IDCloak: A Practical Secure Multi-party Dataset Join Framework for Vertical Privacy-preserving Machine Learning

Shuyu Chen, Guopeng Lin, Haoyu Niu, Lushan Song, Chengxun Hong, Weili Han, *Member, IEEE*

***Abstract*—Vertical privacy-preserving machine learning (vPPML) enables multiple parties to train models on their vertically distributed datasets while keeping datasets private. In vPPML, it is critical to perform the secure dataset join, which aligns features corresponding to intersection IDs across datasets and forms a secret-shared and joint training dataset. However, existing methods for this step could be impractical due to: (1) they are insecure when they expose intersection IDs; or (2) they rely on a strong trust assumption requiring a non-colluding auxiliary server; or (3) they are limited to the two-party setting.**

This paper proposes `IDCloak`, the first practical secure multi-party dataset join framework for vPPML that keeps IDs private without a non-colluding auxiliary server. `IDCloak` consists of two protocols: (1) a circuit-based multi-party private set intersection protocol (cmPSI), which obtains secret-shared flags indicating intersection IDs via an optimized communication structure combining OKVS and OPRF; (2) a secure multi-party feature alignment protocol, which obtains the secret-shared and joint dataset using secret-shared flags, via our proposed efficient secure shuffle protocol. Experiments show that: (1) compared to the state-of-the-art secure two-party dataset join framework (`iPrivJoin`), `IDCloak` demonstrates higher efficiency in the two-party setting and comparable performance when the party number increases; (2) compared to the state-of-the-art cmPSI protocol under honest majority, our proposed cmPSI protocol provides a stronger security guarantee (dishonest majority) while improving efficiency by up to $7.78\times$ in time and $8.73\times$ in communication sizes; (3) our proposed secure shuffle protocol outperforms the state-of-the-art secure shuffle protocol by up to $138.34\times$ in time and $132.13\times$ in communication sizes.

***Index Terms*—Secure multi-party computation, private set intersection, secure dataset join**

I. Introduction

PRIVACY-PRESERVING machine learning (PPML) enables multiple parties to cooperatively train machine learning models on their datasets with privacy preservation. Among various PPML paradigms, multi-party vertical PPML (vPPML), where parties' datasets are vertically distributed, i.e. overlap in sample IDs but have distinct feature sets, has extensive real-world applications across multiple fields such as healthcare, finance [1]–[3]. Multi-party vPPML significantly expands the feature dimension, thereby enhancing the performance of the trained model. For instance, in a healthcare vPPML scenario, there are multiple organizations, e.g., a hospital with patients' health records, a research institute with genetic data, and an insurance company with demographics. They can apply multi-party vPPML to expand the feature dimension, and jointly train a diagnostic model on their vertically distributed datasets with privacy preservation.

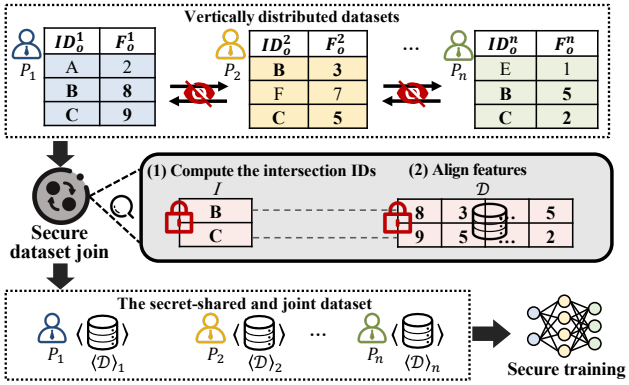


Fig. 1. Illustration of secure dataset join in vertical PPML. Multiple parties $P_1, P_2, \ldots, P_n$, sharing the intersection IDs B and C.

A critical step in vPPML is the secure dataset join. As is shown in Figure 1, this step typically involves two phases: (1) computing the intersection IDs across datasets while keeping IDs private; and (2) aligning features corresponding to these intersection IDs while keeping the datasets private. After the two phases, each party P_i ($i \geq 2$) holds a secret-shared and joint training dataset $\langle \mathcal{D} \rangle_i$ consisting of the aligned features corresponding to the intersection IDs. Then parties can train a machine learning model on $\langle \mathcal{D} \rangle_i$ with privacy preservation.

However, there remains a critical gap in providing a practical secure multi-party dataset join framework for vPPML due to at least one of the following reasons. (1) Most existing secure multi-party dataset join methods for vPPML are insecure since these methods expose the intersection IDs [4]–[7]. Specifically, exposing intersection IDs could disclose sensitive information about individuals. For example, in the healthcare scenario above, exposing intersection IDs discloses patient lists of the hospital. Besides, exposing intersection IDs could suffer from reconstruction attacks, potentially recovering the original training data, as highlighted by Jiang *et al.* [8]. (2) Although Gao *et al.* [9] propose a secure multi-party dataset join method `Peafowl` for vPPML that keeps intersection IDs private, this method relies on a strong trust assumption requiring a non-colluding auxiliary server. This assumption could be impractical in real-world multi-party vPPML settings, where such server typically does not exist. (3) While Liu *et al.* [10] propose a secure dataset joins method `iPrivJoin`

This work was supported in part by the National Natural Science Foundation of China under Grant 92370120, Grant 62172100. (*Corresponding author: Weili Han.*)

Shuyu Chen, Guopeng Lin, Haoyu Niu, Lushan Song, Chengxun Hong, and Weili Han are with the School of Computer Science, Fudan University, Shanghai 10246, China (e-mail: 23110240005@m.fudan.edu.cn; 17302010022@fudan.edu.cn; 23212010019@m.fudan.edu.cn; 19110240022@fudan.edu.cn; 22300240021@m.fudan.edu.cn; wlhan@fudan.edu.cn).

for vPPML that keeps intersection IDs private without requiring a non-colluding auxiliary server, it is limited to the two-party setting.

We note that the secure dataset join in the multi-party setting is notoriously harder to tackle than in the two-party setting: (1) the multi-party setting introduces difficulty in preserving the privacy of intersection IDs across any subset of parties; (2) the multi-party setting introduces the risk of collusion among subsets of parties, which is absent with only two parties; and (3) the multi-party setting increases considerable costs when supporting three or more parties compared to two-party settings [4].

As a result, there remains a critical challenge: *How to achieve practical secure multi-party dataset join for vPPML without a non-colluding auxiliary server?*

A. Our Approaches

To address the above challenge, we propose `IDCloak`, the first practical secure dataset join framework for multi-party vPPML that keeps IDs private without a non-colluding auxiliary server.

High-level Idea. To preserve the privacy of intersection IDs across any subset of parties while resisting collusion attacks by up to $n-1$ parties in semi-honest settings, where n is the number of parties, `IDCloak` consists of our two proposed protocols. (1) An efficient circuit-based multi-party private set intersection protocol (cmPSI), which leverages oblivious key-value store (OKVS) and oblivious pseudorandom function (OPRF) to enable each party to obtain secret-shared flags $\langle \Phi_{b\times 1} \rangle_i$ while keeping IDs private, where b is the size of the input dataset. Specifically, for $j \in b$, if the party P_1's ID vector $ID^1[j]$ is in the intersection, the plaintext flag $\Phi[j]$ is 0, and randomness otherwise. (2) A secure multi-party feature alignment protocol (smFA), which enables each party to obtain the secret-shared and joint dataset while keeping datasets private. Initially, parties use OKVS and secret sharing to obtain a secret-shared dataset consisting of aligned features and redundant data. Specifically, for $j \in [b]$, if the party P_1's ID vector $ID^1[j]$ is in the intersection, the data in the j-th row of the dataset is aligned features, and redundant data otherwise. To remove redundant data, parties execute a secure shuffle protocol on the secret-shared dataset concatenating secret-shared flags $\langle \Phi \rangle$, reconstruct shuffled $\langle \hat{\Phi} \rangle$ into plaintext flags $\hat{\Phi}$, and remove redundant data from the dataset based on whether corresponding entries in $\hat{\Phi}$ are randomness. As the secure shuffle disrupts the original order of flags $\hat{\Phi}$ and each plaintext value in $\hat{\Phi}$ is 0 or randomness, $\hat{\Phi}$ does not disclose the original IDs themselves, ensuring that no party can infer the original intersection IDs.

To enhance the efficiency while ensuring the security of our `IDCloak`, we introduce the following optimizations: (1) for optimizing the cmPSI protocol, we propose an optimized communication structure for transmitting OKVS tables, dynamically configured according to parameters, i.e. the number of parties n, dataset size m, bit length l, network bandwidth, and network latency; and (2) for optimizing the smFA protocol, we propose a novel secure shuffle protocol, which reduces the communication sizes for one party from $O(ndlm \log m)$ to $O(ndlm)$ compared to the SOTA, where d is the feature dimension. Since secure multi-party shuffle dominates the communication and time costs (over 99% in our experiments) in smFA, our novel secure shuffle protocol significantly boosts the practicality of smFA in the multi-party setting.

Overall, `IDCloak` follows a workflow similar to the SOTA two-party framework `iPrivJoin` [10]: first hashes the dataset, securely generates a secret-shared dataset that includes aligned features and redundant data, and finally removes the redundant data through a secure shuffle protocol. The main difference is that we designed a multi-party protocol for each step. Additionally, by reusing the results of OPRF and employing a more lightweight OKVS primitive compared to the oblivious programmable pseudorandom function (OPPRF) used by the `iPrivJoin`, our `IDCloak` achieves lower communication costs and fewer communication rounds, making it more efficient than `iPrivJoin` in two-party settings.

B. Contributions

Our contributions are summarized below:

- To our best knowledge, we propose the first practical secure multi-party dataset join framework for vPPML without a non-colluding auxiliary server, `IDCloak`, which preserves the privacy of intersection IDs across any subset of parties while resisting collusion attacks by up to $n-1$ parties.
- We propose two efficient protocols for `IDCloak`: (1) a cmPSI protocol, which obtains secret-shared flags indicating intersection IDs via an optimized communication structure; and (2) a secure multi-party feature alignment protocol, which obtains the secret-shared and joint dataset using the secret-shared flags via our proposed efficient secure multi-party shuffle protocol.

We evaluate `IDCloak` across various party numbers ($2 \sim 6$) using six real-world datasets, with feature dimensions between 10 and 111 and total dataset sizes between 1353 and 253680. The experimental results show that: (1) in the two-party setting, `IDCloak` outperforms the SOTA secure two-party dataset join framework `iPrivJoin` [10] by $1.69\times \sim 1.92\times$ and $1.50\times \sim 1.72\times$ in terms of time and communication sizes, respectively. Meanwhile `IDCloak` still achieves comparable efficiency to the two-party `iPrivJoin` even as the number of parties increases; (2) our proposed cmPSI protocol with a stronger security guarantee (resists against up to $n-1$ colluding parties) outperforms the SOTA cmPSI protocol [11] (resists against up to $n/2-1$ colluding parties) by $1.39\times \sim 7.78\times$ and $5.58\times \sim 8.73\times$ in terms of time and communication sizes, respectively. (3) our proposed secure multi-party shuffle protocol outperforms the SOTA secure shuffle protocol [12] by $21.44\times \sim 138.34\times$ and $105.69\times \sim 132.13\times$ in terms of time and communication sizes, respectively.

II. Related work

A. Private Set Intersection

1) Multi-party PSI (mPSI): mPSI enables multiple parties to compute the intersection of their ID sets while keeping non-

intersection IDs private. In mPSI-based secure dataset join, parties use mPSI to obtain intersection IDs in plaintext. They can then locally sort the features according to the lexicographical order of these intersection IDs and secretly share these ordered features with other parties. By merging each row of these features, the parties can obtain a secret-shared and joint dataset. Despite significant advances in efficient mPSI protocols [4], [6], [13]–[16], existing mPSI solutions usually expose intersection elements, leading to privacy leakage.

