Efficient Private Filtering and Aggregation for Weighted Set Intersection via Oblivious Encrypted Weight Transfer

Xiaodong Wang
Tsinghua University
Beijing, China
wangxd22@mails.tsinghua.edu.cn

Shengzhe Meng
Tsinghua University
Beijing, China
msz22@mails.tsinghua.edu.cn

Zijie Lu
Xi'an Jiaotong-Liverpool University
Suzhou, China
lzjluzijie@gmail.com

Bei Liang
Beijing Institute of Mathematical Sciences and Applications
Beijing, China
lbei@bimsa.cn

Abstract

Private Set Intersection (PSI) enables parties to compute the intersection of their input item sets while preserving privacy. In many real-world applications, however, each item is accompanied by a sensitive weight, and the ability to privately compute over such weights is crucial. Existing research in this direction is fragmented and driven by application-specific goals, with representative examples including PI-Sum (computing the sum of weights over the intersection), inner-product Private Join and Compute (computing the inner product of weight vectors over the intersection), and Item with Maximum Weight Sum (identifying the intersection item with the maximum combined weight).

In this work, we propose a unified framework for private computation on weighted set intersection. We formalize *Private Filtering and Aggregation for Weighted Set Intersection* (PFA-WSI) as an ideal functionality parameterized by a joint scoring function f and a predicate P, supporting two output modes: (i) *predicate-filtered output*, which reveals a predicate-selected subset of intersection items, and (ii) *aggregated output*, which reveals only aggregate statistics over matched items. By instantiating f and P appropriately, PFA-WSI captures deployed and studied tasks such as PI-Sum, inner-product PJC, and IMWS, and also accommodates richer metrics arising in practice, such as L_1- and L_2-type distance statistics on matched item weights.

To realize PFA-WSI efficiently, we introduce a novel core building block, Oblivious Encrypted Weight Transfer (OEWT), which enables a receiver to obtain encryptions of the sender's weights for intersection items and random-looking ciphertexts otherwise. Building on OEWT and additively homomorphic encryption, we present modular protocol constructions for different instantiations of f and for both output modes. We prove simulation-based security in the semi-honest model and provide detailed communication and computation analyses. Our experiments show that our constructions scale to million-sized sets with practical performance that matches or surpasses the state-of-the-art.

CCS Concepts

• **Applied Cryptography → Secure Multiparty Computation**.

Keywords

secure multiparty computation, private set intersection, additively homomorphic encryption.

1 Introduction

In many domains such as finance and healthcare, data needed for analysis is fragmented across multiple organizations. At the same time, each holder is bound by strict privacy laws and security regulations, which makes joint data analytics highly challenging. Secure multiparty computation (MPC) is a promising cryptographic approach to this problem, as it enables collaborative computation over distributed data while provably protecting the privacy of each party's input.

Compared with generic MPC protocols such as garbled circuits [33], protocols tailored to specific functionalities are often significantly more efficient and closer to practical deployment. Among the functionalities studied in MPC, private set intersection (PSI) is one of the most extensively researched and widely deployed primitives [14, 18, 24, 25]. In standard PSI, two parties, each holding a private set, learn the intersection of their sets and nothing else. Over the last decade, many efficient PSI protocols [7, 20, 29, 30] have been proposed, and their performance is now sufficient for large-scale, real-world use.

However, in many real applications, the relevant data consists not only of identifiers but also of sensitive weights attached to those identifiers, and the ability to privately compute over such weights is crucial. For example, in a financial setting, two large banks each hold a customer list together with a risk score derived from transaction histories. They may wish to perform joint risk assessment, such as finding all common customers whose joint risk score exceeds a threshold, while preserving the privacy of all other customers. In a healthcare setting, two hospitals hold patient populations and each has developed its own risk scoring model for a particular disease. They would like to compare the two models on their overlapping patients by computing a distance statistic (e.g., L_1, L_2) together with the intersection size, while revealing no information beyond the intended output. This latter use case differs from the banking scenario in that the final output should reveal no identifiers at all, only aggregated statistics on the weights. In Fig. 1, we illustrate the two settings using two toy examples.

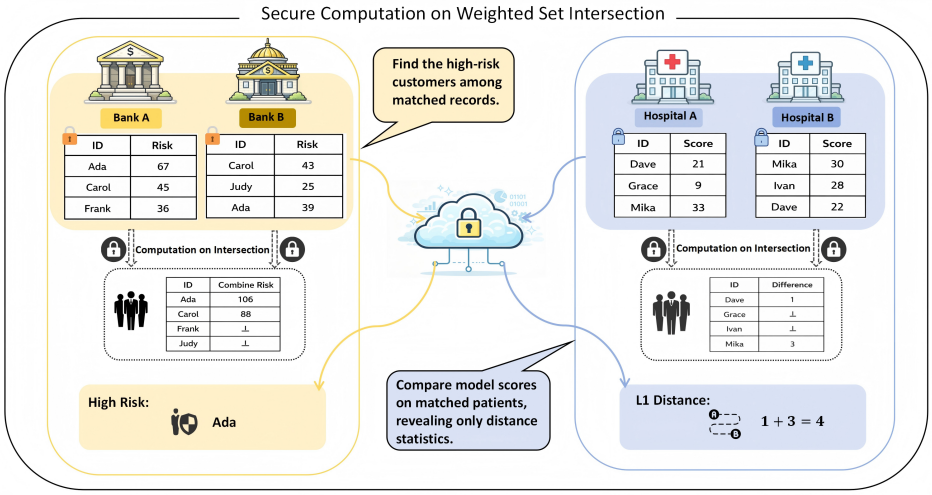

Figure 1: Schematic illustration of the two output modes. On the left, two banks aim to identify the highest-risk customers among their shared clients. On the right, two hospitals aim to assess the agreement between the health scoring models they developed by comparing their scores on overlapping patients.

These examples suggest categorizing private computation on weighted set intersection according to the type of output they reveal. Specifically, such tasks can be viewed as evaluating a function of the two parties' weights on matched identifiers, and then releasing either a predicate-selected subset or an aggregate statistic, which also reflects the current literature. Existing works largely fall into one of these two output types:

Predicate-filtered output. Here, the goal is to output the identifiers (and possibly their weights) of intersection items that satisfy a predicate P applied to their joint scores, as in the banking example above where the parties seek common customers whose joint risk score exceeds a threshold. Another canonical example is the private "best meeting time" problem, where two parties seek the common time slot with the highest total score. Beauregard et al. [1] first developed a protocol for privately identifying the "best" item in the intersection according to the sum of both parties' weights. More recently, Cai et al. [6] formalized this functionality as item with maximum weight sum (IMWS), and proposed two more efficient protocols than [1], particularly when weights lie in a large domain.

Aggregated output. In this case, the protocol does not reveal any identifiers, but only an aggregate function of the associated weights. Existing deployed examples include Google's "PI-Sum with Cardinality" functionality [16] for aggregated conversion measurement, which outputs the total sum of weights over the intersection together with the intersection size. Lepoint et al. [21] subsequently introduced the "inner-product Private Join and Compute" (inner-product PJC) functionality as a generalization of PI-Sum, allowing the parties to obtain the inner-product of their intersection weight vectors. Chida et al. [9] later proposed a more communication- and round-efficient construction for inner-product PJC.

Despite this progress, the current landscape of computation on weighted set intersection is highly fragmented. Prior work often targets a specific functionality (e.g., sum, inner-product, or maximum-weight item selection), and is tailored to a particular cryptographic framework. Consequently, these constructions are inherently limited in scope and do not readily generalize to settings other than the ones for which they were originally developed. Moreover, existing constructions mainly focus on simple linear functions such as sum and inner-product. They cannot directly handle more expressive metrics that arise in applications like model comparison in healthcare, where one may wish to compute, for example, the L_1 or L_2 distance between risk-score vectors over the intersection without revealing any information beyond the intended output. Finally, IMWS is the only predicate-filtered functionality that has been explicitly formalized so far, yet existing IMWS protocols [1, 6] incur quadratic computational complexity in the parties' input sizes, making them difficult to deploy for large datasets.

These observations naturally lead to the following question:

Can we design a general and extensible framework for private computation on weighted set intersection that (i) subsumes existing functionalities such as PI-Sum, inner-product PJC, and IMWS, and (ii) supports richer functions on weights (e.g., L_1/L_2-type distances), while avoiding fully generic circuit based approaches and remaining efficient enough for real-world deployment?

1.1 Our Contributions

We answer this question in the affirmative. We begin by introducing a unified ideal functionality for computation on weighted set intersection.

As shown in Fig. 2, we formalize **Private Filtering and Aggregation for Weighted Set Intersection (PFA-WSI)** as an ideal functionality parameterized by a function f and a predicate P, which supports two practically motivated output modes: (a) *predicate-filtered output*, which reveals a subset of intersection items filtered out by a predicate function P on the joint scores of the matched items; and (b) *aggregated output*, which reveals only aggregate statistics over the weights of intersected items.

FUNCTIONALITY:

Inputs: A set of item-weight pairs $S_1 = \{(x_1, u_1), \ldots, (x_{n_1}, u_{n_1})\}$ from Alice, and $S_2 = \{(y_1, v_1), \ldots, (y_{n_2}, v_{n_2})\}$ from Bob, where $x_i \neq x_j$ and $y_i \neq y_j$ for $i \neq j$.

Parameters: A two-argument function $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$, and a predicate $P : [n_2] \times (\mathbb{Z} \cup \{\perp\})^{n_2} \to \{0, 1\}$.

Outputs: For $j \in [n_2]$, if $\exists (x_i, u_i) \in S_1$ such that $x_i = y_j$, then let $w_j = f(u_i, v_j)$; else let $w_j = \perp$.

(a) **Predicate-Filtered Output.**
Alice outputs $w = (w_1, \ldots, w_{n_2})$, while Bob outputs $Y_P := \{y_j \mid y_j \in Y \wedge P(j, w) = 1\}$.

(b) **Aggregated Output.**
Let $I := \{j \in [n_2] \mid w_j \neq \perp\}$. Alice outputs $|I|$ and $\sum_{j \in I} w_j$.

Figure 2: Ideal functionality $\mathcal{F}_{\text{PFA-WSI}}$ for private filtering and aggregation for weighted set intersection.

This abstraction provides a common language that unifies a broad spectrum of deployed and studied tasks. In particular, PI-Sum fits the aggregated output mode by taking $f(u, v) = v$ (i.e., summing Bob's weights over the intersection).[1] Inner-product PJC is also an aggregated output instance, where $f(u, v) = u \cdot v$. Likewise, distance statistics arise naturally in the same mode: the L_1-type distance corresponds to $f(u, v) = |u - v|$, and the L_2-type distance corresponds to $f(u, v) = (u - v)^2$ (and optionally reporting its square root as needed). On the other hand, IMWS is an instance of the predicate-filtered mode: one sets a scoring function (e.g., $f(u, v) = u + v$ for weight-sum scoring) and chooses P to select the item attaining the maximum score among intersection items.

To realize the PFA-WSI functionality, our contributions are summarized as follows.

• **Oblivious Encrypted Weight Transfer (OEWT).** We introduce OEWT as a core building block that enables a receiver to obtain encryptions of the sender's weights for intersected items, while obtaining random ciphertexts for non-intersected items, without learning which case occurred. To realize OEWT efficiently, we combine the encryption scheme with Oblivious Key-Value Stores (OKVS) by introducing a pseudouniformly embedding technique that maps ciphertexts into the OKVS value space while preserving the uniformity required for OKVS obliviousness and random decoding. And in our implementation, we instantiate OEWT using an efficient binary OKVS.

• **Protocols for predicate-filtered output.** We give an efficient construction for predicate-filtered PFA-WSI by combining OEWT with multi-query Reverse Private Membership Test (mq-RPMT) and a lightweight oblivious transfer step. This construction directly yields IMWS and supports other predicates (e.g., thresholding) via local post-processing. In contrast to prior IMWS protocols [1, 6] with quadratic overhead, our construction achieves linear overhead in the set sizes.

• **Protocols for aggregated output.** We give a modular construction for aggregated output PFA-WSI. To this end, we define a Function-Specific Homomorphic Evaluation phase that allows Bob to obtain, for each intersection item, an encryption of f evaluated on the parties' associated weights, and a random ciphertext otherwise. We provide concrete instantiations of this phase for specific functions f such as L_1 and L_2 distance. We note that the evaluation phase is applicable to the predicate-filtered setting to support functions beyond the simple summation used in IMWS. Moreover, we introduce a new building block called Secure Indicator-Selected Aggregation (SISA), which aggregates these encryptions to produce the final aggregate statistics for Alice. For completeness, we include in the appendix instantiations for inner-product and summation, whose realizations are comparatively simple and direct. As an additional application, we use the resulting PFA-WSI protocols for inner-product and summation to construct a protocol for computing the cosine similarity between the parties' weight vectors over the intersection. Unlike L_1/L_2 distances, cosine similarity captures the similarity of the two vectors' patterns while being invariant to their overall scale.

• **Security and complexity analysis.** We provide simulation-based security proofs for all proposed protocols in the semi-honest model, together with a detailed analysis of their communication and computation complexity.

• **Implementation and evaluation.** We implemented our protocols in C++ and compared them with prior work. Owing to linear complexity, our IMWS protocol requires only 279.61 seconds of computation when both parties hold sets of size 2^{20}, whereas prior protocols [1, 6], which incur quadratic overhead, require more than 20,000 seconds. We further provide a step-by-step breakdown for our protocols that compute L_1- and L_2-type distance statistics over the intersection. The results show that the L_1 and L_2 instantiations have comparable computation costs, but L_1 incurs higher communication. Concretely, when both parties hold sets of size 2^{16}, the L_2 protocol requires 91.55 seconds of running time and 69.21 MiB of communication, whereas the L_1 protocol requires 125.8 seconds and 486.17 MiB. Finally, for inner-product function, our protocol reduces running time by a factor of 1.3-3.7× compared to the state-of-the-art [9], but at the cost of requiring 2× more communication.

