K-1)\right)$, where n_s is the cardinality of the identifier set $\mathcal{U}_s$ shared by the server. The server aggregates the received vectors at a cost of $O\left(n_s LN(K-1)\right)$.

Communication Load Communication during Fed-K-PSI takes place over two phases. First, the server shares its identifier set $\mathcal{U}_s$ of cardinality n_s with the parties, which results in a multicast transmission of $O(n_s)$. Upon computing their linear shares $\mathbf{y}_i^\ell(u)$, $\forall u \in \mathcal{U}_s$ and $\ell \in [L]$, each party sends $n_s LN(K-1)$ elements back to the server.

5 Empirical Evaluation

5.1 Evaluation setting

We benchmark the performance of Fed-K-PSI using Python on a server with AMD EPYC 7502 32-Core CPU Processor. We use the NumPy module for seeded random matrix generations and basic modular arithmetic operation. For reporting the computation time for a module in our protocol (or the end-to-end time), we take the mean and standard deviation over 8 runs each representing 1000 consecutive invocations of the module (or end-to-end computation). For communication, we consider an idealistic simulation where the communication time is proportional to the number of $\mathbb{F}_q$ elements transmitted. In particular, we simulate the communication time taken to run Fed-K-PSI as

$$\frac{\binom{N}{K}(K-1)n_s\log_2(q) + a_N}{BW},$$

where: n_s is the number of identifiers in $\mathcal{U}_s$; $\binom{N}{K}(K-1)$ is the number of messages $\{\mathbf{y}_i^\ell(u)\}_{\ell\in[\binom{N}{K}]}$ from party P_i; the common shared secret a_N across the parties is 100 bits; BW is the assumed communication bandwidth. In all experiments, we set the field size $q = 5051$ which is a prime number.

5.2 Complexity evaluation

In the following, we experimentally evaluate the running time of Fed-K-PSI on $n_s = 10,000$ identifiers in $\mathcal{U}_s$ while studying the dependency of the system complexity on both the N and K parameters. We observe empirically (also through the protocol procedure in Section 3) that the running time is linear in the size of the server identifier set $\mathcal{U}_s$. Thus, in the following subsections, we report communication and computation times per identifier. We start by evaluating the dependency on the number of parties N.

Dependency on N

In Figures 3, we show that the running time of Fed-K-PSI per identifier changes with N when fixing the intersection threshold to $K = 3$. For this evaluation, we set the communication bandwidth to 10 Mbits/sec. Figure 3(a) shows that for the fixed bandwidth and field size ($q = 5051$), the communication time overhead is minimal compared to the computation time which is bottle-necked by encoding. As illustrated in Fig. 3(b), the time taken to construct $\{\mathbf{Z}^\ell\}_{\ell\in[L]}$ at party P_i is the computation bottleneck in the encoding phase. The main reason for this is the fact that $\mathbf{Z}^\ell$ needs to be full-rank for Fed-K-PSI to be correct. Although, we discuss in Remark 2 that a matrix with iid uniformly distributed elements is full-rank with high-probability, in implementation, we need to verify that the generated matrix is full-rank otherwise we re-sample a new matrix. Verifying the rank of the generated matrix is the bottleneck operation in generating $\mathbf{Z}^\ell$.

Dependency on K

To study the effect of varying K on the complexity of our protocol, we consider an experimental setting with $N = 10$ parties, while we vary the target intersection threshold from $K = 2$ to 10 assuming a communication bandwidth of 10 Mbits/sec. Figure 4 illustrates the dependency of Fed-K-PSI on the intersection threshold K. Similar to Figure 3, the time for constructing $\{\mathbf{Z}_i^\ell\}_{\ell\in[L]}$ at party P_i dominates the run time for the encoding phase. Albeit, having different complexity levels, all modules in the encoding phase peak in complexity at $K = N/2 = 5$ and decrease as K approaches 1 or N. This is due to the fact the number of subsets of $[N]$ of K peak at $K = N/2$. Similar to Fig. 3(a), the encoding phase is the bottleneck for the protocol with the current bandwidth as we observe a similar trend in Fig. 4(a).

6 Conclusion

In this work, we proposed the K-PSI problem, a new flavor of the multi-party PSI problem within the federated analytics framework. In K-PSI, a server only knows the identifier of a datapoint (but not its associated value at different parties) and wishes to find the identifiers appearing with repeated values at least K of N parties. To address this problem, we developed a novel protocol named Fed-K-PSI. We showed that Fed-K-PSI is correct and achieves a strong information-theoretic privacy guarantee for identifiers that do not satisfy the query. Finally, we evaluated the complexity of Fed-K-PSI theoretically and through empirical experiments highlighting the effect of the different system parameters on the performance of Fed-K-PSI.