2) Circuit-based PSI (cPSI): The initial cPSI protocol introduced by Huang *et al.* [17] allows two parties to securely obtain the secret-shared intersection of their ID sets while keeping the IDs private. Subsequent research on two-party cPSI [11], [18]–[21] explores leveraging OPPRF or private set membership techniques to extend cPSI for obtaining secret-shared aligned features corresponding to intersection IDs, making it suitable for secure dataset join in vPPML. However, cPSI outputs a joint dataset that includes both features corresponding to intersection IDs and redundant data (i.e. secret-shared zeros). According to the literature [10], training on the joint dataset output by cPSI can lead to a $3\times$ increase in both training time and communication sizes, compared with training on the joint dataset without redundant data.

3) Circuit-based Multi-party PSI (cmPSI): Existing cmPSI protocols [21]–[23] enable parties to obtain secret-shared intersection IDs but do not consider obtaining aligned features corresponding to the intersection IDs.

B. Secure Two-party Dataset Join

Liu *et al.* [10] propose a secure two-party dataset join framework for vPPML, `iPrivJoin`. Similar to cPSI, `iPrivJoin` first employs the OPPRF to construct a secret-shared and joint dataset that includes both aligned features corresponding to intersection IDs and redundant data. To eliminate redundant data, `iPrivJoin` utilizes a secure shuffle protocol. However, `iPrivJoin` is inherently designed for the two-party setting.

C. Secure Multi-party Dataset Join with Non-colluding Auxiliary Server

Gao *et al.* [9] propose `Peafowl`, a secure multi-party dataset join solution, but `Peafowl` requires an auxiliary server to protect intersection IDs and assumes this auxiliary server never colludes with any parties, which is a strong trust assumption. This assumption could be impractical in real-world multi-party vPPML settings, where such server typically does not exist. In contrast to `Peafowl`, our `IDCloak` does not require a non-colluding auxiliary server, thereby achieving stronger security guarantees.

As is shown in Table I, we summarize the strengths and limitations of the representative secure dataset join methods.

TABLE I
COMPARISON OF VARIOUS SECURE DATASET JOIN METHODS. HERE, 'PARTIES' DENOTES THE NUMBER OF SUPPORTED PARTIES, 'ID-PRIVATE' DENOTES WHETHER INTERSECTION IDS ARE KEPT PRIVATE, 'NO REDUNDANT' DENOTES WHETHER REDUNDANT DATA IS AVOIDED IN THE JOINT DATASET, AND 'NO SERVER' DENOTES WHETHER A NON-COLLUDING AUXILIARY SERVER IS NOT REQUIRED. THE BETTER SETTINGS WITHIN EACH COLUMN ARE HIGHLIGHTED IN GREEN.

Method	Parties	ID-Private	No Redundancy	No Server
mPSI	$n \geq 2$	no	**yes**	**yes**
cPSI	$n = 2$	**yes**	no	**yes**
`Peafowl`	$n \geq 2$	**yes**	**yes**	no
`iPrivJoin`	$n = 2$	**yes**	**yes**	**yes**
Ours	$n \geq 2$	**yes**	**yes**	**yes**

TABLE II
NOTATION TABLE.

Symbol	Description
n	The number of parties.
$[x_1, x_2]$	The set $\{x_1, ..., x_2\}$.
$[x]$	The set $\{1, 2, ..., x\}$.
P_i	The i-th party for $i \in [n]$.
X^i	The data belonging to P_i.
ID^i	The IDs in hash table of P_i.
F^i	The features corresponding to ID^i.
I_{id}	The intersection IDs ($I_{id} = \bigcap_{i=1}^{n} ID^i$).
Φ	The flags indicating intersection IDs I_{id}.
$\mathcal{D}$	The dataset consisting of aligned features corresponding to I_{id}.
m	The size of dataset held by each party.
b	The number of bins in hash table.
d_i	The feature dimension of datasets held by P_i.
d	The total feature dimension ($d = \sum_{i=1}^{n} d_i$).
c	The size of I or $\mathcal{D}$.
h	The number of hash functions.
$\mathbb{Z}_{2^l}$	Ring of size l bits; $l = 64$ in this paper.
$\langle x \rangle_i$	The secret-shared value of $x \in \mathbb{Z}_{2^l}$ held by P_i s.t. $x = \sum_{i=1}^{n} \langle x \rangle_i$.
$x \| y$	The concatenation of x and y.
$X \| Y$	The row-by-row concatenation of X and Y.
$\|X\|$	The size of X.
$X_{a \times b}$	The matrix X with size $a \times b$.
$X[i]$	The i-th element or i-th row of X.
$X[i][j]$	The j-th element in the i-th row of X.
κ	The computational security parameter.
λ	The statistical security parameter.

III. OVERVIEW OF IDCLOAK

We summarize the frequently used notations in Table II.

A. Problem Statement

As is shown in figure 2, the secure multi-party dataset join functionality $\mathcal{F}_{smDJoin}$ takes as input from each party P_i ($i \in [n]$) a dataset $(ID_o^i \| F_o^i)$, where ID_o^i is an m-element vector of IDs and F_o^i is an $m \times d_i$ feature matrix. Here, "$\|$" indicates the row-by-row concatenation of sample IDs and their corresponding features. Once $\mathcal{F}_{smDJoin}$ is executed, each P_i obtains a secret-shared and joint dataset $\langle \mathcal{D} \rangle_i$ that consists of the aligned features corresponding to the intersection IDs.

B. Security Model

In this paper, we adopt a semi-honest security model with a dishonest majority. That is, a semi-honest adversary $\mathcal{A}$ can corrupt up to $n-1$ parties and aims to learn extra information from the protocol execution while correctly executing the protocols. We establish semi-honest security in the simulation-based model, with our construction involving multiple sub-

Functionality $\mathcal{F}_{smDJoin}$

Parameters: The number of parties n, the size of the dataset held by each party m, the feature dimension of the dataset held by each party d_i ($i \in [n]$), the total feature dimension d, and the bit length of element l.
Inputs: For each $i \in [n]$, P_i holds dataset $ID_o^i || F_o^i$.
Functionality:

1. Compute intersection $I_{id} = \bigcap_{i=1}^{n} ID_o^i$. Let $c = |I_{id}|$.
2. Set $\mathcal{D}[j] = \{F_o^1[j_1][1], \ldots, F_o^1[j_1][d_1], \ldots, F_o^n[j_n][1], \ldots, F_o^n[j_n][d_n]\}_{\forall j \in [c]}$, where $F_o^i[j_i][u]$ for $u \in [d_i]$ is the corresponding feature value of $ID[j_i] = I[j]$ from P_i for $j_i \in [m]$, $i \in [n]$.
3. Sample $\langle \mathcal{D} \rangle_i \leftarrow \mathbb{Z}_{2^l}^{c \times d}$ such that $\sum_{i=1}^{n} \langle \mathcal{D} \rangle_i = \mathcal{D}$.
4. Return $\langle \mathcal{D} \rangle_i$ to P_i for $i \in [n]$.

Fig. 2. Ideal functionality of secure dataset join.

protocols described using the hybrid model [24]. We give the formal security definition as follows:

DEFINITION 1 (Semi-honest Model)**.** *Let $view_C^{\Pi}(x, y)$ be the views (including the input, random tape, and all received messages) of C in a protocol Π, where C is the set of corrupted parties, x is the input of C and y is the input of the uncorrupted party. Let $out(x, y)$ be the protocol's output of all parties and $\mathcal{F}(x, y)$ be the functionality's output. Π is said to securely compute a functionality $\mathcal{F}$ in the semi-honest model if for any adversary $\mathcal{A}$ there exists a simulator Sim_C such that for all inputs x and y,*

$$\{view_C^{\Pi}(x, y), out(x, y)\} \approx_c \{Sim_C(x, \mathcal{F}_C(x, y)), \mathcal{F}(x, y)\}.$$

IV. PRELIMINARY

A. Cuckoo Hashing & Simple Hashing

Cuckoo hashing [25] relies on h hash functions, denoted $\{H_k : \{0,1\}^{\sigma} \mapsto [b]\}_{k \in [h]}$, to map each of m elements $\{x_j\}_{j \in [m], x_j \in \mathbb{Z}_{2^{\sigma}}}$ to *one* of h potential bins $\{H_k(x_j)\}_{\forall k \in [h]}$, where $b = \omega m$. While cuckoo hashing ensures that each bin holds at most one element, some elements may not find an available bin. Traditional cuckoo hashing employs a stash for these overflow elements. However, Pinkas *et al.* [5] introduce a stash-free variant of Cuckoo hashing that removes the need for this storage. Their findings indicate that with $h = 3$ hash functions and a table size $b = 1.27m$, the probability of failure for an element failing to find an available bin is at most 2^{-40}. We utilize these parameters to implement Cuckoo hashing without a stash.

In contrast, simple hashing uses h hash functions $\{H_k : \{0,1\}^{\sigma} \mapsto [b]\}_{k \in [h]}$ to map each of m elements $\{x_j\}_{j \in [m], x_j \in \mathbb{Z}_{2^{\sigma}}}$ into *all* bins $H_k(x_j)_{\forall k \in [h]}$. Thus, unlike Cuckoo hashing, simple hashing allows each bin to hold multiple elements.

B. Oblivious Key-Value Store

Oblivious key-value store [19] (OKVS) is designed to encode m pairs of key-value pairs where the values are random, such that an adversary cannot determine the original input keys from the encoding. This makes the encoding oblivious to the input keys. The definition is as follows:

DEFINITION 2 (Oblivious Key-Value Store)**.** *An OKVS is parameterized by a key universe $\mathcal{K}$, a value universe $\mathcal{V}$, input length m, output length m' and consists of the two functions:*

- *$\mathcal{S} = Encode(I)$: The encode algorithm receives a set of m key-value pairs $I = \{(k_i, v_i)\}_{\forall i \in [m], (k_i, v_i) \in (\mathcal{K} \times \mathcal{V})}$ and outputs the encoding $\mathcal{S} \in \mathcal{V}^{m'} \cup \{\perp\}$.*
- *$v = Decode(\mathcal{S}, k)$: The decode algorithm receives the encoding $\mathcal{S} \in \mathcal{V}^{m'}$, a key $k \in \mathcal{K}$ and outputs the associated value $v \in \mathcal{V}$.*

An OKVS is computationally oblivious [19], if for any two sets of m distinct keys $\{k_i\}_{\forall i \in [m], k_i \in \mathcal{K}}$ and $\{k_i'\}_{\forall i \in [m], k_i' \in \mathcal{K}}$ and m values $\{v_i\}_{i \in [m]}$ each drawn uniformly at random from $\mathcal{V}$, a computational adversary is not able to distinguish between $S = Encode(\{(k_i, v_i)\}_{i \in [n]})$ and $S' = Encode(\{(k_i', v_i)\}_{i \in [m]})$. Additionally, OKVS has the doubly oblivious [18]–[20] property where the output of the encoding is a uniformly random element from $\mathcal{V}^{m'}$. In addition, another critical property is random decoding, where the decoded value for any non-input key must be indistinguishable from a uniformly random element from $\mathcal{V}$. These properties are particularly useful for our `IDCloak`.

C. Oblivious Pseudorandom Function

As is shown in Figure 3, oblivious pseudorandom function (OPRF) [26] allows $\mathcal{S}$ to learn a pseudorandom function (PRF) key k, while $\mathcal{R}$ learns $F_k(x_1), \ldots, F_k(x_\beta)$ for the inputs $(x_1, \ldots, x_\beta)$. During the OPRF, neither $\mathcal{S}$ nor $\mathcal{R}$ can learn any additional information. Specifically, $\mathcal{S}$ does not learn any input $x_i (i \in [\beta])$ from $\mathcal{R}$ and $\mathcal{R}$ does not learn PRF key k.