1.2 Technique Overview

Oblivious Encrypted Weight Transfer. The core building block underlying our PFA-WSI constructions is the *Oblivious Encrypted Weight Transfer* (OEWT) functionality. In $\mathcal{F}_{\text{OEWT}}$, Alice inputs a set of item–weight pairs $\{(x_i, u_i)\}_{i \in [n_1]}$ and a public key pk, while Bob inputs an item set $\{y_i\}_{i \in [n_2]}$. For each Bob's item y_i, if $y_i = x_j$ for some $j \in [n_1]$, then the functionality $\mathcal{F}_{\text{OEWT}}$ returns to Bob the ciphertext $\mathsf{Enc}_{pk}(u_j)$ of the corresponding weight; otherwise, it returns a ciphertext sampled at random from the ciphertext space.

Our starting point for realizing OEWT is the OKVS data structure. An OKVS encodes a collection of key-value pairs (k_i, v_i) into a compact table T. The obliviousness property guarantees that the keys k_i remain hidden as long as all values v_i are random. A useful additional property is random decoding: informally, if the encoded values are random, then decoding T at any key k not used during encoding yields a value that is uniform over the value space. This behavior matches exactly what we need in OEWT for non-intersection items, where Bob should receive a random ciphertext.

A naive instantiation would therefore let Alice treat her items as OKVS keys and the encryptions of the corresponding weights as

[1] In the rest of the paper, we use Sum to denote the more general case $f(u, v) = u + v$.

OKVS values, encode the resulting key-value pairs into T, and send T to Bob, who decodes at his items. However, the properties of the OKVS listed above crucially rely on the values being uniform in the OKVS value space, which is typically a finite field. In particular, for efficiency reasons, OKVS instantiations over a characteristic-two field (e.g., $\mathbb{F}_{2^m}$) are often more efficient than those over other fields, since their arithmetic can be implemented largely via fast bitwise XOR operations. In contrast, ciphertexts of a public-key encryption scheme typically do not form a finite field, nor are they uniformly distributed over the finite-field value space required by the OKVS. Thus, directly using ciphertexts as OKVS values breaks the assumptions needed for obliviousness and random decoding.

To resolve this mismatch, we introduce the notion *$\mathbb{Z}_\beta$-uniformly embeddable* encryption, which captures encryption schemes whose ciphertexts can be injectively and efficiently encoded into an integer ring $\mathbb{Z}_\beta$ in a way that makes the encodings computationally indistinguishable from uniform over $\mathbb{Z}_\beta$. We show that both Paillier encryption and RLWE-based additively homomorphic encryption schemes satisfy this property. On top of this ring-level embedding, we then "spread" the randomness from $\mathbb{Z}_\beta$ to a prime field $\mathbb{Z}_p$: by adding a random multiple $r\beta$, with r sampled uniformly from $\{0, \ldots, \lfloor p/\beta \rfloor - 1\}$, we obtain values that are statistically close to uniform over $\mathbb{Z}_p$. We prove that when $p \geq 2^\lambda \cdot \beta$, the resulting distribution is $2^{-\lambda}$-close to uniform, and thus also computationally indistinguishable from uniform.

Our approach extends naturally to the binary field $\mathbb{F}_{2^m}$: by encoding elements of $\mathbb{Z}_p$ as m-bit strings (elements of $\mathbb{F}_{2^m}$) and choosing m such that $|\mathbb{F}_{2^m}| \geq 2^\lambda \cdot \beta$, we preserve pseudouniformity of OKVS values in the field representation. The detailed OEWT protocol, along with the formal randomness and security analysis, is given in Sec. 3.

PFA-WSI with predicate-filtered output. Multi-query Reverse Private Membership Test (mq-RPMT) [34] is a functionality which on inputs a set $X = \{x_i\}_{i\in[n_1]}$ from the receiver and a set $Y = \{y_i\}_{i\in[n_2]}$ from the sender, allows the receiver to obtain a bit vector $\overrightarrow{a} = (a_1, \ldots, a_{n_2})$, where $a_i = 1$ iff $y_i \in X$. Combined with $\mathcal{F}_{\text{mq-RPMT}}$ and $\mathcal{F}_{\text{OEWT}}$, we can directly obtain a IMWS protocol. First, Alice and Bob invoke $\mathcal{F}_{\text{OEWT}}$, so that Bob receives, for each y_i, either an encryption of the corresponding weight of Alice's common item, or a random ciphertext. Bob then locally homomorphically adds his own weight v_i to the ciphertext for y_i. Next, they invoke $\mathcal{F}_{\text{mq-RPMT}}$ with inputs X from Alice and Y from Bob, and Alice obtains the indicator vector $\overrightarrow{a}$ with $a_i = \mathbf{1}_{y_i \in X}$. Finally, Alice and Bob run oblivious transfer with selector bits a_i, so that Alice learns a ciphertext of $w_i = u_j + v_i$ when $y_i = x_j$ and learns $\perp$ otherwise. Alice decrypts the non-$\perp$ entries, finds the index j^* of the maximum aggregated weight, and sends j^* to Bob, who finally outputs y_{j^*}.

In the IMWS setting, the predicate P selects the item with maximum aggregated weight, and the underlying aggregation function is $f(u_i, v_j) = u_i + v_j$. More generally, changing the predicate P (e.g., selecting all items whose aggregated weight exceeds a threshold, or the top-k items) only affects Alice's local post-processing of the vector $(w_1, \ldots, w_{n_2})$ and can be handled directly on top of the above protocol. In contrast, changing the aggregation function f (beyond $f(u_i, v_j) = u_i + v_j$) requires an additional *Function-Specific Homomorphic Evaluation* phase, in which Alice and Bob interact so that Bob ends up with encryptions of $f(u_i, v_j)$ for intersection pairs, without leaking extra information to Alice. The detailed analysis is given in Sec. 4.

PFA-WSI with aggregated output. After the Function-Specific Homomorphic Evaluation phase, Bob holds, for each intersection item, an encryption of $f(u_i, v_j)$. To let Alice obtain an aggregated output from these ciphertexts, we introduce a building block called *Secure Indicator-Selected Aggregation* (SISA). In the functionality $\mathcal{F}_{\text{SISA}}$, Alice holds the secret key of an additively homomorphic encryption scheme and an indicator vector $\overrightarrow{a} \in \{0,1\}^{n_2}$, where entries equal to 1 mark the positions to be aggregated. Bob holds a ciphertext vector $\overrightarrow{b} = \{\mathsf{AEnc}_{pk}(v_i)\}_{i\in[n_2]}$. The functionality outputs to Alice the sum of Bob's weights selected by her indicators. We leverage oblivious transfer and random masks technique to implement $\mathcal{F}_{\text{SISA}}$; the full protocol and its analysis are given in Sec. 5.1.

The remaining challenge is to realize the Function-Specific Homomorphic Evaluation phase. For L_2 distance, our idea is that Bob masks the ciphertext of $u_i - v_j$ with a random offset t_j, sends the masked ciphertext to Alice for decryption and squaring, and then, after receiving an encryption of $(u_i - v_j + t_j)^2$ back, uses homomorphic operations to remove the mask and obtain an encryption of $(u_i - v_j)^2$. For L_1 distance, the main difficulty is handling the absolute value without leaking the sign of $u_i - v_j$. Here we invoke an integer comparison functionality: Bob first randomly flips the sign of $u_i - v_j$ and masks it with a random number, so that the sign of the true difference is hidden, and then Alice and Bob run secure comparison to obtain additive shares of the bit $\mathbf{1}_{u_i < v_j}$. Using these shares, Bob (as the OT sender) chooses appropriate OT messages, and Alice (as the receiver) uses her share as the selection bit to obtain a ciphertext of $|u_i - v_j|$ masked by a random number. Alice re-randomizes and sends this ciphertext back; Bob finally removes the mask and obtains an encryption of $|u_i - v_j|$. The concrete protocols and security proofs are presented in Sec. 5.2.

The technical roadmap of the building blocks in our PFA-WSI protocols is outlined in Fig. 3.

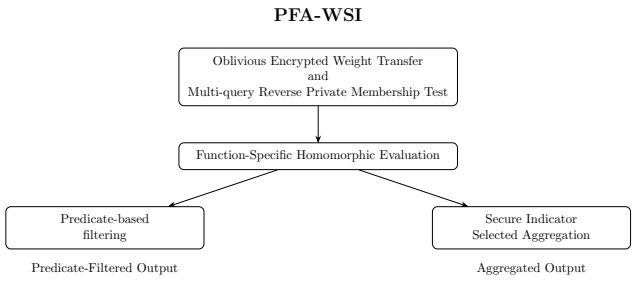

Figure 3: Technical roadmap of the building blocks in our PFA-WSI protocols.

1.3 Related Work

In this section, we review prior work about computing on weights associated with items in the intersection.

Ion et al. [16] study several protocols for Private Intersection-Sum with Cardinality (PI-Sum), motivated by privacy-preserving advertising conversion measurement. In PI-Sum, one party holds item-weight pairs $\{(x_i, u_i)\}$, while the other holds an item set $\{y_j\}$. The goal is to compute the sum of values over the intersection, i.e., $\sum_{i:x_i \in \{y_j\}} u_i$, together with the intersection size $|\{x_i\} \cap \{y_j\}|$. They propose semi-honest protocols based on a Diffie-Hellman style double masking, random oblivious transfer and encrypted Bloom filters, respectively. Then they implemented the protocols and conducted detailed comparisons. We note that PI-Sum aggregates values contributed by only one party; in our framework we also consider a more general summation setting where both parties contribute weights.

Lepoint et al. [21] introduce the inner-product Private Join and Compute (inner-product PJC) functionality. Here both parties hold item-weight pairs, $\{(x_i, u_i)\}$ and $\{(y_j, v_j)\}$, and aim to compute the inner-product of values over the intersection, i.e., $\sum_{(i,j):x_i=y_j} u_i v_j$, together with the intersection size. Their protocol combines PIR techniques with additively homomorphic encryption and is particularly efficient in the unbalanced setting, where the receiver's set is much smaller than the sender's. Chida et al. [9] later design a communication-efficient four-round inner-product PJC protocol, which they implement and benchmark against prior work.

Beauregard et al. [1] first develop the protocol for privately finding the "best" item in the intersection under the score function given by the sum of both parties' weights. Under the DDH assumption, they construct a semi-honest secure protocol. However, their solution requires solving discrete logarithms to recover the weight sums, and thus only supports weights of polynomial size. Cai et al. [6] remove this restriction by using additively homomorphic encryption; their experiments suggest that while the communication cost is slightly higher, computation becomes faster when weight sums are large. However, both of them incur quadratic overhead in the set sizes, which limits scalability to large datasets.

A different strategy is to use circuit-PSI [15, 27], which allows parties to compute more general functions over intersection-related data by securely evaluating a Boolean circuit after private matching. This generic route can support rich functionalities, but typically comes with substantially higher communication and round complexity compared with protocols tailored to specific weighted-intersection tasks. Similarly, in [5, 23], the Meta/Facebook team proposes Private Matching for Compute (PMC) and its delegated variants, enabling parties to perform downstream secure computation on weights associated with matched items. In particular, PS^{3}I outputs additive secret shares of the intersection weights, while Private-ID first assigns pseudorandom UIDs over the union to align records and then hands the aligned data to generic MPC for further computation. As a result, one can in principle realize arbitrary complex functions via generic MPC, at the cost of additional generic MPC overhead.

Most recently, Koirala et al. [19] propose the Encrypted Label Selection and Analytics (ELSA) functionality, which enables a querier to obtain an agreed-upon function of the weight associated with a queried item y held by other parties. They give a semi-honest secure construction based on fully homomorphic encryption and support floating-point computation. However, ELSA is essentially a singleton-query primitive that performs downstream computation for a single queried item, rather than a set-intersection functionality as considered in our work.

2 Preliminaries

In this section, we introduce the notation and primitives used throughout our protocols, including Secure Integer Comparison, Multi-query Reverse Private Membership Test, and Oblivious Key-Value Store. The definitions of semi-honest security, additively homomorphic encryption, and Oblivious Transfer are deferred to Appx. A.

2.1 Notation

We use κ, λ to denote the computational and statistical security parameters, respectively. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a vector $\overrightarrow{v}$, we use v_i to denote its i-th element. For a binary vector $\overrightarrow{a}$, we write $|\overrightarrow{a}|$ for its Hamming weight, i.e., the number of 1's in $\overrightarrow{a}$. For some set S, the notation $s \leftarrow S$ means that s is assigned a uniformly random element from S. We write $\mathrm{U}(S)$ for the uniform distribution on the set S. We use $\mathbf{1}_{a<b}$ to denote the indicator function, which equals 1 if $a < b$ and 0 otherwise. For distributions X and Y, we use $X \overset{\mathrm{c}}{\approx} Y$ to denote computational indistinguishability. By $\mathrm{negl}(\kappa)$ we denote a negligible function.

2.2 Secure Integer Comparison

Secure integer comparison, also known as "Yao's Millionaires' Problem," was introduced by Yao [32] in 1982 and is perhaps the most well-known example of two-party secure computation. In this problem, Alice and Bob hold integers a and b, respectively, and at the end Alice learns the bit $\mathbf{1}_{a<b}$. The ideal functionality $\mathcal{F}_{\mathrm{SIC}}$ is shown in Fig. 4.

FUNCTIONALITY:
Inputs: There are two parties: Alice and Bob. Alice inputs an integer a, and Bob inputs an integer b.
Outputs: Alice outputs $\mathbf{1}_{a<b}$.

Figure 4: Ideal functionality $\mathcal{F}_{\mathrm{SIC}}$ for secure integer comparison.

