Private Computation on Common Fuzzy Records

Kyoohyung Han
Samsung SDS
kh89.han@samsung.com

Seongkwang Kim
Samsung SDS
sk39.kim@samsung.com

Yongha Son*
Sungshin Women's University
yongha.son@sungshin.ac.kr

Abstract

Private computation on common records refers to analyze data from two databases containing shared records without revealing personal information. As a basic requirement for private computation, the databases involved essentially need to be aligned by a common identification system. However, it is hard to expect such common identifiers in real world scenario. For this reason, multiple quasi-identifiers can be used to identify common records. As some quasi-identifiers might be missing or have typos, it is important to support fuzzy records setting. Identifying common records using quasi-identifiers requires manipulation of highly sensitive information, which could be privacy concerns.

This work studies the problem of enabling such data analysis on the fuzzy records of quasi-identifiers. To this end, we propose *ordered threshold-one (OTO)* matching which can be efficiently realized by circuit-based private set intersection (CPSI) protocols and some multiparty computation (MPC) techniques. Furthermore, we introduce some generic encoding techniques from traditional matching rules to the OTO matching. Finally, we achieve a secure efficient private computation protocol which supports various matching rules which have already been widely used.

We also demonstrate the superiority of our proposal with experimental validation. First, we empirically check that our encoding to OTO matching does not affect accuracy a lot for the benchmark datasets found in the fuzzy record matching literature. Second, we implement our protocol and achieve significantly faster performance at the cost of communication overhead compared to previous privacy-preserving record linkage (PPRL) protocols. In the case of 100K records for each dataset, our work shows 147.58MB communication cost, 10.71s setup time, and 1.97s online time, which is 7.78 times faster compared to the previous work (50.12 times faster when considering online time only).

Keywords

secure multiparty computation, privacy-preserving record linkage, private set intersection

1 Introduction

Collaborative data analysis has received increasing interest across various fields, where multiple organizations aim to derive valuable insights and facilitate decision-making by combining their respective databases. Examples include research on rare diseases [27] and investigations into financial crimes [34]. Through the private computation of common records, parties can analyze data corresponding to shared records across databases without disclosing sensitive information. For instance, the effectiveness of online advertisements can be estimated through a collaborative analysis of ad click data and purchase records.

As a prerequisite for such collaboration, the databases involved must be aligned using a common identification system. It might be possible to utilize some universal identifier, such as the social security number (SSN), but it is unrealistic to expect that every database is linked to such a universal identifier. Moreover, there are privacy concerns associated with using universal identifiers. For example, the Health Insurance Portability and Accountability Act (HIPAA) in the U.S. restricts healthcare organizations from disclosing SSNs without the patient's consent [1]. Complicating matters further, many countries lack such permanent personal identifiers [42].

In order to facilitate collaborative data analysis in the absence of a common identification system, it is natural to consider the use of quasi-identifiers (or personally identifiable information), such as names, birth dates, and addresses. However, these records are inherently unstable; they are more susceptible to typos or missing values, and different individuals may share identical field values (e.g., names), while the same person might have different values for certain fields (e.g., addresses, phone numbers). This situation necessitates a framework that can privately determine whether records match, thereby enabling data analysis.

1.1 Private Matching and Analysis

Many studies have focused on privacy-preserving collaborative data analysis in many names. Although the studies equally pursued a join of two datasets in a privacy-preserving way, the objectives of each protocols are subtly different. We explain the problem setting and differences of the related concepts.

Problem Setting. Two parties, which will be called the receiver $\mathcal{R}$ and the sender $\mathcal{S}$ respectively, want to join their dataset X of $\mathcal{R}$ and Y of $\mathcal{S}$ in a certain way. Each dataset is a form of matrix whose rows corresponds to *records* and columns corresponds *quasi-identifiers*. We will call X and Y *record tables*. Here, quasi-identifier is a characteristic of a record which may collide with other record (e.g., date of birth, name, zip code). The quasi-identifiers in each record table is assumed to have possibly blanks or typos. On the other hand, *identifier* is the opposite concept which is linked uniquely with a record (e.g., social security number). We assume that each record table does not include any identifier.

Privacy-preserving Record Linkage. Privacy-preserving record linkage (PPRL) targets to output links of each matched records. They output a function between indices of the two record tables which links the matched records. A notable method of encoding in this

*This work was done while Y. Son was at Samsung SDS.

Proceedings on Privacy Enhancing Technologies 2025(1), 567–583

https://doi.org/10.56553/popets-2025-0031

realm involves the use of Bloom filters (applied to substrings of features) [39, 43, 44], which transform similar records into strings that are close in terms of Hamming distance. Despite this, numerous solutions have naively argued that such encodings can hide the original input records without providing a solid theoretical security framework (e.g., simulation-based security proofs). This oversight has indeed exposed actual vulnerabilities in these solutions [14, 15, 33]. While selected few studies have presented robust security arguments [13, 45], these approaches rely on generic multi-party computation (MPC) for pairwise similarity assessments, thereby unable to circumvent the intrinsic quadratic complexity associated with such computations.

Private ID. Some protocols do not stop at just giving a link between matched records, but rather give a random ID for each record, where the IDs for matched records are same. In this line of research, Private Join and Compute (PJC) developed by Google [23, 29], have facilitated analytics on matching identifiers with practical applications like ad conversion tracking. Similarly, Private Matching for Compute (PMC) from Meta [9] proposes alternative approaches for enabling private computations. Notably, it introduces the concept of private secret shared set intersection (PS^3I), which generates secret shares of data associated with matching identifiers. These shares can then be integrated into one's preferred generic multiparty computation (MPC) framework to carry out any desired computation.

As follow-up efforts to PMC [9], which concentrated on private computation over a single identifier, Buddhavarapu et al. [8] have broadened the scope to encompass multiple identifiers, an approach we refer to as multi-key PMC (MK-PMC). Mouris et al. [32] provide an delegated version of MK-PMC, which allows small parties to delegate the matching protocol.

Private SQL. Mohassel, Rindal, and Rosulek proposed SQL-join operations within a secret-shared state [30]. This work operates under the assumption of an (honest-majority) three-party setting, which is different from the two-party setting. While this presents a relevant context for comparison, our focus remains on the challenges and solutions unique to two-party collaborative environments.

1.2 Our Contribution

This work studies the problem of enabling collaborative data analysis on the fuzzy records of quasi-identifiers, with sub-contributions presented below.

Private Fuzzy Left Join. Combining datasets usually does not end with the join itself, but often requires additional analysis afterward. We focus on a two-party protocol that one party has a payload to share for analysis afterward. So, we assume the sender $\mathcal{S}$ has a additional set of payloads, where each payload is linked with a record. Informally speaking, the objective of our protocol is to secret-share the payloads.

We define a *private fuzzy left join* (PFLJ) as a two-party protocol that computes the join table in a *secret-shared* state, effectively preventing any unwanted information leakage. This includes safeguarding against the disclosure of which records match or do not match; furthermore, even neither party is able to determine the presence of any specific quasi-identifier in the other party's table.

Figure 1 illustrates the PFLJ protocol involving two databases. The first matrix on the upper left represents record table X of $\mathcal{R}$, and the second matrix on the upper right represents record table Y of $\mathcal{S}$. Both record tables arrange columns by quasi-identifiers (Name, DOB, and Zipcode), while Y has additional column of Payload. The protocol identifies similar entries between the two databases, as indicated by bidirectional arrows connecting matching rows.

The lower part of the figure shows the output after performing a join operation on the two databases. The main output matrix on the lower center aligns the index same as X, and it includes the membership information – whether a record in X is linked with a record in Y – and the payload from Y. This output is then split into two secret shares, as shown in the matrices to the left and right of the main output. Each share contains partial information about the membership and payload values, which ensures that sensitive data is protected and can only be fully reconstructed by combining the two shares.

Efficient Protocol for PFLJ. To efficiently realize the PFLJ, we tried to leverage the recent advances circuit-based private set intersection (CPSI) protocols. CPSI with associated payloads [36] offers similar functionality which joins two sets of "identifiers", and outputs the share of membership value and corresponding payload. As this technique does not fit well with databases of quasi-identifiers, we introduce some encoding techniques from quasi-identifiers to highly-distinct columns, dubbed *feature*. The encoding techniques cover wide range of traditional matching rules commonly employed in the literature. Then, we introduce *ordered threshold-one (OTO) matcher*, which judges two records as matched if they share at least one matching feature.

On top of OTO matcher, we construct a secure and efficient protocol for PFLJ, where two records are identified if they share at least one matching feature. Informally speaking, our construction run a CPSI with associated payload for each features, align the outputs of CPSI by permute-and-share protocol (PnS), then share the payload by a generic MPC protocol. As a result, our protocol exhibits a complexity of $O(nN \log N)$, where n denotes the number of encoded features, and N denotes the number of records of each party. While PnS is the only subroutine of quasi-linear complexity with respect to N, PnS does not dominate the total time complexity of the PFLJ protocol in the plausible amount of record ($\sim 10^6$) so that our protocol can be regarded as *de facto* linear complexity. This represents a significant improvement over previous works, such as [8], which required $O(nN)$ public key operations. In contrast, our approach necessitates a constant number of public key operations while also employing $O(nN \log N)$ symmetric key operations, thereby markedly accelerating the protocol's speed. Furthermore, we prove the semi-honest security of our protocol within the two-party computation (2PC) model.

Implementation Results. We validate that our encoding techniques toward OTO matcher, using several benchmark datasets: the DBLP-ACM dataset [2], the European census dataset [3] and the NCVR dataset [4], which have been previously utilized in studies on fuzzy matching and record linkage. The traditional matching rule achieves F1 scores of up to 0.977, 0.985 and 0.979 for each datasets, and OTO matcher with our encoding achieves up to 0.947, 0.965 and 0.976 respectively.

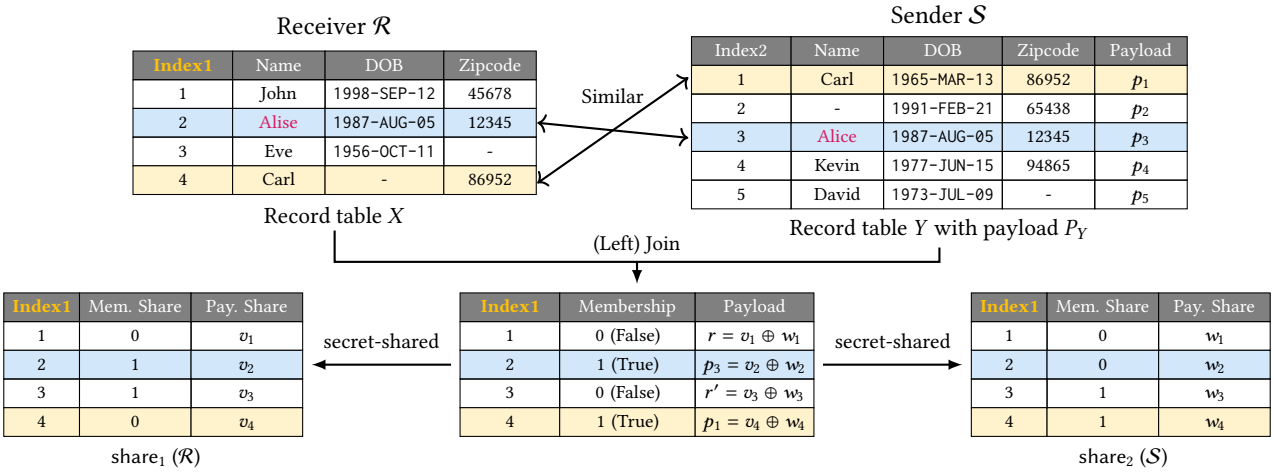


Figure 1: Description of our protocol. Each database may include blanks (denoted by -), or typos. Each party outputs the list of shares of memberships and associated payloads.

In terms of performance, our protocol demonstrates significant improvement in running time, requiring only 12.58 seconds (1.97 seconds online time) over a LAN network to process a dataset of 100K records, with total communication overhead of 147.58MB. This represents a 7.78 times speed increase over previous work [8] (50.12 times increase if we only consider online time), albeit at the cost of a threefold increase in communication volume. Moreover, our protocol's scalability is confirmed by experiments with varying input sizes and under different network conditions, showing an almost linear relationship between running time and input size. For a comprehensive review of these experimental results, we direct readers to Section 6.

1.3 Comparison with PMC

Multi-key Matching for Compute (MK-PMC) [8, 32] is similar concepts with our PFLJ protocol. These papers discussed similar applications to ours, so we especially provide some comparison with the works here.