Functionality $\mathcal{F}_{OPRF}$

Parameters: The number of elements β and the bit length of element l. A PRF $F_{(\cdot)}(\cdot)$.
Receiver's inputs: $\{x_i\}_{\forall i \in [\beta], x_i \in \{0,1\}^l}$
Funcitionality:
- Sender $\mathcal{S}$ obtains the PRF key k.
- Receiver $\mathcal{R}$ obtains $\{F_k(x_i)\}_{\forall i \in [\beta]}$.

Fig. 3. Ideal functionality of OPRF

D. Additive Secret Sharing

Additive secret sharing (ASS) [3], [27] is a cryptographic technique that splits a private value into multiple shares, allowing the original value to be reconstructed by summing these shares. It is formally defined as follows:

- **Secret-shared values**: an l-bit value x is additively secret-shared among n parties as shares $\langle x \rangle_1, \ldots, \langle x \rangle_n$, where each $\langle x \rangle_i \in \mathbb{Z}_{2^l}$ for $i \in [n]$. The value x can be reconstructed as: $\sum_{i=1}^{n} \langle x \rangle_i \equiv x \pmod{2^l}$.
- **Sharing** $Shr(x, i, Q)$: party P_i randomly samples $r_j \in \mathbb{Z}_{2^l}$ for all $j \in [|Q| - 1]$, sets its own share $\langle x \rangle_i = x - \sum_{j=1}^{|Q|-1} r_j$ and sends r_j to party $P_{Q[j]}$, who sets $\langle x \rangle_{Q[j]} = r_j$.
- **Reconstruction** $Rec(x)$: each party P_i ($\forall i \in [n]$) sends its share $\langle x \rangle_i$ to all other parties. Once all shares are gathered, each party P_i ($\forall i \in [n]$) reconstruct the original value by computing $x = \sum_{i=1}^{n} \langle x \rangle_i$.

E. Oblivious Shuffle & Secure Shuffle

As is shown in Figure 4, the oblivious shuffle allows one party P_1 who holds an matrix $X_{m\times d}$ and receives $\langle\hat{X}\rangle_1$, while the other party P_2 who holds a random permutation $\pi : [m] \to [m]$ and obtains $\langle\hat{X}\rangle_2$, ensuring that $\hat{X} = \pi(X)$. During this oblivious shuffle process, the permutation provider P_2 learns nothing about X, and the data provider P_1 learns nothing about π. In this paper, we adopt the oblivious shuffle protocol $\Pi_{\mathrm{O-Shuffle}}$ proposed in [10], which realizes $\mathcal{F}_{\textit{O-Shuffle}}$ with the same security guarantees as ours.

Functionality $\mathcal{F}_{\textit{O-Shuffle}}$

Parameters: The rows of matrix m, the columns of matrix d, the bit length of element l.
Inputs: P_1 holds X. P_2 holds a permutation $\pi : [m] \to [m]$.
Functionality:
1. Sample $\langle\hat{X}\rangle_1 \leftarrow \mathbb{Z}_{2^l}^{m\times d}$, $\langle\hat{X}\rangle_2 \leftarrow \mathbb{Z}_{2^l}^{m\times d}$ satisfying
$$\langle\hat{X}[\pi[i]]\rangle_1 + \langle\hat{X}[\pi[i]]\rangle_2 = X[i]$$
2. Return $\langle\hat{X}\rangle_1$ to P_1 and $\langle\hat{X}\rangle_2$ to P_2.

Fig. 4. Ideal functionality of oblivious shuffle

As is shown in Figure 5, the secure shuffle allows the random permutation of share matrix $\langle X\rangle$ from all parties, resulting in refreshed secret-shared matrix $\langle X'\rangle = \langle\pi(X)\rangle$, while keeping the permutation π unknown to the parties.

Functionality $\mathcal{F}_{\textit{smShuffle}}$

Parameters: The number of parties n, the rows of matrix m, the columns of matrix d, The bit length of element l.
Inputs: For each $i \in [n]$, P_i holds $\langle X_{m\times d}\rangle_i$.
Functionality:
1. Sample a random permutation $\pi : [m] \to [m]$. For $\forall i \in [n], \forall j \in [m]$, sample $\langle X'\rangle_i \leftarrow \mathbb{Z}_{2^l}^{m\times d}$ satisfying
$$\bigoplus_{i=1}^{n}\langle X'[\pi[j]]\rangle_i = \left(\bigoplus_{i=1}^{n}\langle X\rangle_i\right)$$
2. Return $\langle X'\rangle_i$ to P_i for $i \in [n]$.

Fig. 5. Ideal functionality of secure multi-party shuffle

V. Design

A. Setup Phase

During the setup phase, a randomly selected party, denoted as P_1, applies stash-less cuckoo hashing on its dataset $\textit{ID}_o^1\|F_o^1$, while the remaining parties, denoted as $P_i (i \geq 2)$ apply simple hashing on their respective dataset $\textit{ID}_o^i\|F_o^i$. Each party maps its dataset based on the unique $id \in \textit{ID}_o^i$ using h public hash functions $\{H_k : \{0,1\}^l \mapsto [b]\}_{k\in[h]}$ along with a hash table $(B_1, \ldots, B_b)$. The specific hashing operations are as follows:

- P_1 applies cuckoo hashing using h hash functions $\{H_k : \{0,1\}^l \mapsto [b]\}_{k\in[h]}$, where $b = \omega m$ and $\omega > 1$, that maps each row $(id_j^1, f_{j,1}^1, \ldots, f_{j,d_1}^1) \in \textit{ID}_o^1\|F_o^1$ for $j \in [m]$ to *one of the bins* $\{B_{H_k(id_j^1)}\}_{k\in[h]}$. After hashing, P_1 obtains $\textit{ID}^1\|F^1$, where $\textit{ID}^1[j] = (id^1\|j)$ for $j \in \{H_k(id^1)\}_{k\in[h]} \cap id^1 \in \textit{ID}_o^1$ and $F^1[j]$ is the feature matrix corresponding to $\textit{ID}^1$. Since $b > m$ (where $m = |\textit{ID}_o^1|$), P_1 fills each empty j-th bin with uniformly random values: $\textit{ID}^1[j] = r_x\|r_i$ where $r_x \in \mathbb{Z}_{2^l}$, $r_i \in \mathbb{Z}_{2^l} \setminus [b]$ and $F^1[j] = (r_1, \ldots, r_{d_1})$ where $r_1, \ldots, r_{d_1} \in \mathbb{Z}_{2^l}$.
- Each $P_i (i \geq 2)$ applies simple hashing using h hash functions $\{H_k : \{0,1\}^l \mapsto [b]\}_{k\in[h]}$ that maps each row $(id_j^i, f_{j,1}^i, \ldots, f_{j,d_i}^i) \in \textit{ID}_o^i\|F_o^i$ for $j \in [m]$ to *all bins* $\{B_{H_k(id_j^i)}\}_{k\in[h]}$. After hashing, P_i obtains $\textit{ID}^i\|F^i$, where $\textit{ID}^i[j] = \{(id^i\|j)\}_{id^i\in\textit{ID}_o^i, j\in\{H_1(id^i),\ldots,H_h(id^i)\}}$ and F^i represents the features corresponding to $\textit{ID}^i$.

In particular, for cuckoo hashing, we follow the approach in [5], [28], which guarantees that each element $x \in X$ is placed in the hash table with an overwhelming probability. After hashing, each bin in P_1's hash table holds exactly one unique transformed ID, whereas each bin in P_i's (for $i \geq 2$) hash table may contain zero or multiple transformed IDs. Furthermore, rather than storing the original ID directly, parties store its concatenation with the bin index j (i.e., $id\|j$).

B. Private ID Intersection

1) Definition: As is shown in Figure 6, functionality $\mathcal{F}_{\textit{cmPSI}}$ enables each party P_i ($i \in [n]$) to hold $\textit{ID}^i$ processed by hashing and to obtain secret-shared flags $\langle\Phi_{b\times 1}\rangle_i$. For each $j \in [b]$, $\Phi[j]$ ($= \sum_{i=1}^n\langle\Phi\rangle_i$) indicates whether $id_j (= \textit{ID}^1[j])$ is an element in intersection IDs I_{id}. In particular, $\Phi[j]$ is 0 if $id_j \in I_{id}$. Or, $\Phi[j]$ is a randomness if $id_j \notin I_{id}$.

Functionality $\mathcal{F}_{\textit{cmPSI}}$

Parameters: The number of parties n, the size of dataset b, the bit length of element l.
Inputs: For each $i \in [n]$, P_i holds dataset $\textit{ID}^i$ with size b.
Functionality:
1. Compute $I_{id} = \cap_{i=1}^n(\textit{ID}^i)$.
2. For each $j \in [b]$, if $\textit{ID}^1[j] \in I_{id}$, set $\Phi[j] = 0$, else sample $\Phi[j]$ uniformly from $\mathbb{Z}_{2^l}$.
3. Sample $\langle\Phi\rangle_i \leftarrow \mathbb{Z}_{2^l}^{b\times 1}$ satisfying $\Phi = \sum_{i=1}^n\langle\Phi\rangle_i$.
4. Return $\langle\Phi\rangle_i$ to P_i for $i \in [n]$.

Fig. 6. Ideal functionality of circuit-based multi-party PSI

To illustrate the core concept, we first present our proposed cmPSI protocol in a ring-based communication structure. We then demonstrate how substantial performance gains can be achieved by optimizing the communication structure while maintaining security guarantees.

2) cmPSI with Ring-based Communication Structure: Holding $\textit{ID}^1$ or $\textit{ID}^i$ (with size b) processed by hashing, each pair P_1 and P_i ($i \in [2, n]$) invokes an OPRF instance. Specifically, in each OPRF instance, P_1 is the receiver and gets $y_j^i = F_{k_i}(\textit{ID}^1[j])$ for $\forall j \in [b]$, while $P_i (i \geq 2)$ is the sender, gets the PRF key k_i and computes PRF values $U^i[j] = \{F_{k_i}(\textit{ID}^i[j][u])\}_{\forall u\in[|\textit{ID}^i[j]|]}$ for $\forall j \in [b]$. Subsequently, parties adopt a ring-based communication structure to iteratively transmit the OKVS tables from P_n to P_1 while preserving privacy. Concretely, each party $P_i (i \geq 2)$ generates independent randomness $r_j^i \in \mathbb{Z}_{2^l}$ for $\forall j \in [b]$. The transmission begins with P_n, who encodes the key-value pairs $I^n = \{(\textit{ID}^n[j][u], U^n[j][u] + r_j^n)\}_{\forall j\in[b], \forall u\in[|\textit{ID}^n[j]|]}$ into an OKVS table $S^n = \textit{Encode}(I^n)$. In our construction, each row of PRF values $U^n[j]$ (for row $j \in [b]$) is masked by the same random value r_j^n. Crucially, due to the OPRF interaction

between P_1 and P_n, party P_1 can learn at most one PRF output per row j, if and only if $ID^1[j]$ matches $id^n_{j,u} \in ID^n[j]$. Consequently, values $\{U^n[j][u] + r^n_j)\}_{j\in[b],u\in[|U^n[j]|]}$ are indistinguishable from uniformly random. Furthermore, under the obliviousness guarantees of the OKVS, no other party gains additional information from $S^n = Encode(I^n)$. Thus P_n can sends S^n to P_{n-1} with privacy preservation. After receiving S^n, P_{n-1} decodes S^n to retrieve $V^{n-1}[j] = \{v^i_{j,u} = Decode(S^n, ID^{n-1}[j][u])\}_{\forall u\in[|ID^{n-1}[j]|]}$ for $\forall j \in [b]$, then constructs new key-value pairs $I^{n-1} = \{(ID^{n-1}[j][u], U^{n-1}[j][u] + V^{n-1}[j][u] + r^{n-1}_j)\}_{\forall j\in[b],\forall u\in[|ID^{n-1}[j]|]}$. P^{n-1} then encodes new key-value pairs into the OKVS table $S^{n-1} = Encode(I^{n-1})$ and sends S^{n-1} to the next party P_{n-2}. By the same obliviousness property, other parties cannot infer additional information from S^{n-1}. This process is repeated iteratively from P_{n-1} to P_2, with each P_i $(i \geq 2)$ receiving an OKVS table S^{i+1}, decoding it to retrieve the intermediate results V^i, constructing new key-value pairs $I^i = \{(ID^i[j][u], U^i[j][u] + V^i[j][u] + r^i_j)\}_{\forall j\in[b],\forall u\in[|ID^i[j]|]}$, encoding the OKVS table $S^i = Encode(I^i)$ and sending S^i to the next party P_{i-1}. Finally, upon receiving S^2 from P_2, P_1 decodes S^2 to retrieve $V^1 = \{v^1_j = Decode(S^2, ID^1[j])\}_{\forall j\in[b]}$ and computes its output $\langle\Phi_{b\times 1}\rangle_1$ by subtracting V^1 from the sum of all received y^i_j values, that is $\langle\Phi\rangle_1 = (\sum_{i=2}^n y^i_1 - V^1[1], \ldots, \sum_{i=2}^n y^i_b - V^1[b])$. Each other party $P_i(\forall i \geq 2)$ sets its output $\langle\Phi\rangle_i = (r^i_1, r^i_2, \ldots, r^i_b)$.

