Abstract Federated learning has emerged as a promising approach for collaborative model training among multiple parties. A key challenge in production federated learning is aligning common records from heterogeneous datasets held by different parties, known as entity alignment. Specifically, production scenarios often encounter extreme data size unbalance - a client may have tens of records while the server has billions. Existing methods based on private set intersection (PSI) have efficiency bottlenecks encoding billions of server records. In this paper, we propose α-ESF, a practical entity alignment framework that breaks the efficiency bottleneck under extreme dataset unbalance. It exploits the asymmetry in the client- and server-side privacy to prune the server-side dataset, and incorporates a novel PSI-oriented index to reduce the encoding overhead during entity alignment execution. Uniquely, the time complexity of α-ESF is independent of the server-side dataset size. We evaluate α-ESF on a large-scale dataset from a real-world mobile telecom operator. Experiments show that α-ESF can reduce the entity alignment delay by at least two orders of magnitude than the state-of-the-arts under billion-scale dataset unbalance.

Keywords Entity Alignment, Federated Learning, Private Set Intersection

1 Introduction

Federated learning has emerged as a promising paradigm for collaborative model training that facilitates cooperation among multiple parties while ensuring data privacy [1–3]. This approach holds significant potential for a diverse range of analytics applications involving sensitive data [4]. For instance, in the field of medical big data analysis, federated learning has been successfully employed for tasks like disease prediction and diagnosis, enabling the utilization of patients' data without exposing their private medical information to external services [5]. Furthermore, banks and insurance companies have leveraged federated learning to develop accurate machine learning models for tasks such as risk assessment and customer recommendation [6].

The effectiveness of federated learning heavily relies on the successful alignment of entities across parties [7–9]. In this context, the alignment refers to the process of harmonizing heterogeneous data from different parties to identify shared entities for collective model training. An example scenario involves a bank and an insurance company aiming to collaborate on training a predictive model for estimating user insurance expenses. However, direct sharing of users' identifiers between the two parties is prohibited due to privacy regulations. In such cases, aligning the user entities becomes crucial to allow the bank to contribute relevant records for the same users during joint model training [10].

Private set intersection (PSI) is a fundamental technique that enables the identification of the intersection between two datasets owned by different parties without revealing any additional information to either party. PSI plays a vital role in aligning entities across parties while preserving the confidentiality of the underlying data [11]. By employing PSI, parties involved in federated learning can securely identify the entities that exist in both of their datasets. This process facilitates collaborative modeling specifically for those shared entities, enabling effective cooperation while preserving data privacy and confidentiality.

Limitations of Prior Arts. Despite extensive research on PSI in the community of secure multi-party computation (SMC) [12], it is still challeng-

ing to perform entity alignment with PSI efficiently in federated learning. It is because in federated learning, the data across multiple parties can be extremely unbalanced.

Our measurement study (Sec. 2.2) shows that state-of-the-art PSI solutions [13, 14] and the optimizations dedicated to unbalanced PSI [15, 16] fail to deliver satisfactory efficiency in case of unbalanced entity alignment in federated learning.

Our Approach. We break the efficiency bottleneck and propose a new entity alignment solution dedicated to data unbalanced federated learning via optimizations at two levels.

(i) Detach the large data size from the execution time. We harness the asymmetry in privacy requirements, an overlooked opportunity, to improve the efficiency of entity alignment on unbalanced data. The privacy asymmetry refers to the observation that the parties of many real-world applications impose different levels of privacy. For example, it is common for the insurance companies in the motivation example to reveal the last several digits of users' phone number [17], whereas the bank would not disclose any information at all. With such asymmetry, the party with large data size can utilize the revealed information about the other party to prune its own dataset, thus making the entity alignment time independent of the larger data size.

(ii) Reduce the encoding overhead via pre-computation. Traditional PSI based entity alignment solutions implicitly assume a single request, and perform encoding on both the parties per request. In practice, the server-side encoding may be shared and reused across requests (either requests from the same client or different clients). Therefore, we propose to pre-compute the server-side encoding (*e.g.*, encryption) and index the encryption for fast entity alignment. With a PSI-oriented index, we convert the per-query server-side encoding overhead as a one-off preparation step that is reused among multiple queries, which notably reduces the encoding overhead in entity alignment.

We implement the above optimizations as α-ESF, a new entity alignment solution delivers high efficiency under data unbalanced federated learning. It is featured with *(i)* α*-indistinguishability*, a client-side privacy requirement metric, allows quantitative server-side dataset pruning and is easily configurable in real-life client-side privacy control mechanisms; and *(ii) Bucket-ESF*, a novel PSI-oriented index that complies with the asymmetric privacy requirements while reduces encoding workload during entity alignment. On this basis, α-ESF operates in two phases: *pruning* and *verification*. In the pruning phase, we prune the server-side dataset based on the revealed information about the client, which is controlled by α-indistinguishability. In the verification phase, entity alignment is performed on the client-side dataset and the pruned server-side dataset leveraging the Bucket-ESF index. Importantly, the time complexity of α-ESF is determined by the *client-side dataset size only*, which drastically improves the efficiency of entity alignment solutions in case of extreme dataset unbalance.

Contributions and Roadmap. Our main contributions and results are summarized as follows.

- We utilize α-indistinguishability to quantify and control the client-side privacy requirement in entity alignment. To the best of our knowledge, this is the first work that exploits the asymmetry in privacy requirements to boost the efficiency of entity alignment in federated learning.
- We design a novel PSI-oriented index called

Bucket-ESF for fast entity alignment. It seamlessly integrates the asymmetric privacy, and converts the per-query server-side encoding overhead into a one-off preparation step.

- We propose a new PSI framework named α-ESF, which reduces the computation time, *i.e.*, the efficiency bottleneck of unbalanced PSI, from $O(n_S + n_C)$ in state-of-the-art solutions [13–16] to only $O(\sqrt{\alpha} \cdot n_C)$, where n_C and n_S are the sizes of client- and server-side datasets, respectively ($n_C \ll n_S$).
- We conduct evaluations on both synthetic and real-world billion-scale datasets. The experimental results show that our α-ESF solution is at least 2 orders of magnitude faster than the prior arts [13–16] in case of billion-scale dataset unbalance.

In the rest of this paper, we formulate the entity alignment problem in Sec. 2 and introduce the key techniques and overview of our solution framework in Sec. 3. We elaborate on the detailed design in Sec. 4-6 and present the evaluations in Sec. 7. We review related work in Sec. 8, and finally conclude in Sec. 9.

2 Problem Statement

In this section, we introduce the entity alignment problem in federated learning (Sec. 2.1) and present a measurement study on the limitations of prior PSI based entity alignment solutions (Sec. 2.2).