Although the both two papers considered the OTO matcher (called *multi-key matching* in those papers), they lack a connection to real-world fuzzy matching. Our paper shows a connection from conventional fuzzy matching techniques to the OTO matcher, utilizing methods such as concatenation and locality-sensitive hashing.

The timing efficiency of our protocol executing the OTO matcher surpasses the protocol in those papers. The disparity between the performances stems from their reliance on the specific characteristics of public key operations based on Diffie-Hellman, which are inherently computation-intensive. In contrast, our approach leverages recent advancements in PSI-related techniques, which heavily uses symmetric key operations rather than public key operations.

While the protocol in [32] outputs a similar format with the PFLJ protocol (shares of payloads), the protocol in [8] only outputs private IDs in plain. The distribution of shares allows further private computation over the payload. If some users want to invoke a private computation after the protocol in [8], they need to secret-share their data.

The protocol in [32] has additional private *delegation* functionality, which allows low-computational clients to delegate computational burden to some server (called helper). In contrast, our research is specifically tailored to the collaboration between two parties, and is not readily adaptable to scenarios involving delegation. Although the delegation scenario may be desirable for some scenario, two-party protocols still have useful applications in the real world such as a cross-border collaboration, or the case where two large companies with substantial resources and data want to collaborate. Therefore, we believe our protocol with much faster running time still has some advantages over the delegation protocol of [32].

1.4 Applications

Our PFLJ protocol outputs two lists of membership shares and payload shares from two record tables (see Figure 1). By manipulating the outputted lists, the PFLJ protocol can be transformed into related concepts introduced in Section 1.1. We explain how the PFLJ protocol can be extended to other protocols.

Privacy-Preserving Record Linkage. The goal of privacy-preserving record linkage (PPRL) protocols is to output links between records which are identified as a same entity. The link might can be form of index pairs, where one of the index is from X and the other is from Y (e.g., $(4, 1)$ in Figure 1). To achieve these pairs, two parties ($\mathcal{R}$ and $\mathcal{S}$) needs to invoke the PFLJ protocol with payload to be the exact index of the record (e.g., $p_j = j$ in Figure 1). After the protocol, $\mathcal{S}$ sends its payload share to $\mathcal{R}$. Then, $\mathcal{R}$ can link the payload – index of Y – to corresponding index in X. Communication overhead is $N \log N$ bits where N is the number of records for each record table. The overhead is negligible compared to the PFLJ protocol.

Many previous PPRL solutions involving Bloom filter are found out to either be vulnerable [14, 15, 33] or lack practical efficiency [13, 28, 45] because of their inherent $O(N^2)$ time complexity. As the latter ones usually are combined with generic multiparty computation (MPC) protocols such as Yao's garbled circuit, they lead to

poor performance that is quite far from practice (e.g., a few hours for matching 10K data in LAN environment).

One comparable previous work is the study of Adir et al. [6], which achieved linear complexity by combining a private set intersection (PSI) with a locality-sensitive hashing (LSH) encoding. It is hard to fairly compare efficiency between our work and this solution as the Adir et al. did not publicly open the source codes. Our PFLJ protocol seems to provide a more efficient PPRL solution as [6] requires 273s time and 1.44 GB communication for 2^{16} records in the LAN setting. Furthermore, although PSI hides the information of non-matching records, this protocol still leaks some unwanted information; the PSI receiver knows the exact intersected LSH evaluations for each record, which indicates the similarity between records from the number of intersections.

Private-ID. The goal of private-ID (PID) protocols is to build common identifiers for each record, where the identifier does not include any private information (usually randomly sampled) and agree on the matched records. If two parties (receiver $\mathcal{R}$ and sender $\mathcal{S}$) invoke the PFLJ protocol with randomly sampled payload (i.e., p_i's in Figure 1 are randomly sampled), $\mathcal{R}$ and $\mathcal{S}$ shares the random payloads for common records at the end of the protocol. After the protocol, $\mathcal{S}$ sends the payload shares (w_i's) so that $\mathcal{R}$ recovers the payload corresponding to the same records (e.g., p_1 and p_3 in Figure 1). Both parties can agree on common identifiers by setting the payloads as the identifiers; $\mathcal{R}$ gets random ID (r, p_3, r', p_1) without knowing what is matched.

Performance overhead from the transformation is negligible; $\mathcal{S}$ needs to additionally send all the payload shares of $\mathcal{S}$. If the probability of ID collision is required to be less than 2^{-40}, then communication cost of $(40+2\log N)\cdot N$ bits will be added where N is the number of records in each record table. This communication overhead is less than 1% of the communication cost of the PFLJ protocol, and so we compare communication and computation cost with a PID protocol [8] in Section 6 without the additional communication overhead.

2 Backgrounds

2.1 Notations

For an integer n, we denote the set $\{1, \ldots, n\}$ by $[n]$. An n-dimensional vector $v = (v_1, \ldots, v_n)$ is succinctly represented by $v = (v_i)_{i\in[n]}$. For a matrix X of size $N \times n$, the i-th row and k-th column are denoted by X_i and $X[k]$, respectively. The matrix X is often regarded as a length-N vector of rows, thus we express $X = (X_i)_{i\in[N]}$. For a condition con, function $\mathbf{1}(\mathsf{con})$ outputs 1 if the corresponding input satisfies con, or 0 otherwise. We denote the statistical security parameter by λ, and the computational security parameter by κ. For an element x and a vector v, if $x = v[i]$ for some i, then we abuse the notation $x \in v$ for convenience.

2.2 Oblivious Transfer

The oblivious transfer (OT) functionality involves two parties: a sender, who inputs two messages m_0 and m_1, and a receiver, who inputs a choice bit $b \in \{0, 1\}$. The objective is for the receiver to obtain the message m_b corresponding to their choice bit, without revealing any additional information to either party. Specifically, the sender does not learn b, and the receiver does not learn the alternative message m_{1-b}.

We particularly consider two specialized forms of OT:

- A *correlated* OT (COT_ℓ) wherein the sender inputs a single correlation $\delta \in \{0,1\}^\ell$ to establish $m_0 = r$ and $m_1 = r \oplus \delta$ for a randomly chosen $r \in \{0,1\}^\ell$.
- A *random* OT (ROT_ℓ) in which the sender does not provide any input but receives two random messages m_0 and m_1, each of length ℓ, as outputs.

To avoid any confusion, we refer to the original OT, where the sender inputs two messages m_0 and m_1, as *standard* OT (SOT_ℓ).

The subprotocols employed in this work necessitate a substantial number of OT calls, prompting the consideration of the *OT extension* framework [24]. This framework starts with a small (polynomial in κ) number of *base* OTs and extends them to a large number of ROT_κ instances. According to the state-of-the-art OT extension protocols [18, 47], the communication overhead is remarkably low, for instance, less than 0.1 bit per ROT.

2.3 Circuit-based PSI

Circuit-based private set intersection (CPSI) is a 2-party protocol which computes intersection information in a secret-shared manner [10, 16, 25, 36, 37, 41]. In the protocol, two parties, a receiver and a sender, who own input sets X^* and Y^* respectively,[1] computes Boolean shares of $\mathbf{1}(x \in X^* \cap Y^*)$ for each $x \in X^*$. In this work, we further consider a useful variant [36] of this concept, which not only computes the intersection information $\mathbf{1}(x \in X^* \cap Y^*)$ but also generates secret shares of the associated payloads related to Y^*, denoted as $P_{Y^*} = \{p_y : y \in Y^*\}$.

Although it seems natural that the functionality retains its alignment of the output secret shares same with the input set X^*, recent CPSI constructions [36, 37, 41] exhibit an output ordering that may appear random. This characteristic stems from the use of the cuckoo hashing technique [7], which is essential for achieving linear complexity within the protocol. As a result, the ideal functionality of CPSI includes an output alignment represented by the index map $\mathsf{idx} : X^* \to [M]$ for some $M \geq |X^*|$, where M is a bit larger than $|X^*|$ (e.g., $M = 1.3|X^*|$) to ensure that the cuckoo hashing works well. We describe the ideal functionality of CPSI including the random alignment in Figure 2. It is important to note that the index map idx is randomly determined during the protocol execution, ensuring that neither party can influence its outcome. Moreover, only $\mathcal{R}$ gains knowledge of the index map $\mathsf{idx} : X^* \to [M]$.

While the inclusion of the index map in the CPSI protocol might seem to add unnecessary complexity, it is important to recognize its essential role in the technical underpinnings of our main protocol construction, as detailed in Section 5.

2.4 Permute-and-Share

Permute-and-Share (PnS) is a functionality that obliviously shuffles the input, as defined in Figure 3. Given a permutation $\pi : [N] \to [N]$ and a vector x of length N, this functionality outputs to each party additive shares of $\pi(x)$, where $\pi(x) := (x_{\pi(1)}, \ldots, x_{\pi(N)})$.

[1] We additionally mark asterisks for readers not to confuse with record tables X and Y which appear throughout this paper.

Parameters: Set sizes $|X^*|$ and $|Y^*|$, output size $M \geq |X^*|$, and bit-length ℓ of associated payloads.
Input: Set X^* from one party $\mathcal{R}$, and set Y^* along with the associated payload $P_{Y^*} = \{p_y \in \{0,1\}^\ell : y \in Y^*\}$ from the other party $\mathcal{S}$.
Functionality: The functionality samples a random injective function $\mathsf{idx} : X^* \to [M]$, and define $b \in \{0,1\}^M$ and $v \in (\{0,1\}^\ell)^M$ so that

$$b_{\mathsf{idx}(x)} = 1, v_{\mathsf{idx}(x)} = p_x \quad \text{if } x \in X^* \cap Y^*$$
$$b_i = 0, v_i \leftarrow_\$ \{0,1\}^\ell \quad \text{otherwise.}$$

It then sends the index map idx to $\mathcal{R}$, and the secret shares of b and w to each party.

Figure 2: Ideal functionality $\mathcal{F}_{\mathsf{cpsi}}$ of circuit-PSI (with associated payload).

Several protocols [12, 31] have been proposed to realize this functionality. Notably, the protocol from [12] exhibits advantages over [31] for input vector components of bit-length ℓ that is significantly larger than the computational security parameter κ. However, in this paper, we primarily focus on the protocol from [31] as it suffices for our requirements where $\ell \leq \kappa$. This protocol necessitates $N \log N$ instances of $\mathsf{COT}_{2\ell}$, resulting in a communication cost of $2\ell N \log N$ bits.

Parameters: A permutation target size N and an item length ℓ.
Input: A sender with a permutation π over $[N]$, and a receiver with input vector $x \in (\{0,1\}^\ell)^N$.
Functionality: The functionality samples a random vector $r \in (\{0,1\}^\ell)^N$, and sends r to the sender and $\pi(x) \oplus r$ to the receiver.

Figure 3: Ideal functionality $\mathcal{F}_{\mathsf{pns}}$ of Permute-and-Share.

3 Private Fuzzy Left Join

This section provides a rigorous definition of the concepts depicted in Figure 1, alongside the related assumptions and threat models.

Terminologies. Throughout this paper, *a record* having n quasi-identifiers is represented by a vector of n (bit-)strings. Correspondingly, *a record table* containing N records, each with n quasi-identifiers X_i for $i \in [N]$, is represented by a matrix $X = (X_i)_{i\in[N]}$ of dimensions $N \times n$.

3.1 Fuzzy Matching and Unique Matcher

As a formal argument concerning *similar* records, we define a *(fuzzy-)matching rule* R as a binary function $R : (x, y) \mapsto \{0, 1\}$, where $R(x, y) = 1$ indicates that x and y are judged to be identical entities. A typical example involves the use of a proper similarity measure $S : (x, y) \mapsto [0, 1]$ and a threshold $t > 0$ to define $R(x, y) := \mathbf{1}(S(x, y) \geq t)$.

As our objective is to fuzzy-match two record tables X and Y, there could be a situation where a single record X_i corresponds to multiple records Y_j within the other table, i.e., $R(X_i, Y_j) = 1$ for many $j \in [N_Y]$. In such instances, the standard SQL left join operation indiscriminately incorporates every matching pair into the resultant table. This approach, however, leads to the repeated occurrence of certain indexes in the output table, inadvertently revealing the number of matched indices for a specific record X_i, thereby compromising privacy.