The key security points of the cmPSI protocol are summarized as follows: (1) using independent random masks per bin for secret sharing: for each j ($j \in [b]$), an independent random mask r^i_j is introduced into the key-value pairs I^i used by P_i $(i \geq 2)$ when encoding the OKVS table. This ensures that each party obtains secret-shared flags indicating intersection elements without exposing the underlying IDs; (2) storing transformed IDs $id\|j$ in ID^i and using OPRF on ID^i to prevent collusion-based brute force attacks: adding the PRF value of $id\|j$ within the values of key-value pairs I^i, and P_1 invokes OPRF with each other party P_i $(i \geq 2)$ to remove the PRF values of intersection IDs, preventing colluding parties from inferring other parties' ID values. For example, if IDs store the original IDs instead of $id\|j$, consider parties P_1, P_2, and P_3, where P_1 holds IDs a, e, P_2 holds ID a, and P_3 holds IDs a, e, with IDs a and e hashed into the same bin for P_3. If P_1 and P_2 collude, they can infer whether P_3 has e by verifying that $Decode(S^3, a) - F_{k_3}(a)$ equals $Decode(S^3, e) - F_{k_3}(e)$. However, by using OPRF on the concatenated value $id\|j$, the above attack is effectively prevented, since each hash table bin of P_1 has only one element, P_1 obtains $F_{k_3}(a\|j)$ and $F_{k_3}(e\|j')$ for $j' \neq j$ but does not obtain $F_{k_3}(e\|j)$.

3) cmPSI with Optimized Communication Structure: Based on our earlier security analysis, an adversary cannot reconstruct the original IDs from an encoded OKVS table via brute-force attacks. Therefore, it is secure to transmit OKVS tables to any party, enabling efficient parallel transmissions that ultimately converge at P_1.

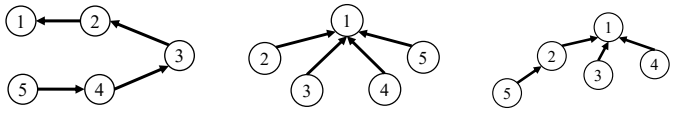
As is shown in Figure 7, a scenario for cmPSI involving 5 parties. The leftmost diagram in Figure 7 illustrates a ring-based communication structure for transmitting OKVS tables

Protocol 1: Π_{cmPSI}

Parameters: The number of parties n, the size of hash table b, the bit length of element l, the security parameter κ and the statistical security parameter λ. An OKVS scheme $Encode(\cdot)$, $Decode(\cdot,\cdot)$. An OPRF functionality $\mathcal{F}_{oprf}$. A PRF $F_{(\cdot)}(\cdot)$.
Inputs: P_i holds ID^i for $i \in [n]$.
Outputs: P_i obtains $\langle\Phi\rangle_i$, such that if $ID^1[j] \in I_{id}(\forall j \in [b])$, $\Phi[j]$ is zero; otherwise $\Phi[j]$ is randomness.
Offline: $P_i(i \geq 2)$ samples uniformly random and independent values $(r^i_1, r^i_2, \ldots, r^i_b) \in \mathbb{Z}_{2^l}$.
Online:
1. **Performing OPRF:**
   1) For $i \in [2, n]$, each (P_1, P_i) pair invokes an instance of $\mathcal{F}_{OPRF}$, where:
      - $P_i(i \geq 2)$ is the sender and gets PRF key k_i.
      - P_1 is the receiver, inputs ID^1 and receives $Y^i = (y^i_1, \cdots, y^i_b)$ where $y^i_j = F_{k_i}(ID^1[j])$ for $\forall j \in [b]$.
   2) For $i \in [2, n]$, each P_i computes $U^i[j] = \{F_{k_i}(ID^i[j][u])\}_{\forall u\in[|ID^i[j]|]}$ for $\forall j \in [b]$.
2. **Transmiting OKVS tables:**
   1) Each $P_i(i \in [n])$ uses the optimized communication structure generation algorithm1 to obtain the set of child parties $P_i.children$ and its parent party $P_i.parent$ (the child party sends the OKVS table to the parent party).
   2) For $i = n$ to 2:
      i) If $P_i.children \neq \emptyset$, P_i receives $\{S^{i^*}\}_{\forall i^*\in P_i.children}$ from all child parties and gets $V^{i,i^*} = \{v^{i,i^*}_{j,u} = Decode(S^{i^*}, ID^i[j][u])\}_{\forall j\in[b],\forall u\in[|ID^i[j]|]}$ for $\forall i^* \in P_i.children$.
      ii) If $i \geq 2$, P_i sets $I^i = \{(ID^i[j][u], \sum_{i^*\in P_i.children} V^{i,i^*}[j][u] + U^i[j][u] + r^i_j)\}_{\forall j\in[b],\forall u\in[|ID^i[j]|]}$. P_i builds OKVS table $S^i = Encode(I^i)$ and sends S^i to $P_i.parent$.
3. **Setting outputs:**
   - P_1 sets $\langle\Phi\rangle_1 = (\sum_{i=2}^n y^i_1 - V^1[1], \ldots, \sum_{i=2}^n y^i_b - V^1[b])$.
   - $P_i(i \geq 2)$ sets $\langle\Phi\rangle_i = (r^i_1, r^i_2, \ldots, r^i_b)$.

in our proposed cmPSI protocol. We denote the transmission time for an OKVS table (excluding latency) as t_s and the network latency as t_l. The total time required to transmit all the OKVS tables via a ring-based structure is $4(t_s + t_l)$, assuming that the local encoding and decoding times are negligible. The middle part of Figure 7 illustrates the simplest star-based parallel structure, in which every party $P_i(i \geq 2)$ sends its OKVS table S^i to P_1. The total time required to transmit all the OKVS tables via the star-based structure is $4t_s + t_l$, which is $(n-2)t_l$ less than that of the ring-based structure. We notice that if P_1 first receives P_4's OKVS table, then parties P_2,P_3, and P_5 remain idle. As is shown in the rightmost part of Figure 7, this idle time can be effectively leveraged by having P_5 send its OKVS table to P_2. P_2 subsequently decodes the OKVS table S^5 to retrieve $V^{2,5}[j] = \{v^{2,5}_{j,u} = Decode(S^5, U^2[j][u])\}_{\forall u\in[|U^2[j]|]}$ for $\forall j \in [b]$, and add the retrieved values into values of its own key-value pairs, constructing new pairs $I^2 = \{(ID^2[j][u], U^2[j][u] + V^{2,5}[j][u] + r^2_j)\}_{\forall j\in[b],\forall u\in[|ID^2[j]|]}$. P_2 encodes the OKVS table $S^2 = Encode(I^2)$ and sends S^2 to party P_1. As all parties possess datasets of equal size, P_1 and P_2 simultaneously receive the OKVS table from P_4 and P_5, respectively. Subsequently, P_1 receives the OKVS tables S^3 and S^2 from P_3 and P_2, respectively. The time required to

transmit all the OKVS tables via this optimized structure is $3t_s + t_l$, which is t_s less than that of the star-based structure.



(a) Ring-based structure (b) Star-based structure (c) Optimized structure

Fig. 7. Comparison of different communication structures for transmitting OKVS tables. Each circle denotes a party and the number i in the circle denotes P_i ($i \in [n]$). Arrow indicates the transmission direction of the OKVS table. Each party must finish receiving the OKVS tables before sending.

We propose an optimized communication structure generation algorithm (detailed in Algorithm 1 in the Appendix) that takes as input the number of parties n, the number of designated root party $leader\_num$ ($leader\_num$ is 1 in our proposed cmPSI protocol), the time required to send one OKVS table t_s, and the network delay t_l. It outputs the root node of the optimized communication structure, where each node represents a party and contains its identifier, parent, and children. Each party must finish receiving all the OKVS tables from its child party before it sends the OKVS table to the parent party. The algorithm follows a greedy approach to maximize communication utilization while minimizing the latency for each party. As is shown in Protocol 1, we construct the final cmPSI protocol by using this optimized communication structure to efficiently transmit the OKVS tables in step 2.

THEOREM 1. *The protocol Π_{cmPSI} (Protocol 1) securely realizes $\mathcal{F}_{cmPSI}$ (Figure 6) in the random oracle and $\mathcal{F}_{OPRF}$-hybrid model, against semi-honest adversary $\mathcal{A}$ who can corrupt $n-1$ parties.*

Proof. We exhibit simulators Sim_C for simulating the view of the corrupt parties including P_1 and excluding P_1, respectively, and prove that the simulated view is indistinguishable from the real one via standard hybrid arguments. Let C be the set of corrupted $n-1$ parties.

Case 1 ($P_1 \in C$). The Sim_C samples randomness $y'_j \leftarrow \mathbb{Z}_{2^l}$ for $\forall j \in [b]$ and invokes the OPRF receiver's simulator $Sim^R_{OPRF}(Y^i, Y'^i)$, where $Y'^i = \{y'_1, \ldots, y'_b\}$ and sends Y'^i to P_1 on behalf of $P_i(\notin C)$. If $P_i(\notin C)$ has child parties, Sim_C receives the OKVS table from P_i's child parties. Next, Sim_C samples uniformly random OKVS table $S'^i \leftarrow \mathbb{Z}_{2^l}^{m' \times 1}$ where m' is the size of the OKVS table when encoding $h \cdot m$ elements. And Sim_C sends S'^i to $P_i(\notin C)$'s parent party. We argue that the view output by Sim_C is indistinguishable from the real one. First, we define three hybrid transcripts T_0,T_1,T_2, where T_0 is the real view of C, and T_2 is the output of Sim_C.

1. $Hybrid_0$.The first hybrid is the real interaction in Protocol 1. Here, an honest party P_i uses real inputs and interacts with corrupt parties C. Let T_0 denote the real view of C.
2. $Hybrid_1$. Let T_1 be the same as T_0, except that the OPRF execution is replaced by the OPRF receiver's simulator. $Sim^R_{OPRF}(Y^i, Y'^i)$, where Y'^i has b elements $\{y'_1, \ldots, y'_b\}$. The simulator security of OPRF guarantees that this view is indistinguishable from T_0.
3. $Hybrid_2$. Let T_2 be the same as T_1, except that the OKVS table S'^i for $P_i(\notin C)$ is sampled uniformly from $\leftarrow \mathbb{Z}_{2^l}^{m' \times 1}$. In real execution, $P_i(\notin C)$ sets $I^i = \{(ID^i[j][u], \sum_{P_{i^*} \in P_i.children} V^{i^*}[j][u] + U^i[j][u] + r^i_j)\}_{\forall j \in [b], \forall u \in [|ID^i[j]|]}$ and builds the OKVS table $S^i = Encode(I^i)$. If $x = ID^i[j][u]$ is not the intersection element of P_1 and $P_i(\notin C)$, the value $I^i(x)$ is uniformly random from $\mathbb{Z}_{2^l}$ because it contains the PRF value $U^i[j][u] = F_{k_i}(x)$. If x is the intersection element of P_1 and P_i, P_1 holds $U^i[j][u] = F_{k_i}(x)$, but r^i_j is a uniform sample by P_i, so the values $I^i(x)$ are still uniformly at random from $\mathbb{Z}_{2^l}$. Owing to the double obliviousness and random decoding property of the OKVS, the simulated S'^i has the same distribution as S in the real protocol. Hence, T_1 and T_2 are statistically indistinguishable. This hybrid is exactly the view output by the simulator.