2.1 Entity Alignment Problem

We consider the client-server architecture commonly adopted in real federated learning applications, which involve two parties, the *server* $\mathcal{S}$ and the *client* C. The server S holds a private set $D_S = \{s_1, s_2, \cdots, s_{n_S}\}$ with size n_S. Each element s_i is a sample in domain $\mathcal{X}$. The client C has a private set $D_C = \{c_1, c_2, \cdots, c_{n_C}\}$ with size n_C, where $c_j \in \mathcal{X}$ is sampled from the same domain.

The entity alignment in many federated learning applications have two characteristics. *(i) Unbalanced dataset sizes.* The server-side dataset is orders of magnitude larger than the client-side, *i.e.*, $n_S \gg n_C$, where n_S can be at billion scale and n_C is often no more than several hundred. *(ii) Asymmetric privacy requirements.* Stringent privacy guarantee is imposed on the server-side yet less restrictive privacy control is applied to the client-side. For example, the last few digits of a user's phone number are often revealed to the bank for entity alignment [17].

As previous studies [12, 13, 15], we assume the semi-honest (*a.k.a.* honest-but-curious) adversary model. That is, the two parties (server and client) will honestly execute the procedures assigned to them, but they may keep all the intermediate computations and received messages, and analyze the messages to try to learn extra information about the other party.

Finally, we can define the following entity alignment problem.

Definition 1 (Entity Alignment)**.** Given two private sets D_S and D_C held by the client C and the server $\mathcal{S}$ respectively ($n_C \ll n_S$), the entity alignment problem aims to find the intersection of these two sets $D_S \cap D_C$ and return it to client C. The server may only infer limited information about client specified by the client-side privacy control, while the client should not learn any extra information about the server expect the resulting intersection set.

Example 1. Assume domain $\mathcal{X}$ is all the 16-bit integers $\{0, \cdots, 65535\}$. The server holds all elements in form $4x + 1$, *i.e.*, $D_S = \{1, 5, \cdots, 12353,$

$\cdots$, 65533}. The client holds two elements D_C = {12353, 12363}. The result of the entity alignment problem is $D_S \cap D_C$ = {12353}. In practice, C can selectively expose several digits of D_C, *e.g.*, the last digit is 3 for all elements in D_C, while the server should keep all its dataset confidential except the result 12353.

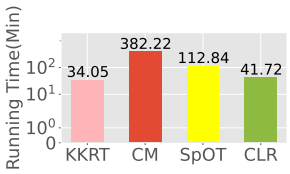
2.2 Motivation Study

We now show through measurements that prior PSI based solutions [13–16] are highly inefficient on entity alignment, which motivates our study.

Setups. Existing PSI based solutions roughly fall into two categories, one agnostic to dataset unbalance, and the other dedicated to dataset unbalance. In this measurement, we choose KKRT [13] and CM [14], two state-of-the-art generic PSI solutions, and CLR [15] and SpOT [16], two latest unbalanced PSI solutions. We measure the running time of a PSI query on a client-side dataset size of 100 and a server-side dataset size of 1 billion, which is typical for real-world federated learning applications to banks [18]. Other experimental details are in Sec. 7.1.

Results. Fig. 1a plots the overall running time for entity alignment. Even with a client-side dataset size of only 100, it still takes more than 30 minutes to process entity alignment over the billion-scale server-side dataset. Fig. 1b further shows the running time breakdown. We can observe that the communication time is negligible compared to the computation time. The communication time is only 3.01%, 0.33%, 0.07%, and 0.11% of the overall running time for KKRT [13], CM [14], SpOT [16] and CLR [15], respectively.

Discussions. We make the following notes from the measurements.



(a) Running time

PSI Solution	Comp.	Comm.
KKRT [13]	96.99%	3.01%
CM [14]	99.67%	0.33%
SpOT [16]	99.93%	0.07%
CLR [15]	99.89%	0.11%

(b) Breakdown

Fig. 1 The overall running time and breakdown of prior arts in unbalanced entity alignment scenario.

(i) Prior PSI solutions are inefficient with billion-scale dataset unbalance. Generic solutions such as KKRT [13] and CM [14] are inefficient because they encode (*e.g.*, pseudo-randomly evaluate or encrypt) all the server-side elements for each entity alignment request. The state-of-the-art unbalanced PSI solutions SpOT [16] and CLR [15] are also inefficient because they mainly reduce the communication cost, which is negligible compared with the computation cost. In summary, the efficiency bottleneck lies in the computation cost (mainly encoding) of server-side dataset.

(ii) None of the prior arts have utilized the asymmetric privacy requirements in practice, *i.e.*, the less stringent client-side privacy constraint than the server-side, to trade privacy for efficiency.

(iii) Existing studies neglect the optimization opportunities in practice. Once deployed, the server-side often processes multiple requests from the clients. This may allow preparation and reuse of computation-intensive operations across requests, and thus reduce the overall execution time.

3 α-ESF Overview

This section presents an overview of α-ESF, a practical framework for efficient entity alignment even in case of extreme dataset unbalance.

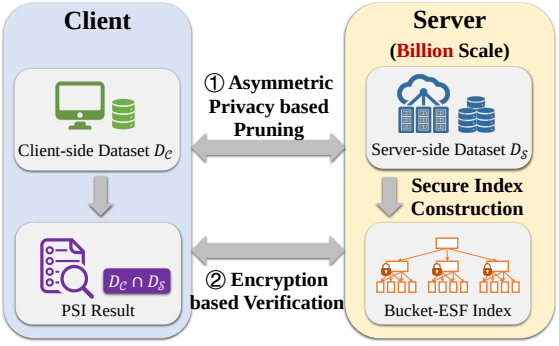


Fig. 2 Overview of α-ESF framework.

Principles and Challenges. As previously mentioned, the efficiency bottleneck of entity alignment lies in the computation time of *encoding* on billion-scale *server-side dataset*. Our solution breaks this bottleneck from two aspects.

(i) Exploit indexing to convert per-query server-side encoding overhead to one-off pre-processing. In practice, the server-side usually processes entity alignment requests from multiple clients. This allows pre-computing the server-side encoding and sharing the results among clients to amortize the server-side encoding overhead. The challenge is how to design indexing techniques that allow fast search on the encoded data while satisfying the predefined privacy requirements.

(ii) Harness asymmetric privacy requirements to prune server-side dataset for entity alignment. Real-world federated learning applications often impose less stringent privacy control on the client-side, which holds the potentials to prune server-side dataset. The challenge is how to quantify the client-side privacy requirement and configure server-side dataset pruning to balance privacy and efficiency.

Solution Workflow. We implement the principles and address the challenges above via α-ESF, which consists of a one-off *secure index construction* step, and a two-phase processing framework (*asymmetric privacy based pruning* and *encryption based verification*) for entity alignment (see Fig. 2).