Thus, to prevent information leakage regarding the count of matched records, it is desirable that the output maintains a predetermined size. This approach necessitates the selection of at most one corresponding record in Y for each record X_i, ensuring that the actual number of matches remains undisclosed. To formalize this criterion for selecting a unique matching record, we introduce the following concept:

Definition 1. *A unique matcher is a probabilistic algorithm that takes as input a record X_i and a target record table $Y = (Y_j)_{j\in[N_Y]}$. It returns either an index $j \in [N_Y]$ of the record Y_j that is most likely to be matched with X_i, or $\perp$ to indicate that there is no matching record for X_i in Y. This algorithm is denoted by* matcher*, and the output of* matcher *on input (X_i, Y) is concisely expressed as* $\mathsf{matcher}(X_i, Y) \in [N_Y] \cup \{\perp\}$.

In practice, unique matchers are run with a public seed unless they are deterministic; for the sender and the receiver to securely realize a unique matcher, the seed should be shared between the two parties. For convenience, we omit the random seed unless it is not confusing.

A matching rule R can be transformed into a unique matcher by arbitrarily selecting one index among the j's satisfying $R(X_i, Y_j) = 1$. If the matching rule R is based on some similarity measure S, the transformed unique matcher identifies the index j yielding the maximal similarity $S(X_i, Y_j)$ where one index among such js is randomly selected in case of a tie.

3.2 Private Fuzzy Left Join

Consider two parties $\mathcal{R}$ and $\mathcal{S}$ possessing record tables $X = (X_i)_{i\in[N_X]}$ and $Y = (Y_j)_{j\in[N_Y]}$ respectively, where $\mathcal{S}$ additionally has a set of associated payloads $P_Y = \{p_j \in \{0,1\}^\ell : j \in [N_Y]\}$. Assuming the presence of a unique matcher, denoted by matcher, we define the membership vector $b = (b_i)_{i\in[N_X]} \in \{0,1\}^{N_X}$ and the payload vector $v = (v_i)_{i\in[N_X]} \in (\{0,1\}^\ell)^{N_X}$ as follows:

$$b_i = 1, \quad v_i = p_j \quad \text{if } j = \mathsf{matcher}(X_i, Y) \neq \perp,$$
$$b_i = 0, \quad v_i \leftarrow_\$ \{0,1\}^\ell \quad \text{otherwise.}$$

Then, *private fuzzy left join* (PFLJ) refers to a two-party interactive protocol enabling the parties to obtain secret shares of two vectors $b \in \{0,1\}^{N_X}$ and $v \in (\{0,1\}^\ell)^{N_X}$. We consider the following basic assumptions and threat model for PFLJ:

- The both parties are assumed to be connected by a secure channel for interaction, which allows us to focus solely on threats where one party attempts to extract additional information about the other party's input.
- Regarding adversarial behavior, we adopt the *semi-honest model*, wherein an adversarial party follows the protocol specifications honestly but seeks to derive information beyond the permitted output.

- Finally, we assume that the both parties have previously agreed upon a unique matcher. Consequently, the process of privately training and developing matcher (or the matching rule R) falls outside the scope of this paper.

3.3 Overview of Our Approach

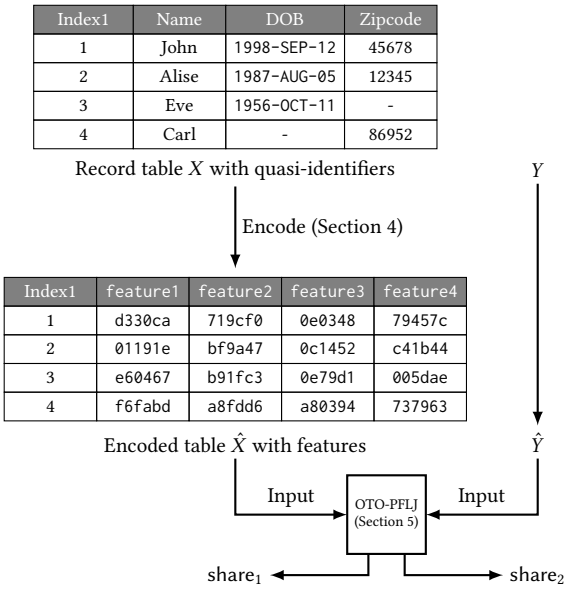


Figure 4: Our Approach for Private Fuzzy Left Join.

Figure 4 illustrates our approach for performing a Private Fuzzy Left Join (PFLJ) protocol. The process begins with two record tables, X and Y, containing quasi-identifiers such as names, dates of birth, and zip codes. These tables are depicted at the top of the figure, with X on the left and Y on the right. Each record in these tables is encoded into a set of features, creating the encoded tables $\hat{X}$ and $\hat{Y}$, as shown in the second row. This encoding process, whose details will be presented in Section 4, transforms the original data into a format that is suitable for a special matcher what we call the ordered threshold-one (OTO) matcher. The encoded tables $\hat{X}$ and $\hat{Y}$ are then input into the OTO-PFLJ protocol, indicated in the central block of the figure (Section 5).

The Ordered-Threshold-One Matcher. The ordered threshold-one (OTO) matcher determines if a record matches by evaluating the membership information of each feature column. To substantiate the utility of the OTO matcher, we discuss how widely-used matchers in the literature can be transformed into the OTO matcher with appropriate encoding techniques: Similarity-based matchers can be transformed using a suitable Locality-Sensitive Hashing (LSH) family that aligns with the specific similarity (or distance) metric, and equality-based matchers such as scoring-based approach [20] can be efficiently encoded via concatenation. The details can be found in Section 4.

Efficient PFLJ for OTO. In Section 5, we propose a highly efficient PFLJ protocol that is specifically designed for the OTO matcher. This protocol inherently requires multiple executions of the CPSI protocol across each feature column, given that the OTO matcher can be represented by column-wise membership information. However, we emphasize that our protocol is way more than a mere aggregation of CPSI applications, primarily due to alignment complications introduced by the index map concept inherent in CPSI. To address this alignment issue, we employ permute-and-share (PnS) protocols, thereby unifying the alignment of CPSI outputs. Subsequently, we obtain secret-shared vectors, each corresponding to the membership (and the associated payload) information for every feature column. The culmination of our protocol involves the secure amalgamation of these vectors into a single vector of shares. This process involves secure evaluations of MUX gates, which are efficiently executed utilizing OT.

4 Ordered Threshold-One Matcher

This section proposes a unique matcher which is called ordered threshold-one (OTO) matcher, and the generic data encoding methods that approximate the traditional fuzzy-matcher on the original data into OTO matcher on the encoded data. Figure 5 presents an overview of the encoding process leading to the OTO matcher, which will be elaborated in the rest of this section.

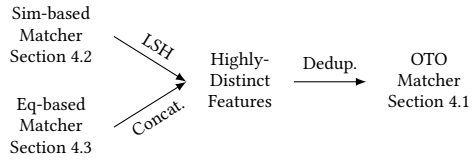


Figure 5: Encodings from traditional rules to OTO matcher.

4.1 Ordered Threshold-One Matcher

Let us consider a simple matching rule that determines two records, X_i and Y_j, to refer to the same entity if and only if there exists at least one identical feature between them. That is, we consider the matching rule $R : (X_i, Y_j) \mapsto 1$ if and only if $X_i[k] = Y_j[k]$ for some k. This straightforward approach, however, may lead to multiple candidate records Y_j for a single record X_i, necessitating a method to select the most appropriate candidate.

To address this, we introduce the *ordered-threshold-one* (OTO) unique matcher, which selects the candidate Y_j associated with the smallest matching feature k. In scenarios where multiple candidates Y_j share the smallest matching feature k, a random candidate among these is chosen. The formal definition of the OTO matcher is presented below:

Definition 2. *Given two tables X and Y of sizes $N_X \times n$ and $N_Y \times n$ respectively, the ordered threshold-one (OTO) matcher* $\mathsf{matcher}_{\mathrm{OTO}} : [N_X] \to [N_Y] \cup \{\perp\}$ *is defined as follows:*

- *For each record X_i, let $C_{i,k} = \{j \in [N_Y] : X_i[k] = Y_j[k]\}$ and $C_i = \bigcup_k C_{i,k}$.*
- *If $C_i = \emptyset$, then* $\mathsf{matcher}_{\mathrm{OTO}}(i) := \perp$.
- *Otherwise,* $\mathsf{matcher}_{\mathrm{OTO}}(i) := j \xleftarrow{\$} C_{i,k}$ *for the smallest k such that $C_{i,k} \neq \emptyset$.*

Equivalence to Membership-based Matcher. Given that the OTO matcher evaluates pairs of records, a direct application of this approach for every potential pair would inevitably lead to a quadratic number of comparisons. To circumvent this computational inefficiency, we introduce an observation that converts pairwise comparisons into feature-wise membership tests. The essence of this strategy lies in the realization where the feature-wise membership tests can be securely conducted using the CPSI protocol. Lemma 1 shows how the OTO matcher can be succinctly represented through membership information.

Lemma 1. *For given two deduplicated tables X and Y with number of rows N_X and N_Y, the membership matcher* $\mathsf{matcher}_{\mathrm{mem}} : [N_X] \to [N_Y] \cup \{\perp\}$ *can be defined as follows:*

- *For each record X_i, let $M_i := \{k : X_i[k] \in Y[k]\}$.*
- *If $M_i = \emptyset$,* $\mathsf{matcher}_{\mathrm{mem}}(i) := \perp$.
- *Else,* $\mathsf{matcher}_{\mathrm{mem}}(i) := j \in [N_Y]$ *by the smallest index j in M_i.*

If there is no duplicate entry in each record table (i.e., $X_i[k] \neq X_j[k]$, $Y_i[k] \neq Y_j[k]$ for any $i \neq j$), then it holds that for all $i \in [N_X]$, $\mathsf{matcher}_{\mathrm{mem}}(i) = \mathsf{matcher}_{\mathrm{OTO}}(i)$.

Proof. Suppose $\mathsf{matcher}_{\mathrm{OTO}}(\alpha) = \beta$, it means that $X_\alpha[k] = Y_\beta[k]$ for some k. Because k is the smallest index,there is no $(i, j) \in [N_X] \times [N_Y]$ that $X_i[k'] = Y_j[k']$ for $k' < k$. This means that the membership results $M_1, M_2, \ldots, M_{k-1}$ should be zero for $i = \alpha$, and $M_k = 1$ as $X_\alpha[k] = Y_\beta[k]$. If there is no duplication, β is the only one matching. So, $\mathsf{matcher}_{\mathrm{mem}}(\alpha)$ should be β. For the opposite side, we suppose $\mathsf{matcher}_{\mathrm{mem}}(\alpha) = \beta$. This means that $X_\alpha[k] = Y_\beta[k]$ for some k, and there is no $(i, j) \in [N_X] \times [N_Y]$ that $X_i[k'] = Y_j[k']$ for $k' < k$, which implies $\mathsf{matcher}_{\mathrm{OTO}}(\alpha) = \beta$. □

On Feature Duplication. In order to utilize Lemma 1 that efficiently computes OTO matcher, we need to ensure that there is no record having duplicated features. However, it is quite implausible assumption for usual quasi-identifiers. To remove duplication, we use a two-step solution: encoding and deduplication. The encoding step, which will be explained in following subsections, combines multiple quasi-identifiers to a highly distinctive feature. As the encoded features are highly distinctive, the duplication rates are expected to quite small, but not zero. Then, the deduplication step artificially remove the duplication of the encoded features so that there is no remaining duplicate feature. The detailed process of the deduplication step is described in Algorithm 4 in Section F.

This deduplication could harm the accuracy of fuzzy-matching, especially when there is extensive duplication within some column. However, we argue that the deduplication step does not degrade the accuracy of the matching combined with our proposed the encoding step, because our encoding step already effectively reduces the number of duplication. For the justification, we provided the impact of deduplication on the accuracy through experimental results on benchmark datasets in Section 4.4.

4.2 Encoding from Similarity-Based Matcher

Two records which are in fact the same entity possibly have no exactly matched quasi-identifiers, due to some typos, blanks, or changes of quasi-identifiers (e.g., change of phone number). In this case, we measure the similarity of two strings, defined as a function $S : \mathcal{D} \times \mathcal{D} \to [0, 1]$ for a proper domain of input strings (or sets) $\mathcal{D}$.[2] Given a similarity measure S, a similarity-based rule decides whether two records are matched or not by checking that similarity of two data is above a threshold t. Formally, the similarity-based matching rule R with a similarity measure S can be written by

$$R_{\mathsf{sim}}(x, y) = \mathbf{1}(S(x, y) \geq t).$$

To make this rule into the unique matcher, we usually picks the candidate records among them whose similarity is of the largest value, and a random selection in the event of a tie.