Secure integer comparison is typically implemented via relatively lightweight comparison circuits. In our implementation, we use OT extension to generate the correlated randomness needed for evaluating AND gates, and then evaluate the comparison circuit under the GMW protocol [22] using only symmetric-key operations.

2.3 Multi-query Reverse Private Membership Test

Multi-query reverse private membership test (mq-RPMT) [34] is a functionality where the receiver holds an input vector $(x_1, \ldots, x_{n_1})$ and the sender holds a set Y. As a result, the receiver learns only the bit vector $\overrightarrow{a} = (a_1, \ldots, a_{n_2})$, where e_i indicates whether $y_i \in X$. The ideal functionality $\mathcal{F}_{\mathrm{mqRPMT}}$ is presented in Fig. 5.

The notion of mq-RPMT was first introduced by Zhang et al. [34], who proposed two constructions: one based on symmetric-key

FUNCTIONALITY:

Inputs: There are two parties: the Sender $\mathcal{S}$ and the Receiver $\mathcal{R}$. $\mathcal{S}$ inputs a set $Y = \{y_1, \ldots, y_{n_2}\}$, while $\mathcal{R}$ inputs a set $X = \{x_1, \ldots, x_{n_1}\}$.

Outputs: $\mathcal{R}$ outputs a vector $\overrightarrow{a} = \{a_1, \ldots, a_{n_2}\} \in \{0,1\}^{n_2}$, where $a_i = 1$ if $y_i \in X$, and $a_i = 0$ otherwise.

Figure 5: Ideal functionality $\mathcal{F}_{\text{mq-RPMT}}$ for multi-query reverse private membership test.

encryption combined with general 2PC techniques, and another based on re-randomizable public-key encryption. Subsequently, Chen et al. [8] presented more efficient mq-RPMT protocols under the DDH assumption.

2.4 Oblivious Key-Value Store (OKVS)

Definition 2.1. A **Key-Value Store (KVS)** is parameterized by a set $\mathcal{K}$ of keys, a set $\mathcal{V}$ of values, and consists of two algorithms:

- Encode: Takes as input a set of (k_i, v_i) key-value pairs and outputs an object T (or, with statistically small probability, an error indicator $\perp$).
- Decode: Takes as input an object T, a key k, and outputs a value v.

A KVS is correct if, for all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys:

$$(k, v) \in A \text{ and } \perp \neq T \leftarrow \mathsf{Encode}(A) \Longrightarrow \mathsf{Decode}(T, k) = v.$$

The concept of an oblivious key-value store (OKVS) was introduced by Garimella et al. [13] to capture the properties of data structures commonly used in PSI protocols. At a high level, an OKVS enables the encoding of n pairs of key-value pairs in such a way that an adversary cannot reverse-engineer the original input keys from the encoding, assuming the input values are random.

Definition 2.2. A KVS is an **oblivious KVS (OKVS)** if, for all distinct $\{k_1^0, \ldots, k_n^0\}$ and distinct $\{k_1^1, \ldots, k_n^1\}$, if Encode does not output $\perp$ for $(k_1^0, \ldots, k_n^0)$ or $(k_1^1, \ldots, k_n^1)$, then the output of $\mathsf{Exp}^{\mathcal{A}}(\mathcal{K} = (k_1^0, \ldots, k_n^0))$ is computationally indistinguishable to that of $\mathsf{Exp}^{\mathcal{A}}(\mathcal{K} = (k_1^1, \ldots, k_n^1))$, where:

$\mathsf{Exp}^{\mathcal{A}}(\mathcal{K} = (k_1, \ldots, k_n))$:
(1) for $i \in [n]$: choose uniform $v_i \leftarrow \mathcal{V}$
(2) return $\mathcal{A}(\mathsf{Encode}(\{(k_1, v_1), \ldots (k_n, v_n)\}))$

In our construction, we need to use the additional property of random decodings, where the decoded value for a non-input key must be indistinguishable from a uniformly random element from $\mathcal{V}$.

Definition 2.3. An OKVS satisfies **random decoding** if, for all sets of n distinct keys $\{k_1, \ldots, k_n\} \subseteq \mathcal{K}$, n values $v_1, \ldots, v_m$ each drawn uniformly at random from $\mathcal{V}$, the output of $\mathsf{Decode}(T, k)$ for $k \notin \{k_1, \ldots, k_n\}$ is statistically indistinguishable from a uniformly random element in $\mathcal{V}$ where $T \leftarrow \mathsf{Encode}(\{(k_1, v_1), \ldots, (k_n, v_n)\})$.

FUNCTIONALITY:

Inputs: Alice inputs a set of item-weight pairs $\{(x_1, u_1), \ldots, (x_{n_1}, u_{n_1})\}$ and a public key pk. Bob inputs a set $\{y_1, \ldots, y_{n_2}\}$. ($x_i \neq x_j$ and $y_i \neq y_j$ for $i \neq j$)

Outputs: Bob outputs a vector $\overrightarrow{b} = (b_1, \ldots, b_{n_2}) \in C^{n_2}$, where C is the ciphertext space of the encryption scheme. For each $i \in [n_2]$, $b_i \leftarrow \mathsf{Enc}_{pk}(u_j)$ if $y_i = x_j$ for some $j \in [n_1]$, and $b_i \leftarrow C$ otherwise.

Figure 6: Ideal functionality $\mathcal{F}_{\text{OEWT}}$ for oblivious encrypted weight transfer.

3 Oblivious Encrypted Weight Transfer

3.1 Ideal Functionality

In the functionality $\mathcal{F}_{\text{OEWT}}$ (Fig. 6), Bob receives an encryption of the corresponding weight for items in the intersection, and a random ciphertext for non-intersection items. By the IND-CPA security of the encryption scheme, Bob is oblivious to which case occurs.

Specifically, Alice inputs the item–weight pairs $\{(x_i, u_i)\}_{i \in [n_1]}$ and a public key pk. Bob inputs an item set $\{y_i\}_{i \in [n_2]}$. Denote C as the ciphertext space of the encryption scheme, Bob obtains a vector $\overrightarrow{b} \in C^{n_2}$, where

$$b_i \leftarrow \begin{cases} \mathsf{Enc}_{pk}(u_j), & \text{if } \exists\, j \in [n_1] \text{ s.t. } y_i = x_j, \\ \mathrm{U}(C), & \text{otherwise.} \end{cases}$$

3.2 Oblivious Encrypted Weight Transfer Protocol

We construct our protocol using an OKVS. A natural first approach is to take Alice's items as OKVS keys and the encryptions of the corresponding weights as values: Alice encodes these key–value pairs and sends the resulting OKVS to Bob. When Bob decodes on input y_i, if $y_i = x_j$ for some $j \in [n_1]$, he obtains the ciphertext $\mathsf{Enc}_{pk}(u_j)$ associated with x_j; if $y_i \notin X$, then by the random-decoding property, he obtains a uniformly random element in the value space.

However, this straightforward approach is incompatible with the OKVS randomness requirement. The obliviousness property demands that the encoded values be uniformly random over the value space, and OKVS constructions typically assume that this value space is a finite field.[2] In contrast, ciphertexts of public-key encryption scheme do not, in general, follow the uniform distribution over such a field. Furthermore, a uniformly random field element obtained via random decoding does not necessarily correspond to a random ciphertext.

To resolve this mismatch, we introduce the notion $\mathbb{Z}_\beta$*-uniformly embeddable* encryption, which specifies how ciphertexts can be encoded into an integer ring $\mathbb{Z}_\beta$ while preserving computational pseudouniformity. We show that both Paillier and standard RLWE-based public-key encryption satisfy this notion. We then lift the resulting ring-level embedding to an appropriate finite-field value

[2] In fact, a few OKVS constructions, such as RB-OKVS [2], theoretically support value spaces that form finite abelian groups at the abstract level. In practice, however, implementations almost always instantiate the value space as a characteristic-two finite field, since it offers substantially better efficiency.

space, again preserving pseudouniformity, which yields our final protocol construction.

Definition 3.1 ($\mathbb{Z}_\beta$-uniformly embeddable). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an public-key encryption scheme with plaintext space $\mathcal{M}$, and ciphertext space C. We say that Π is $\mathbb{Z}_\beta$-uniformly embeddable if there exist a modulus $\beta \in \mathbb{N}$ and PPT algorithms

$$\tau : C \to \mathbb{Z}_\beta, \qquad \tau^{-1} : \mathbb{Z}_\beta \to C \cup \{\perp\},$$

such that:

(1) **Injectivity.** τ is injective and, for all $c \in C$, $\tau^{-1}(\tau(c)) = c$. For any $x \notin \mathrm{Im}(\tau)$ we have $\tau^{-1}(x) = \perp$.
(2) **Pseudouniformity.** Given the public key pk. For every message $m \in \mathcal{M}$ and every PPT distinguisher $\mathcal{D}$,

$$\left| \Pr_{c \leftarrow \mathsf{Enc}_{pk}(m)} \left[\mathcal{D}(pk, \tau(c)) = 1\right] - \Pr_{\gamma \leftarrow \mathrm{U}(\mathbb{Z}_\beta)} \left[\mathcal{D}(pk, \gamma) = 1\right] \right| \le \mathrm{negl}(\kappa)$$

In this case, we call τ a $\mathbb{Z}_\beta$-uniform ciphertext encoding for Π.

Remark 3.2. By the pseudouniformity definition, a random $x \in \mathbb{Z}_\beta$ lies in $\mathrm{Im}(\tau)$ with all but negligible probability; in other words, the probability that τ^{-1} outputs $\perp$ is negligible.

The *injectivity* condition in Def. 3.1 guarantees that ciphertexts can be efficiently recovered from their encodings, while *pseudouniformity* ensures that, for any adversary not holding the secret key sk, the image of a ciphertext under τ is computationally indistinguishable from a uniformly random element of $\mathbb{Z}_\beta$. As a concrete example, we now show that Paillier encryption satisfies this definition.

Theorem 3.3. *Paillier encryption is $\mathbb{Z}_{N^2}$-uniformly embeddable, where N denotes the Paillier modulus.*

It can be proved under the Decisional Composite Residuosity (DCR) assumption. A detailed proof is provided in Appx. C.1.

We further show that, for RLWE-based public-key encryption, there exists a modulus β such that the scheme is $\mathbb{Z}_\beta$-uniformly embeddable. Let $R_q = \mathbb{Z}_q[x]/(g(x))$ be a polynomial ring of degree n, so that a ciphertext $(c_0, c_1) \in R_q^2$ can be viewed as a vector in $\mathbb{Z}_q^{2n}$ via its coefficient representation. We define τ by packing these $2n$ coefficients in base q into a single integer in $\mathbb{Z}_{q^{2n}}$ (with the obvious base-q decoding as τ^{-1}); under the standard RLWE assumption, the resulting encoding $\tau(c)$ of any ciphertext c is computationally indistinguishable from uniform over $\mathbb{Z}_{q^{2n}}$. Thus, RLWE-based encryption is $\mathbb{Z}_{q^{2n}}$-uniformly embeddable.

We now use a $\mathbb{Z}_\beta$-uniformly embeddable encryption scheme as a starting point to lift from the integer ring $\mathbb{Z}_\beta$ to the prime field $\mathbb{Z}_p$, while preserving uniformity. Our key observation is that a uniform distribution over $\mathbb{Z}_\beta$ can be "spread" to the interval $\mathbb{Z}_{\left\lfloor \frac{p}{\beta} \right\rfloor \cdot \beta} \subseteq \mathbb{Z}_p$ by adding a random multiple of β: the resulting distribution has statistical distance from the uniform distribution over $\mathbb{Z}_p$ equal to the tail probability $\Pr[z \in \{\left\lfloor \frac{p}{\beta} \right\rfloor \cdot \beta, \ldots, p-1\}]$, which is at most $2^{-\lambda}$ whenever $p \ge 2^\lambda \cdot \beta$.

Concretely, Alice samples $r_i \leftarrow \{0, \ldots, \left\lfloor \frac{p}{\beta} \right\rfloor - 1\}$ for each $i \in [n_1]$, and forms the key–value pairs

$$Q = \left\{ \left(x_i,\ \tau(\mathsf{Enc}_{pk}(u_i)) + r_i\beta\right) \right\}_{i \in [n_1]} \subseteq \mathbb{Z}_p.$$

PROTOCOL:

Inputs: Alice inputs a set of item-weight pairs $\{(x_1, u_1), \ldots, (x_{n_1}, u_{n_1})\}$ and public key pk. Bob inputs a set $\{y_1, \ldots, y_{n_2}\}$.

Parameters: The computational and statistical security parameters κ, λ. The OKVS scheme (Encode, Decode) with the value space $\mathcal{V} = \mathbb{Z}_p$, where p is a prime number. A public-key encryption scheme Π with ciphertext space C. It admits a $\mathbb{Z}_\beta$-uniform ciphertext encoding $\tau : C \to \mathbb{Z}_\beta$. It holds that $p \ge 2^\lambda \cdot \beta$.

Protocol Steps:

(1) For $i \in [n_1]$, Alice randomly samples $r_i \leftarrow \{0, \ldots, \left\lfloor \frac{p}{\beta} \right\rfloor - 1\}$, computes $c_i \leftarrow \mathsf{Enc}_{pk}(u_i)$ and $t_i = \tau(c_i) \in \mathbb{Z}_\beta$.
(2) Alice computes key–value pairs

$$Q = \{(x_i, t_i + r_i \cdot \beta)\}_{i \in [n_1]},$$

and encodes the OKVS as $T \leftarrow \mathsf{Encode}(Q)$. She then sends T to Bob.
(3) For each $i \in [n_2]$, Bob decodes the OKVS as $z_i = \mathsf{Decode}(T, y_i) \in \mathbb{Z}_p$ and computes $b_i' = z_i \bmod \beta \in \mathbb{Z}_\beta$. He then sets $b_i = \tau^{-1}(b_i')$.
(4) Bob outputs $\overrightarrow{b} = (b_1, \ldots, b_{n_2})$.