- *Secure Index Construction (Sec. 4).* We devise *Bucket-ESF*, a novel PSI-oriented index that allows fast search while complying with privacy constraints (α-indistinguishability in our case). The index construction is a one-off effort that serves multiple entity alignment requests.
- *Asymmetric Privacy based Pruning (Sec. 5).* We propose to apply *α-indistinguishability* as a metric to quantify the client-side privacy requirement. On this basis, we design pruning strategies that reduce the server-side dataset for further processing while ensuring the α-indistinguishability of the client. This module is the core that detaches the server-side data size from execution time, which notably improves the efficiency of unbalanced entity alignment.
- *Encryption based Verification (Sec. 6).* We propose a homomorphic encryption based verification procedure upon the pruned Bucket-ESF index. It returns the entity alignment result to the client without revealing any extra information to either party. We also propose compression techniques to reduce the encryption overhead.

4 Secure Index Construction

As a prerequisite for α-ESF framework, the server constructs a PSI-oriented index of its billion scale dataset, called Bucket-ESF. To ensure data security, the index construction relies on a homomorphic encryption scheme, named Paillier Encryption. In the following, we first introduce the preliminary of *Paillier Encryption* and then elaborate on our *Bucket-ESF index*.

4.1 Preliminary: Paillier Encryption

Paillier [19] is a partially homomorphic encryption scheme, which allows efficient secure computations over encrypted data. Given a plaintext u, $\mathcal{E}(u)$ is used to denote its ciphertext by Paillier encryption. As next, we introduce the functionalities of Paillier encryption and refer to [20] for detailed implementation.

- The Paillier encryption supports an addition between two ciphertexts $\mathcal{E}(u_1)$ and $\mathcal{E}(u_2)$, *i.e.*, $\mathcal{E}(u_1 + u_2) = \mathcal{E}(u_1) \oplus \mathcal{E}(u_2)$, where $\oplus$ denotes the homomorphic addition operation.
- The Paillier encryption supports a multiplication between a ciphertext $\mathcal{E}(u_1)$ and a plaintext u_2, *i.e.*, $\mathcal{E}(u_1 \cdot u_2) = \mathcal{E}(u_1)^{u_2}$.

Paillier encryption has been proved and demonstrated to guarantee semantic security [20], *i.e.*, for a given ciphertext, an attacker cannot deduce any sensitive information about the plaintext.

In our framework, the public key of Paillier, which is shared between the client and server, is used in the encryption and secure computations. By contrast, Paillier's private key, which is held by the server only, is used to perform decryption.

4.2 Bucket-ESF Index

As shown in Fig. 3, the Bucket-ESF index is a *two-level* structure. It enables a fast search over ciphertexts while complying with the privacy constraints. The *first level* of Bucket-ESF partitions the server-side entire dataset into $|B|$ buckets and the elements inside each bucket are determined by a hash function $\mathcal{H}_b : \mathcal{X} \rightarrow B$. The *second level* of Bucket-ESF is an encrypted data structure, called EncSum Filter, which supports fast and secure membership test. As next, we introduce the details of EncSum Filter and illustrate the construction of Bucket-ESF index.

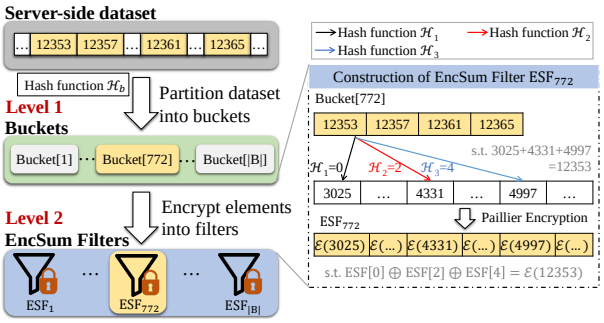


Fig. 3 Two-level structure of Bucket-ESF index.

EncSum Filter. Our EncSum Filter is defined by an array $\mathsf{ESF}[1..N]$ and several hash functions $\mathcal{H}_1$, $\mathcal{H}_2, \mathcal{H}_3$ (since our results show three hash functions is enough to process billion-scale dataset, we assume three hash functions here). For a given element u, the EncSum Filter stores the Paillier ciphertext $\mathcal{E}(u)$ into the array ESF according to the values of functions $\mathcal{H}_1$ to $\mathcal{H}_3$. Specifically, $\mathcal{E}(u)$ is defined as

$$\mathcal{E}(u) = \bigoplus_{z=1}^{3} \mathsf{ESF}[\mathcal{H}_z(u)]. \tag{1}$$

where $\oplus$ is the Paillier's addition operator. Given a set of elements $\{u_1, \cdots, u_n\}$, the EncSum Filter can be built as follows.

- We initialize the array $\mathsf{ESF}[1..N]$ as null ($\perp$).
- We find appropriate hash functions $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$ until each element u_i can be successfully divided and inserted into ESF while satisfying the condition in Eq. (1). Specifically, the $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ are first randomly generated from a family of hash functions. Then a topological sort is used to check whether all elements can be inserted into ESF successfully. If not, the hash functions will be regenerated randomly until the check becomes true.

Algorithm 1: Secure Index Construction

Input: Private set D_S, bucket set B

Output: The Bucket-ESF index

1 Choose a hash function $\mathcal{H}_b : \mathcal{X} \to B$ randomly
2 Initialize $Bucket[t] \leftarrow \emptyset$ for all buckets $t \in B$
3 **foreach** *element* $s_i \in D_S$ **do**
4 $\quad Bucket[\mathcal{H}_b(s_i)] \leftarrow Bucket[\mathcal{H}_b(s_i)] \cup \{s_i\}$
5 **foreach** *bucket* $t \in B$ **do**
6 $\quad \mathsf{ESF}_t \leftarrow$ build EncSum Filter for $Bucket[t]$
7 **return** *the* 1^{st} *level* $\mathcal{H}_b$ *and the* 2^{nd} *level* $\{\mathsf{ESF}_t\}$

- For any remaining $\mathsf{ESF}[j] = \perp$ after inserting all elements, we replace $\mathsf{ESF}[j]$ with a randomly chosen Paillier ciphertext.

It has been proven that, when n is large, we can find such hash functions with probability $\Omega(1)$ [21]. Our experimental study shows that, if we set the size N of EncSum Filter to $1.23n + 64$, where n is the number of elements in the server-side dataset, the probability of successful construction is nearly 100%. It is better than the traditional membership test data structures such as Bloom Filter, which requires 20 hash functions and $N = 28.8n$ memory cost to achieve the same rate of successful construction [22].

Index Construction. Alg. 1 shows the construction of the Bucket-ESF index. At the first level of Bucket-ESF index, a hash function $\mathcal{H}_b : \mathcal{X} \to B$ is randomly chosen by the server, which partitions the whole server-side dataset into $|B|$ buckets. Each element s_i held by the server is assigned to the $\mathcal{H}_b(s_i)$-th bucket. The server then constructs the EncSum Filter ESF_t for the elements in each bucket $t \in B$ as the second level of Bucket-ESF index. Finally, the bucket hash function $\mathcal{H}_b$ and the EncSum Filters $\{\mathsf{ESF}_1, \cdots, \mathsf{ESF}_{|B|}\}$ are assembled as the Bucket-ESF index.