Case 2 ($P_1 \notin C$). For each $P_i(i \geq 2)$, Sim_C generates random key k_i. It then invokes the OPRF sender's simulator $Sim^S_{OPRF}(k_i)$ and appends the output to the view. Sim_C receives the OKVS table from P_1's children parties on behalf of P_1. We argue that the view output by Sim_C is indistinguishable from the real one. We first define two hybrid transcripts T_0,T_1, where T_0 is the real view of C, and T_1 is the output of Sim_C.

1. $Hybrid_0$.The first hybrid is the real interaction in Protocol 1. Here, an honest party P_i uses real inputs and interacts with the corrupt parties C. Let T_0 denote the real view of C.
2. $Hybrid_1$. Let T_1 be the same as T_0, except that the OPPF execution is replaced by the OPRF sender's simulator $Sim^S_{OPRF}(k_i)$ for every $i \in [2, n]$. The simulator security of OPRF guarantees this view is indistinguishable from T_0. This hybrid is exactly the view output by the Sim_C.

□

C. Secure Feature Alignment

1) Definition: We formally define secure feature alignment functionality $\mathcal{F}_{smFA}$ in Figure 8. Using the secure feature alignment protocol, each party $P_i(i \in [n])$ holds $(\langle\Phi\rangle_i, ID^i, F^i)$ and obtains secret-shared and joint training dataset $\langle\mathcal{D}\rangle_i$ consisting of aligned features.

Functionality $\mathcal{F}_{smFA}$

Parameters: The number of parties n, the bit length of element l, the feature dimension of dataset held by each party d_i ($i \in [n]$).
Inputs: For each $i \in [n]$, P_i holds $(\langle\Phi\rangle_i, ID^i, F^i)$.
Functionality:
1. Initialize empty $\mathcal{D}$.
2. For $j = 0$ to b:
   If $\Phi[j] == 0$: $\mathcal{D}$.append($F^1[j][1], \ldots, F^1[j][d_1], \ldots, F^n[j][j_n][1], \ldots, F^n[j][j_n][d_n]$) for $j_i \in [|ID^i[j]|]$, where $ID^i[j][j_i] = ID^1[j]$.
3. Return $\langle\mathcal{D}\rangle_i$ to P_i for $i \in [n]$.

Fig. 8. Ideal functionality of secure multi-party feature alignment

2) Construction: Protecting intersection IDs in `IDCloak` introduces a significant challenge because without revealing which IDs are in the intersection, parties cannot directly determine which features need to be aligned or how they should be ordered. To address this issue, the smFA protocol

has two milestones. First, parties use OKVS and secret sharing to obtain a secret-shared dataset consisting of aligned features and redundant data. Second, parties use a secure multi-party shuffle to remove redundant data corresponding to non-intersection IDs, thus yielding a secret-shared and joint dataset consisting of aligned features. The smFA protocol Π_{smFA} is described formally in Protocol 3.

The first milestone corresponds to steps 1–3 of Π_{smFA} (Protocol 3). In the offline phase, each party P_i ($i \geq 2$) samples a matrix of uniformly random and independent values $R^i_{b\times d_i} = [r^i_{j,k}]_{\forall j\in[b],\forall k\in[d_i],r^i_{j,k}\in\mathbb{Z}_{2^l}}$. In the online phase, P_1 secret shares its feature matrix F^1 with all other parties P_i ($i \geq 2$) (step 1). Next, each P_i ($i \in [2,n]$) secret shares R^i among all parties except P_1, denoted as $\langle R^i\rangle = Shr(R^i, i, [n] \setminus \{1\})$ (step 2-(1)). Subsequently, each party $P_i(i \geq 2)$ constructs an OKVS table $S^i_f = Encode(I^i_f)$, where $I^i_f = \{(U^i[j][u], H_o(U^i[j][u]) \oplus ((F^i[j][u][1] - r^i_{j,1})\| \ldots \|(F^i[j][u][d_i] - r^i_{j,d_i})))\}_{\forall j\in[b],\forall u\in[|ID^i[j]|]}$ (step 2-(2)). Here, $U^i = F_{k_i}(ID^i)$ is P_i's PRF value (obtained from Π_{cmPSI}), and $H_o : \{0,1\}^\kappa \to \{0,1\}^*$ is public random oracles whose output bit length '$*$' is $l \cdot d_i$. Each $P_i(i \geq 2)$ sends the OKVS table S^i_f to P_1 (step 2-(2)). After receiving S^i_f, P_1 encodes S^i_f using the OPRF values Y^i (obtained from Π_{cmPSI}) to retrieve $V^i_f = \{v^i_j = Decode(S^i_f, Y^i[j])\}_{\forall j\in[b]}$ (step 2-(3)). P_1 XORs these result with $H_o(Y^i)$ to derive masked features $V^i_f = V^i_f \oplus H_o(Y^i)$. In step 3, P_1 sets $\langle Z\rangle_1 = (\langle F^1\rangle_1, V^2_f, V^3_f, \ldots, V^n_f)$, while each P_i ($i > 2$) sets $\langle Z\rangle_i = (\langle F^1\rangle_i, \langle R^2\rangle_i, \ldots, \langle R^n\rangle_i)$. Through these steps, the parties collectively obtain a secret-shared dataset $\langle Z\rangle$ that includes aligned features and redundant data.

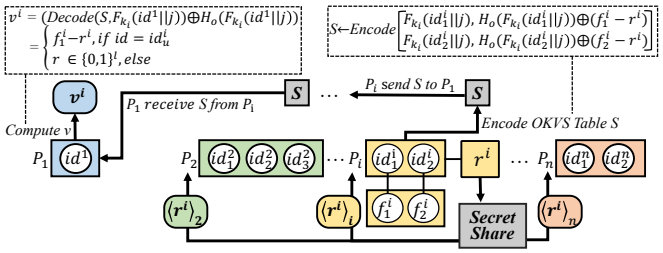


Fig. 9. Examples of sharing features of party $P_i(i \geq 2)$ in j-th bin (index j omitted for simplicity) with privacy-preserving. P_i's bin contains two ids id^i_1, id^i_2 and the corresponding features are f^i_1, f^i_2. P_i samples a random value r^i locally and secretly shares r^i to $P_j(\forall j \in [2,n])$, who each obtain share $\langle r^i\rangle_j$. P_i sends the OKVS table S to P_1, who decodes S and computes v^i. The dashed boxes highlight computations corresponding directly to the adjacent text, and ellipses indicate outputs held by each party.

To analyze the correctness of the above steps, consider the j-th bin in Figure 9. For simplicity, we omit the index j. Party P_i's dataset has a single feature dimension, containing two ID values id^i_1, id^i_2 in the j-th bin. If $id^1 = id^i_u$ for some $u \in [|ID^i[j]|]$, P_1 computes

$$\begin{aligned} v^i &= (Decode(S, F_{k_i}(id^1\|j)) \oplus (H_o(F_{k_i}(id^1)) \\ &= (Decode(S, F_{k_i}(id^i_u\|j)) \oplus (H_o(F_{k_i}(id^i_u)) \\ &= H_o(F_{k_i}(id^i_u)) \oplus (f^i_u - r^i) \oplus (H_o(F_{k_i}(id^i_u)) \\ &= f^i_u - r^i \end{aligned} \tag{1}$$

Adding the shares $\langle r^i\rangle$(held by $P_2, \ldots, P_n$) to v^i (held by P_1) reconstructs f^i_u, meaning that all parties hold a share $\langle z\rangle^1$ of P_i's feature f^i_u for intersection ID. If $id^1 \neq id^i_u$, the decoded output is a random value. Adding shares $\langle r^i\rangle$ (held by $P_2, \ldots, P_n$) to v^i reconstructs a random value, implying that each party obtains a secret-shared random value $\langle z\rangle$. Thus, all parties obtain $\langle Z_{b\times d}\rangle_i$, where if $ID^1[j] \in I$, the j-th row of $\langle Z\rangle$ consists of secret-shared aligned features from all parties corresponding to this ID; otherwise, redundant data.

Regarding security, note that for the key-value pairs I^i_f encoded into P_i's OKVS table S^i_f ($i \geq 2$), party P_i set $U^i = F_{k_i}(id^i_u)$ as the keys of I^i_f. And P_i mask each feature value using a random number and $H_o(U^i)$, thus forming the values $\{H_o(U^i[j][u]) \oplus ((F^i[j][u][1] - r^i_{j,1})\| \ldots \|(F^i[j][u][d_i] - r^i_{j,d_i}))\}_{\forall j\in[b],\forall u\in[|ID^i[j]|]}$ of I^i_f. This design prevents P_1 from inferring whether any decoded entry from S^i_f is a random value or a masked feature. Moreover, since P_i uses OKVS to encode the entire dataset F^i across the simple hash table, S^i_f does not reveal the number of elements in each hash bin.

For the second milestone for removing redundant data in step 4 of Π_{smFA}, parties employ our proposed secure shuffling protocol (Protocol 2) on $\langle Z\rangle\|\langle\Phi\rangle$, reconstruct shuffled flags $\langle\hat{\Phi}\rangle$ into plaintext $\hat{\Phi}$, and then remove redundant data from $\langle Z\rangle$ if the corresponding entries in $\hat{\Phi}$ are randomness.

Protocol 2: $\Pi_{\mathrm{smShuffle}}$

Inputs: P_i holds a secret-shared matrix $\langle X\rangle_i$ with size $b \times d$.
Outputs: P_i obtains a shuffled secret-shared matrix $\langle X'\rangle_i$, such that $X' = \pi(X)$. where $\pi : [b] \to [b]$ is an unknown permutation to all parties.
Offline: $P_i(\forall i \in [n])$ samples uniformly random independent values $R^{i,i'} = [r^{i,i'}_{j,u}]_{\forall j\in[b],\forall u\in[d],r^{i,i'}_{j,u}\in\mathbb{Z}_{2^l}}$ for $\forall i' \in [n] \setminus \{i\}$) and a random permutation $\pi^i : [b] \to [b]$. Each pair of P_i and $P_{i'}(\forall i, i' \in [n], i \neq i')$ invokes $\Pi_{\mathrm{O-Shuffle}}$:
- P_i inputs π^i and obtains $\langle\hat{R}^{i',i}\rangle_i$.
- $P_{i'}$ inputs $R^{i',i}$ and obtains $\langle\hat{R}^{i',i}\rangle_{i'}$.

Online:
1. $P_i(\forall i \in [n])$ sets $\langle\hat{X}\rangle_i = \langle X\rangle_i$.
2. For $i = 1$ to n:
   1) $P_{i'}$ $(\forall i' \in [n] \setminus \{i\})$ computes $W^{i',i} = \langle\hat{X}\rangle_{i'} - R^{i',i}$ and sends $W^{i',i}$ to P_i. $P_{i'}$ updates $\langle\hat{X}\rangle_{i'} = \langle\hat{R}^{i',i}\rangle_{i'}$.
   2) Upon receiving all $W^{i',i}$ from $P_{i'}$ ($i' \in [n] \setminus \{i\}$), P_i computes $W^i = \sum_{i'\in[n]\setminus\{i\}} W^{i',i} + \langle\hat{X}\rangle_i$. Then, P_i applies its permutation π^i to obtain $\hat{W}^i = \pi^i(W^i)$ and updates $\langle\hat{X}\rangle_i = \hat{W}^i + \sum_{i'\in[n]\setminus\{i\}}\langle\hat{R}^{i',i}\rangle_i$.