Figure 7: The oblivious encrypted weight transfer protocol.

She encodes Q into an OKVS and sends to Bob. Bob decodes on his items y_i, reduces the decoded value modulo β, and applies τ^{-1} to recover the corresponding ciphertexts. The full protocol is given in Fig. 7.

We formalize the above statistical-distance argument in Lem. 3.4, and then, in Thm. 3.5, we prove that the randomness of the values stored in the OKVS; and finally, in Thm. 3.6, we establish the security of our OEWT protocol.

Lemma 3.4. *Let r be uniformly sampled from $\{0, \ldots, \lfloor \frac{p}{\beta} \rfloor - 1\}$, and let γ be uniformly sampled from $\{0, \ldots, \beta - 1\}$. If $p \ge 2^\lambda \cdot \beta$, then the distribution of $\zeta = \gamma + r \cdot \beta$ is statistically indistinguishable from the uniform distribution over $\{0, \ldots, p-1\}$.*

The proof requires only simple probabilistic analysis. The detailed proof is provided in Appx.C.1.

Theorem 3.5 (The randomness of the OKVS values). *If the encryption scheme Π is $\mathbb{Z}_\beta$-uniformly embeddable (via (τ, τ^{-1})), then in Fig. 7 the values stored in the OKVS T are computationally indistinguishable from the uniform distribution over $\mathbb{Z}_p$.*

This follows from Lem. 3.4 and the definition of $\mathbb{Z}_\beta$-uniform embeddability. The detailed proof is provided in Appx.C.1.

Theorem 3.6. *Assume that the OKVS satisfies the random decoding property, the protocol in Fig. 7 securely realizes the functionality $\mathcal{F}_{\mathrm{OEWT}}$ in the semi-honest model.*

The detailed proof is provided in Appx.C.1.

Transfer to binary OKVS. In practice, OKVS instantiations with value space $\mathbb{F}_{2^m}$ are often more efficient, since operations are mainly bit-wise (such as XOR), whereas arithmetic over $\mathbb{Z}_p$ is typically slower. We can therefore further improve the efficiency of our protocol by moving the value space from $\mathbb{Z}_p$ to $\mathbb{F}_{2^m}$. Concretely, in

the protocol of Fig. 7, the OKVS values are represented as m-bit strings (the binary encoding of elements of $\mathbb{F}_{2^m}$) before encoding, and after decoding, these bit strings are reinterpreted as integers for the subsequent computations. As our previous argument shows, as long as $|\mathbb{F}_{2^m}| = 2^m \geq 2^\lambda \cdot \beta$, the pseudorandomness of the OKVS values still holds.

Discussion. In recent fuzzy PSI constructions [12, 31], PKE ciphertexts are also encoded as OKVS values. When implementing their PKE schemes with ECC-ElGamal over Curve25519, the group elements (ciphertexts) are represented as 256-bit Ristretto encodings which are fed to the OKVS as values. However, if the OKVS is instantiated over the full space of 256-bit strings, then by decoding property, a non-encoded key yields a uniformly random 256-bit string. With overwhelming probability, it does not correspond to a valid Ristretto/ElGamal ciphertext encoding, which is incompatible with the security requirement. In contrast, our OEWT-based construction resolves this mismatch: by first embedding ciphertexts into an integer ring $\mathbb{Z}_\beta$ via a $\mathbb{Z}_\beta$-uniform ciphertext encoding and then lifting to a field $\mathbb{Z}_p$, we ensure that OKVS values are pseudouniform over $\mathbb{Z}_p$ and that non-key decodings remain computationally indistinguishable from embedded ciphertexts.

4 PFA-WSI with Predicate-Filtered Output

4.1 IMWS Protocol

We construct our IMWS protocol starting from the functionality $\mathcal{F}_{\text{OEWT}}$. In $\mathcal{F}_{\text{OEWT}}$, for each of Bob's items y_i ($i \in [n_2]$) he receives $b_i \leftarrow \mathsf{AEnc}_{pk}(u_j)$ if $y_i = x_j$ for some $j \in [n_1]$, and $b_i \leftarrow C$ otherwise. Using additively homomorphic operations, Bob can then compute

$$m_i = \mathsf{ARef}\big(\mathsf{ASum}(\{b_i, \mathsf{AEnc}_{pk}(v_i)\})\big),$$

which is a refreshed ciphertext of $u_j + v_i$ when $y_i = x_j$.

A naive approach would be for Bob to send m_i directly to Alice for decryption. However, in that case, Alice would decrypt m_i for all i, including indices with $y_i \notin X$, and thus learn additional information about Bob's non-intersection items. More concretely, in $\mathcal{F}_{\text{OEWT}}$ the ciphertexts that Bob receives on non-intersection items are modeled as random from Bob's point of view, but their randomness may be correlated with Alice's inputs. If Bob simply forwards these ciphertexts back to Alice, this correlation may leak extra information.

We leverage $\mathcal{F}_{\text{mq-RPMT}}$ and $\mathcal{F}_{\text{OT}}$ to avoid this leakage. Concretely, Alice and Bob first invoke $\mathcal{F}_{\text{mq-RPMT}}$ with inputs $X = \{x_i\}_{i\in[n_1]}$ from Alice and $Y = \{y_j\}_{j\in[n_2]}$ from Bob. Alice receives the indicator vector $\{\mathbf{1}_{y_i \in X}\}_{i\in[n_2]}$, aligned with Bob's ordering of the items. Next, as above, they invoke $\mathcal{F}_{\text{OEWT}}$ and Bob locally computes m_i for all $i \in [n_2]$. They then invoke $\mathcal{F}_{\text{OT}}$ with Alice as the receiver and Bob as the sender: for each i, Bob's input is the pair $(\perp, m_i)$ and Alice's choice bit is $\mathbf{1}_{y_i \in X}$. If $y_i \in X$, Alice receives m_i, decrypts it, and obtains $w_i = u_j + v_i$; if $y_i \notin X$, she instead receives $\perp$ and sets $w_i = \perp$. Finally, Alice identifies the index maximizing w_i and sends it to Bob, and Bob outputs the corresponding item.

The detailed IMWS protocol is presented in Fig. 8.

Theorem 4.1. *Assume that the additively homomorphic encryption scheme is IND-CPA secure, the protocol in Fig. 8 securely realizes the functionality $\mathcal{F}_{\text{IMWS}}$ in the $\mathcal{F}_{\text{mq-RPMT}}$, $\mathcal{F}_{\text{OEWT}}$, $\mathcal{F}_{\text{OT}}$ hybrid model in the presence of semi-honest adversaries.*

PROTOCOL:

Inputs: Alice has input set $\{(x_i, u_i)\}_{i\in[n_1]}$ and Bob has input set $\{(y_j, v_j)\}_{j\in[n_2]}$.

Parameters: The computational and statistical security parameters κ, λ. The additively homomorphic encryption scheme (AGen, AEnc, ADec, ASum, ARef) which is $\mathbb{Z}_\beta$-uniformly embeddable. The functionalities $\mathcal{F}_{\text{mq-RPMT}}$, $\mathcal{F}_{\text{OEWT}}$, $\mathcal{F}_{\text{OT}}$.

Protocol Steps:

(1) Alice generates an additively homomorphic encryption keypair $(pk, sk) \leftarrow \mathsf{AGen}(\kappa)$ and sends pk to Bob.
(2) Alice and Bob invoke $\mathcal{F}_{\text{mq-RPMT}}$ with inputs $\{x_i\}_{i\in[n_1]}$ from Alice and $\{y_j\}_{j\in[n_2]}$ from Bob. The functionality returns to Alice the indicator vector $\overrightarrow{a} = (a_1, \ldots, a_{n_2})$.
(3) Alice and Bob invoke $\mathcal{F}_{\text{OEWT}}$ with inputs $(\{(x_i, u_i)\}_{i\in[n_1]}, pk)$ from Alice and $\{y_j\}_{j\in[n_2]}$ from Bob. The functionality returns to Bob the vector $\overrightarrow{b} = (b_1, \ldots, b_{n_2})$.
(4) For $i \in [n_2]$, Bob computes
$$m_i = \mathsf{ARef}(\mathsf{ASum}(\{b_i, \mathsf{AEnc}_{pk}(v_i)\})).$$
(5) Alice and Bob invoke the 1-out-of-2 OT functionality $\mathcal{F}_{\text{OT}}$ n_2 times, with Alice acting as the receiver with input bit a_i during the i-th invocation, and Bob as the sender with input $(\perp, m_i)$. Denote Alice's output as c_i.
(6) For $i \in [n_2]$, if $c_i = \perp$, let $w_i = \perp$. Otherwise, Alice decrypts c_i using sk and sets $w_i = \mathsf{ADec}_{sk}(c_i)$.
(7) Alice outputs $(w_1, \ldots, w_{n_2})$, and sends $j^* := \arg\max_{j\in[n_2]} w_j$ to Bob. Then Bob outputs y_{j^*}.

Figure 8: The IMWS protocol.

The detailed proof is provided in Appx. C.2.

4.2 Complexity Analysis

In our IMWS protocol, invoking $\mathcal{F}_{\text{mq-RPMT}}$ from Chen et al. [8] incurs linear communication and computation overhead to both parties' input size. When invoking our OEWT protocol in Fig. 7, Alice is required to encrypt n_1 ciphertexts, encode an OKVS with n_1 key-value pairs, and send it to Bob, who decodes it using n_2 keys as the vector $\overrightarrow{b}$. For $i \in [n_2]$, Bob adds b_i and v_i via the homomorphic encryption scheme and re-randomizes those ciphertexts. Both parties will then invoke n_2 1-out-of-2 OTs, and Alice will decrypt the received ciphertexts. The total communication of our protocol scales linearly with the input sizes of both parties, whereas existing IMWS protocols from Beauregard et al.[1] and Cai et al. [6] require quadratic communication complexity for Alice. The detailed communication and computation of our IMWS protocol are shown in Tab. 1.

4.3 Other Variants

In this section, we consider other predicate-filtered output protocols, illustrating the generality of our construction.

Different predicates P. In $\mathcal{F}_{\text{IMWS}}$, we only considered the predicate that selects the item with the largest weight sum. In practice,

Table 1: Communication and computation of our IMWS protocol and the protocols in [1, 6]. Let n_1, n_2 be the input sizes of Alice and Bob.

Protocol	Communication	Computation	
		Alice	Bob
FOCI [1]	$O(n_1 + n_2)$	$O(n_1 n_2)$	$O(n_1 + n_2)$
OPRF-based [6]	$O(n_1 + n_2)$	$O(n_1 n_2)$	$O(n_1 + n_2)$
DDH-based [6]	$O(n_1 + n_2)$	$O(n_1 n_2)$	$O(n_1 + n_2)$
Ours (Fig. 8)	$O(n_1 + n_2)$	$O(n_1 + n_2)$	$O(n_1 + n_2)$

however, other predicates are common, such as selecting the minimum, or returning all elements whose weight exceeds a given threshold. Our protocol directly supports any such predicate-based variant: once Alice has obtained $(w_1, \ldots, w_{n_2})$, she simply applies the desired predicate P to compute the set of indices and sends these indices to Bob, who outputs the corresponding predicate-filtered items.

Different functions f. In $\mathcal{F}_{\text{IMWS}}$, we only considered the function $f(u_i, v_j) = u_i + v_j$. In many applications, other functions f are more appropriate. For example, when computing joint probabilities, one should use the product $f(u_i, v_j) = u_i \cdot v_j$. Realizing this functionality only requires replacing the homomorphic addition in Step 4 by homomorphic plaintext–ciphertext multiplication, i.e., computing $m_i \leftarrow \mathsf{ARef}\big(\mathsf{AMultP}(v_i, b_i)\big)$.

In interest-matching scenarios, Alice and Bob may each input a set of (hobby, rating) pairs as item–weight pairs, and wish to identify all hobbies whose ratings are sufficiently close. In this case, the natural score is $f(u_i, v_j) = |u_i - v_j|$. To realize this functionality, Bob can no longer compute a ciphertext of $f(u_i, v_j)$ locally as in the additive case; instead, we design an interactive protocol in which Alice and Bob run a few rounds of communication and Bob eventually obtains a ciphertext of $f(u_i, v_j)$, while Alice learns nothing. The concrete construction appears in the next section, in the "Function-Specific Homomorphic Evaluation" phase of Fig. 11.

5 PFA-WSI with Aggregated Output

In this section, we construct PFA-WSI protocols with aggregated output. We first introduce the building block named Secure Indicator Selected Aggregation (SISA) in Sec. 5.1, which allows Alice to securely obtain an aggregate of the selected Bob's ciphertexts. Then, in Sec. 5.2, we use $\mathcal{F}_{\text{OEWT}}$ together with $\mathcal{F}_{\text{SISA}}$ to construct PFA-WSI protocols with aggregated output for $f \in \{\mathrm{D}_{L_2}, \mathrm{D}_{L_1}\}$. Finally, in Sec. 5.3 we analyze the theoretical complexity of our constructions.

5.1 Secure Indicator-Selected Aggregation

In the functionality $\mathcal{F}_{\text{SISA}}$ (Fig. 9), Alice holds the secret key of an additively homomorphic encryption scheme and an indicator vector $\overrightarrow{a} \in \{0,1\}^{n_2}$, where entries equal to 1 mark the positions to be aggregated. Bob holds a ciphertext vector $\overrightarrow{b} = \{\mathsf{AEnc}_{pk}(v_1), \ldots, \mathsf{AEnc}_{pk}(v_{n_2})\}$. The functionality outputs Alice the sum of Bob's weights selected by her indicators, namely $S = \sum_{\substack{i \in [n_2] \\ a_i = 1}} v_i$.