Complexity Analysis. In Alg. 1, the server takes $O(n_S)$ time to partition its dataset into $|B|$ buckets (lines 2-4). The construction of all EncSum Filters (lines 5-6) also takes $O(n_S)$ time. Thus, the total time complexity is $O(n_S)$. The memory cost of the index is also $O(n_S)$. Note that the Bucket-ESF index construction is a one-off effort, which means its time cost is independent of entity alignment response.

5 Asymmetric Privacy based Pruning

This section introduces α-indistinguishability, a metric to quantify client-side privacy (*i.e.*, asymmetric privacy) requirement, and then proposes our pruning strategies based on this metric.

5.1 α-indistinguishability: Metric for Client-side Privacy Requirement

The asymmetric privacy in the entity alignment problem enables the trade off between privacy and efficiency. For example, the client can achieve more efficient entity alignment response by selecting their preferred privacy requirement. Intuitively, the first question is how to define the privacy requirement of the client-side in entity alignment.

In our work, the client-side privacy requirement can be measured by the probability that whether an element is in the client-side dataset from the server-side view [23, 24]. Specifically, from the server-side view, if the probability that a server's element s appears in the client-side dataset D_C is not greater than $1/\alpha$, where the client-defined parameter α denotes the privacy preservation level, then the privacy of this client can still be protected. For example, by setting $\alpha = 10^4$, the server can only infer whether an element belongs to the client-side

dataset with a probability 0.01%, which is acceptable in many real-life applications like identity verification for risk control [18]. Accordingly, we propose a metric called α-indistinguishability, which is formally defined as follows.

Definition 2 (α-Indistinguishability)**.** Given a private dataset D_C held by a client C, it is α-indistinguishable for the server $\mathcal{S}$ if:

$$\forall s \in D_{\mathcal{S}}, \Pr(s \in D_C) \leq \frac{1}{\alpha}. \tag{2}$$

This metric performs an easy configuration mechanism for the client to achieve controllable privacy leakage in entity alignment by setting the parameter α. For example, in the scenario of federated user recommendation, a financial company can achieve $\alpha = 10^4$ indistinguishability by hiding the internal 4 digits of the debtors' phone numbers.

5.2 Pruning Strategies

By utilizing our Bucket-ESF index, we introduce how to prune the billion-scale server-side dataset while complying with the privacy requirement of α-indistinguishability. In our secure index, each bucket contains around $R = |\mathcal{X}|/|B|$ possible elements. Thus, for any client's element $c_j \in D_C$, the client can prune the server-side dataset with the bucket $\mathcal{H}_b(c_j)$ and another several obscured buckets that are used to hide the true bucket, to make sure the total number of possible elements in these buckets are no smaller than α. In this way, the pruning method can satisfy α-indistinguishability for the client.

Basic Pruning. Alg. 2 illustrates the basic pruning method. Specifically, the client first requests the hash function $\mathcal{H}_b$ that is used by the server to partition its dataset into buckets. Each client's element needs $L = \lceil \alpha/R \rceil$ buckets (of totally α possible elements) to satisfy the privacy requirement of

Algorithm 2: Pruning Strategy

Input: Private set $D_{\mathcal{S}}$ and D_C, the parameter α and the Bucket-ESF index

Output: EncSum Filters $\{\mathsf{ESF}_{p_i}\}$ of the Bucket-ESF index

1 **Client** C **executes:**
2 Get the hash function $\mathcal{H}_b$ of server's Bucket-ESF index
3 $PrunESF \leftarrow \emptyset$
4 $R \leftarrow \frac{|\mathcal{X}|}{|B|}$, $L \leftarrow \lceil \frac{\alpha}{R} \rceil$
5 **foreach** $c_j \in D_C$ **do**
6 **if** $\mathcal{H}_b(c_j) \notin PrunESF$ **then**
7 Randomly generate $L-1$ distinct dummy buckets $d_2, \cdots, d_L \in B$
8 $PrunESF \leftarrow PrunESF \cup \{\mathcal{H}_b(c_j), d_2, \cdots, d_L\}$
9 Send the pruned buckets $PrunESF$ to server
10 **Server** $\mathcal{S}$ **executes:**
11 **foreach** $p_i \in PrunESF$ **do**
12 Send the p_i-th pruned Filter ESF_{p_i} to client C

α-indistinguishability, where those required buckets are denoted by $PrunESF$ (lines 3-4). Then, the $\mathcal{H}_b(c_j)$-th bucket at the server-side may also contain the element c_j in the client-side dataset. Thus, those buckets should be included in the candidate set $PrunESF$ (lines 5-6). Moreover, for each client's element, another $L-1$ obscured bucket IDs will be added to the candidates to preserve the client's privacy (lines 7-8). Finally, the client sends all the candidate bucket IDs to the server to request the corresponding EncSum Filters at the server side (lines 9-12). Overall, this algorithm takes $O(n_C \cdot L)$ time, and the total communication cost is bounded by the total size of the requested EncSum Filters, $O(n_C \cdot L \cdot R) = O(n_C \cdot \alpha)$.

Overhead Optimized Pruning. Next, we discuss how to tune the parameters L and R to achieve the optimized pruning overhead. The key insight is that, after the client receives EncSum Filters which are in a serialization form, it also needs to deserialize the EncSum Filters containing elements in D_C. Thus, the time cost of deserialization is $O(n_C \cdot R)$ and the communication cost of sending the bucket numbers (line 9) is $O(n_C \cdot L)$. Denote the time for communicating one bit as T_c and the time for deserializing one bit as T_d, the total overhead is $O(T_c \cdot n_C \cdot L + T_d \cdot n_C \cdot R)$. Due to $L \cdot R = \alpha$, the optimized pruning overhead can be achieved when

$$L = \sqrt{\frac{T_d \cdot \alpha}{T_c}}, \quad R = \sqrt{\frac{T_c \cdot \alpha}{T_d}}. \tag{3}$$

Under this condition, the time complexity is $O(\sqrt{\alpha} \cdot n_C)$. To achieve such optimized pruning, the server can build the Bucket-ESF index with the bucket size defined in Eq. (4) based on the pre-defined parameter α, communication cost and deserialization cost (per bit),

$$|B| = |\mathcal{X}| \cdot \sqrt{\frac{T_d}{T_c \cdot \alpha}}. \tag{4}$$

Theorem 1. The asymmetric privacy based pruning (Alg. 2) can meet the privacy requirement of client, i.e., α-indistinguishability.