In the following, we introduce some similarity measures for strings that we mainly focus on. For a string s, let $\mathsf{qgram}_q(s)$ be the set of every q-gram of s; every q-length substrings of s. For example, $\mathsf{qgram}_2(\text{"}hello\text{"}) = \{\text{"}he\text{"}, \text{"}el\text{"}, \text{"}ll\text{"}, \text{"}lo\text{"}\}$.

Example 1 (Jaccard similarity). *For given two sets X, Y, Jaccard similarity is defined by*

$$S_J(X, Y) := |X \cap Y|/|X \cup Y|.$$

Jaccard similarity for two strings x and y is defined with a parameter q by

$$S^{\mathsf{str}}_{J,q}(x, y) = S_J(\mathsf{qgram}_q(x), \mathsf{qgram}_q(y)).$$

Example 2 (Cosine/Angular similarity). *For two vectors $x, y \in \mathbb{R}^n$ with non-negative components, cosine similarity is defined by*

$$S_C(x, y) := \langle x, y\rangle/(\|x\| \cdot \|y\|)$$

and angular similarity is defined by

$$S_A(x, y) := 1 - \arccos(S_C(x, y))/\pi.$$

Cosine (and angular) similarity for two strings x and y is defined with a vectorizer *map $V : \{0, 1\}^* \to \mathbb{R}^n$ that embeds an input string into a numerical vector, precisely*

$$S^{\mathsf{str}}_C(x, y) = S_C(V(x), V(y)),\ S^{\mathsf{str}}_A(x, y) = S_A(V(x), V(y)).$$

Locality Sensitive Hashing. The locality sensitive hashing (LSH) family for similarity measure S is defined as follows [11].

Definition 3 (LSH family for a similarity). *Let S be a similarity measure defined on $\mathcal{D}$. A function family $\mathcal{H}$ is LSH of S if*

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] = S(x, y) \quad \text{for all } x, y \in \mathcal{D}.$$

Our interest Jaccard similarity and the angular similarity have the following LSH families.

Example 3 (LSH for Jaccard similarity). *For a family of random function $\mathcal{H}$, define a function* $\mathrm{MinHash}_h$ *for $h \in \mathcal{H}$ as*

$$\mathrm{MinHash}_h(X) = \min_{x \in X} h(x).$$

It holds that $\Pr[\mathrm{MinHash}_h(X) = \mathrm{MinHash}_h(Y)] = S_J(X, Y)$.

Example 4 (LSH for angular similarity). *For a vector $a \in \mathbb{R}^n$, define a function h_a by*

$$h_a(x) := \mathrm{sgn}(\langle a, x\rangle),$$

where sgn *is the sign function. It holds that* $\Pr_r[h_a(x) = h_a(y)] = S_A(x, y)$.

[2] In the literature, the term 'similarity' sometimes includes distance functions in metric spaces. However, we restrict the meaning of this term to have range $[0, 1]$.

It is common to use LSH with an AND-OR amplificiation technique. The AND amplification considers a r-tuple of LSH functions $h'(x) = (h_1(x), h_2(x), \ldots, h_r(x))$ for $h_i \in \mathcal{H}$, and then check whether *all* components of $h'(x)$ collide, whose probability is

$$\Pr[h'(x) = h'(y)] = p^r$$

where $p = S(x, y)$. Then OR amplification considers b independent functions h'_j of AND amplification with r LSHs, and checks whether *at least one* $h'_j(x)$ collide. The probability of such event is computed by

$$\Pr[h'_j(x) = h'_j(y) \text{ for at least one } j] = 1 - (1 - p^r)^b \qquad (1)$$

where $p = S(x, y)$. Since the hash value $h'_j(x)$ itself is too long, it is hashed by a collision-resistant hash function with smaller output size in practice.

Our Encoding Method. For a similarity-based matcher with similarity S having LSH family $\mathcal{H}$ and a matching threshold t, our encoding understands each record as a long string, and converts it using LSH with appropriate AND-OR amplification parameters r and b, whose concrete choices are discussed below. More precisely, our encoding maps a string record x into $\left(h'_1(x), \cdots, h'_b(x)\right)$ where each $h'_j(x)$ is an r-AND amplification of LSH function. Such encoded records are judged to be matched if there is at least one same hash value $h'_j(x)$.

Concrete Choices of AND-OR Parameters. As two strings x and y of similarity $p := S(x, y)$ are judged as a matched pair with probability $1-(1-p^r)^b$ (see (1)), we will observe the function $C_{r,b}(p) := 1-(1-p^r)^b$ over $p \in [0, 1]$, and call it by (r, b)-*curve*. Considering that the similarity-based matcher can be understood with the step function $\mathbf{1}(p \geq t)$, it is natural to focus on (r, b)-curves that *inflects* at t, precisely whose the second derivative at t is zero, i.e., $C''_{r,b}(t) = 0$. Figure 6 presents some examples of such (r, b)-curves inflects at $t = 0.7$.

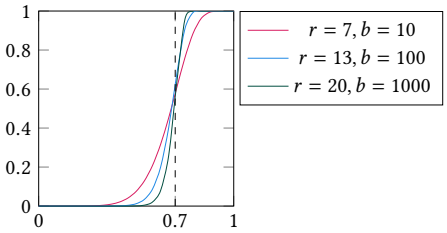


Figure 6: For a matching threshold $t = 0.7$, we take AND-OR parameters (r, b) for LSH so that the curve $1 - (1 - p^r)^b$ has the inflection point at $t = 0.7$.

Not surprisingly, appropriate choice of larger r and b provide steeper (r, b)-curve, and hence it would be better to take large r and b to make our LSH encoding close to the original similarity-based matcher. Meanwhile, the OR parameter b corresponds to the number of CPSI calls (see Section 5) which is paramount to the efficiency of our protocol. Considering this, we choose some (r, b) pairs for each t as follows.

(1) Pick (r, b) candidates such that $C''_{r,b}(t) = 0$[3] restricting the maximum b-value B.
(2) Among the candidates, choose (r, b) with the maximum derivative $C'_{r,b}(t)$, while expecting it makes the steepest curve.

Table 1 shows some example (r, b) parameters for each t, where different OR parameters b for the same threshold t provide a trade-off between the accuracy and running time. Note that those values are just examples, and any other (r, b) can be utilized as a fine-tuning. In particular, if the matching threshold t is quite large, for example $t = 0.9$, very small parameters such as $(r, b) = (12, 3)$ might be fine because it provides a sufficiently steep curve; see the $t = 0.9$ and $(12, 3)$ row for NCVR dataset in Table 2.

t	B		
	30	50	100
0.5	(5, 28)	(6, 50)	(7, 100)
0.6	(7, 30)	(8, 50)	(9, 99)
0.7	(10, 30)	(11, 50)	(13, 100)
0.8	(16, 30)	(18, 50)	(22, 100)
0.9	(35, 30)	(40, 50)	(46, 100)

Table 1: LSH AND-OR Parameters (r, b) for each threshold t

4.3 Encoding from Equality-Based Matcher

Equality-based rule determines whether records correspond to one another by checking the equality of each quasi-identifier. More precisely, this rule assigns a specific score value s_k for each k-th quasi-identifier, and then determines whether two records refer to the same entity by checking whether the sum of the scores of matching features exceeds some predetermined threshold t. Formally, assuming each record has m quasi-identifiers, the corresponding matching rule R of two records $x = \{x_k\}_{k\in[m]}$ and $y = \{y_k\}_{k\in[m]}$ is defined by

$$R_{\text{eq}}(x, y) = \mathbf{1}\left(\sum_{k\in[m]} \mathbf{1}(x_k = y_k) \cdot s_k \geq t\right).$$

To make this rule into a unique matcher, one can choose the record whose sum of the scores is maximal. In the event of a tie, a random selection is made from among the tied candidates.

This method might seem overly restrictive; it is incapable of accommodating even minor discrepancies within quasi-identifiers. It is found in the Fellegi-Sunter model [20], which has gained widespread acceptance in the field of record linkage for its robustness and effectiveness. Furthermore, this method often combined with a strategic reassemble of quasi-identifiers. Indeed, some record linkage or entity resolution in the real world [5, 19] are categorized by this equality-based matching with threshold 1, combined with adequate reassembling of quasi-identifiers (see Example 5).

Example 5 (SEER-Medicare[5]). *In this dataset published by National Cancer Institute, two cancer patient records are judged as the same person under several criteria. For example, one criterion checks whether the last 4 digits of SSN, first name (in Soundex), last*

[3] In fact, as r and b should be integers, $C''_{r,b}(t)$ scarcely becomes exactly 0; we allow (r, b) satisfying $C''_{r,b}(t) \approx 0$.

name (in Soundex), DOB-day and DOB-month are all same or not. The criterion is equivalent to check whether a single feature, which is a concatenation of last 4 digits of SSN, first name (in Soundex), last name (in Soundex), DOB-day, and DOB-month, is same or not.

Our Encoding Method. We encode an equality-based matcher to OTO matcher as follows. First, enumerate all possible combination of quasi-identifiers that makes $\sum_{k\in[m]} \mathbf{1}(x_k = y_k) \cdot s_k \geq t$. Then, for each combination of quasi-identifiers, define a feature by concatenating all quasi-identifiers. The ordering of derived features can be naturally assigned by the total score of the combination. One might think this encoding results in excessively large numbers of feature, since there could be total 2^m possible combinations for m quasi-identifiers. However, since quasi-identifiers that are less important are given low scores, or are rarely used in matching criteria, the number of quasi-identifier combinations that exceed the threshold is not as large as 2^m. See also the experiments with the real data in Section 4.4.

Example 6 (Eq-based to OTO matcher). *Assume that an equality-based matcher considers three quasi-identifiers f_1, f_2, f_3 assigned by score $1, 2, 4$, with the score threshold $t = 3$. Then the derived features for OTO matcher would be $[f_1\|f_2\|f_3]$, $[f_2\|f_3]$, $[f_1\|f_3]$, $[f_3]$, and $[f_1\|f_2]$, which are ordered by the total score in decreasing order.*

4.4 Experimental Result on Accuracy

Our encoding process from the traditional matchers (both equality based matchers and similarity-based matchers) incurs some accuracy changes, especially due to the probabilistic nature of LSH or the de-duplication step. In this section, we present experimental results that measure the effect of our encoding on the accuracy, using the following benchmark datasets found in the fuzzy matching literature.

- DBLP-ACM [2]: This dataset consists bibliographic data from two databases DBLP and ACM, which comprises 2,614 and 2,294 records of 4 quasi-identifiers: title, authors, venue, and year. We use 'title' and 'author' for our experiments.
- European Census [3]: This is synthetically generated dataset that simulates the census data of three institutes. We especially use the data from two institutes named cis and census, which comprises 24,613 and 25,343 records of 12 quasi-identifiers that represent different individuals, respectively. We select 9 quasi-identifiers for the experiments that are listed in Section A.
- NCVR (North Carolina Voter Registration) [4]: We selected snapshots from November 2014 and November 2017. The original data contains over 10M individuals for each dataset with 90 quasi-identifiers. For our test, we select 10 quasi-identifiers that are listed in Section A.

Table 2 and Table 3 shows some accuracy metrics for several matchers discussed in this section. We refer Section C for the definitions of accuracy metrics and our method to measure accuracy.

Similarity-based Matchers v.s. Our Encodings. We especially focus on Jaccard and angular similarity measures and the corresponding LSH family, which is widely used for quasi-identifiers of variable-length string.[4] In detail, to measure the similarities between strings, we use q-grams of records for Jaccard similarity, and we utilizes a vectorizer map V by a variant of one-hot encoding of q-grams of the input record for angular similarity.[5] For both Jaccard and angular similarity, we fix $q = 2$ for for q-grams, which is widely used setting for analyzing string similarity. Finally, we investigate the optimal matching threshold t that provides the best F1 score, whose values can be found in Table 2.

We then examine the accuracy change of our encoding process (LSH and deduplication) from the original similarity-based matcher, using the AND-OR parameters (r, b) presented in Table 1. In Table 2, we compare the accuracy results between similarity-based matcher and OTO matcher with LSH encoding.

Recall that in Section 4.2, we suggest some (r, b) pairs for each matching threshold t with some theoretic explanation. These suggestions, however, lead to a significant amount of accuracy degradation as in Table 2 shows, which implies that those parameters are too small to effectively simulate the step function $\mathbf{1}(p \leq t)$. Interestingly, as an unexpected observation, we found that (r, b) from another t' that is slightly larger than t rather provides better F1 score, which is quite close to the original similarity-based matcher with threshold t especially for Jaccard similarity-based matcher.