FUNCTIONALITY:

Inputs: Alice inputs the indicator vector $\overrightarrow{a} \in \{0,1\}^{n_2}$ and an additively homomorphic encryption key pair (pk, sk). Bob inputs the public key pk and a ciphertext vector $\overrightarrow{b} = \{\mathsf{AEnc}_{pk}(v_1), \ldots, \mathsf{AEnc}_{pk}(v_{n_2})\}$.

Outputs: Alice outputs $S = \sum_{\substack{i \in [n_2] \\ a_i = 1}} v_i$.

Figure 9: Ideal functionality $\mathcal{F}_{\text{SISA}}$ for secure indicator-selected aggregation.

PROTOCOL:

Inputs: Alice inputs the indicator vector $\overrightarrow{a} \in \{0,1\}^{n_2}$ and an additively homomorphic encryption key pair (pk, sk). Bob inputs the public key pk and a ciphertext vector $\overrightarrow{b} = \{\mathsf{AEnc}_{pk}(v_1), \ldots, \mathsf{AEnc}_{pk}(v_{n_2})\}$.

Parameters: The computational and statistical security parameters κ, λ. The additively homomorphic encryption scheme (AGen, AEnc, ADec, ASum, ARef) with plaintext space $\mathcal{M} = \mathbb{Z}_\alpha$.

Protocol Steps:

(1) Bob samples n_2 random values $\{r_i \in \mathbb{Z}_\alpha\}_{i \in [n_2]}$ and computes $m_{i,0} = r_i$ and $m_{i,1} = \mathsf{ARef}(\mathsf{ASum}(\{b_i, \mathsf{AEnc}_{pk}(r_i)\}))$.

(2) Alice and Bob invoke the 1-out-of-2 OT functionality $\mathcal{F}_{\text{OT}}$ n_2 times, with Alice acting as the receiver with input bit a_i during the i-th invocation, and Bob as the sender with input $(m_{i,0}, m_{i,1})$. Denote Alice's output as c_i.

(3) Bob computes $R = \sum_{i=1}^{n_2} r_i$ and sends it to Alice.

(4) Alice computes $S = \sum_{\substack{i \in [n_2] \\ a_i = 1}} \mathsf{ADec}_{sk}(c_i) + \sum_{\substack{i \in [n_2] \\ a_i = 0}} c_i - R$ and outputs S.

Figure 10: The secure indicator-selected aggregation protocol.

We realize $\mathcal{F}_{\text{SISA}}$ using random masking combined with Oblivious Transfer. Concretely, for each $i \in [n_2]$, Bob samples a random mask $r_i \leftarrow \mathbb{Z}_\alpha$ and defines

$$m_{i,0} = r_i, \qquad m_{i,1} = \mathsf{ARef}\big(\mathsf{ASum}(\{b_i, \mathsf{AEnc}_{pk}(r_i)\})\big).$$

Alice and Bob then invoke $\mathcal{F}_{\text{OT}}$, where Alice acts as the receiver with choice bit a_i, and Bob acts as the sender with input pair $(m_{i,0}, m_{i,1})$. Let denote Alice's OT output as c_i. Bob then sends $R = \sum_{i=1}^{n_2} r_i$ to Alice. Alice then computes and outputs

$$S = \sum_{\substack{i \in [n_2] \\ a_i = 1}} \mathsf{ADec}_{sk}(c_i) + \sum_{\substack{i \in [n_2] \\ a_i = 0}} c_i - R.$$

The detailed SISA protocol is presented in Fig. 10.

Correctness follows directly from the OT functionality: for each $i \in [n_2]$, if $a_i = 0$ then $c_i = r_i$, and if $a_i = 1$ then $c_i = \mathsf{ARef}(\mathsf{ASum}(b_i, \mathsf{AEnc}_{pk}(r_i)))$. Therefore,

$$\begin{aligned} S &= \sum_{\substack{i \in [n_2] \\ a_i = 1}} (v_i + r_i) + \sum_{\substack{i \in [n_2] \\ a_i = 0}} r_i - \sum_{i=1}^{n_2} r_i \\ &= \sum_{\substack{i \in [n_2] \\ a_i = 1}} v_i \end{aligned}$$

which matches the functionality of $\mathcal{F}_{\text{SISA}}$.

Theorem 5.1. *Assume that the additively homomorphic encryption scheme is IND-CPA secure, the protocol in Fig. 10 securely realizes the functionality $\mathcal{F}_{\text{SISA}}$ in the $\mathcal{F}_{\text{OT}}$ hybrid model in the presence of semi-honest adversaries.*

The detailed proof is provided in Appx. C.3.

5.2 PFA-WSI Protocols with Aggregated Output

We construct our PFA-WSI protocols with aggregated output from $\mathcal{F}_{\text{OEWT}}$ and $\mathcal{F}_{\text{SISA}}$. The protocol consists of three phases.

- In the *mq-RPMT and Encrypted Intersection Weight Transfer* phase, the same as the IMWS protocol: Alice and Bob invoke $\mathcal{F}_{\text{mq-RPMT}}$ and $\mathcal{F}_{\text{OEWT}}$, Alice obtains the indicator vector $\overrightarrow{a}$ and Bob obtains the ciphertext vector $\overrightarrow{b}$.
- In the *Function-Specific Homomorphic Evaluation* phase, depending on the chosen function f, Alice and Bob run a few rounds of interaction after which Bob holds ciphertexts encrypting $f(u_i, v_j)$ for the intersection pairs, while Alice learns nothing.
- In the *Secure Intersection Aggregation* phase, Alice uses $\overrightarrow{a}$ as her selection vector and invokes $\mathcal{F}_{\text{SISA}}$ to aggregate Bob's ciphertexts from the previous phase, obtaining the final result $S^{(f)}$.

Thus, the main design challenge is to construct the *Function-Specific Homomorphic Evaluation* phase for different functions f. We remark that, in principle, a fully homomorphic encryption scheme would allow us to implement any arithmetic f, but at the cost of a significant loss in efficiency. Our goal instead is to realize these functionalities using only an efficient additively homomorphic encryption scheme. In what follows, we give concrete instantiations for $f \in \{\mathrm{D}_{L_2}, \mathrm{D}_{L_1}\}$. The L_2 distance penalizes large discrepancies more heavily through squaring, whereas the L_1 distance aggregates absolute differences and is typically more robust to outliers.

Case $f = \mathrm{D}_{L_2}$: In this case $f(u_i, v_j) = (u_i - v_j)^2$. Since an additively homomorphic encryption scheme cannot directly evaluate ciphertext–ciphertext products, Alice and Bob should interact to obtain a ciphertext of $f(u_i, v_j)$.

Concretely, for each $j \in [n_2]$, Bob samples a random mask t_j and computes

$$d_j = \mathsf{ASum}\big(\{\ b_j,\ \mathsf{AEnc}_{pk}(-v_j + t_j)\ \}\big) = \mathsf{AEnc}_{pk}(u_i - v_j + t_j),$$

re-randomizes it as d'_j, and sends d'_j to Alice. Alice decrypts, squares the plaintext, and re-encrypts, obtaining

$$e_j = \mathsf{AEnc}_{pk}\big((\mathsf{ADec}_{sk}(d'_j))^2\big) = \mathsf{AEnc}_{pk}\big((u_i - v_j + t_j)^2\big),$$

which she sends back to Bob. Bob then removes the effect of the mask using the identity $(u_i - v_j)^2 = (u_i - v_j + t_j)^2 + t_j^2 - 2t_j(u_i - v_j + t_j)$, and sets

$$\begin{aligned} c_j^{(\mathrm{D}_{L_2})} &= \mathsf{ASum}\Big(\big\{\ e_j,\ \mathsf{AEnc}_{pk}(t_j^2),\ \mathsf{AMultP}(d_j, -2t_j)\ \big\}\Big) \\ &= \mathsf{AEnc}_{pk}\big((u_i - v_j)^2\big). \end{aligned}$$

Case $f = \mathrm{D}_{L_1}$: In this case, $f(u_i, v_j) = |u_i - v_j|$. The main challenge is how to handle the absolute value without leaking the sign of $\Delta_j = u_i - v_j$. A direct approach would be to use the secure integer comparison functionality $\mathcal{F}_{\text{SIC}}$ to learn the sign of Δ_j, but this would reveal additional information about the relative magnitudes of the weights.

To avoid this leakage, we first randomize the sign. Bob samples a random mask t_j and a random sign $\tau_j \in \{-1, +1\}$, and sends Alice a ciphertext of the randomly flipped and masked difference:

$$\begin{aligned} e_j &= \mathsf{ARef}\Big(\mathsf{ASum}\big(\{\mathsf{AmultP}(\tau_j, d_j),\ \mathsf{AEnc}_{pk}(t_j)\}\big)\Big) \\ &= \mathsf{AEnc}_{pk}(\tau_j \Delta_j + t_j). \end{aligned}$$

Alice decrypts this value and, together with Bob's t_j, they invoke $\mathcal{F}_{\text{SIC}}$ to learn only the bit $\eta_j = \mathbf{1}_{\tau_j \Delta_j < 0}$, which does not reveal the true sign of Δ_j due to the random τ_j.

Given η_j, we then let Alice obliviously obtain a masked encryption of $|\Delta_j|$ via OT. Bob samples another random mask s_j and computes

$$D_j^+ = \mathsf{AEnc}_{pk}(\Delta_j + s_j), \qquad D_j^- = \mathsf{AEnc}_{pk}(-\Delta_j + s_j).$$

Alice and Bob invoke $\mathcal{F}_{\text{OT}}$, where Alice is the receiver with choice bit η_j, and Bob is the sender with input

$$(m_0, m_1) = \begin{cases} (D_j^+, D_j^-), & \text{if } \tau_j = +1, \\ (D_j^-, D_j^+), & \text{if } \tau_j = -1. \end{cases}$$

A simple case analysis shows that, in all cases, Alice receives

$$m_{\eta_j} = \mathsf{AEnc}_{pk}(|\Delta_j| + s_j).$$

She re-randomizes m_{η_j} and sends it back to Bob. Bob finally removes the mask s_j and sets

$$c_j^{(\mathrm{D}_{L_1})} = \mathsf{ASum}\big(\{m_{\eta_j}, \mathsf{AEnc}_{pk}(-s_j)\}\big) = \mathsf{AEnc}_{pk}(|\Delta_j|),$$

which is an encryption of $f(u_i, v_j)$ as desired.

Our detailed aggregated output PFA-WSI protocol is presented in Fig. 11.

Theorem 5.2. *Assume that the additively homomorphic encryption scheme is IND-CPA secure. Then the protocol in Fig. 11 securely realizes $\mathcal{F}_{\text{PFA-WSI}}$ with aggregated output for $f \in \{\mathrm{D}_{L_2}, \mathrm{D}_{L_1}\}$, in the hybrid model with $\mathcal{F}_{\text{mq-RPMT}}$, $\mathcal{F}_{\text{OEWT}}$, $\mathcal{F}_{\text{SIC}}$, $\mathcal{F}_{\text{OT}}$ and $\mathcal{F}_{\text{SISA}}$ against semi-honest adversaries.*

The detailed proof is provided in Appx. C.4.

Other variants. Our protocol can be easily extended to handle intersection inner-product [9, 21] and intersection sum [16]. The detailed analysis and experimental results for these variants can be found in Appx. B.1.

Beyond $f \in \{\mathrm{D}_{L_2}, \mathrm{D}_{L_1}\}$, another widely used similarity measure is the cosine similarity, which is insensitive to differences in users' rating scales and instead captures similarity of rating patterns. Computing cosine similarity does not fit directly into the single-aggregate PFA-WSI functionality, but we can build a privacy-preserving protocol for cosine similarity by combining our protocols for intersection inner product and intersection sum. We present the concrete construction and its analysis in Appx. B.2.

5.3 Complexity Analysis

In Tab. 2, we provide a step-by-step analysis of the computation and communication complexity of the protocol in Fig. 11. Overall, each step of our protocol has linear complexity in the input sizes. The complexity of Step 2, which invokes $\mathcal{F}_{\text{mq-RPMT}}$, and Step 3, which invokes $\mathcal{F}_{\text{OEWT}}$, matches that of the corresponding steps in our IMWS

PROTOCOL:

Inputs: Alice has input set $\{(x_i, u_i)\}_{i\in[n_1]}$ and Bob has input set $\{(y_j, v_j)\}_{j\in[n_2]}$.

Parameters: The computational and statistical security parameters κ, λ. The additively homomorphic encryption scheme (AGen, AEnc, ADec, ASum, AMultP, ARef) with the plaintext space $\mathbb{Z}_\alpha$. The functionalities $\mathcal{F}_{\text{mq-RPMT}}$, $\mathcal{F}_{\text{OEWT}}$, $\mathcal{F}_{\text{SIC}}$, $\mathcal{F}_{\text{OT}}$ and $\mathcal{F}_{\text{SISA}}$.

Protocol Steps:

(1) Alice generates an additively homomorphic encryption key-pair $(pk, sk) \leftarrow \mathsf{AGen}(\kappa)$ and sends pk to Bob.

[mq-RPMT and Encrypted Intersection Weight Transfer]

(2) Alice and Bob invoke $\mathcal{F}_{\text{mq-RPMT}}$ with inputs $\{x_i\}_{i\in[n_1]}$ from Alice and $\{y_j\}_{j\in[n_2]}$ from Bob. The functionality returns to Alice the indicator vector $\overrightarrow{a}$.

(3) Alice and Bob invoke $\mathcal{F}_{\text{OEWT}}$ with inputs $(\{(x_i, u_i)\}_{i\in[n_1]}, pk)$ from Alice and $\{y_j\}_{j\in[n_2]}$ from Bob. The functionality returns to Bob the vector $\overrightarrow{b}$.

[Function-Specific Homomorphic Evaluation]

(4) Conditioned on the function f, Bob (with Alice when necessary) computes the ciphertexts $c_j^{(f)}$ and re-randomizes them with ARef.