In our proposed secure multi-party shuffle $\Pi_{\mathrm{smShuffle}}$ (Protocol 2), each party $P_i(i \in [n])$ samples a private random permutation π^i and generates random masks $R^{i,i'}(i' \in [n] \setminus \{i\})$ for each other party in the offline phase. Then, through a two-party protocol $\Pi_{\mathrm{O-Shuffle}}$, parties convert these masks into secret-shared, permuted forms $\langle\hat{R}^{i,i'}\rangle(i' \in [n] \setminus \{i\})$. In the online phase, the parties iteratively update their shares in n rounds. In each round, a designated party P_i ($i \in [n]$) collects masked inputs $W^{i',i} = \langle\hat{X}\rangle_{i'} - R^{i',i}$ from all others $P_{i'}(i' \in [n] \setminus \{i\})$, applies its private permutation π^i to the aggregated value

[1]Here, v^i is $V^i_f[j]$, z is $Z[j][\sum_{i'=1}^{i-1} d_{i'}]$, r^i is r^i_j, id^i is id^i_j, as we omit j for simplicity.

$\hat{W}^i = \pi^i(W^i) = \pi^i(\sum_{i' \in [n] \setminus \{i\}} W^{i',i} + \langle \hat{X} \rangle_i)$, and updates its share $\langle \hat{X} \rangle_i = \hat{W}^i + \sum_{i' \in [n] \setminus \{i\}} \langle \hat{R}^{i',i} \rangle_i$, while the others $P_{i'}$ update their shares $\langle \hat{X} \rangle_{i'} = \langle \hat{R}^{i',i} \rangle_{i'}$. After all rounds, the joint output is a secret-shared matrix, which is a randomly shuffled version of the original input, with the shuffle pattern and data remaining completely hidden from all the parties.

Since the secure shuffle disrupts the original order of flags $\hat{\Phi}$ and each plaintext entry in $\hat{\Phi}$ is either 0 or randomness, ensuring that no party can infer the original intersection IDs from $\hat{\Phi}$. Then, the parties can remove redundant data: if $\hat{\Phi}[j] = 0$, it indicates that the corresponding row $\langle Z[j] \rangle$ belongs to an intersection ID, and thus $\langle Z[j] \rangle$ is appended in final secret-shared dataset $\langle \mathcal{D} \rangle$. Finally, each party P_i obtains the secret-shared and joint dataset $\langle \mathcal{D}_{c \times d} \rangle_i$ consisting of secret-shared and aligned features corresponding to the intersection IDs I_{id}.

Protocol 3: Π_{smFA}

Parameters: The number of parties n, the size of input dataset b, the bit length of element l, and the feature dimension d_i for party $P_i (i \in [n])$, the security parameter κ and the statistical security parameter λ. An OKVS scheme *Encode*$(\cdot)$, *Decode*$(\cdot,\cdot)$. Random oracles $H_o : \{0,1\}^\kappa \to \{0,1\}^*$.
Inputs: P_i holds $(\langle \Phi \rangle_i, ID^i, F^i)$ for $i \in [n]$. P_1 holds OPRF values $\{Y^i = F_{k_i}(ID^1)\}_{\forall i \in [2,n]}$ and $P_i (i \geq 2)$ holds PRF values $U^i = F_{k_i}(ID^i)$ from Π_{cmPSI}.
Outputs: P_i obtains $\langle \mathcal{D}_{c \times d} \rangle_i$ consisting of aligned features.
Offline:
- Each $P_i (i \geq 2)$ uniformly and independently samples each element of matrix $R^i_{b \times d_i} = [r^i_{j,k}]_{\forall j \in [b], \forall k \in [d_i], r^i_{j,k} \in \mathbb{Z}_{2^l}}$.
- $P_i (\forall i \in [n])$ executes offline operation in $\Pi_{\text{smShuffle}}$.

Online:
1. **Sharing features of** P_1**:** $\langle F^1 \rangle = Shr(F^1, 1, [n])$.
2. **Sharing features of** $P_i (i \geq 2)$**:**
   1) For $i \in [2,n]$, $\langle R^i \rangle = Shr(R^i, i, [n] \setminus \{1\})$.
   2) $P_i (i \geq 2)$ builds OKVS table $S^i_f = Encode(I^i_f)$, where $I^i_f = \{(U^i[j][u], H_o(U^i[j][u]) \oplus ((F^i[j][u][1] - r^i_{j,1}) \| \ldots \| (F^i[j][u][d_i] - r^i_{j,d_1})))\}_{\forall j \in [b], \forall u \in [|U^i[j]|]}$ and sends S^i_f to P_1.
   3) After receiving S^i_f from $P_i (i \geq 2)$, P_1 gets $V^i_f = \{v^i_j = Decode(S^i_f, Y^i[j])\}_{\forall j \in [b]}$ and computes $V^i_f = V^i_f \oplus H_o(Y^i)$ for $\forall i \in [2,n]$.
3. **Merging features:**
   - P_1 sets $\langle Z \rangle_1 = (\langle F^1 \rangle_1, V^2_f, V^3_f, \ldots, V^n_f)$.
   - For $i \geq 2$, P_i sets $\langle Z \rangle_i = (\langle F^1 \rangle_i, \langle R^2 \rangle_i, \ldots, \langle R^n \rangle_i)$.
4. **Removing redundant data:**
   1) $\langle \hat{Z} \rangle \| \langle \hat{\Phi} \rangle = \Pi_{\text{smShuffle}}(\langle Z \rangle \| \langle \Phi \rangle)$.
   2) $\hat{\Phi} = Rec(\langle \hat{\Phi} \rangle)$.
   3) For $j = 0$ to b: If $\hat{\Phi}[j] == 0$: $\langle \mathcal{D} \rangle$.append$(\langle \hat{Z}[j] \rangle)$.

THEOREM 2. *The protocol* Π_{smFA} *(Protocol 3) is securely realized* $\mathcal{F}_{smFA}$ *in random oracle model, against a semi-honest adversary* $\mathcal{A}$ *who can corrupt* $n-1$ *parties.*

Proof. We exhibit simulators *Sim*$_C$ for simulating the view of the corrupt parties, including P_1 and excluding P_1, respectively, and prove that the simulated view is indistinguishable from the real one via standard hybrid arguments. Let C be the set of corrupted $n-1$ parties.

Case 1 ($P_1 \in C$). The corrupted parties obtain shares in steps 1, 2-(1), and 4, thus *Sim*$_C$ can pick shares of random value on behalf of P_i ($\notin C$) in these steps. *Sim*$_C$ samples uniformly random OKVS table $S'^i \leftarrow \mathbb{Z}_{2^l}^{m' \times d_i}$ where m' is the size of OKVS table when encoding $h \cdot m$ number of elements. And *Sim*$_C$ sends S'^i to P_1 in step 2-(2). We argue that the view output by *Sim*$_C$ is indistinguishable from the real one. We first define three hybrid transcripts T_0,T_1,T_2, where T_0 is the real view of C, and T_2 is the output of *Sim*$_C$.

1. *Hybrid*$_0$.The first hybrid is the real interaction in Protocol 3. Here, an honest party P_i uses real inputs and interacts with the corrupt parties C. Let T_0 denote the real view of C.
2. *Hybrid*$_1$. Let T_1 be the same as T_0, except that the OKVS table S'^i for $P_i (\notin C)$ is sampled uniformly from $\leftarrow \mathbb{Z}_{2^l}^{m' \times d_i}$. By the double obliviousness and random decoding property of OKVS, the simulated S'^i has the same distribution as S in the real protocol. Hence, T_1 and T_0 are statistically indistinguishable.
3. *Hybrid*$_2$. Let T_1 be the same as T_0, except that the secret sharing operation is replaced by the secret share simulator. The underlying secret sharing guarantee T_2 is indistinguishable from T_1.This hybrid is exactly the view output by the simulator *Sim*$_C$.

Case 2 ($P_1 \notin C$). The corrupted parties obtain shares in steps 1, 2-(1), and 4, thus the *Sim*$_C$ can pick shares of random value to the view on behalf of P_1 in these steps. *Sim*$_C$ receive OKVS table from P_i on behalf of P_1 in step 2-(3). We argue that the view output by *Sim*$_C$ is indistinguishable from the real one. We first define two hybrid transcripts T_0, T_2, where T_0 is the real view of C, and T_1 is the output of *Sim*$_C$.

1. *Hybrid*$_0$.The first hybrid is the real interaction in Protocol 3. Here, an honest party P_i uses real inputs and interacts with the corrupt parties C. Let T_0 denote the real view of C.
2. *Hybrid*$_1$. Let T_1 be the same as T_0, except that the secret sharing operation is replaced by the secret share simulator. The underlying secret sharing guarantees that T_1 is indistinguishable from T_0.This hybrid is exactly the view output by the simulator *Sim*$_C$.

□

D. Communication Complexity

As is shown in Table III, we compare the communication costs of our `IDCloak` with `iPrivJoin`.Thee communication costs of key cryptographic primitives are defined as follows: C_{oprf}, C_{okvs},C_{opprf} are $\mathcal{O}(m(\lambda + \log m))$, $\mathcal{O}(\kappa m)$ and $\mathcal{O}(m(\lambda + \log m + \kappa))$, respectively.

TABLE III
COMPARISON OF COMMUNICATION SIZES BETWEEN OUR IDCLOAK VS. THE SOTA IPRIVJOIN, WHERE n REPRESENTS THE NUMBER OF PARTICIPATING PARTIES, d_i DENOTES THE FEATURE DIMENSION OF PARTY P_i, d DENOTES THE TOTAL FEATURE DIMENSION, m DENOTES THE DATASET SIZE AND l DENOTES THE BIT LENGTH.

Framework	Communication Sizes
`iPrivJoin`	$\mathcal{O}(C_{oprf} + (1+d_2)C_{opprf} + (d_1 + 2d + 1)ml)$
Ours ($n = 2$)	$\mathcal{O}(C_{oprf} + (1+d_2)C_{okvs} + (d_1 + 2d + 1)ml)$
Ours	$\mathcal{O}(C_{oprf} + (n - 1 + \sum_{i=2}^{n}(d_i))C_{okvs} + (d_1 + n^2 d - nd + 1)ml)$

VI. EVALUATION

A. Experiment Setting

1) Environment: We perform all experiments on a single Linux server equipped with 20-core 2.4 GHz Intel Xeon CPUs

and 1T RAM. We used a single process to simulate a single party. Additionally, we apply tc tool[2] to simulate LAN setting with a bandwidth of 1GBps and sub-millisecond round-trip time (RTT) latency and WAN setting with 40Mbps bandwidth and 40ms RTT latency, following prior work [29]. Without specifications, the default network is the WAN setting.

2) Implementation: We implement `IDCloak` in C++, integrating essential components such as OPRF, OKVS, cuckoo hashing, and simple hashing from [20]. In particular, for OPRF and OKVS, we leverage the optimized 3H-GCT algorithm from [20], setting the cluster size to 2^{14} and the weight parameter to 3. For hashing, we configure the hash table size as $b = 1.27m$ and utilize $h = 3$ hash functions. Additionally, we set the bit-length parameter to $l = 64$ and define the computational ring as $\mathbb{Z}_{2^{64}}$. The protocol operates with a computational security parameter of $\kappa = 128$ and statistical security parameter of $\lambda = 40$. The source code of `IDCloak` is publicly available[3].