References

[Abadi et al.(2020)] Aydin Abadi, Sotirios Terzis, and Changyu Dong. 2020. Feather: Lightweight Multi-party Updatable Delegated Private Set Intersection. *IACR Cryptol. ePrint Arch.* 2020 (2020), 407.

[Bhowmick et al.(2021)] Abhishek Bhowmick, Dan Boneh, Steve Myers, and Kunal Talwar Karl Tarbe. 2021. The Apple PSI System.

[Borel(2012)] Armand Borel. 2012. *Linear algebraic groups*. Vol. 126. Springer Science & Business Media, New York.

[Branco et al.(2021)] Pedro Branco, Nico Döttling, and Sihang Pu. 2021. Multiparty Cardinality Testing for Threshold Private Intersection. In *Public-Key Cryptography – PKC 2021*, Juan A. Garay (Ed.). Springer International Publishing, Cham, 32–60.

[Chen et al.(2017)] Hao Chen, Kim Laine, and Peter Rindal. 2017. Fast Private Set Intersection from Homomorphic Encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1243–1255. https://doi.org/10.1145/3133956.3134061

[Dachman-Soled et al.(2009)] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. 2009. Efficient Robust Private Set Intersection. In *Applied Cryptography and Network Security*, Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 125–142.

[Diffie and Hellman(1976)] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE transactions on Information Theory* 22, 6 (1976), 644–654.

[Dong et al.(2013)] Changyu Dong, Liqun Chen, and Zikai Wen. 2013. When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security* (Berlin, Germany) *(CCS '13)*. Association for Computing Machinery, New York, NY, USA, 789–800. https://doi.org/10.1145/2508859.2516701

[Duong et al.(2020)] Thai Duong, Duong Hieu Phan, and Ni Trieu. 2020. Catalic: Delegated PSI Cardinality with Applications to Contact Tracing. In *Advances in Cryptology – ASIACRYPT 2020*, Shiho Moriai and Huaxiong Wang (Eds.). Springer International Publishing, Cham, 870–899.

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

[Freedman et al.(2004)] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. 2004. Efficient Private Matching and Set Intersection. In *Advances in Cryptology - EUROCRYPT 2004*, Christian Cachin and Jan L. Camenisch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–19.

[Ghosh and Simkin(2019)] Satrajit Ghosh and Mark Simkin. 2019. The communication complexity of threshold private set intersection. In *Annual International Cryptology Conference*. Springer, 3–29.

[Huang et al.(2012)] Yan Huang, David Evans, and Jonathan Katz. 2012. Private set intersection: Are garbled circuits better than custom protocols?. In *NDSS*.

[Huberman et al.(1999)] Bernardo A Huberman, Matt Franklin, and Tad Hogg. 1999. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM conference on Electronic commerce*. 78–86.

[Ion et al.(2017)] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. 2017. Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. *IACR Cryptol. ePrint Arch.* 2017 (2017), 738.

[Kales et al.(2019)] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. 2019. Mobile private contact discovery at scale. In *28th USENIX Security Symposium USENIX Security 19)*. 1447–1464.

[Kamara et al.(2014)] Seny Kamara, Payman Mohassel, Mariana Raykova, and Saeed Sadeghian. 2014. Scaling private set intersection to billion-element sets. In *International Conference on Financial Cryptography and Data Security*. Springer, 195–215.

[Kolesnikov et al.(2017)] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. 2017. Practical multi-party private set intersection from symmetric-key techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1257–1272.

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

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

[Pinkas et al.(2014)] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2014. Faster private set intersection based on OT extension. In *23rd USENIX Security Symposium USENIX Security 14)*. 797–812.

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

[Rindal and Rosulek(2017)] Peter Rindal and Mike Rosulek. 2017. Improved Private Set Intersection Against Malicious Adversaries. In *Advances in Cryptology – EUROCRYPT 2017*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer International Publishing, Cham, 235–259.

[Segal et al.(2017)] Aaron Segal, Antonio Marcedone, Benjamin Kreuter, Daniel Ramage, H. Brendan McMahan, Karn Seth, K. A. Bonawitz, Sarvar Patel, and Vladimir Ivanov. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. https://eprint.iacr.org/2017/281.pdf

[So et al.(2022)] Jinhyun So, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E Ali, Basak Guler, and Salman Avestimehr. 2022. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems* 4 (2022).