Case $f = \mathrm{D}_{L_2}$**:**

(a) For $j \in [n_2]$, Bob samples random $t_j \leftarrow \mathbb{Z}_\alpha$ and computes $d_j = \mathsf{ASum}\Big(\{b_j, \mathsf{AEnc}_{pk}(-v_j + t_j)\}\Big)$ and sends $d'_j = \mathsf{ARef}(d_j)$ to Alice.

(b) Alice decrypts d'_j and sends Bob $e_j = \mathsf{AEnc}_{pk}((\mathsf{ADec}_{sk}(d'_j))^2)$.

(c) Bob computes $c_j^{(\mathrm{D}_{L_2})} = \mathsf{ASum}\Big(\big\{e_j, \mathsf{AEnc}_{pk}(t_j^2), \mathsf{AMultP}(d_j, -2t_j)\big\}\Big)$.

Case $f = \mathrm{D}_{L_1}$**:**

(a) For $j \in [n_2]$, Bob computes $d_j = \mathsf{ASum}(\{b_j, \mathsf{AEnc}_{pk}(-v_j)\} = \mathsf{AEnc}_{pk}(\Delta_j)$.

(b) Bob samples $t_j \leftarrow \mathbb{Z}_\alpha$ and a random sign $\tau_j \in \{-1, +1\}$, and sends Alice $e_j = \mathsf{ARef}\Big(\mathsf{ASum}\big(\{\mathsf{AmultP}(\tau_j, d_j), \mathsf{AEnc}_{pk}(t_j)\}\big)\Big) = \mathsf{AEnc}_{pk}(\tau_j\Delta_j + t_j)$.

(c) Alice computes $e'_j = \mathsf{ADec}_{sk}(e_j)$. Alice and Bob invoke $\mathcal{F}_{\text{SIC}}$ with inputs e'_j from Alice and t_j from Bob. Alice receives $\eta_j \in \{0, 1\}$ where $\eta_j = \mathbf{1}_{\tau_j\Delta_j<0}$.

(d) Bob samples $s_j \leftarrow \mathbb{Z}_\alpha$ and computes $D_j^+ = \mathsf{AEnc}_{pk}(\Delta_j + s_j)$, $D_j^- = \mathsf{AEnc}_{pk}(-\Delta_j + s_j)$. Alice and Bob invoke $\mathcal{F}_{\text{OT}}$, with Alice acting as the receiver with input bit η_j, and Bob as the sender with input

$$(m_0, m_1) = \begin{cases} (D_j^+, D_j^-), & \text{if } \tau_j = +1, \\ (D_j^-, D_j^+), & \text{if } \tau_j = -1. \end{cases}$$

Alice obtains $m_{\eta_j} = \mathsf{AEnc}_{pk}(|\Delta_j| + s_j)$, re-randomizes it and sends to Bob.

(e) Bob removes the mask and sets

$$c_j^{(\mathrm{D}_{L_1})} = \mathsf{ASum}(\{m_{\eta_j}, \mathsf{AEnc}_{pk}(-s_j)\}) = \mathsf{AEnc}_{pk}(|\Delta_j|).$$

[Secure Intersection Aggregation]

(5) Alice and Bob invoke $\mathcal{F}_{\text{SISA}}$ with inputs $(\overrightarrow{a}, (pk, sk))$ from Alice and $(pk, \overrightarrow{c}^{(f)})$ from Bob. The functionality returns $S^{(f)}$ to Alice. Alice outputs the intersection cardinality $|\overrightarrow{a}|$ and the aggregated value $S^{(f)}$ (or $\sqrt{S^{(f)}}$ when $f = \mathrm{D}_{L_2}$).

Figure 11: The PFA-WSI protocol with aggregated output.

protocol (Fig. 8). In contrast, the cost of the Function-Specific Homomorphic Evaluation phase depends on the instantiated function f. In this phase, implementing $f = \mathrm{D}_{L_1}$ incurs additional linear overhead compared with $f = \mathrm{D}_{L_2}$, since handling the absolute value requires a linear number of invocations of $\mathcal{F}_{\text{OT}}$ and $\mathcal{F}_{\text{SIC}}$. Finally, when realizing $\mathcal{F}_{\text{SISA}}$ via the protocol in Fig. 10, the overhead consists of a linear number of additively homomorphic encryption operations together with a linear number of invocations of $\mathcal{F}_{\text{OT}}$.

6 Implementation and Evaluation

6.1 Implementation Details

We implemented[3] our protocols in C++. All experiments were run on a desktop computer with an AMD 3950X CPU and 32GiB RAM.

We use RB-OKVS [2] as our Oblivious Key-Value Store, libOTe[4] for oblivious transfer, paillier cryptolib[5] for Paillier encryption (using the same parameters as in [6, 16]), and the Kunlun library[6] for mq-RPMT.

We report LAN measurements. For the WAN setting, we can estimate the runtime by translating the total communication volume into transmission time using the available bandwidth; the additional RTT overhead is negligible at our problem sizes and round complexity, so it does not materially affect the estimate.

We generate triples via IKNP OT extension [17] and then implement Secure Integer Comparison by evaluating a comparison circuit under the GMW protocol [22]; this choice incurs a relatively high linear communication cost in our L_1-distance protocol. While

[3] Our implementation is available at: https://github.com/lzjluzijie/pfa-wsi.

[4] https://github.com/osu-crypto/libOTe

[5] https://github.com/intel/paillier cryptolib

[6] https://github.com/yuchen1024/Kunlun

Table 2: Communication and computation costs of our PFA-WSI protocol for $f \in \{\mathrm{D}_{L_2}, \mathrm{D}_{L_1}\}$. Alice and Bob hold inputs of sizes n_1 and n_2, respectively. We use a $\mathbb{Z}_\beta$-uniformly embeddable AHE scheme with plaintext space $\mathbb{Z}_\alpha$, and denote by ϵ the OKVS expansion factor. Notation: Exp = group exponentiations; GE = group elements; AC = AHE ciphertexts; Comm(OT/SIC) and Comp(OT/SIC) denote the per-instance communication and computation of $\mathcal{F}_{\mathrm{OT}}$ and $\mathcal{F}_{\mathrm{SIC}}$.

Step		Communication	Computation	
			Alice	Bob
$\mathcal{F}_{\text{mq-RPMT}}$ [8]		$(2n_1 + n_2)$ GE	$(n_1 + n_2)$ Exp	$(n_1 + n_2)$ Exp
$\mathcal{F}_{\text{OEWT}}$ (Fig. 7)		$\epsilon n_1 \lceil \log \beta \rceil$ bits	n_1 AEnc 1 OKVS Encode	n_2 OKVS Decode
Function-Specific Homomorphic Evaluation	$f = \mathrm{D}_{L_2}$	$2n_2$ AC	n_2 ADec, AEnc	$3n_2$ ASum $2n_2$ AEnc n_2 ARef, AMultP
	$f = \mathrm{D}_{L_1}$	$2n_2$ AC n_2 Comm(OT) n_2 Comm(SIC)	n_2 ADec, ARef n_2 Comp(OT) n_2 Comp(SIC)	$5n_2$ AEnc $3n_2$ ASum n_2 ARef, AMultP n_2 Comp(OT) n_2 Comp(SIC)
$\mathcal{F}_{\text{SISA}}$ (Fig. 10)		n_2 Comm(OT) $+\lceil \log \alpha \rceil$ bits	n_2 ADec n_2 Comp(OT)	n_2 AEnc, ASum, ARef n_2 Comp(OT)

more advanced techniques like Silent OT [3] could reduce this overhead, we consider such optimizations orthogonal to our main contribution.

6.2 Benchmark Comparison

In Tab. 4, we report results for running our IMWS protocol, the DDH-based and OPRF-based protocols from [6], and the FOCI protocol from [1], all under the same experimental setup. We consider set sizes $n_1 = n_2 \in \{2^{12}, 2^{16}, 2^{20}\}$.

The results show that our protocol achieves the fastest runtime across all tested set sizes. In particular, due to its linear complexity, when $n_1 = n_2 = 2^{20}$ our protocol requires only 279.61 seconds of computation, whereas the prior protocols, which incur quadratic overhead, require more than 20,000 seconds. In terms of communication, our protocol is comparable to the lower-communication DDH-based protocol of [6], and reduces communication by roughly a factor of two compared with their OPRF-based protocol. On the other hand, our protocol incurs about 3× higher communication than the FOCI protocol of [1]. This is mainly because our construction transmits additively homomorphic ciphertexts, which are substantially larger than the elliptic-curve group elements exchanged in FOCI.

In Tab. 3, we report the runtime and communication of our protocols for $f \in \{\mathrm{D}_{L_1}, \mathrm{D}_{L_2}\}$ at set sizes $n_1 = n_2 \in \{2^{12}, 2^{16}, 2^{20}\}$. We break down the cost of each protocol component, including mq-RPMT, OEWT, the Function-Specific Homomorphic Evaluation phase (Func), and Secure Indicator-Selected Aggregation (SISA).

The empirical results align with our theoretical complexity analysis: each phase of our protocols scales linearly in the input sizes. As shown in the table, the computational costs of the D_{L_1} and D_{L_2} instantiations are comparable. However, D_{L_1} incurs substantially higher communication in the Func phase due to our IKNP OT approach for generating triples. Overall, for million-scale sets, both protocols complete in roughly half an hour, with communication of about 1 GB for D_{L_2} and about 7 GB for D_{L_1}, which is compatible with practical deployment. In particular, since multiplication triples can be generated in an offline preprocessing phase, the online communication can be substantially reduced.

We also implemented and evaluated our protocols for $f \in \{\text{IP}, \text{Sum}\}$ and compared their performance with related work; details are deferred to Appx. B.1.

7 Conclusion

In this work, we propose a unified framework for private computation on weighted set intersection. Our PFA-WSI functionality supports two output modes: predicate-filtered output and aggregated output. This functionality subsumes prior tasks such as PI-Sum, inner-product PJC, and IMWS, and extends naturally to richer applications, including computing distance statistics between the parties' weight vectors on the intersection. We constructed efficient semi-honest secure PFA-WSI protocols based on additively homomorphic encryption, oblivious key-value store, and oblivious transfer. Experimental results show that the proposed constructions scale to million-scale sets with practical runtimes, matching or surpassing the state-of-the-art, while achieving competitive communication.

Several directions remain open for future work:

- **Multi-party generalizations.** Extend PFA-WSI beyond the two-party setting to support richer multi-party applications.
- **Supporting additional functions.** Develop protocols for a broader class of joint scoring functions f, guided by concrete application requirements.
- **Stronger adversarial models.** Move beyond semi-honest security and design maliciously secure variants of PFA-WSI with practical efficiency.

Ethical Considerations

Our protocols enable parties to compute functions on weighted set intersection using private data without revealing the underlying inputs. To encourage responsible use, the assumptions and limitations of the technique should be clearly communicated to non-technical stakeholders. For instance, before obtaining user authorization, any deployed system should disclose in a clear and standardized manner the exact protocol outputs and the adopted security model. We believe that, when implemented within a transparent governance and compliance framework, this approach offers a net-positive impact.

Moreover, all experimental data in this paper are randomly generated and involve no real users or sensitive information. Overall, we do not believe this work raises significant ethical concerns beyond those inherent to privacy-preserving analytics systems.

Open Science

To support open science and reproducibility, we have released the code associated with this work in an anonymous repository, accessible at https://github.com/lzjluzijie/pfa-wsi.

Acknowledgments

We thank the CCS'26 reviewers for their valuable comments and feedback. This research was supported by the National Key R&D Program of China (2023YFC3305500).

Table 3: Runtime (in seconds) and communication (in MiB) of our protocols for $f \in \{\mathrm{D}_{L_1}, \mathrm{D}_{L_2}\}$.

Protocol	n	Time (s)					Comm. (MiB)				
		Total	mq-RPMT	OEWT	Func	SISA	Total	mq-RPMT	OEWT	Func	SISA
D_{L_2}	2^{12}	6.73	1.07	1.03	1.67	2.95	4.33	0.56	0.89	1.63	1.26
	2^{16}	91.55	1.82	15.72	27.17	46.95	69.21	8.90	14.30	26.00	20.01
	2^{20}	1455.11	13.56	253.83	439.10	750.81	1107.21	142.40	228.80	416.00	320.01
D_{L_1}	2^{12}	8.98	1.08	1.05	2.77	4.11	30.41	0.56	0.89	25.19	3.76
	2^{16}	125.80	1.85	15.94	43.45	64.67	486.17	8.90	14.30	402.96	60.01
	2^{20}	1985.18	11.59	252.72	692.61	1027.68	7778.65	142.40	228.80	6447.44	960.01

Table 4: Comparison of computation and communication for our IMWS protocol (Fig. 8) and the protocols of [1, 6].

n	Time (s)				Comm. (MiB)			
	Ours (Fig. 8)	DDH [6]	OPRF [6]	FOCI [1]	Ours (Fig. 8)	DDH [6]	OPRF [6]	FOCI [1]
2^{12}	**2.16**	3.68	4.48	2.43	2.33	2.56	4.61	**0.70**
2^{16}	**18.28**	133.77	151.17	85.10	37.20	41.51	74.51	**11.25**
2^{20}	**279.61**	$> 2 \times 10^4$	$> 2 \times 10^4$	$> 2 \times 10^4$	595.20	672.10	1206.68	**180.00**

References

[1] Tyler Beauregard, Janabel Xia, and Mike Rosulek. 2022. Finding One Common Item, Privately. In *International Conference on Security and Cryptography for Networks*. Springer, 462–480.

[2] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. 2023. {Near-Optimal} Oblivious {Key-Value} Stores for Efficient {PSI},{PSU} and {Volume-Hiding}{Multi-Maps}. In *32nd USENIX Security Symposium (USENIX Security 23)*. 301–318.