3) Baseline: To the best of our knowledge, `iPrivJoin` is the only framework with identical functionality to `IDCloak`, except that it is limited to the two-party setting. `iPrivJoin`, like our scheme, first computes the intersection IDs, aligns the features, and uses shuffle to remove redundant data. Since `iPrivJoin` is not open-sourced, we re-implement it[4]. Both `iPrivJoin` and `IDCloak` are evaluated under identical experimental settings, except for the number of parties $n = 2$ for `iPrivJoin`. We do not include mPSI-based dataset join methods as our baselines because they expose intersection IDs in plaintext, whereas `IDCloak` keeps them private for stronger privacy protection.

4) Datasets: As is shown in Table IV, we employ six widely-used real-world datasets, *phishing*, *myocardial infarction complications*, *Appliances energy prediction*, *bank marketing*, *Give me some credit*, *Health Indicators* all of which are available in the UCI Machine Learning Repository [30] or Kaggle [31]. These datasets have 10 to 111 features and contain between 1353 and 253680 samples. We partition each dataset vertically, ensuring that all parties hold the same number of samples as in the original dataset, but with uniformly distributed disjoint feature subsets. Specifically, 80% of the samples retain randomly selected intersection IDs, whereas the remaining 20% consist of randomly generated samples.

TABLE IV
DATASETS USED FOR END-TO-END EXPERIMENTS

Datasets	#Features (d)	Dataset size (m)
Phishing	10	1353
Myocardial	111	1700
Energy	29	19735
Bank	17	45211
Credit	12	150000
Health	21	253680

[2]https://man7.org/linux/man-pages/man8/tc.8.html
[3]`IDCloak` is available at: https://github.com/chenshuyuhhh/IDCloak.git.
[4]`iPrivJoin` is available at: https://github.com/chenshuyuhhh/iPrivJoin.git.

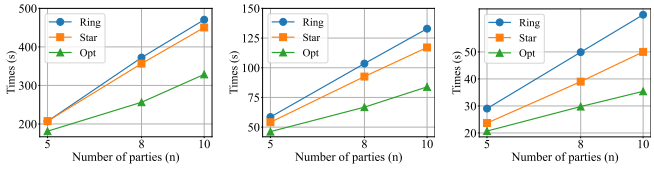
B. Evaluation of `IDCloak` on Real-world Datasets

We first conduct end-to-end online efficiency experiments by varying the number of parties (n) from 2 to 6. As is shown in Table V, we can summarize the following:

- In the two-party setting, `IDCloak` outperforms `iPrivJoin` by $1.69\times \sim 1.92\times$ and $1.50\times \sim 1.72\times$ in terms of time and communication sizes, respectively. This efficiency gain primarily stems from `IDCloak` using lightweight OKVS, whereas `iPrivJoin` relies on the more communication-expensive OPPRF (Section V-D provides a detailed comparison of the communication complexity).
- In multi-party settings involving 3 to 6 parties, `IDCloak` maintains efficiency comparable to `iPrivJoin`. This is attributed to the scalability of our multi-party protocol, which preserves performance as the number of parties increases. Notably, `IDCloak` demonstrates linear scalability with respect to the number of parties, highlighting its practicality for multi-party settings.

C. Evaluation of cmPSI Protocol

1) Evaluation of Optimization for cmPSI Protocol: We conduct experiments to evaluate the performance of our proposed cmPSI protocol using an optimized communication structure, comparing it against ring-based and star-based communication structures under varying network conditions of 10Mbps, 40Mbps, and 100Mbps bandwidth, with a fixed network latency of 40ms. As is shown in Figure 10, the cmPSI protocol with optimized communication structure outperforms the ring-based communication structure and the star-based communication structure by $1.14\times \sim 1.80\times$ and $1.14\times \sim 1.41\times$, respectively. Furthermore, the performance advantage of the optimized communication structure becomes increasingly significant as the number of parties increases.



(a) 10Mbps bandwidth (b) 40Mbps bandwidth (c) 100Mbps bandwidth

Fig. 10. Comparison of time (in seconds) for different communication structures in cmPSI protocol under different network settings. 'Ring', 'Star', and 'Opt' denote a ring-based communication structure, a star-based communication structure, and optimized communication structure, respectively.

2) Evaluation of SOTA Comparison for cmPSI Protocol: We conduct experiments to compare our proposed cmPSI protocol with the SOTA cmPSI protocol [23], which also keeps the intersection IDs private. Our experiments configure each protocol to resist the maximum number of colluding parties that it can tolerate. Specifically, our proposed cmPSI protocol provides stronger security (resists against up to $n-1$ colluding parties), whereas [23] only resists against up to $n/2$ colluding parties. Moreover, to align its output representation with [23], we add a secure zero-equality operator [12] to the output of our proposed cmPSI protocol. Specifically, in our protocol,

TABLE V
IDCLOAK VS. THE TWO-PARTY FRAMEWORK IPRIVJOIN [10] ON SIX REAL-WORLD DATASETS.

Framework	n	Time (s)						Communication sizes (MB)					
		Phishing	Myocardial	Energy	Bank	Credit	Health	Phishing	Myocardial	Energy	Bank	Credit	Health
iPrivJoin	2	3.13	8.36	22.82	31.62	70.61	192.53	2.88	13.13	42.14	62.35	146.46	401.53
IDCloak	2	1.73	4.37	12.49	16.50	41.72	109.13	1.82	8.74	25.91	36.29	89.38	240.14
	3	2.05	5.61	17.99	21.68	60.74	162.79	3.82	19.96	59.42	83.06	205.11	560.17
	4	2.37	7.22	23.63	28.96	79.93	212.31	6.09	34.45	104.18	145.91	356.62	975.59
	5	2.64	8.97	29.20	37.76	99.36	265.19	8.69	52.63	159.00	223.02	540.55	1502.14
	6	3.02	10.85	35.49	47.59	120.60	328.47	11.41	74.42	225.85	314.31	772.74	2129.98

when an ID in P_1's hash bin belongs to the intersection, all other parties receive a secret-shared flag of 0; otherwise, they receive a secret-shared flag of random values. In contrast, [23] assigns a secret-shared flag of 1 indicating intersection IDs.

TABLE VI
OUR CMPSI VS. SOTA CMPSI PROTOCOL [23] ON DATASETS WITH SIZES $m \in \{2^{16}, 2^{20}\}$, AND INVOLVING 3 TO 10 PARTIES. 'COMM' IS SHORT FOR COMMUNICATION.

	m	Protocol	3	5	8	10
LAN Time (s)	2^{16}	[23]	1.46	2.04	2.89	3.47
		Our cmPSI	1.05	1.43	1.99	2.34
	2^{20}	[23]	22.40	33.62	48.12	57.60
		Our cmPSI	16.09	19.16	26.16	28.35
WAN Time (s)	2^{16}	[23]	16.74	30.78	49.87	63.72
		Our cmPSI	6.60	8.47	11.37	13.08
	2^{20}	[23]	278.95	518.67	875.99	1130.24
		Our cmPSI	56.27	84.90	118.53	145.25
Comm Sizes (MB)	2^{16}	[23]	82.09	171.49	312.14	414.87
		Our cmPSI	14.72	29.44	51.52	66.25
	2^{20}	[23]	1412.08	2941.12	5339.59	8158.22
		Our cmPSI	207.66	415.31	726.80	934.45

As is shown in Table VI, we can summarize the following: Our proposed cmPSI protocol, which offers stronger security, outperforms SOTA cmPSI [23] by $2.54\times \sim 7.78\times$ and $1.39\times \sim 2.03\times$ in the WAN and LAN settings, respectively, in terms of time. These improvements primarily stem from the reduced communication overhead in our protocol, which achieves a $5.58\times \sim 8.73\times$ reduction compared to cmPSI [23]. Specifically, cmPSI [23] relies on PSM primitives, which impose higher communication costs than the OKVS and OPRF primitives used in our protocol. In addition, whereas the SOTA cmPSI protocol employs a star communication structure, our proposed protocol adopts an optimized communication structure that enhances the efficiency without compromising security. Notably, as the number of parties n increases or the dataset size m grows, the advantages of our protocol become even more pronounced.

D. Evaluation of smFA Protocol

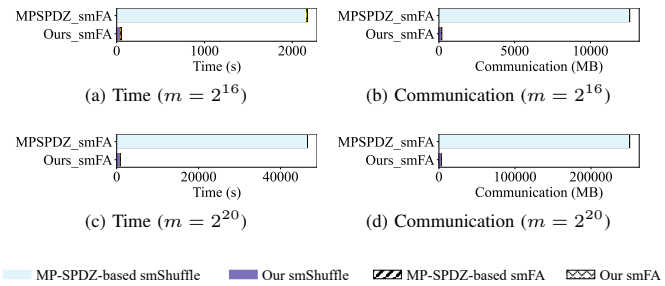
1) Evaluation of SOTA Comparison for Secure Shuffle Protocol: We evaluate the efficiency of our proposed secure multi-party shuffle protocol, a core component of the smFA protocol, against the SOTA secure multi-party shuffle protocol implemented by MP-SPDZ [12]. As shown in Table VII, our proposed secure shuffle protocol achieves $21.44\times \sim 41.47\times$ and $61.62\times \sim 138.34\times$ speedup over the SOTA protocol [12] in the LAN and WAN settings, respectively. These improvements primarily stem from significantly lower communication sizes and rounds, precisely, a $105.69\times \sim 132.13\times$ reduction in communication sizes compared to [12]. Specifically, our proposed secure shuffle protocol incurs the communication sizes of $\mathcal{O}(ndlm)$ per party over n communication rounds, whereas the SOTA secure shuffle protocol [12] leverages Waksman networks [32] for shuffling, resulting in larger communication sizes of $\mathcal{O}(ndlm \log m)$ and a requirement of $\mathcal{O}(\log m)$ rounds per party. Consequently, our protocol demonstrates a more pronounced efficiency advantage as the number of parties and the dataset size increase. For instance, in the WAN setting with $n = 5$, increasing m from 2^{16} to 2^{20} results in an efficiency improvement over [12] from $104.50\times$ to $138.34\times$. Similarly, when $m = 2^{20}$, increasing n from 3 to 5 boosts the efficiency improvement from $81.89\times$ to $138.34\times$.

TABLE VII
THE ONLINE TIME (IN SECONDS) AND COMMUNICATION SIZE (IN MBS) OF OUR PROPOSED SECURE MULTI-PARTY SHUFFLE PROTOCOL VS. SOTA SECURE MULTI-PARTY SHUFFLE PROTOCOL [12] ON DATASETS WITH SIZES $m \in \{2^{16}, 2^{20}\}$, AND INVOLVING 3 AND 5 PARTIES. 'PRO' AND 'COMM' ARE SHORT FOR PROTOCOL AND COMMUNICATION. RESPECTIVELY.

m	Pro	LAN Time (s)		WAN Time (s)		Comm sizes (MB)	
		3	5	3	5	3	5
2^{16}	[12]	38.79	144.37	2154.36	11204.60	12484.92	68456.40
	Ours	1.81	6.49	34.96	107.22	118.11	647.70
2^{20}	[12]	1330.87	3316.92	46267.61	239100.02	249687.80	1369105.00
	Ours	32.09	113.24	564.98	1728.33	1889.76	10363.20

2) Evaluation of Efficiency Improvement for smFA using our secure shuffle: We evaluate the efficiency improvements of the smFA protocol from using the SOTA secure multi-party shuffle provided by MP-SPDZ [12] to use our proposed secure shuffle protocol. As is shown in Figure 11, our proposed secure shuffle protocol significantly reduces the time and communication sizes of the smFA protocol by $40.54\times \sim 53.58\times$ and $65.63\times \sim 81.29\times$ respectively, compared with integrating the SOTA secure shuffle in the smFA protocol. The improvements occur because the shuffle operation alone contributes to over 99% of the total time and communication sizes when using the SOTA secure multi-party shuffle protocol.