Considering a record pair (x, y) with a similarity $p = S(x, y)$, the step functions deterministically judge the pairs as matched if $p > t$, and not matched if $p < t$. However, an (r, b)-curve retains two kind of errors compared to the step functions; firstly even if $p > t$, they can be falsely judged as non-match with probability $1 - C_{r,b}(p)$, and secondly, even if $p < t$, they can be falsely judged as match with probability $C_{r,b}(p)$. From the observation, we inferred the reason of this phenomenon in the following. For a fixed x in some record table, there are much more y's in the opposite-side record table such that $p = S(x, y) < t$, which implies that the second kind of error (of probability $C_{r,b}(p)$) occurs much frequently. Therefore, we can expect better accuracy if we slide the (r, b)-curve to the right side by setting slightly larger t' than t, despite some sacrifice in the $p > t$ range.

Meanwhile, our LSH encoding for angular similarity still shows significant differences of F1 score compare to the original similarity-based matcher, except NCVR dataset. We can conclude that the matcher based on angular similarity is not an efficient choice of our LSH encoding; we provide some discussion on this phenomenon in Section B.

Equality-based Matchers v.s. Our Encodings. To make a baseline for comparison with equality-based matchers, we manually investigate the best scoring strategy for each datasets, by mimicking the real world's fuzzy matching methodology [5]. The details for scoring strategy can be found in Section A. The datasets assessing human data – European Census and NCVR – consists of various short quasi-identifiers, where we can successfully build some equality-based

[4] One might be curious about with edit distance, also known as Levenshtein distance, which is also popularly used to measure string similarity. However, to our best knowledge, the LSH family for edit distance has been unknown so far, so we forgo edit distance-based similarity measure in our encoding target.

[5] Specifically, we adopt `TfidfVectorizer` map found in Scikit-learn. For more details, we refer to the website https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

	DBLP-ACM			European Census			NCVR†		
	t	(r,b)	F1	t	(r,b)	F1	t	(r,b)	F1
(Baseline) Jaccard Sim.-based Matcher	0.5	-	0.977	0.6	-	0.985	0.7	-	0.964
(Ours) OTO Matchers with Jaccard LSH encoding & dedup.		(5, 28)	0.778		(7, 30)	0.902		(10, 30)	0.727
	0.5	(6, 50)	0.836	0.6	(8, 50)	0.931	0.7	(11, 50)	0.818
		(7, 100)	0.876		(9, 99)	0.944		(13, 100)	0.879
		(10, 30)	0.940		(10, 30)	0.953		(12, 3)	0.945
	0.7	(11, 50)	0.941	0.7	(11, 50)	0.960	0.9	(35, 30)	0.954
		(13, 100)	0.947		(13, 100)	0.965		(40, 50)	0.955
(Baseline) Angular Sim-based Matcher	0.8	-	0.976	0.7	-	0.947	0.9	-	0.955
(Ours) OTO Matchers with Angular LSH encoding & dedup.		(16, 30)	0.602		(10, 30)	0.035		(35, 30)	0.897
	0.8	(18, 50)	0.639	0.7	(11, 50)	0.064	0.9	(40, 50)	0.931
		(22, 100)	0.814		(13, 100)	0.149		(46, 100)	0.952
		(35, 30)	0.897		(16, 30)	0.340			
	0.9	(40, 50)	0.890	0.8	(18, 50)	0.441			
		(46, 100)	0.905		(22, 100)	0.712			

† We used only the first 10K rows of each datasets for similarity-based matcher tests.

Table 2: Accuracy metrics for similarity-based matchers discussed in Section 4. t indicates the similarity threshold, (r,b) indicates the AND-OR parameter for LSH amplification. The colors of the F1 score cells vary from 0.5 to 1, where F1 scores below 0.5 are not colored.

	European Census		NCVR	
	# feat.	F1	# feat.	F1
(Baseline) Equality-based Matcher	-	0.949	-	0.979
(Ours) OTO Matcher with concat. encoding & dedup.	6	0.948	3	0.976

Table 3: Accuracy metrics for equality-based matchers discussed in Section 4. # feat. shows the number of (concatenated) features after encoding.

matchers (or scoring method) of good accuracy. However, the DBLP-ACM dataset consists of only 4 numbers of long quasi-identifiers such as titles or authors, and we cannot find good equality-based matcher for this dataset, which explains why Table 2 omit the equality-based matcher rows for the DBLP-ACM dataset.

The concatenation encoding itself perfectly preserves the matching result (or accuracy), so the only point where the accuracy can change is deduplication process. The two bottom rows of Table 2 shows the effect of deduplication is quite small, compared to the similarity-based matcher cases.

Finally, we emphasize that our concatenation-based encoding for the equality-based matcher ends with much smaller number of encoded features than LSH-based encoding, while providing quite a decent accuracy; the accuracy metrics are even better than the LSH encoding for NCVR dataset. Although this equality-based method seems a bit less generic than similarity-based methods, we believe that there are many real-world cases that utilize the equality-based method (e.g. Example 5), where our encoding method could be useful.

5 Private Fuzzy Left Join Protocol for Ordered-Threshold-One Matcher

In this section, we propose a cryptographic realization of the private fuzzy left join tailored for the OTO matcher. We start with the assumption that both parties have preprocessed their raw record tables X and Y, containing quasi-identifiers, into *encoded tables* $\hat{X}$ and $\hat{Y}$, respectively. It is important to note that $\hat{X}$ and $\hat{Y}$ are specifically processed to eliminate any duplication within each column vector. To avoid any potential confusion in terminology, we will consistently use the notation $\hat{X}$ and $\hat{Y}$ throughout this section. Furthermore, we will refer to each column of $\hat{X}$ and $\hat{Y}$ as a *feature* to distinguish these from the *quasi-identifiers* found in each column of the raw tables X and Y.

Thanks to Lemma 1, we can consider $\mathsf{matcher}_{\mathrm{mem}}$ instead of $\mathsf{matcher}_{\mathrm{OTO}}$ for the encoded table $\hat{X}$ and $\hat{Y}$. More formally, we consider an ideal functionality as illustrated in Figure 7. This section is dedicated to proposing a protocol that securely realizes this ideal functionality. We note that the size of final output is M which is larger than the number of rows in the left encoded table $\hat{X}$. The index map $\mathsf{idx} : [N_X] \to [M]$ in the final output at receiver side is mapping from the each row of encoded table to the secret shares.

Since $\mathsf{matcher}_{\mathrm{mem}}$ is defined upon the feature-wise membership result $M_i = \{k : \hat{X}_i[k] \in \hat{Y}[k]\}$, we can utilize CPSI. Our protocol consists of the following two main stages:

- Generate a membership share table B of size $N_X \times n$ such that consists of membership shares $b_i[k] = \mathbf{1}(\hat{X}_i[k] \in \hat{Y}[k])$
- For each $i \in [N_X]$ and the smallest k such that $b_i[k]$, output the corresponding payload share.

We elaborate each step in the following sections, and then provide a full protocol.

5.1 Generating Membership Share

Share Generation. To generate the secret shares, both parties agree on the output table size $M = O(N_X)$ and run CPSI n times where $\mathcal{R}$ inputs $\hat{X}[k]$, and $\mathcal{S}$ inputs $\hat{Y}[k]$ and the payload P_Y for $k = 1, \ldots, n$. Then, CPSI outputs secret shares of the membership Boolean vector $B[k] = (b_i[k])_{i\in[M]}$ and the payload candidate vector $V[k] = (v_i[k])_{i\in[M]}$, and the receiver further obtains the index map idx_k :

Parameters: Row sizes N_X and N_Y, and number of columns (features) n, and payload bit-length ℓ.
Input: Encoded tables $\hat{X}$ of size $N_X \times n$ from one party $\mathcal{R}$ and $\hat{Y}$ of size $N_Y \times n$ along with the payloads $P_Y = \{p_j \in \{0,1\}^\ell : j \in [N_Y]\}$ from the other party $\mathcal{S}$.
Functionality: The ideal functionality first compute $\mathsf{matcher}_{\mathrm{OTO}}$ defined as Definition 2, and define $b = (b_i) \in \{0,1\}^M$ and $v = (v_i) (\in (\{0,1\}^\ell)^{N_X}$ so that

$$b_{\mathsf{idx}(x)} = 1,\ w_{\mathsf{idx}(x)} = v_j \qquad \text{if } \mathsf{matcher}_{\mathrm{OTO}}(i) = j \neq \perp$$
$$b_i = 0,\ w_i \leftarrow \{0,1\}^\ell \qquad \text{otherwise.}$$

for a index map $\mathsf{idx} : [N_X] \to [M]$. It then returns the secret shares of b and w to each party; precisely, it samples random vectors $s_b \in \{0,1\}^T$ and $s_w \in (\{0,1\}^\ell)^T$. Then sends s_b and s_w to the sender, and $b \oplus s_b$ and $w \oplus s_w$ and idx to the receiver.

Figure 7: Ideal functionality of Private Left-Join with the OTO matcher.

$[N_X] \to [M]$. From the definition of CPSI, those outputs satisfy

$$b_i[k] = 1, v_i[k] = v_j \quad \text{if } \exists j \in [N_Y] \text{ s.t } \hat{X}_i[k] = \hat{Y}_j[k] \in \hat{Y}[k]$$

Otherwise, we have $b_i[k] = 0$ and $v_i[k]$ is just a random string. For convenience, we denote the membership secret share by $B^*[k] = (b_i^*[k])_{i\in[M]}$ and the payload secret share $V^*[k] = (v_i^*[k])_{i\in[M]}$, for $* \in \{\mathcal{S}, \mathcal{R}\}$.

One might think that this process is a straight-forward application of CPSI on each feature column $\hat{X}[k]$ and $\hat{Y}[k]$, since CPSI exactly computes the secret shares of $b_i[k]$. However, as a technical detail, recall from Figure 2 that CPSI outputs the secret shares in a random order $\mathsf{idx}_k : [N_X] \to [M]$ only known to $\mathcal{R}$. Therefore, there should be an additional procedure to unify the alignment of the secret shares for both $\mathcal{R}$ and $\mathcal{S}$. The most naive approach would be letting $\mathcal{R}$ send the index map idx_k to $\mathcal{S}$, so that $\mathcal{S}$ locally adjusts the ordering of each CPSI output vector. However, this naive solution leads to unwanted information leakage, whose details are presented in [22]. We resolve this problem by employing permute-and-share (PnS) functionality to let $\mathcal{R}$ obliviously permutes $\mathcal{S}$'s list of shares.

Details for Alignment Unification. The receiver starts with an arbitrary global index $\mathsf{gIdx} : [N_X] \to [M]$ at the beginning of the protocol. After each CPSI execution where the receiver obtains $\mathsf{idx}_k : [N_X] \to [M]$, it also sets a random permutation $\pi_k : [M] \to [M]$ satisfying $\pi_k(\mathsf{gIdx}(i)) = \mathsf{idx}_k(i)$ for every $i \in [N_X]$.Then two parties engage PnS on inputs π_k from the receiver and $B^{\mathcal{S}}[k]$ from the sender. By definition, it outputs $\overline{B}_k^{\mathcal{R}} = (\overline{b}_i^{\mathcal{R}}[k])_{i\in[M]}$ and $\overline{B}^{\mathcal{S}}[k] = (\overline{b}_i^{\mathcal{S}}[k])_{i\in[M]}$ for each party such that $\overline{B}^{\mathcal{S}}[k] \oplus \overline{B}^{\mathcal{R}}[k] = \pi_k(B^{\mathcal{S}}[k])$. The receiver then adjusts its shares as

$$\overline{B}^{\mathcal{R}}[k] \leftarrow \overline{B}^{\mathcal{R}}[k] \oplus \pi_k(B^{\mathcal{R}}[k]) \tag{2}$$

to have

$$\overline{B}^{\mathcal{S}}[k] \oplus \overline{B}^{\mathcal{R}}[k] = \pi_k(B^{\mathcal{S}}[k]) \oplus \pi_k(B^{\mathcal{R}}[k]) = \pi_k(B[k]).$$

Finally, it holds that for every $i \in [N_X]$,

$$\overline{b}_{\mathsf{gIdx}(i)}[k] = b_{\pi_k(\mathsf{gIdx}_k(i))}[k] = b_{\mathsf{idx}_k(i)}[k] = \mathbf{1}(\hat{X}_i[k] \in \hat{Y}[k]), \tag{3}$$

which ensures the correctness of the alignment.