Proof. From the view of the server, it only obtains the candidate bucket set $PrunES\,F$ sent by the client. Firstly, the server can detect whether a bucket in the candidate set contains an actual element of the client with a probability $1/L$. Secondly, each bucket has $R = |\mathcal{X}|/|B|$ possible elements, so the possibility of correctly detecting the client's element in a certain bucket is $1/R$. Since $L = \lceil \alpha/R \rceil$, the probability that the server can infer whether an element s is in D_C is bounded by

$$\Pr(s \in D_C) = \frac{1}{L} \cdot \frac{1}{R} = \frac{1}{\lceil \alpha/R \rceil \cdot R} \leq \frac{1}{\alpha}. \tag{5}$$

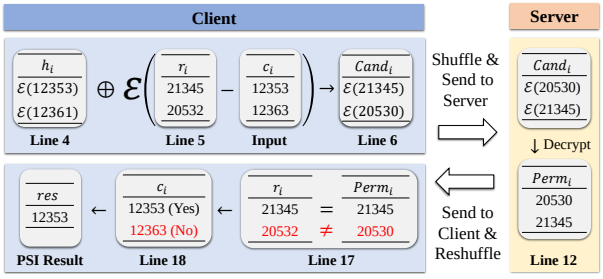


Fig. 4 Example of encryption based verification.

□

6 Encryption based Verification

After the client receives the EncSum Filters, this section presents how to correctly and efficiently achieve the entity alignment between server and client.

The entity alignment is now conducted on the client-side dataset D_C and the received EncSum Filters $\{\mathsf{ESF}_{p_i}\}$. For each element c_j, the client needs to test whether c_j exists in the $\mathcal{H}_b(c_j)$-th EncSum Filter. Specifically, the ciphertext stored in the EncSum Filter is

$$h_j = \bigoplus_{z=1}^{3} \mathsf{ESF}_{\mathcal{H}_b(c_j)}[\mathcal{H}_z(c_j)]. \tag{6}$$

The client needs to test whether the value of h_j defined in Eq. (6) is the corresponding ciphertext of c_j. Since only the server holds the private key that can perform decryption, the client can add the ciphertext $\mathcal{E}(-c_j)$ to h_j by utilizing the homomorphic property of Paillier encryption, and hand over the decryption to the server.

Algorithm Details and Analysis. Alg. 3 illustrates the procedure of our encryption based verification. For each client-side element c_j, the corresponding ciphertext from server h_j is extracted from the EncSum Filters $\mathsf{ESF}_{\mathcal{H}_b(c_j)}$ (line 4). To avoid the server inferring extra information about c_j, the client can add another randomly chosen value r_j to c_j and

Algorithm 3: Verification Workflow

Input: Private set D_c, pruned EncSum Filters ESF_{p_i}

Output: The private set intersection res

1 **Client** C **executes:**
2 Initialize an array $cand[1..n_C]$
3 **for** $j \leftarrow 1$ ***to*** n_C **do**
4 $h_j \leftarrow \oplus_{z=1}^{3} ESF_{\mathcal{H}_{b(c_j)}}[\mathcal{H}_z(c_j)]$
5 $r_j \leftarrow$ a randomly chosen integer
6 $cand[j] \leftarrow h_j \oplus \mathcal{E}(r_j - c_j)$
7 Generate a random permutation π
8 Shuffle $cand$ by π and send it to the server
9 **Server** $\mathcal{S}$ **executes:**
10 Initialize $perm$ as an array of $[1..|cand|]$
11 **for** $i \leftarrow 1$ ***to*** $|cand|$ **do**
12 $perm[i] \leftarrow$ Paillier.decrypt($cand[i]$)
13 send $perm$ to client
14 **Client** C **executes:**
15 $res \leftarrow \emptyset$
16 **for** $j \leftarrow 1$ *to* $|perm|$ **do**
17 **if** $perm[j] = r_{\pi(j)}$ **then**
18 $res \leftarrow res \cup \{c_{\pi(j)}\}$

compute the $h_j \oplus \mathcal{E}(r_j - c_j)$ (lines 5-6). It then shuffles the ciphertexts by a random permutation π to protect the original order (line 8). These steps executed by the client take $O(n_C)$ time. The server only needs to decrypt the ciphertext $cand[j]$ and returns the corresponding plaintext $perm[j]$ to the client, which also takes $O(n_C)$ time (lines 9-13). Finally, the client can check whether $perm[j]$ equals $r_{\pi(j)}$. Since $cand[j] = h_{\pi(j)} \oplus \mathcal{E}(r_{\pi(j)} - c_{\pi(j)})$, $perm[j] = r_{\pi(j)}$ implies that $h_{\pi(j)}$ is the ciphertext of $c_{\pi(j)}$, which means $c_{\pi(j)}$ is in the entity alignment result. Fig. 4 is an illustration of our encryption based verification. In summary, the total time complexity and communication cost are $O(n_C)$, which are only determined by the size of the client-side dataset. Notice that the time complexity of our solution is notably lower than that of the state-of-the-arts [13, 15, 16], *i.e.*, $O(n_S + n_C) = O(n_S)$ in the unbalanced PSI based solutions.

Ciphertext Compression based Optimization. The encryption based verification (Alg. 3) can be further accelerated by our compression technique. In practice, the security parameter κ of Paillier encryption is often at least 512 bits, which means we can concatenate multi-integers as one to perform encryption and decryption together. For example, we can represent each encrypted element in $\mathcal{X}$ into a 50-bit integer, *i.e.*, we can compress 10 elements into a big integer to process Paillier's encryption and computation. With such a compression technique, the computation and communication cost of the verification phase can be significantly reduced.

Theorem 2. Our α-ESF framework (Alg. 2 and Alg. 3) can securely align the entities under the semi-honest model.

Proof. The security of our entity alignment solution (Alg. 2 and Alg. 3) depends on the underlying encryption scheme. For the *client's view*, it only obtains the ciphertext h_j from the EncSum Filters and the decrypted result about c_j. Since the encryption scheme Paillier is semantically secure [20], the ciphertexts h_j and the plaintext data from the server s_i are indistinguishable. Thus, the client cannot learn any information about the server except for the query answer (*i.e.*, the intersection set). For the *server's view*, it can get the pruned bucket IDs in Alg. 2 and the decrypted set $\{s_{\pi(j)} - c_{\pi(j)} + r_{\pi(j)}\}$ in Alg. 3. Since the server cannot get the random permutation π and variable r, it can learn neither the element c_i nor the intersection size. And the security of bucket IDs is guaranteed by α-indistinguish-

ability (Theorem 1). Thus, the α-ESF is secure under the semi-honest adversary model. □

7 Experimental Study

This section presents the evaluations of our entity alignment solution α-ESF.

7.1 Experiment Setup

Datasets. We evaluate our solutions on real and synthetic datasets.

- **Mobile Telecom Operator (MTO)**. It is the operating data from a leading mobile telecom operator in China, which contains 1.08 billion identities (*e.g.*, 11-digit phone numbers) from 31 provinces (server-side) and an identity verification query request with 226 records (client-side). To simulate real-world entity alignment scenario, we vary the size of client-side dataset from 1 to 200 by random sampling and denote the corresponding real datasets as MTO#1, MTO#50, MTO#100 and MTO#200, respectively.
- **Synthetic Datasets (SYN)**. For the scalability test, We randomly generate synthetic datasets as well. In SYN, we vary the size of the server-side data from *50 million* to *5 billion* and the size of the client-side data from one to ten thousand. Each element in the datasets is an 11-digit integer.