3) Evaluation of Efficiency for smFA Protocol: We evaluate the efficiency of our proposed smFA protocol under varying numbers of parties (n), dataset sizes (m), and feature dimensions (d). As is shown in Figure 12, the time and communication sizes of the smFA protocol exhibit linear growth with respect to the number of parties, dataset size, and

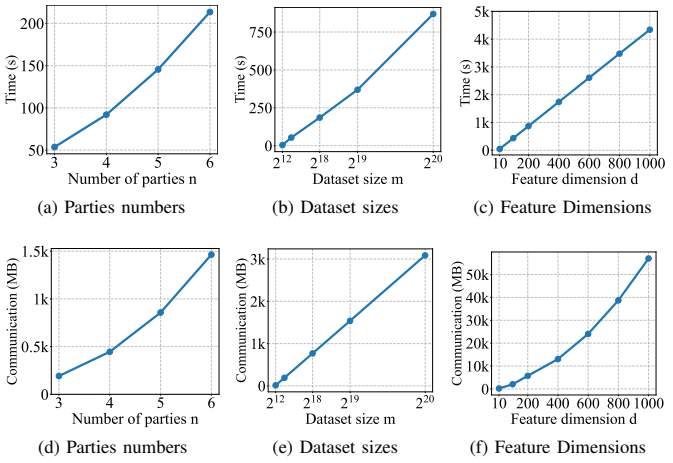


(a) Time ($m = 2^{16}$) (b) Communication ($m = 2^{16}$)

(c) Time ($m = 2^{20}$) (d) Communication ($m = 2^{20}$)

Fig. 11. Online time (in seconds) and communication sizes (in MBs) of the smFA protocol using our proposed secure multi-party shuffle vs. the smFA protocol using the SOTA secure multi-party shuffle [12]. Evaluations are conducted on datasets with varying dataset sizes $m \in \{2^{16}, 2^{20}\}$, fixed feature dimension $d_i = 10 (i \in [n])$, and a fixed number of parties $n = 3$.

feature dimensions. These results highlight the practicality of the smFA protocol in multi-party settings.



(a) Parties numbers (b) Dataset sizes (c) Feature Dimensions

(d) Parties numbers (e) Dataset sizes (f) Feature Dimensions

Fig. 12. Online Time (in seconds) and communication sizes (in MB) of our smFA protocol under varying parameters. When varying n, parameters $d_i = 10$ and $m = 2^{16}$ are fixed; when varying m, parameters $n = 3$ and $d_i = 10$ are fixed; when varying d_i, parameters $n = 3$ and $m = 2^{16}$ are fixed. Note: 'k' denotes 1000.

VII. Discussion

Revealing the Size of the Intersection Set. Although `IDCloak` reveals the intersection size c, this information is essential for practical applications [33], [34]. In vPPML, parties often rely on c to decide whether to continue training; if c is below a certain threshold (e.g., $c < 50\% \cdot n$), they can terminate early to avoid wasting effort. Thus, revealing c in `IDCloak` is reasonable. Moreover, `IDCloak` provides c in the cmPSI protocol, allowing parties to end the process promptly if the intersection size is too small.

Supporting vPPML Frameworks with Different Schemes. Our `IDCloak` generates training dataset in ASS, a commonly used scheme in vPPML frameworks [3], [12]. However, practical applications may require other schemes, such as boolean secret sharing or homomorphic encryption, depending on the training setup. To address this, `IDCloak` can use existing conversion protocols [27], [29] to transform ASS dataset into the required scheme.

VIII. Conclusion

In this paper, we investigate that existing frameworks for secure dataset join in vPPML could be impractical because they are insecure when they expose the intersection IDs; or they rely on a strong trust assumption requiring a non-colluding auxiliary server; or they are limited to the two-party setting. To resolve the problem, we propose `IDCloak`, the first practical secure multi-party dataset join framework for vPPML without a non-colluding auxiliary server. `IDCloak` consists of two efficient protocols. First, our proposed circuit-based multi-party private set intersection (cmPSI) protocol securely computes secret-shared flags indicating intersection IDs. Second, our proposed secure multi-party feature alignment protocol leverages our proposed secure multi-party shuffle protocol to construct the secret-shared and joint dataset based on these secret-shared flags. Our experiments show that in the two-party setting, `IDCloak` outperforms `iPrivJoin`, a state-of-the-art secure two-party dataset join framework, and maintains comparable efficiency even as the number of parties increases. Furthermore, compared to the SOTA protocol, our proposed cmPSI protocol offers a stronger security guarantee while improving efficiency by up to $7.78\times$ in time and $8.73\times$ in communication sizes. Additionally, our secure multi-party shuffle protocol outperforms the SOTA protocol by up to $138.34\times$ in time and $132.13\times$ in communication sizes.

References

[1] C. Chen, J. Zhou, L. Wang, X. Wu, W. Fang, J. Tan, L. Wang, A. X. Liu, H. Wang, and C. Hong, "When homomorphic encryption marries secret sharing: Secure large-scale sparse logistic regression and applications in risk control," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2652–2662.

[2] G. Lin, W. Han, W. Ruan, R. Zhou, L. Song, B. Li, and Y. Shao, "Ents: An efficient three-party training framework for decision trees by communication optimization," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4376–4390.

[3] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[4] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1257–1272.

[5] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 2, pp. 1–35, 2018.

[6] A. Bay, Z. Erkin, M. Alishahi, and J. Vos, "Multi-party private set intersection protocols for practical applications." in *SECRYPT*, 2021, pp. 515–522.

[7] M. Wu, T. H. Yuen, and K. Y. Chan, "O-Ring and K-Star: Efficient multi-party private set intersection," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 6489–6506.

[8] X. Jiang, X. Zhou, and J. Grossklags, "Comprehensive analysis of privacy leakage in vertical federated learning during prediction." *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 2, pp. 263–281, 2022.

[9] Y. Gao, H. Deng, Z. Zhu, X. Chen, Y. Xie, P. Duan, and P. Chen, "Peafowl: Private entity alignment in multi-party privacy-preserving machine learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 2706–2720, 2025.

[10] Y. Liu, B. Zhang, Y. Ma, Z. Ma, and Z. Wu, "iprivjoin: An id-private data join framework for privacy-preserving machine learning," *IEEE Transactions on Information Forensics and Security*, 2023.

[11] N. Chandran, D. Gupta, and A. Shah, "Circuit-psi with linear complexity via relaxed batch opprf," *Cryptology ePrint Archive*, 2021.

[12] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[13] R. Inbar, E. Omri, and B. Pinkas, "Efficient scalable multiparty private set-intersection via garbled bloom filters," in *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*. Springer, 2018, pp. 235–252.

[14] A. Kavousi, J. Mohajeri, and M. Salmasizadeh, "Efficient scalable multi-party private set intersection using oblivious prf," in *Security and Trust Management: 17th International Workshop, STM 2021, Darmstadt, Germany, October 8, 2021, Proceedings 17*. Springer, 2021, pp. 81–99.

[15] A. Bay, Z. Erkin, J.-H. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1–15, 2021.

[16] M. Wu, T. H. Yuen, and K. Y. Chan, "O-ring and k-star: Efficient multi-party private set intersection," in *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, 2024.

[17] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS*, 2012.

[18] P. Rindal and P. Schoppmann, "Vole-psi: fast oprf and circuit-psi from vector-ole," in *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II*. Springer, 2021, pp. 901–930.

[19] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*. Springer, 2021, pp. 395–425.

[20] P. Rindal and S. Raghuraman, "Blazing fast psi from improved okvs and subfield vole." *IACR Cryptol. ePrint Arch.*, vol. 2022, p. 320, 2022.

[21] J. P. Ma and S. S. Chow, "Secure-computation-friendly private set intersection from oblivious compact graph evaluation," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 1086–1097.

[22] Y. Li, D. Ghosh, P. Gupta, S. Mehrotra, N. Panwar, and S. Sharma, "Prism: Private verifiable set computation over multi-owner outsourced databases," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1116–1128.

[23] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty psi and extensions to circuit/quorum psi," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1182–1204.

[24] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[25] R. Pagh and F. F. Rodler, "Cuckoo hashing," in *Algorithms—ESA 2001: 9th Annual European Symposium Århus, Denmark, August 28–31, 2001 Proceedings*. Springer, 2001, pp. 121–133.

[26] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2*. Springer, 2005, pp. 303–324.

[27] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *NDSS*, 2015.

[28] D. Demmler, P. Rindal, M. Rosulek, and N. Trieu, "Pir-psi: scaling private contact discovery," *Cryptology ePrint Archive*, 2018.

[29] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 35–52.

[30] M. Kelly, R. Longjohn, and K. Nottingham, "The uci machine learning repository," 2023. [Online]. Available: https://archive.ics.uci.edu

[31] W. C. Credit Fusion, "Give me some credit," 2011. [Online]. Available: https://kaggle.com/competitions/GiveMeSomeCredit

[32] A. Waksman, "A permutation network," *Journal of the ACM (JACM)*, vol. 15, no. 1, pp. 159–163, 1968.

[33] S. Ghosh and M. Simkin, "The communication complexity of threshold private set intersection," in *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II*. Springer, 2019, pp. 3–29.

[34] S. Badrinarayanan, P. Miao, S. Raghuraman, and P. Rindal, "Multi-party threshold private set intersection with sublinear communication," in *Public-Key Cryptography–PKC 2021: 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10–13, 2021, Proceedings, Part II*. Springer, 2021, pp. 349–379.

APPENDIX

A. Algorithm for optimized communication structure of cmPSI

As is shown in algorithm 1, it aims to construct an efficient communication structure for our cmPSI protocol, based on the parameters, i.e. the number of parties n, the number of designated root party $leader\_num$, the time for sending one OKVS table t_s, and the network delay (t_l). The core idea is to maximize node utilization while minimizing communication delay. Node is a data structure that has the num, children, and parent, where the 'num' indicates the identifier of a party, the 'parent' indicates the parties to whom this Node will send data, the 'child' indicates the parties whose data this Node will receive. The algorithm begins by computing k, which determines the number of children each node should have in the structure. Nodes are then grouped into sets of size k, with the last node in each group acting as the parent for the others. This grouping process is iterated until a single root node remains, which represents the leader. If the resulting root does not match the designated leader, a breadth-first search is used to locate and swap in the correct leader node.

Algorithm 1 Optimized communication structure generation

Input: The number of parties n, the number of root party $leader\_num$, the time for sending one OKVS table t_s, the time of network delay t_l.

Output: Root node $node$ of the communication structure.

$k = \lceil \frac{t_L}{t_s} \rceil + 1$, $c = n \bmod k$, $nodes, nodes2 = \emptyset$
for $i = 0$ to $n - 1$ **do**
  $nodes.append(Node(i))$
end for
while $n > 1$ **do**
  $c = n \bmod k$
  for $i = 0$ to $n - k - c$ step k **do**
    for $j = 0$ to $k - 1$ **do**
      $nodes[i + k - 1].add\_child(nodes[i + j])$
      $nodes[i + j].parent = nodes[i + k - 1]$
    end for
    $nodes2.append(nodes[i + k - 1])$
  end for
  for $i = n - k - c$ to $n - 1$ **do**
    $nodes[n - 1].add\_child(nodes[i])$
    $nodes[i].parent = nodes[n - 1]$
  end for
  $nodes2.append(nodes[n - 1])$
  $nodes = nodes2$, $nodes2 = \emptyset$
  $n = len(nodes)$
end while
$node = nodes[0]$
if $node.num == leader\_num$ **then**
  Return
end if
$queue =$ deque($[node]$), $t\_node =$ None
while $queue \neq \emptyset$ **do**
  $current = queue.popleft()$
  if $current.num == leader\_num$ **then**
    $t\_node = current$; Break
  end if
  for $child \in current.child$ **do**
    $queue.append(child)$
  end for
end while
if $t\_node \neq$ None **then**
  $node.num, t\_node.num = t\_node.num, node.num$
end if
Return $node$