Augmenting Payload Shares. As our desired output is to generate secret shares of the corresponding payload, we need to remember specific matching index $j \in [N_Y]$ (or the corresponding payload) as well as the membership Boolean $b_i[k]$. To securely realize this, we let the sender further inputs the payload set P_Y for each CPSI execution, so that two parties obtains the secret shares of the payload candidate vector v_k defined by

$$v_i[k] := \begin{cases} v_j & \text{if } b_i[k] = 1 \text{ by } \hat{X}_i[k] = \hat{Y}_j[k] \\ \mathsf{random} & \text{if } b_i[k] = 0. \end{cases}$$

These shares also need to be re-aligned with respect to the global index gIdx, which can be done in the exactly same way by considering the concatenated vector $(B^*[k] \| V^*[k])$ instead of $B^*[k]$.

5.2 Aggregation of Shares

Two parties now obtain the secret shares of the membership results $b_i[k]$ and corresponding payloads $v_i[k]$. Then the desired output can be obtained by selecting one payload share $v_i[k]$ for the smallest index k such that $b_i[k] = 1$. It can be easily checked that this can be done by the recursive evaluation of the following MUX-gate[6] from $k = n$ down to $k = 1$:

$$v_i \leftarrow \mathsf{MUX}(v_i[k], v_i, b_i[k]), \tag{4}$$

where $\mathsf{MUX}(v_0, v_1, b)$ outputs v_b. Then it only remains to specify how to evaluate the MUX-gate in a secret shared state. We consider a simple protocol called SSMUX that performs this using two OTs, whose details are presented by Algorithm 1.

Algorithm 1 Secret-Shared MUX (SSMUX).

Input: Secret shares of selector bit b, and two input strings v_0, v_1, say b^*, v_0^*, v_1^* for $* \in \{S, R\}$.
Output: Secret shares of v_b, say v_b^* for $* \in \{S, R\}$.

// 1st OT
$\mathcal{S}$ picks random masking $r^{\mathcal{S}}$, and sets $m_{b^{\mathcal{S}}} = v_0^{\mathcal{S}} \oplus r^{\mathcal{S}}$ and $m_{1-b^{\mathcal{S}}} = v_1^{\mathcal{S}} \oplus r^{\mathcal{S}}$.
Two parties execute SOT with messages m_0, m_1 from $\mathcal{S}$ and choice $b^{\mathcal{R}}$ from $\mathcal{R}$, and $\mathcal{R}$ receives $m_{b^{\mathcal{R}}}$.
// 2nd OT
$\mathcal{R}$ picks random masking $r^{\mathcal{R}}$ and sets $m'_{b^{\mathcal{R}}} = v_0^{\mathcal{R}} \oplus r^{\mathcal{R}}$ and $m'_{1-b^{\mathcal{R}}} = v_1^{\mathcal{R}} \oplus r^{\mathcal{R}}$. Two parties execute SOT with messages m'_0, m'_1 from $\mathcal{R}$ and choice $b^{\mathcal{S}}$ from $\mathcal{S}$, and $\mathcal{S}$ receives $m'_{b^{\mathcal{S}}}$
// Finalize
$\mathcal{S}$ outputs $v_b^{\mathcal{S}} := r^{\mathcal{S}} \oplus m'_{b^{\mathcal{S}}}$, and $\mathcal{R}$ outputs $v_b^{\mathcal{R}} := r^{\mathcal{R}} \oplus m_{b^{\mathcal{R}}}$.

Lemma 2. *Algorithm 1 securely realizes a MUX evaluation in a secret-shared state against semi-honest adversaries in the* SOT_ℓ*-hybrid model.*

Proof. See Section E.1. □

[6] We abuse the terminology *multiplexer* (MUX) to take two strings as inputs, rather than bit inputs of the standard notion for MUX.

5.3 Full Protocol and Cost Analysis

Algorithm 2 shows the full description of our protocol, and Theorem 1 shows that it securely realizes the ideal functionality in the semi-honest model.

Algorithm 2 Private Fuzzy Left Join for the OTO matcher.

Input: $\mathcal{R}$ with an encoded table $\hat{X}$ of size $N_X \times n$
$\mathcal{S}$ with an encoded table $\hat{Y}$ of size $N_Y \times n$, and a payload set $P_Y = \{p_y \in \{0,1\}^\ell : y \in Y\}$

// Generating Matching Shares

1: Both parties agree on the CPSI output size $M = O(N_{\mathcal{R}})$
2: **for** $k = 1$ to n **do**
3: Both parties run CPSI where $\mathcal{R}$ plays the receiver role with input $\hat{X}[k]$ and $\mathcal{S}$ plays the sender role with input $\hat{Y}[k]$ and P_Y. Each party obtains $(B_k^* \| V_k^*)$ for $* \in \{\mathcal{S}, \mathcal{R}\}$, and $\mathcal{R}$ further obtain an index map idx_k.

// Unification of Alignment

4: $\mathcal{R}$ sets a random global index map $\mathsf{gIdx} : [N_X] \to [M]$
5: **for** $k = 1$ to n **do**
6: $\mathcal{R}$ picks a permutation $\pi_k : [M] \to [M]$ satisfying $\pi_k(\mathsf{idx}_k(x)) = \mathsf{gIdx}(x)$ for every $x \in [N_X]$
7: Both parties run PnS, where $\mathcal{R}$ plays the sender role with input π_k, and $\mathcal{S}$ plays the receiver role with input $(B_k^{\mathcal{S}} \| V_k^{\mathcal{S}})$. Each party obtains $(\overline{B}_k^* \| \overline{V}_k^*)$ for $* \in \{\mathcal{S}, \mathcal{R}\}$.
8: For $j \in [M]$, $\mathcal{R}$ updates $(\overline{B}_k^{\mathcal{R}} \| \overline{V}_k^{\mathcal{R}})$ as in Equation (2).

// Aggregation of Shares

9: Each party randomly samples $V^* \leftarrow (\{0,1\}^\ell)^M$ and set $b^* = 0^M \in \{0,1\}^M$
10: **for** $k = n$ down to 1 **do**
11: Both parties update V^* by the output of SSMUX on selector shares $\bar{B}_k^*$ and input strings $\bar{V}_k^*$ and V^*, and update b^* by the output of secrete shared OR computation using GMW protocol and input strings $\bar{B}_k^*$ and b^*.
12: Both parties
13: Each party outputs (b^*, V^*) for $* \in \{\mathcal{S}, \mathcal{R}\}$, and receiver outputs $\{\mathsf{gIdx}\}$ additionally.

Theorem 1. *The protocol described in Algorithm 2 is a secure realization of the ideal functionality given in Figure 7 against semi-honest adversary, assuming that the CPSI protocol and the* PnS *protocol are secure.*

Proof. See Section E.2. □

Cost Analysis. The asymptotic complexity analysis can be found in Section G, due to the space limit. The overall computation cost is $O(\ell n N)$ ROTs and $O(nN)$ OKVS operation, and the communication cost is dominated by $O(\ell n N \log N)$.

6 Experiments

This section provides benchmark setup and implementation results of our PFLJ protocol on fuzzy records. The source code is available at https://github.com/samsungsds-research-papers/FuzzyPC.

Environment. The experiments were conducted on a single machine equipped with 3.50 GHz Intel Xeon processors and 128 GB of RAM. Network environments were simulated using the Linux `tc` command, with LAN configured with 5 Gbps bandwidth and 0.3 ms latency, and WAN configured with 100 Mbps bandwidth and 40 ms latency (80 ms round-trip time). All experiments were executed in a single-threaded environment unless otherwise specified. Our source codes are developed in C++20 and leverage several libraries for cryptographic operations and network simulation, including the libOTe library [40], the vole-psi library [46], the xxHash library [17], and the Kuku library [26]. Notably, the xxHash library is utilized exclusively for scenarios requiring a non-cryptographic hash function.

MPC protocols. For OT extension (OTe) and vector-oblivious linear evaluation (VOLE), we adopt the protocol described in [18], which bases its security on the hardness of decoding structured LDPC codes. In particular, we utilize the ExConv7x24 encoding outlined in [38]. Additionally, the oblivious key-value store (OKVS) algorithm presented in [37] is integrated into our deployed CPSI protocol [37].

General Parameters. Throughout this section, we assume a statistical security parameter $\lambda = 40$ and a computational security parameter $\kappa = 128$. For cuckoo hashing within the CPSI protocol, we utilize $\gamma = 3$ hash functions and set the table size $M \approx 1.3N_Y$, where N_Y denotes the number of records.

Setup and Online Separation. In multi-party cryptographic protocols, the *setup* (or offline) phase is characterized as a preparatory process that occurs prior to the receipt of specific input data. In our protocol, the random OT extension is an operation that can be efficiently executed during this setup phase. Consequently, we separately report the costs associated with both the setup and online phases within the timing reports presented in this section.

6.1 Experiments with Various Datasets

In this section, we test overall performance of our protocol for the given three datasets; DBLP-ACM, European Census, and NCVR, whose descriptions are presented in Section 4.4. Table 4 summarizes the performance of two different matcher (equality-based and Jaccard LSH) on three datasets (European, NCVR, and DBLP-ACM) with respect to communication cost, setup and online times, and accuracy metrics. Note that our method achieves the same accuracy metrics as the OTO matcher with deduplication, which demonstrates the correctness of our protocol.

6.2 Comparison with MK-PMC

We compare with [8], which presents a close result to our protocol. Their work outputs a random identifier and provides a robust security proof, albeit with the incidental leakage of the linkage graph shape. For a fair comparison, we executed the open-sourced implementation of their protocol Private-ID MultiKey[7] within our benchmark environment. Notably, we adapted the implementation to run with the given number of threads for consistency with our testing conditions, as the original code is designed to utilize maximal threads.

Table 5 compares the performance of our protocol and [8], on the LAN/WAN network across dataset sizes (N) and features (n). The performance metrics include communication cost, setup time, and online time; PID protocol has no computation that can be

[7] Available at https://github.com/facebookresearch/Private-ID.

Data	Matcher	n (# Features)	Comm. (MB)	Network	Setup (s)	Online (s)	Prec./Recall/F1
European	Equality-based	6	76.21 MB	LAN	4.47	1.13	0.999/0.902/0.948
				WAN	12.37	15.27	
	Jaccard LSH	100	1413.24 MB	LAN	75.9	22.02	0.992/0.938/0.965
				WAN	188.72	273.47	
NCVR	Equality-based	3	1615.02 MB	LAN	169.0	23.69	0.992/0.960/0.976
				WAN	172.77	169.13	
	Jaccard LSH	3	1615.02 MB	LAN	167.6	23.7	0.996/0.895/0.943
				WAN	172.21	169.37	
DBLP-ACM	Jaccard LSH	50	91.92	LAN	4.95	1.68	0.959/0.935/0.947
				WAN	53.55	65.85	

Table 4: Communication cost, setup and online time, and accuracy measures for European, NCVR, and DBLP-ACM.

N	n	Method	Comm. (MB)	LAN		WAN	
				Setup	Online	Setup	Online
10^5	3	Ours	147.58	10.71	1.97	16.2	18.9
		PID	43.48	98.74		112.19	
	5	Ours	277.73	18.11	3.48	27.2	34.6
		PID	61.79	143.34		161.54	
	7	Ours	407.86	25.55	4.89	38.1	50.9
		PID	80.10	190.37		210.92	
	9	Ours	538.01	32.89	6.41	48.3	66.7
		PID	98.42	240.62		265.42	
10^6	3	Ours	1615.02	168.43	23.73	173.4	168.9
		PID	434.87	1037.8		1093.9	
	5	Ours	3074.86	285.6	44.22	294.9	317.4
		PID	617.98	1512.7		1597.59	
	7	Ours	4534.7	403.0	62.3	415.1	467.9
		PID	801.08	2009.5		2118.6	
	9	Ours	6006.9	529.9	82.2	559.2	616.6
		PID	984.19	2544.20		2672.5	

Table 5: Comparison with Private-ID (PID) from [8] on LAN network. N is the number of records in each record table where $N = N_X = N_Y$, and n is the number of features (after encoding).