[3] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. 2019. Efficient two-round OT extension and silent non-interactive secure computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 291–308.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 1–36.

[5] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. 2020. Private matching for compute. *Cryptology ePrint Archive* (2020).

[6] Hongyuan Cai, Xiaodong Wang, Zijie Lu, and Bei Liang. 2025. Privately Compute the Item with Maximal Weight Sum in Set Intersection. In *International Conference on Applied Cryptography and Network Security*. Springer, 195–223.

[7] Melissa Chase and Peihan Miao. 2020. Private set intersection in the internet setting from lightweight oblivious PRF. In *Annual International Cryptology Conference*. Springer, 34–63.

[8] Yu Chen, Min Zhang, Cong Zhang, Minglang Dong, and Weiran Liu. 2024. Private set operations from multi-query reverse private membership test. In *IACR international conference on public-key cryptography*. Springer, 387–416.

[9] Koji Chida, Koki Hamada, Atsunori Ichikawa, Masanobu Kii, and Junichi Tomida. 2023. Communication-efficient inner product private join and compute with cardinality. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. 678–688.

[10] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[11] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012).

[12] Ying Gao, Lin Qi, Xiang Liu, Yuanchao Luo, and Longxin Wang. 2024. Efficient fuzzy private set intersection from fuzzy mapping. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 36–68.

[13] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2021. Oblivious key-value stores and amplification for private set intersection. In *CRYPTO 2021, Proceedings, Part II 41*. Springer, 395–425.

[14] Laura Hetz, Thomas Schneider, and Christian Weinert. 2023. Scaling mobile private contact discovery to billions of users. In *European Symposium on Research in Computer Security*. Springer, 455–476.

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

[16] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. 2020. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 370–389.

[17] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. 2003. Extending oblivious transfers efficiently. In *Annual International Cryptology Conference*. Springer, 145–161.

[18] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. 2019. Mobile private contact discovery at scale. In *USENIX Security 19*. 1447–1464.

[19] Nirajan Koirala, Seunghun Paik, Sam Martin, Helena Berens, Tasha Januszewicz, Jonathan Takeshita, Jae Hong Seo, and Taeho Jung. 2025. Select-Then-Compute: Encrypted Label Selection and Analytics over Distributed Datasets using FHE. *Cryptology ePrint Archive* (2025).

[20] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. 2016. Efficient batched oblivious PRF with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 818–829.

[21] Tancrede Lepoint, Sarvar Patel, Mariana Raykova, Karn Seth, and Ni Trieu. 2021. Private join and compute from PIR with default. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 605–634.

[22] Silvio Micali, Oded Goldreich, and Avi Wigderson. 1987. How to play any mental game. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*. ACM New York, 218–229.

[23] Dimitris Mouris, Daniel Masny, Ni Trieu, Shubho Sengupta, Prasad Buddhavarapu, and Benjamin Case. 2024. Delegated private matching for compute. *Proceedings on Privacy Enhancing Technologies* 2024, 2 (2024).

[24] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. 2010. {BotGrep}: Finding {P2P} Bots with Structured Graph Analysis. In *USENIX Security 10*.

[25] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, Dan Boneh, et al. 2011. Location privacy via private proximity testing.. In *NDSS*, Vol. 11.

[26] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*. Springer, 223–238.

[27] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. 2019. Efficient circuit-based PSI with linear communication. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 122–153.

[28] Michael O Rabin. 2005. How to exchange secrets with oblivious transfer. *Cryptology ePrint Archive* (2005).

[29] Srinivasan Raghuraman and Peter Rindal. 2022. Blazing fast PSI from improved OKVS and subfield VOLE. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*. 2505–2517.

[30] Peter Rindal and Phillipp Schoppmann. 2021. VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 901–930.

[31] Aron van Baarsen and Sihang Pu. 2024. Fuzzy private set intersection with large hyperballs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 340–369.

[32] Andrew C.-C. Yao. 1982. Protocols for Secure Computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 160–164.

[33] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*. IEEE, 162–167.

[34] Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, and Dongdai Lin. 2023. Linear Private Set Union from Multi-Query Reverse Private Membership Test. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association.

A Preliminaries

A.1 Security Model

We consider the semi-honest security model in this paper, which guarantees that a party can never learn any information about the other party's input other than its own output as long as it follows the protocol.

Definition A.1. Let Π be a two-party protocol computing the functionality $f = (f_1, f_2)$ and $\mathsf{View}_i^{\Pi}(x, y)$ be the view of P_i (the entire distribution that P_i can see), $\mathsf{Out}^{\Pi}(x, y) = (\mathsf{Out}_1^{\Pi}(x, y), \mathsf{Out}_2^{\Pi}(x, y))$ be the output of the protocol where x and y are inputs of P_1 and P_2, respectively. We say Π has semi-honest security if there exist probabilistic polynomial-time (PPT) simulators S_1, S_2, and the following holds for all inputs x, y:

$$\left(\mathsf{View}_1^{\Pi}(x, y), \mathsf{Out}^{\Pi}(x, y)\right) \overset{c}{\approx} \left(\mathsf{S}_1\left(1^{\lambda}, x, f_1(x, y)\right), f(x, y)\right),$$
$$\left(\mathsf{View}_2^{\Pi}(x, y), \mathsf{Out}^{\Pi}(x, y)\right) \overset{c}{\approx} \left(\mathsf{S}_2\left(1^{\lambda}, y, f_2(x, y)\right), f(x, y)\right).$$

A.2 Additively Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data without first having to decrypt it. An additively homomorphic encryption scheme consists of the following probabilistic polynomial-time algorithms:

- AGen: Given a security parameter κ, $\mathsf{AGen}(\kappa)$ outputs a public-private key pair (pk, sk), and specifies a message space $\mathcal{M}$.
- AEnc: Given the public key pk and a plaintext message $m \in \mathcal{M}$, one can compute a ciphertext $\mathsf{AEnc}_{pk}(m)$, an encryption of m under pk.
- ADec: Given the secret key sk and a ciphertext $\mathsf{AEnc}_{pk}(m)$, ADec is to recover a plaintext m.
- ASum: Given the public key pk and a set of ciphertexts $\{\mathsf{AEnc}_{pk}(m_i)\}$ which are the encryption of messages $\{m_i\}$, one can homomorphically compute a ciphertext which is the encryption of the sum of the underlying messages:
$$\mathsf{AEnc}_{pk}\left(\sum_i m_i\right) = \mathsf{ASum}\left(\{\mathsf{AEnc}_{pk}(m_i)\}_i\right).$$
- AMultP: Given the public key pk, a scalar a, and a ciphertext $\mathsf{AEnc}_{pk}(m)$, output a ciphertext of $a \cdot m$ (equivalently, the homomorphic sum of a encryptions of m):
$$\mathsf{AEnc}_{pk}(a \cdot m) = \mathsf{AMultP}_{pk}\left(a, \mathsf{AEnc}_{pk}(m)\right).$$
- ARef: Given the public key pk and a ciphertext $\mathsf{AEnc}_{pk}(m)$, one can randomize the ciphertext using a randomized procedure denoted as ARef.

We depend on the standard concept of CPA security in encryption, which essentially implies that, without the private key sk, encrypted messages are computationally indistinguishable from one another. The commonly used additively homomorphic encryption schemes that satisfy the above properties include Paillier encryption [26], Exponential ElGamal encryption [10] and Ring-LWE-based encryption schemes [4, 11].

A.3 Oblivious Transfer

Oblivious Transfer (OT) is a central cryptographic primitive in the area of secure computation, which was introduced by Rabin [28]. 1-out-of-2 OT refers to the setting where a sender has two input strings (m_0, m_1) and a receiver has an input choice bit $b \in \{0, 1\}$. As the result of the OT protocol, the receiver learns m_b without learning anything about m_{1-b} while the sender learns nothing about b. The ideal functionality $\mathcal{F}_{\mathrm{OT}}$ is presented in Fig. 12.

FUNCTIONALITY:

Inputs: There are two parties: the Sender $\mathcal{S}$ and the Receiver $\mathcal{R}$. $\mathcal{S}$ inputs two messages (m_0, m_1). $\mathcal{R}$ inputs a chosen bit $b \in \{0, 1\}$.
Outputs: The receiver $\mathcal{R}$ outputs m_b.

Figure 12: Ideal functionality $\mathcal{F}_{\mathrm{OT}}$ for oblivious transfer.

B Other Output Functions

B.1 Inner Product and Summation

The PFA-WSI protocol with aggregated output in Fig. 11 can be easily extended to support intersection inner-product [9, 21] and intersection sum [16]. (We note that the protocol of [16] only allows one party's items to carry weights, whereas our protocol more generally supports weights on both sides.) In these cases, we take $f(u_i, v_j) = u_i \cdot v_j$ for the inner product and $f(u_i, v_j) = u_i + v_j$ for the summation. Concretely, it suffices to augment the *Function-Specific Homomorphic Evaluation* phase in Fig. 11 with the following cases:

Case $f = \mathrm{IP}$**:**
For $j \in [n_2]$, Bob computes $c_j^{(\mathrm{IP})} = \mathsf{AMultP}(v_j, b_j)$.

Case $f = \mathrm{Sum}$**:**
For $j \in [n_2]$, Bob computes $c_j^{(\mathrm{Sum})} = \mathsf{ASum}(\{b_j, \mathsf{AEnc}_{pk}(v_j)\})$.

Recall that if $y_j = x_i$, then by the functionality of $\mathcal{F}_{\mathrm{OEWT}}$ we have $b_j = \mathsf{AEnc}_{pk}(u_i)$, and hence $c_j^{(\mathrm{IP})} = \mathsf{AEnc}_{pk}(v_j \cdot u_i)$, $c_j^{(\mathrm{Sum})} = \mathsf{AEnc}_{pk}(v_j + u_i)$. Thus the correctness of the protocol is guaranteed. Moreover, this step only requires Bob to perform local homomorphic computations, so no extra information is leaked and security is preserved.

Performance Comparison. In Tab. 5, we compare the computation and communication costs of our protocols for $f \in \{\mathrm{IP}, \mathrm{Sum}\}$ with related work [9, 16, 21]. Both [9, 21] implement the $f = \mathrm{IP}$ functionality. In contrast, [16] considers a setting where only one party's items carry weights, whereas our $f = \mathrm{Sum}$ protocol addresses the more general case where both parties hold weights. Since the implementations of prior work [9, 21] are not publicly available, we compare against the experimental results reported in [9]. Specifically, the inner-product PJC results of [21] were obtained on an Intel Xeon CPU E5-2696 v3 @ 2.30GHz; the CHI$^+$23 [9] and PI-Sum [16] were evaluated on an Intel Core i9-9900K CPU @ 3.60GHz; and our experiments were run on an AMD 3950X CPU @ 3.50GHz.

The results show that our f = IP protocol uses about 2× more communication than CHI$^+$23 [9], while reducing computation time by a factor of 1.3-3.7×. Although the inner-product PJC protocol of [21] was evaluated on a less powerful CPU, it incurs substantially higher computation time, and its communication grows significantly for larger set sizes. Comparing our f = Sum protocol with PI-Sum [16], the CPUs are of comparable performance; despite incurring roughly 4× higher communication to support weights from both parties, our protocol is about 2× faster in computation.

B.2 Cosine Similarity

In Fig. 11 we constructed protocols for the L_1 and L_2 distances. In practice, another important similarity measure is the cosine similarity, which captures the similarity of the *pattern* between two vectors while being invariant to their overall scale. For example, in preference matching applications, different users may use very different absolute rating scales, but the cosine similarity of their rating vectors is still a meaningful indicator of how similar their preferences are.

Concretely, given inputs $\{(x_i, u_i)\}_{i\in[n_1]}$ and $\{(y_j, v_j)\}_{j\in[n_2]}$, let

$$I = \left\{ j \in [n_2] : \exists\, i \in [n_1] \text{ with } y_j = x_i \right\}$$

be the index set of intersection items on Bob's side, and for each $j \in I$ let $i(j)$ denote the unique index such that $y_j = x_{i(j)}$. We define the intersection weight vectors $\mathbf{u}_I = \{u_{i(j)}\}_{j\in I}$, $\mathbf{v}_I = \{v_j\}_{j\in I}$, and their cosine similarity as

$$\mathrm{CosSim}(\mathbf{u}_I, \mathbf{v}_I) = \frac{\sum_{j\in I} u_{i(j)}\, v_j}{\sqrt{\sum_{j\in I} u_{i(j)}^2}\,\sqrt{\sum_{j\in I} v_j^2}}.$$

We observe that computing the cosine similarity over the intersection requires (i) the inner-product of the intersection weights and (ii) the sums of squared weights on each side. This task is not directly captured by our PFA-WSI functionality with a single aggregated output. However, starting from our protocols for inner product and summation, and combining them with random-masking techniques, we can construct a secure protocol for computing the cosine similarity over the intersection.

Concretely, we assume that all weights lie in an integer ring $\mathbb{Z}_\alpha$. A naive approach that lets Alice learn the intersection inner product and both denominator sums in the cosine similarity would reveal more information than the cosine value itself. Instead, we use random masking so that Alice only learns appropriately masked shares of the numerator and denominator.

First, Alice and Bob independently sample random elements $r_1, r_2 \leftarrow \mathbb{Z}_\alpha$. They first invoke the PFA-WSI functionality with f = Sum and Bob as the receiver, running it on Alice's squared weights scaled by r_1^2, so that Bob learns the masked sum $p = r_1^2 \sum_{j\in I} u_{i(j)}^2$. Next, they invoke PFA-WSI again with f = Sum and Alice as the receiver, this time on Bob's squared weights scaled by pr_2^2, yielding $q = pr_2^2 \sum_{j\in I} v_j^2$. Finally, they invoke PFA-WSI with f = IP on weights scaled by r_1 (on Alice's side) and r_2 (on Bob's side), so that Alice obtains $k = r_1 r_2 \sum_{j\in I} u_{i(j)}\, v_j$. Alice can then recover the cosine similarity as

$$\mathrm{CosSim}(\mathbf{u}_I, \mathbf{v}_I) = \frac{k}{\sqrt{q}} = \frac{r_1 r_2 \sum_{j\in I} u_{i(j)}\, v_j}{\sqrt{pr_2^2 \sum_{j\in I} v_j^2}} = \frac{\sum_{j\in I} u_{i(j)}\, v_j}{\sqrt{\sum_{j\in I} u_{i(j)}^2}\,\sqrt{\sum_{j\in I} v_j^2}}.$$

The full protocol is given in Fig. 13.