Baselines. We compare our α-ESF with existing balanced PSI based solutions (KKRT [13], CM [14], EC-DH [25] and PaXoS [26]) and unbalanced PSI based solutions (SpOT [16] and CLR [15]). The details are as follows.

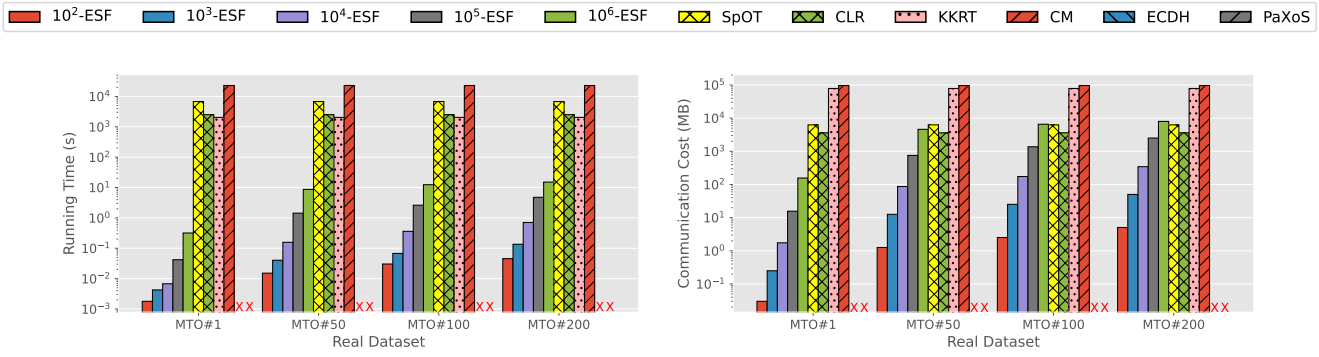
- **KKRT** [13]. It is one of the state-of-the-art entity alignment solution with balanced PSI techniques implemented with a novel oblivious pseudo-random function.
- **CM** [14]. It is one of the state-of-the-art entity alignment solutions with balanced PSI techniques which balances the network traffic and computation overhead.
- **ECDH** [25]. It is a frequently-used entity alignment solution in industry [27,28], which is based on the classic Diffie–Hellman key exchange.
- **PaXoS** [26]. It utilizes a new oblivious key-value store called probe-and-XOR of strings for entity alignment.
- **SpOT** [16]. It is one of the state-of-the-art entity alignment solution based on unbalanced PSI that applies sparse OT for low communication cost.
- **CLR** [15]. It is one of the state-of-the-art entity alignment solution based on unbalanced PSI. It is designed to reduce communication cost based on fully homomorphic encryption.

Evaluation Metrics. In the experimental study, we mainly focus on the *efficiency* of different entity alignment solutions, which is assessed by the running time, *i.e.*, total time to perform entity alignment (the time to pre-construct the Bucket-ESF index is excluded). Besides, we also measure the communication cost of the compared solutions.

Implementation. We conduct all experiments on two machines, one as the server and the other as the client. Both the server and client have 24 2.40GHz Intel(R) Xeon(R) Gold 6240R CPU processors and 1TB memory with Centos 7.9 OS. The network bandwidth between these machines is up to 10Gbps. All the algorithms are implemented by GNU C++ with NTL library [29] for big integer computation and CryptoTools library [30] for network connec-

Table 1 The running time and communication cost on billion-scale mobile telecom operation (MTO) datasets. We mark the "best performance" as **green**. "N.A." denotes the solutions are crashed or time out in experiments.

Dataset	Running Time (Sec)							Communication Cost (MB)						
	KKRT	CM	ECDH	PaXoS	SpOT	CLR	Ours	KKRT	CM	ECDH	PaXoS	SpOT	CLR	Ours
MTO#1	2033	22547	N.A.	N.A.	6273	2503	**0.01**	78643	96656	N.A.	N.A.	6266	3591	**1.74**
MTO#50	2056	22711	N.A.	N.A.	6318	2503	**0.18**	78643	96656	N.A.	N.A.	6266	3591	**86.3**
MTO#100	2082	22933	N.A.	N.A.	6814	2504	**0.38**	78643	96656	N.A.	N.A.	6266	3591	**173**
MTO#200	2112	23284	N.A.	N.A.	6917	2504	**0.70**	78643	96656	N.A.	N.A.	7311	3591	**344**



(a) Running time

(b) Communication cost

Fig. 5 Experimental results of varying α on real datasets. Note that ECDH and PaXoS fail to terminate on billion-scale data.

tion. The key size of Paillier encryption is $\kappa = 512$ bits.

7.2 Results on Billion-Scale Mobile Telecom Operator Datasets

For real-life efficiency evaluation, we deploy the compared algorithms on a leading mobile telecom operator in China, which holds 1.08 billion records of users' phone numbers from 31 provinces. Following the identity verification applications [17], we set the client-side privacy requirement α as 10^4. The construction of the Bucket-ESF index in our α-ESF solution takes 2.9 hours with 164 GB memory cost. We demonstrate the efficiency of our solution from two perspectives: *(i) overall performance compared to baselines*, and *(ii) performance under different privacy requirements of the client*.

Overall Performance. Table 1 lists the running time and communication cost of all compared algorithms on the mobile telecom operator datasets. We can make the following observations. ***(i)* Our α-ESF is 3 to 6 orders of magnitude faster than the baselines in running time over billion-scale server-side dataset**. Take dataset MTO#200 as a example. our α-ESF is 9880×, 3016×, 3576× and 33262× faster than SpOT, KKRT, CLR and CM, respectively. The running time of α-ESF is always less than 1 second, while other baselines take more than 30 minutes for entity alignment. It is because their time cost is mainly determined by the billion-scale of the server-side dataset. PaXoS is crashed and ECDH cannot terminate in 12 hours during our experiments. ***(ii)* Our α-ESF also outperforms the baselines in the communication cost.** The communication cost of baselines is between 3.5GB and 94.4GB. Existing balanced PSI based solutions

(KKRT and CM) take the highest communication cost, because they require encoding and sending all the billion-scale records to the client. Prior unbalanced PSI based solutions (SpOT and CLR) achieve relatively better communication cost due to their specialized optimizations for communication. By contrast, the communication overhead of our α-ESF framework mildly increases with the expansion of the client-side data size.