N	n	#Thr.	LAN			WAN		
			PID	Ours		PID	Ours	
			total	online	total	total	online	total
10^5	3	1	98.74	1.97	12.68	112.19	18.9	35.1
		2	62.39			72.30		
		4	40.14			49.34		
		8	27.68			36.67		
10^6	3	1	1037.8	23.73	192.2	1093.9	168.9	342.3
		2	645.55			707.7		
		4	411.62			471.2		
		8	286.34			345.2		

Table 6: Timings of PID [8] with multi-threading, versus our single thread timing. N is the number of records in each record table where $N = N_X = N_Y$. "#Thr." stands for the number of used threads, and n is the number of features (after encoding).

done without input data, so the running time of previous work is setup and online combined time. Our protocol shows higher communication costs but lower setup and online times compared to the PID method. For example, over LAN network with $N = 10^5$ and $n = 3$, our method has a communication cost of 147.58 MB, setup time of 10.71s, and online time of 1.97s, whereas the PID method has a communication cost of 43.48 MB but a higher time of 98.74s with no separation of setup and online. Over the WAN environment, our protocol takes 16.2s for setup, and 18.9s for online phase, whereas PID takes 112.19s for overall protocol. For financial cost analysis on a cloud computing scenario, the PID protocol is expected to cost less than ours regardless of network environment, as communication tends to cost more than computation; for example, an AWS machine costs 0.0137 USD for our protocol and costs 0.0072 USD for the PID protocol.[8]

The PID protocol extensively utilized multi-threading, as their underlying computations are highly friendly for it. However, due to the complex computations and frequent interactions between parties, our method has several challenges in leveraging the benefits of multi-threading. Table 6 compares PID timing results with multi-threading from 1 to 8 threads with our single thread timing. It shows that the timing gap between our protocol, and Table 6 narrowed from 7.78x to 2.18x for 1 to 8 threads even for LAN network. Although we observe no further gain by setting more threads due to the limitation of our deployed machine, we have to say that PID protocol could still be better than our protocol, provided with sufficient multi-threading resources.

To conclude, although our protocol requires higher communication cost than the PID protocol, the advantage on setup and online times results in better running time even for WAN network of 100Mbps bandwidth and 80ms RTT. However, we clarify that the PID protocol has a definite advantage on having smaller communication cost and good multi-threading performance, which makes it better than our protocol when the network environment is much slower or each party has sufficient computational resource for multi-threading.

[8]Both $\mathcal{R}$ and $\mathcal{S}$ are assumed to use `m7a.medium` in Eastern United States. Protocol runs for 3 features, and 10^5 records each.

References

[1] Federal and State Laws Restrict Use of SSNs, yet Gaps Remain. https://www.gao.gov/assets/gao-05-1016t.pdf, 2005. United States Government Accountability Office.

[2] Benchmark datasets for entity resolution, 2010. https://dbs.uni-leipzig.de/research/projects/benchmark-datasets-for-entity-resolution.

[3] Synthetic European Census Data, 2011. https://cros-legacy.ec.europa.eu/content/job-training_en.
[4] North Carolina Voter Registration (NCVR) database snapshots, 2014. https://dl.ncsbe.gov/index.html?prefix=data/Snapshots/.
[5] SEER-Medicare database, 2020. https://healthcaredelivery.cancer.gov/seermedicare/overview/linked.html.
[6] Adir, A., Aharoni, E., Drucker, N., Kushnir, E., Masalha, R., Mirkin, M., and Soceanu, O. Privacy-preserving record linkage using local sensitive hash and private set intersection, 2022.
[7] Arbitman, Y., Naor, M., and Segev, G. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *FOCS 2010* (2010), IEEE, pp. 787–796.
[8] Buddhavarapu, P., Case, B. M., Gore, L., Knox, A., Mohassel, P., Sengupta, S., Taubeneck, E., and Xue, M. Multi-key Private Matching for Compute. Cryptology ePrint Archive, Paper 2021/770, 2021. https://eprint.iacr.org/2021/770.
[9] Buddhavarapu, P., Knox, A., Mohassel, P., Sengupta, S., Taubeneck, E., and Vlaskin, V. Private matching for compute. Cryptology ePrint Archive, Paper 2020/599, 2020. https://eprint.iacr.org/2020/599.
[10] Chandran, N., Gupta, D., and Shah, A. Circuit-PSI with Linear Complexity via Relaxed Batch OPPRF. *PoPETs 2022* (2022), 353–372.
[11] Charikar, M. S. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2002), STOC '02, Association for Computing Machinery, p. 380–388.
[12] Chase, M., Ghosh, E., and Poburinnaya, O. Secret-shared shuffle. In *ASIACRYPT 2020* (Cham, 2020), Springer, pp. 342–372.
[13] Chen, F., Jiang, X., Wang, S., Schilling, L. M., Meeker, D., Ong, T., Matheny, M. E., Doctor, J. N., Ohno-Machado, L., and Vaidya, J. Perfectly Secure and Efficient Two-Party Electronic-Health-Record Linkage. *IEEE Internet Computing 22*, 2 (2018), 32–41.
[14] Christen, P., Ranbaduge, T., Vatsalan, D., and Schnell, R. Precise and fast cryptanalysis for bloom filter based privacy-preserving record linkage. *IEEE Transactions on Knowledge and Data Engineering 31*, 11 (2019), 2164–2177.
[15] Christen, P., Schnell, R., Vatsalan, D., and Ranbaduge, T. Efficient Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage. In *Advances in Knowledge Discovery and Data Mining* (Cham, 2017), J. Kim, K. Shim, L. Cao, J.-G. Lee, X. Lin, and Y.-S. Moon, Eds., Springer International Publishing, pp. 628–640.
[16] Ciampi, M., and Orlandi, C. Combining Private Set-Intersection with Secure Two-Party Computation. In *Security and Cryptography for Networks* (Cham, 2018), D. Catalano and R. De Prisco, Eds., Springer International Publishing, pp. 464–482.
[17] Collet, Y. xxhash: Extremely fast hash algorithm. *GitHub https://github.com/Cyan4973/xxHash* (2023).
[18] Couteau, G., Rindal, P., and Raghuraman, S. Silver: Silent vole and oblivious transfer from hardness of decoding structured ldpc codes. In *CRYPTO 2021* (Cham, 2021), Springer, pp. 502–534.
[19] Dusetzina, S. B., Tyree, S., Meyer, A.-M., Meyer, A., Green, L., and Carpenter, W. R. Linking data for health services research: a framework and instructional guide.
[20] Fellegi, I. P., and Sunter, A. B. A Theory for Record Linkage. *Journal of the American Statistical Association 64*, 328 (1969), 1183–1210.
[21] Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., and Yanai, A. Oblivious key-value stores and amplification for private set intersection. In *CRYPTO 2021* (2021), Springer, pp. 395–425.
[22] Huang, C., Zhang, F., Tan, M., Hou, C., Zhao, Y., Rao, H., Cheng, Y., Li, Z., and Liu, Z. Idash 2022 track 4: Psi-circuit and oblivious switching network-based privacy-preserving record linkage (team angelfl), 2022.
[23] Ion, M., Kreuter, B., Nergiz, A. E., Patel, S., Saxena, S., Seth, K., Raykova, M., Shanahan, D., and Yung, M. On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality. In *EuroS&P 2020* (2020), IEEE, pp. 370–389.
[24] Ishai, Y., Kilian, J., Nissim, K., and Petrank, E. Extending Oblivious Transfers Efficiently. In *CRYPTO 2003* (2003), Springer, pp. 145–161.
[25] Karakoç, F., and Küpçü, A. Linear Complexity Private Set Intersection for Secure Two-Party Protocols. In *Cryptology and Network Security* (Cham, 2020), S. Krenn, H. Shulman, and S. Vaudenay, Eds., Springer International Publishing, pp. 409–429.
[26] Microsoft Kuku. https://github.com/microsoft/Kuku, 2021. Microsoft Research, Redmond, WA.
[27] Kussel, T., Brenner, T., Tremper, G., Schepers, J., Lablans, M., and Hamacher, K. Record linkage based patient intersection cardinality for rare disease studies using Mainzelliste and secure multi-party computation. *Journal of Translational Medicine 20*, 1 (Oct 2022), 458.
[28] Lazrig, I., Ong, T. C., Ray, I., Ray, I., Jiang, X., and Vaidya, J. Privacy preserving probabilistic record linkage without trusted third party. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)* (2018), pp. 1–10.
[29] Lepoint, T., Patel, S., Raykova, M., Seth, K., and Trieu, N. Private join and compute from pir with default. In *ASIACRYPT* (2021), Springer, pp. 605–634.
[30] Mohassel, P., Rindal, P., and Rosulek, M. Fast database joins and psi for secret shared data. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020), pp. 1271–1287.
[31] Mohassel, P., and Sadeghian, S. How to hide circuits in MPC an efficient framework for private function evaluation. In *EUROCRYPT 2013* (Cham, 2013), Springer, pp. 557–574.
[32] Mouris, D., Masny, D., Trieu, N., Sengupta, S., Buddhavarapu, P., and Case, B. Delegated private matching for compute. Cryptology ePrint Archive, Paper 2023/012, 2023. https://eprint.iacr.org/2023/012.
[33] Niedermeyer, F., Steinmetzer, S., Kroll, M., and Schnell, R. Cryptanalysis of Basic Bloom Filters Used for Privacy Preserving Record Linkage. *Journal of Privacy and Confidentiality 6*, 2 (Dec. 2014).
[34] Onar, S. Ç., Öztaysi, B., and Kahraman, C. Record linkage using fuzzy sets for detecting suspicious financial ransactions. In *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)* (2015), Atlantis Press, pp. 241–246.
[35] Pinkas, B., Schneider, T., Segev, G., and Zohner, M. Phasing: Private Set Intersection Using Permutation-based Hashing. In *USENIX 2015* (2015), USENIX Association.
[36] Pinkas, B., Schneider, T., Tkachenko, O., and Yanai, A. Efficient Circuit-based PSI with Linear Communication. In *EUROCRYPT 2019* (Cham, 2019), Springer, pp. 122–153.
[37] Raghuraman, S., and Rindal, P. Blazing fast psi from improved okvs and subfield vole. In *CCS 2022* (New York, NY, USA, 2022), ACM, p. 2505–2517.
[38] Raghuraman, S., Rindal, P., and Tanguy, T. Expand-convolute codes for pseudorandom correlation generators from lpn. In *Annual International Cryptology Conference* (2023), Springer, pp. 602–632.
[39] Randall, S., Wichmann, H., Brown, A., Boyd, J., Eitelhuber, T., Merchant, A., and Ferrante, A. A blinded evaluation of privacy preserving record linkage with Bloom filters. *BMC Medical Research Methodology 22*, 1 (Jan 2022), 22.
[40] Rindal, P. libote: an efficient, portable, and easy to use oblivious transfer library, 2022.
[41] Rindal, P., and Schoppmann, P. VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE. In *EUROCRYPT 2021* (Cham, 2021), Springer, pp. 901–930.
[42] Schnell, R. Privacy-preserving record linkage in the context of a national statistics institut, 2021.
[43] Schnell, R., Bachteler, T., and Reiher, J. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making 9*, 1 (Aug 2009), 41.
[44] Schnell, R., and Borgs, C. Randomized response and balanced bloom filters for privacy preserving record linkage. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (2016), pp. 218–224.
[45] Stammler, S., Kussel, T., Schoppmann, P., Stampe, F., Tremper, G., Katzenbeisser, S., Hamacher, K., and Lablans, M. Mainzelliste SecureEpiLinker (MainSEL): privacy-preserving record linkage using secure multi-party computation. *Bioinformatics 38*, 6 (09 2020), 1657–1668.
[46] Visa-Research. volepsi: Efficient private set intersection base on vole, 2022.
[47] Yang, K., Weng, C., Lan, X., Zhang, J., and Wang, X. Ferret: Fast extension for correlated ot with small communication. In *CCS 2020* (New York, NY, USA, 2020), ACM, pp. 1607–1626.

Appendix A Quasi-Identifiers Details

European Census. Among the raw data with 12 quasi-identifiers, we used the following 9 quasi-identifiers: `PERNAME1`, `PERNAME2`, `DOB_DAY`, `DOB_MON`, `DOB_YEAR`, `ENUMPC`, `ENUMCAP`.

For the similarity-based matcher and LSH encoding, we concatenate all 9 quasi-identifiers as one long string. For the equality-based matcher, we set the match threshold 1, and assign score 1 on the following combinations of quasi-identifiers.

- `PERNAME1` (Soundex), `PERNAME2` (Soundex), 2 out of 3 DOB (date of birth) parts, sex.
- `ENUMPC`, `ENUMCAP`, 2 out of 3 DOB (date of birth) parts.

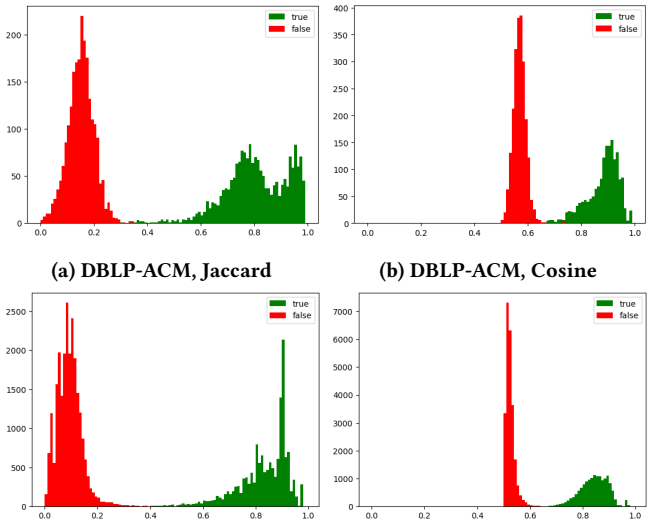
NCVR Among the raw data with 90 quasi-identifiers, we used the following 10 quasi-identifiers: `first name`, `middle name`, `last name`, `house number`, `street number`, `zip code`, `birth place`, `birth year`, `phone area code`, `phone number`.

For the similarity-based matcher and LSH encoding, we concatenate all 10 quasi-identifiers as one long string. For the equality-based matcher, we set the match threshold 1, and assign score 1

on the following combinations of quasi-identifiers: `first name`, `middle name`, `last name` and one of the following sets:

- `house number`, `street number`, `zip code`
- `birth place`, `birth year`
- `phone area code`, `phone number`

Appendix B Angular Similarity and LSH



(a) DBLP-ACM, Jaccard (b) DBLP-ACM, Cosine

(c) European Census, Jaccard (d) European Census, Cosine

Figure 8: Similarity distributions for true match paris (green) and false match pairs (red). For the false match pair, we randomly sample 10000 pairs.

The underlying idea of LSH encoding with AND-OR amplification is to judge two strings x and y of similarity $p := S(x, y)$ with probability $1 - (1 - p^r)^b$. To compute the angular similarity between strings, we relied on a vectorizer map V that converts a string s into some numerical vector $V(s) \in \mathbb{R}^n$. The problem is that our deployed vectorize map is (a variant of) one hot encoding of q-grams, which always makes $V(s)$ has non-negative component. This makes any two strings x and y has at least 0.5 angular similarity; whereas Jaccard similarity of two strings can vary from 0 to 1. As a visual explanation, Figure 8 shows the distributions of similarity for true match pairs and false match pairs. Considering the AND-OR parameters (r, b) is chosen so that the curve $1 - (1 - p^r)^b$ distinguishes the green and red parts, r and b should be large enough to have a steep curve than Jaccard similarity if the same F1 score is required.

Appendix C Accuracy Metrics

To measure the quality of our matching, we use the unique payload in the given dataset to check whether the linkage is correct or not. When an entity in the receiver's data is matched by some entity in the sender's data, we define the result as correct if the result payload is the same as the sender's payload. Precisely, Algorithm 3 shows how we compute true-positive to false-negative.

Algorithm 3 Accuracy Measurement

Input: Linkage information link : $[N_{\mathcal{R}}] \to [N_{\mathcal{S}}] \cup \{\perp\}$, the receiver's received payload $V_{\mathcal{S}}$, and the sender's original payload $V_{\mathcal{R}}$
Output: (tp, tn, fp, fn)

```
for i ∈ [N_R] do
    if link(i) ≠⊥ then
        if V_R[i] = V_S[link(i)] then
            tp+ = 1
        else
            fn+ = 1
    else
        if V_R[i] ∈ V_S then
            // Wrong ID is assigned
            fp+ = 1
        else
            tn+ = 1
return (tp, tn, fp, fn))
```

Then the accuracy metrics such as Prec, Recall and F1 are defined by

$$\text{Prec} = \frac{\text{tp}}{\text{tp} + \text{fp}}, \quad \text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}, \quad \text{F1} = \frac{2}{\text{Prec}^{-1} + \text{Recall}^{-1}}.$$

Appendix D Experiments with Various Input Sizes

This section reports the experimental outcomes for various input settings including the number of records and the number of features. To check the performance in various input settings, we randomly generate for the given number of records N_X, N_Y and the number of features n.

Figure 9 presents a detailed analysis of the performance by plotting the time taken for various components (CPSI, PnS, AS) against the number of input data points, and the number of features after encoding. These figures provide insights of how each component's processing time scales with different input sizes and the number of features counts.

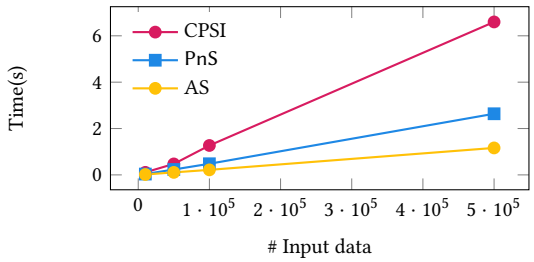
Both graphs clearly demonstrate that the CPSI component is the most time-consuming part of the process. The total processing time seems to grow linearly (which is theoretically not true for huge N) in both the number of records and the number of features.

Appendix E Missing Proofs

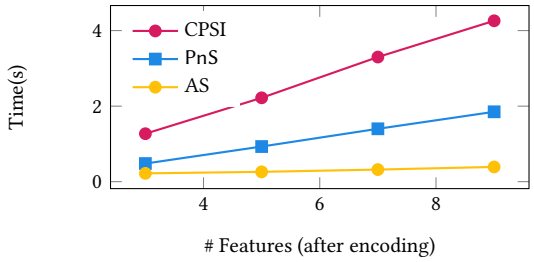
E.1 Proof for Lemma 2

Proof. The security proof is immediate; each SOT output reveals nothing since it is randomly masked, and the intermediate view of SOT can be simulated by the SOT simulator.

We proceed to the correctness proof. The output of the first OT $m_{b^{\mathcal{R}}}$ is $v_0^{\mathcal{S}} \oplus r^{\mathcal{S}}$ if $b^{\mathcal{R}} = b^{\mathcal{S}}$, and $v_1^{\mathcal{S}} \oplus r^{\mathcal{S}}$ if $b^{\mathcal{R}} = 1 \oplus b^{\mathcal{S}}$, which is exactly the same with $v_b^{\mathcal{S}} \oplus r^{\mathcal{S}}$. Then, the first OT generates the secret shares of $v_b^{\mathcal{S}}$, where $\mathcal{S}$ obtains $r^{\mathcal{S}}$ and $\mathcal{R}$ obtains $v_b^{\mathcal{S}} \oplus r^{\mathcal{S}}$. Similarly, the second OT generates the secret shares of $v_b^{\mathcal{R}}$. At the



(a) Various input sizes $N_X = N_Y$ with three features ($n = 3$).



(b) Various number of features with $N_X = N_Y = 10^5$.

Figure 9: Timings for each part according to the size of inputs and the number of features on LAN network.

finalize step, two parties locally add each share of $v_b^{\mathcal{S}}$ and $v_b^{\mathcal{R}}$, which becomes secret shares of v_b. □

E.2 Proof for Theorem 1

Proof. As the correctness is already discussed in the protocol detail explanation so far, we only need to prove the security. For a corrupt receiver, it is sufficient to simulate the views consisting of the outputs of CPSI, PnS, and SSMUX. The simulation can be done by replacing all the outputs with uniform random. Precisely, the simulator $\mathcal{T}$ interacts with $\mathcal{R}$ as follows.

- $\mathcal{T}$ plays the role of $\mathcal{F}_{\mathsf{cpsi}}$. When $\mathcal{R}$ sends $Y[k]$ to $\mathcal{F}_{\mathsf{cpsi}}$, $\mathcal{T}$ sends random shares $B_k^{\mathcal{R}} \in \{0,1\}^M$ and $V_k^{\mathcal{R}} \in (\{0,1\}^\ell)^M$.
- $\mathcal{T}$ plays the role of $\mathcal{F}_{\mathsf{pns}}$. When $\mathcal{R}$ sends π_k to $\mathcal{F}_{\mathsf{pns}}$, $\mathcal{T}$ sends random shares $\overline{B}_k^{\mathcal{R}} \in \{0,1\}^M$ and $\overline{V}_k^{\mathcal{R}} \in (\{0,1\}^\ell)^M$.
- $\mathcal{T}$ plays the role of $\mathcal{F}_{\mathsf{ssmux}}$. When $\mathcal{R}$ sends the input shares to $\mathcal{F}_{\mathsf{ssmux}}$, $\mathcal{T}$ sends a random share v^*.

To prove that this simulation is indistinguishable from the real protocol, we consider the following hybrid worlds.

- Hybrid 0: The same as the real protocol.
- Hybrid 1: $\mathcal{T}$ in this world plays the role of $\mathcal{F}_{\mathsf{cpsi}}$ and $\mathcal{F}_{\mathsf{pns}}$. We already assumed that CPSI and PnS are secure, and those protocol outputs a uniformly random shares by definition. So, this world is indistinguishable from the previous world.
- Hybrid 2: Now, $\mathcal{T}$ in this world additionally plays the role of $\mathcal{F}_{\mathsf{ssmux}}$. As Lemma 2 already proved that SSMUX is secure, it is indistinguishable from the previous world. One can observe that this world is same as the simulation.

For a corrupt sender, it is sufficient to simulate the views consisting of the outputs of CPSI, PnS, and SSMUX, whose proof is almost similar to that for a corrupt receiver. □

Appendix F Deduplication Process

Algorithm 4 Deduplication.

Input: An encoded record table $\hat{X}$ of size $N_X \times n$

Set $\mathcal{I} = \emptyset$
for $k = 1$ to n **do**
  for $i = 1$ to N_X **do**
    if $i \notin \mathcal{I}$ **then**
      Compute $D = \{j \in [N_X] | \hat{X}_i[k] = \hat{X}_j[k]\}$.
      Choose $j^* \leftarrow_{\$} D$.
      For $j \in D \backslash \{j^*\}$, replace $\hat{X}_j[k]$ to a random string from $\{0,1\}^\lambda$ until $\hat{X}_j[k] \notin X[k]$.
      Update $\mathcal{I} = \mathcal{I} \cup D$.
return $\hat{X}$

Appendix G Details for Cost Analysis

Table 7 summarizes the analysis below. Given the statistical security parameter λ, we choose the table size $M = \varepsilon \cdot N_X$ for some $\varepsilon > 1$ so that cuckoo hashing with γ hash functions fails with probability less than $2^{-\lambda}$. As a typical example, $\lambda = 40$ and $\gamma = 3$ yields $M \approx 1.3N_X$ [35], so we regard $M = O(N_X)$ below.

For the CPSI step, we employ the OPPRF-based protocols [36, 37, 41]. Here we simply give a rough summary of the cost, and refer to the original papers for the details [36, 37, 41]. As the first step, both parties compute and interact with each other using a special linear system solver called oblivious key-value store (OKVS) [21, 37], whose computation complexity is $O(N_Y + N_X)$ and communication complexity is $O(\kappa N_Y + \gamma \ell N_X)$. Next, the second part consists of $O(\lambda M)$ times of COT_1, whose computation cost is $O(\lambda M)$ times of ROT and communication cost is $O(\lambda M)$. This process will be executed n times in our protocol.

For the alignment step, we employ the switching network-based protocol of PnS [31]. It requires $O(M \log M)$ times of $\mathsf{COT}_{2\ell_{\mathsf{pns}}}$ where ℓ_{pns} is the bit-length of each input vector entry, whose computation cost is $O(M \log M)$ times of ROT and communication is $O(\ell M \log M)$ bits. Usually, the input vector is of the form $(B^{\mathcal{S}}[k] \| V^{\mathcal{S}}[k])$ and we have $\ell_{\mathsf{pns}} = 1 + \ell$. If there is no need to generate ID shares for some k, the input vector could be the Boolean vector $B^{\mathcal{S}}[k]$, and we have only $\ell_{\mathsf{pns}} = 1$. This process is executed n times (or $n - 1$ times considering the global index optimization).

The final share aggregation step requires n times of SSMUX calls per each row of the secret share table. As one SSMUX requires two SOT_ℓ calls, we count the computation cost by $2n$ times ROT, and the communication cost by $4\ell + 2$. As we have $M = \varepsilon N_X$ rows, the total computation cost is counted as $O(nN)$ times of ROT, and the communication cost is $O(\ell n N)$.

		CPSI	PnS	AS
Comp.	# ROTs	$O(\lambda nN)$	$O(nN \log N)$	$O(nN)$
	Other	$O(nN)$ OKVS	-	-
Comm.		$O(n(\kappa + \ell)N)$	$O(n\ell N \log N)$	$O(\ell nN)$

Table 7: Asymptotic complexities of each step (in the worst case) of our protocol, assuming $O(N) = N_Y \approx N_X$. κ is the computational security parameter, and ℓ is the bit-length of the payload.