Performance under Different Privacy Requirements α. We further vary the client-side privacy requirement from 100 to 1 million to evaluate the performance of our α-ESF solutions. As shown in Fig. 5, our α-ESF still notably outperforms all the baselines in running time. Take MTO#200 as an example. As α increases, the running time of α-ESF increases slightly (from 0.05s to 15.1s). When the client-side privacy requirement is $\alpha = 10^6$, our α-ESF is still 134× faster than the runner-up solution (KKRT). The communication cost of α-ESF is relatively sensitive to the privacy requirements. Specifically, 10^2-ESF is 1240×, 710×, 15572× and 19139× lower than SpOT, CLR, KKRT and CM respectively. The 10^6-ESF takes similar communication cost to the baselines, SpOT and CLR. This is because our solution needs to transmit more Enc-Sum Filters from the server to the client as α increases, which leads to the increase of communication overhead.

Performance on Effectiveness. Fig. 6 show the model performance of vertical federated learning with our data alignment solution. To simulate the real world applications, we perform the federated training among the server and three clients. We can find that, by involving the server-side data based on our entity alignment solution, the performances of all clients and the federated model increase significantly. The accuracy gain of the entity alignment based federated learning is up to 3.93%, which demonstrates the effectiveness of the entity alignment in federated learning.

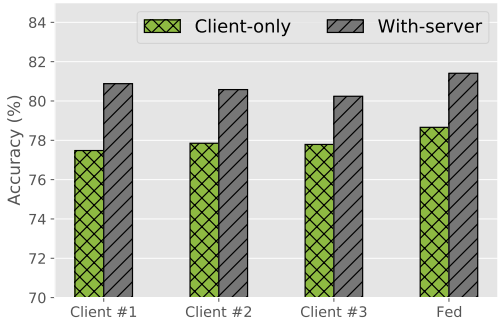


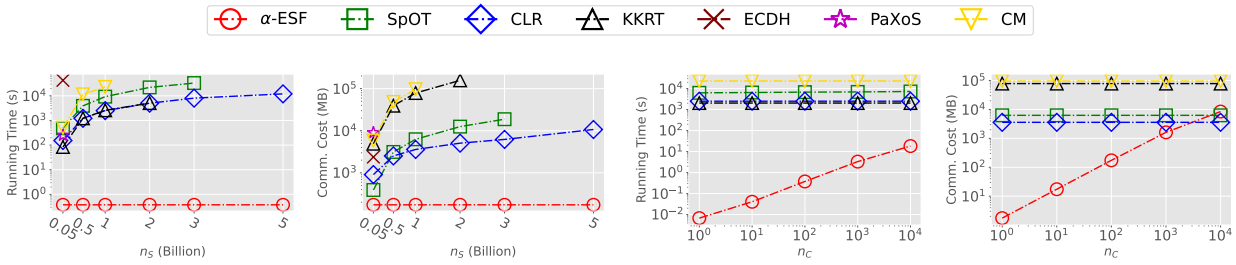
Fig. 6 Model performance with entity alignment solution.

7.3 Scalability Test on Synthetic Datasets

This subsection conducts the scalability test on datasets SYN and evaluates the impacts of different data scales. For default parameters, we set the server-side data size n_S as 1 billion, the client-side data size as 100, and the client-side privacy requirement α as 10^4.

Varying size of sever-side data n_S. Fig. 7a presents the results of varying n_S. In terms of running time, our α-ESF outperforms all compared baselines and KKRT is the runner-up. During the test, KKRT is up to 13436× slower than our α-ESF. When n_S is more than 50 million, PaXoS and ECDH are crashed or timed out, respectively. As n_S increases, other baselines become inefficient because their time complexities are determined by n_S. The running time of α-ESF remains stable because its time complexity is mainly determined by n_C. As for communication overhead, our α-ESF is not affected by n_S and performs the best at billion scale. By contrast, the communication cost of baselines linearly grows as n_S increases.

Varying size of client-side data n_C. As shown in Fig. 7b, the running time and communication cost



(a) Performance of varying the server-side data size n_S **(b)** Performance of varying the client-side data size n_C

Fig. 7 Scalability test on synthetic datasets (SYN).

of α-ESF are linear to n_C. Prior solutions perform stable because they need to encode and transmit all server-side data to the client. Our α-ESF still dominates all baselines in running time. Specifically, α-ESF is at least 396×, 113×, 136× and 1252× faster than SpOT, KKRT, CLR and CM, respectively. When $n_C = 10^4$, the communication cost of α-ESF is slightly higher than SpOT and CLR, but it only takes a small part of the overall efficiency overhead. Overall, it indicates that our α-ESF is more suitable for the unbalanced scenario.

8 Related Work

Entity alignment, as a fundamental pre-processing step in federated learning, has been studied in both the artificial intelligence and database committee. [31–37]. However, most of these studies assume that the data is insensitive for the service provider and hence needs no privacy protection. Recently, *private set intersection* (PSI) attracts widespread attention in entity alignment problem, since data privacy is becoming more important than ever. In the following, we review existing solutions to entity alignment from two categories: *balanced PSI based solutions* and *unbalanced PSI based solutions*. Approximate PSI query [23, 38, 39] is out of our scope because we focus on applications such as identity verification that require accurate results.

Balanced PSI based Entity Alignment Solutions. As one of the earliest studies on PSI, ECDH [25] is an easy-to-implement solution to balanced PSI based on the Diffie-Hellman key exchange [20]. However, its computation overhead can be prohibitive on large-scale datasets [12]. State-of-the-art balanced PSI solutions are designed based on oblivious transfer (OT) techniques [13,14,26,40]. Specifically, Kolesnikov *et al.* [13] propose a novel oblivious pseudo-random function based solution. PaXoS [26] and CM [14] further optimize the performance of balanced PSI in some specific settings. However, the performance of their solutions is not satisfactory under billion-scale dataset unbalance based on our experiments.

Unbalanced PSI based Entity Alignment Solutions. Most unbalanced PSI solutions rely on homomorphic encryption techniques. Freedman *et al.* [41] propose an unbalanced PSI solution with a partially homomorphic encryption scheme. Subsequently, Chen *et al.* [15] propose a communication-efficient unbalanced PSI solution by leveraging a fully homomorphic encryption scheme. SpOT [16] is a new OT based solution to unbalanced PSI with low communication cost. Kales *et al.* [42] construct an two phase unbalanced PSI solutions against malicious attackers in mobile device, where clients download a large cuckoo filter in the setup phase and then execute private set intersection. However,

if the server holds a set with billion elements, the client needs to download a large filter making such solution impractical [42]. In summary, existing solutions to unbalanced PSI mainly focus on reducing the communication cost, while the computation overhead can become the efficiency bottleneck in real-life applications.

9 Conclusion

In this paper, we explored the entity alignment problem in federated learning. Existing solutions are inefficient in case of extreme dataset unbalance because they require encoding on a billion-scale server-side dataset. In response, we propose α-ESF, an efficient entity alignment algorithm catered for extreme dataset unbalance. It is featured with two core techniques: α-indistinguishability-based server-side dataset pruning, and a novel PSI-oriented index for rapid search on server-side encryption. Unlike prior arts whose efficiency is restricted by the billion-scale server-side data size, the time complexity of our solution is mainly determined by the small-scale client-side data size. Extensive experiments show that our α-ESF is at least 2 orders of magnitude faster than prior arts in case of billion-scale dataset unbalance. We envision our work as a leap toward practical production federated learning.

References

1. Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Transactions on Privacy and Security, 2019, 10(2): 12:1–12:19
2. Tong Y, Zeng Y, Zhou Z, Liu B, Shi Y, Li S, Xu K, Lv W. Federated computing: Query, learning, and beyond. IEEE Data Engineering Bulletin, 2023, 46(1): 9–26
3. Zhang K, Song X, Zhang C, Yu S. Challenges and future directions of secure federated learning: a survey. Frontiers of Computer Science, 2022, 16(5): 165817
4. Song Z, Gu Y, Wang Z, Yu G. DRPS: efficient disk-resident parameter servers for distributed machine learning. Frontiers of Computer Science, 2022, 16(4): 164321
5. Chen Y, Qin X, Wang J, Yu C, Gao W. Fedhealth: A federated transfer learning framework for wearable healthcare. IEEE Intelligent Systems, 2020, 35(4): 83–93
6. Zhang Y, Shi Y, Zhou Z, Xue C, Xu Y, Xu K, Du J. Efficient and secure skyline queries over vertical data federation. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(9): 9269–9280
7. Cheng K, Fan T, Jin Y, Liu Y, Chen T, Papadopoulos D, Yang Q. Secureboost: A lossless federated learning framework. IEEE Intelligent Systems, 2021, 36(6): 87–98
8. Wu Z, Li Q, He B. A coupled design of exploiting record similarity for practical vertical federated learning. In: NeurIPS. 2022
9. Hu Y, Shen D, Nie T, Kou Y, Yu G. Biomedical entity linking based on less labeled data. Frontiers of Computer Science, 2022, 16(3): 163343
10. Liu F, Zheng Z, Shi Y, Tong Y, Zhang Y. A survey on federated learning: A perspective from secure multiparty computation. Frontiers of Computer Science, 2023
11. Agrawal R, Evfimievski A V, Srikant R. Information sharing across private databases. In: Proceedings of the 2003 ACM International Conference on Management of Data (SIGMOD). 2003, 86–97
12. Pinkas B, Schneider T, Zohner M. Scalable private set intersection based on OT extension. ACM Transactions on Privacy and Security, 2018, 21(2): 7:1–7:35
13. Kolesnikov V, Kumaresan R, Rosulek M, Trieu N. Efficient batched oblivious PRF with applications to private set intersection. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS). 2016, 818–829
14. Chase M, Miao P. Private set intersection in the internet setting from lightweight oblivious PRF. In: Proceedings of the 40th Annual International Cryptology Conference (CRYPTO). 2020, 34–63
15. Chen H, Laine K, Rindal P. Fast private set intersection from homomorphic encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS). 2017, 1243–1255
16. Pinkas B, Rosulek M, Trieu N, Yanai A. Spot-light: Lightweight private set intersection from sparse OT extension. In: Proceedings of the 39th Annual International Cryptology Conference (CRYPTO). 2019, 401–431

17. Zhao K, Li H, Gong Z, Cui J. Sades: An interactive system for sensitivity-aware desensitization towards tabular data. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM). 2021, 4828–4832
18. Group C M C. Operating data of china mobile, 2022
19. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT). 1999, 223–238
20. Katz J, Lindell Y. Introduction to Modern Cryptography, Second Edition. CRC Press, 2014
21. Botelho F C, Pagh R, Ziviani N. Simple and space-efficient minimal perfect hash functions. In: Proceedings of the 10th International Workshop on Algorithms and Data Structures (WADS). 2007, 139–150
22. Luo L, Guo D, Ma R T B, Rottenstreich O, Luo X. Optimizing bloom filter: Challenges, solutions, and comparisons. IEEE Communications Surveys & Tutorials, 2019, 21(2): 1912–1949
23. Wang S, Chen C, Zhang G. Similarity-based privacy protection for publishing k-anonymous trajectories. Frontiers of Computer Science, 2022, 16(3): 163605
24. Wang H, Xu Z, Zhang X, Peng X, Li K. An optimal differentially private data release mechanism with constrained error. Frontiers of Computer Science, 2022, 16(3): 161608
25. Meadows C A. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: Proceedings of the 1986 IEEE Symposium on Security and Privacy (S & P). 1986, 134–137
26. Pinkas B, Rosulek M, Trieu N, Yanai A. PSI from paxos: Fast, malicious private set intersection. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). 2020, 739–767
27. FATE . An industrial grade federated learning framework, 2022
28. SecretFlow . A unified framework for privacy-preserving data analysis and machine learning, 2022
29. NTL . A library for doing number theory, 2022
30. CryptoTools . A collection of tools for building cryptographic protocols, 2022
31. Ding B, König A C. Fast set intersection in memory. Proceedings of the VLDB Endowment, 2011, 4(4): 255–266
32. Zhou J, Bao Z, Wang W, Ling T W, Chen Z, Lin X, Guo J. Fast SLCA and ELCA computation for XML keyword queries based on set intersection. In: Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE). 2012, 905–916
33. Eppstein D, Goodrich M T, Mitzenmacher M, Torres M R. 2-3 cuckoo filters for faster triangle listing and set intersection. In: Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS). 2017, 247–260
34. Zhang J, Lu Y, Spampinato D G, Franchetti F. FESIA: A fast and simd-efficient set intersection approach on modern cpus. In: Proceedings of the 36th IEEE International Conference on Data Engineering (ICDE). 2020, 1465–1476
35. Inoue H, Ohara M, Taura K. Faster set intersection with SIMD instructions by reducing branch mispredictions. Proceedings of the VLDB Endowment, 2014, 8(3): 293–304
36. Pagh R, Stöckel M, Woodruff D P. Is min-wise hashing optimal for summarizing set intersection? In: Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS). 2014, 109–120
37. Blanusa J, Stoica R, Ienne P, Atasu K. Many-core clique enumeration with fast set intersections. Proceedings of the VLDB Endowment, 2020, 13(11): 2676–2690
38. Inan A, Kantarcioglu M, Ghinita G, Bertino E. Private record matching using differential privacy. In: Proceedings of the 13th International Conference on Extending Database Technology (EDBT). 2010, 123–134
39. He X, Machanavajjhala A, Flynn C J, Srivastava D. Composing differential privacy and secure computation: A case study on scaling private record linkage. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS). 2017, 1389–1406
40. Yang Y, Dong X, Cao Z, Shen J, Li R, Yang Y, Dou S. EMPSI: efficient multiparty private set intersection (with cardinality). Frontiers of Computer Science, 2024, 18(1): 181804
41. Freedman M J, Nissim K, Pinkas B. Efficient private matching and set intersection. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). 2004, 1–19
42. Kales D, Rechberger C, Schneider T, Senker M, Weinert C. Mobile private contact discovery at scale. In: Proceedings of the 28th USENIX Security Symposium. 2019, 1447–1464