By the randomness of r_1, the value p learned by Bob is a randomly masked version of Alice's intersection sum and is itself distributed as a random element of $\mathbb{Z}_\alpha$. Likewise, due to the randomness of r_2, the values q and k observed by Alice are individually masked and reveal no information beyond the ratio $\frac{k}{\sqrt{q}}$, which equals the final cosine similarity. Therefore, the protocol preserves the privacy.

PROTOCOL:

Inputs: Alice has input set $\{(x_i, u_i)\}_{i\in[n_1]}$ and Bob has input set $\{(y_j, v_j)\}_{j\in[n_2]}$.

Parameters: The computational and statistical security parameters κ, λ. The value space $\mathbb{Z}_\alpha$. The functionality $\mathcal{F}_{\text{PFA-WSI}}$ with aggregated output for $f \in \{\text{IP}, \text{Sum}\}$.

Protocol Steps:

(1) Alice and Bob independently sample random integers r_1 and r_2 from $\mathbb{Z}_\alpha$.

(2) Alice (as the receiver) and Bob invoke $\mathcal{F}_{\text{PFA-WSI}}$ with f = Sum and inputs $\{(x_i, r_1^2 u_i^2)\}_{i\in[n_1]}$ from Alice and $\{(y_j, 0)\}_{j\in[n_2]}$ from Bob. The functionality returns to Bob the intersection cardinality $|I|$ and the intersection sum $p = r_1^2 \sum_{j\in I} u_{i(j)}^2$.

(3) Alice (as the receiver) and Bob invoke $\mathcal{F}_{\text{PFA-WSI}}$ with f = Sum and inputs $\{(x_i, 0)\}_{i\in[n_1]}$ from Alice and $\{(y_j, pr_2^2 v_j^2)\}_{j\in[n_2]}$ from Bob. The functionality returns to Alice the intersection cardinality $|I|$ and the intersection sum $q = pr_2^2 \sum_{j\in I} v_j^2$.

(4) Alice (as the receiver) and Bob invoke $\mathcal{F}_{\text{PFA-WSI}}$ with f = IP and inputs $\{(x_i, r_1 u_i)\}_{i\in[n_1]}$ from Alice and $\{(y_j, r_2 v_j)\}_{j\in[n_2]}$ from Bob. The functionality returns to Alice the intersection cardinality $|I|$ and the intersection inner product $k = r_1 r_2 \sum_{j\in I} u_{i(j)}\, v_j$.

(5) Alice outputs the intersection cardinality $|I|$ and the cosine similarity $\mathrm{CosSim}(\mathbf{u}_I, \mathbf{v}_I) = \frac{k}{\sqrt{q}}$, and Bob outputs $|I|$.

Figure 13: The private intersection cosine similarity protocol.

C Proofs

C.1 Proofs for OEWT (Sec. 3.2)

Theorem 3.3. *Paillier encryption is $\mathbb{Z}_{N^2}$-uniformly embeddable, where N denotes the Paillier modulus.*

Proof. In Paillier encryption, let $(pk, sk) \leftarrow \mathrm{AGen}(\kappa)$ with $pk = (N, g)$, $N = pq$ for distinct odd primes p, q, and $g = N + 1$. The plaintext space is $\mathcal{M} = \mathbb{Z}_N$ and the ciphertext space is $C = \mathbb{Z}_{N^2}^*$.

Table 5: Runtime (in seconds) and communication (in MiB) of our protocols for $f \in \{\mathrm{IP}, \mathrm{Sum}\}$ and related work.

Input Size		IP (Ours)		Sum (Ours)		CHI$^+$23 [9]		inner-product PJC [21]		PI-Sum [16]	
n_1	n_2	Time (s)	Comm. (MiB)	Time (s)	Comm. (MiB)	Time (s)	Comm. (MiB)	Time (s)	Comm. (MiB)	Time (s)	Comm. (MiB)
2^{16}	2^8	4.03	20.06	3.36	20.06	11.80	8.77	172.3	25.75	6.05	5.15
	2^{12}	4.98	20.90	4.37	20.90	12.63	9.25	139.3	114.44	6.23	5.34
	2^{16}	20.11	34.31	19.54	34.31	25.92	16.40	131.1	763.89	12.4	8.77
2^{20}	2^8	50.03	320.06	38.28	320.06	191.6	139.24	3026	27.66	93.6	82.11
	2^{12}	51.21	320.90	38.67	320.90	192.4	139.24	2134	203.13	93.7	82.21
	2^{16}	67.53	334.31	54.15	334.31	205.4	146.87	2228	1736.64	99.8	85.64

Set $\beta = N^2$ and define

$$\tau(c) = c \ (\mathrm{mod} N^2), \quad \tau^{-1}(x) = \begin{cases} x, & \gcd(x, N) = 1, \\ \perp, & \text{otherwise.} \end{cases}$$

Then τ is injective and $\tau^{-1}(\tau(c)) = c$ for all $c \in C$.

For pseudouniformity, fix any $m \in \mathbb{Z}_N$ and write $H = \{r^N \bmod N^2 : r \in \mathbb{Z}_N^*\}$. For $r \leftarrow \mathbb{Z}_N^*$ uniform,

$$c = \mathsf{Enc}_{pk}(m; r) = g^m r^N \bmod N^2$$

is uniform over the coset $g^m H$. Under the Decisional Composite Residuosity (DCR) assumption, we have

$$\mathrm{U}(g^m H) \overset{\mathrm{c}}{\approx} \mathrm{U}(\mathbb{Z}_{N^2}^*).$$

Therefore, for every PPT distinguisher $\mathcal{D}$,

$$\left| \Pr_{c \leftarrow \mathsf{Enc}_{pk}(m)} \left[\mathcal{D}\big(pk, \tau(c)\big) = 1\right] - \Pr_{\gamma \leftarrow \mathrm{U}(\mathbb{Z}_{N^2}^*)} \left[\mathcal{D}\big(pk, \gamma\big) = 1\right] \right| \leq \mathrm{negl}(\kappa).$$

Finally, the statistical gap between $\mathrm{U}(\mathbb{Z}_{N^2}^*)$ and $\mathrm{U}(\mathbb{Z}_{N^2})$ equals

$$\Delta\big(\mathrm{U}(\mathbb{Z}_{N^2}^*), \mathrm{U}(\mathbb{Z}_{N^2})\big) = 1 - \frac{\varphi(N^2)}{N^2} = \frac{1}{p} + \frac{1}{q} - \frac{1}{pq},$$

which is negligible. By the triangle inequality,

$$\left| \Pr_{c \leftarrow \mathsf{Enc}_{pk}(m)} \left[\mathcal{D}\big(pk, \tau(c)\big) = 1\right] - \Pr_{\gamma \leftarrow \mathrm{U}(\mathbb{Z}_{N^2}^*)} \left[\mathcal{D}\big(pk, \gamma\big) = 1\right] \right| \leq \mathrm{negl}(\kappa).$$

establishing pseudouniformity. Hence Paillier is $\mathbb{Z}_{N^2}$-uniformly embeddable. □

Lemma 3.4. *Let r be uniformly sampled from $\{0, \ldots, \lfloor \frac{p}{\beta} \rfloor - 1\}$, and let γ be uniformly sampled from $\{0, \ldots, \beta - 1\}$. If $p \geq 2^\lambda \cdot \beta$, then the distribution of $\zeta = \gamma + r \cdot \beta$ is statistically indistinguishable from the uniform distribution over $\{0, \ldots, p - 1\}$.*

Proof. Let $q = \lfloor \frac{p}{\beta} \rfloor$, and write $p = q\beta + \delta, \quad 0 \leq \delta < \beta$.

Define the two distributions on $\{0, 1, \ldots, p - 1\}$:

- D_0 is uniform on $\{0, \ldots, p - 1\}$, so $\Pr[D_0 = x] = \frac{1}{p}$ for every x.
- D_1 is the distribution of $\zeta = \gamma + r \cdot \beta$, where $\gamma \leftarrow \{0, \ldots, \beta - 1\}$ and $r \leftarrow \{0, \ldots, q - 1\}$.

Since γ and r are independent and $r \cdot \beta \in \{0, \beta, 2\beta, \ldots, (q-1) \cdot \beta\}$, ζ is uniform over $\{0, 1, \ldots, q\beta - 1\}$.

Thus

$$\Pr[D_1 = x] = \begin{cases} \frac{1}{q\beta}, & 0 \leq x < q\beta, \\ 0, & q\beta \leq x < p. \end{cases}$$

The statistical distance between D_0 and D_1 is

$$\begin{aligned} \Delta(D_0, D_1) &= \tfrac{1}{2} \sum_{x=0}^{p-1} \left| \Pr[D_0 = x] - \Pr[D_1 = x] \right| \\ &= \tfrac{1}{2} \Big(q\beta \cdot \left| \tfrac{1}{p} - \tfrac{1}{q\beta} \right| + \delta \cdot \left| \tfrac{1}{p} - 0 \right| \Big). \end{aligned}$$

Since $q\beta = p - \delta$, we have

$$\left| \tfrac{1}{p} - \tfrac{1}{q\beta} \right| = \tfrac{\delta}{p(p-\delta)} \quad \Longrightarrow \quad q\beta \cdot \left| \tfrac{1}{p} - \tfrac{1}{q\beta} \right| = \tfrac{\delta}{p}.$$

Hence $\Delta(D_0, D_1) = \frac{1}{2}\left(\frac{\delta}{p} + \frac{\delta}{p}\right) = \frac{\delta}{p} \leq \frac{\beta - 1}{p} < \frac{\beta}{p}$.

By the assumption $p \geq 2^\lambda \cdot \beta$, it follows that $\Delta(D_0, D_1) < \frac{\beta}{p} \leq 2^{-\lambda}$, which proves that the distribution of $\zeta = \gamma + \beta \cdot r$ is statistically indistinguishable from the uniform distribution on $\{0, \ldots, p - 1\}$. □

Theorem 3.5 (The randomness of the OKVS values). *If the encryption scheme Π is $\mathbb{Z}_\beta$-uniformly embeddable (via (τ, τ^{-1})), then in Fig. 7 the values stored in the OKVS T are computationally indistinguishable from the uniform distribution over $\mathbb{Z}_p$.*

Proof. Fix any $i \in [n_1]$. Let

$$t_i = \tau\big(\mathsf{Enc}_{pk}(u_i)\big) \in \mathbb{Z}_\beta, \qquad r_i \leftarrow \left\{0, \ldots, \lfloor \tfrac{p}{\beta} \rfloor - 1\right\},$$

and define $z_i = t_i + r_i\beta \in \mathbb{Z}_p$.

Let $\gamma \leftarrow \mathrm{U}(\mathbb{Z}_\beta)$ be independent of r_i, and set $\zeta = \gamma + r_i\beta$.

By Lem. 3.4, the distributions of ζ and $\mathrm{U}(\mathbb{Z}_p)$ are statistically indistinguishable; hence they are also computationally indistinguishable, i.e., for any PPT distinguisher $\mathcal{D}$ we have

$$\left| \Pr[\mathcal{D}(\zeta) = 1] - \Pr_{\eta \leftarrow \mathbb{Z}_p} [\mathcal{D}(\eta) = 1] \right| \leq \mathrm{negl}(\kappa).$$

Next, the transformation $\phi : \mathbb{Z}_\beta \to \mathbb{Z}_p, x \mapsto x + r_i\beta$, is PPT. By closure of computational indistinguishability under PPT transformations, it suffices to prove $t_i \overset{\mathrm{c}}{\approx} \gamma$. By Def. 3.1 (pseudouniformity),

for every message m we have $\tau(\mathsf{Enc}_{pk}(m)) \overset{c}{\approx} \mathrm{U}(\mathbb{Z}_\beta)$; in particular, $t_i = \tau(\mathsf{Enc}_{pk}(u_i)) \overset{c}{\approx} \gamma$. Therefore,

$$z_i = \phi(t_i) \overset{c}{\approx} \phi(\gamma) = \zeta.$$

Finally, for any PPT distinguisher $\mathcal{D}$ we have

$$\begin{aligned}
&\left|\Pr[\mathcal{D}(z_i) = 1] - \Pr[\mathcal{D}(\mathrm{U}(\mathbb{Z}_p)) = 1]\right| \\
&\le \left|\Pr[\mathcal{D}(z_i) = 1] - \Pr[\mathcal{D}(\zeta) = 1]\right| \\
&+ \left|\Pr[\mathcal{D}(\zeta) = 1] - \Pr_{\eta \leftarrow \mathbb{Z}_p}[\mathcal{D}(\eta) = 1]\right| \\
&\le \mathrm{negl}(\kappa).
\end{aligned}$$

Since i was arbitrary, the same holds for every OKVS value. □

Theorem 3.6. *Assume that the OKVS satisfies the random decoding property, the protocol in Fig. 7 securely realizes the functionality $\mathcal{F}_{\mathrm{OEWT}}$ in the semi-honest model.*

Proof. Alice receives no messages in the protocol. Hence, it suffices to prove security against the corrupt Bob.

Bob's view consists only of the OKVS T. By Thm. 3.5, the values stored in T are computationally indistinguishable from uniform over $\mathbb{Z}_p$. Hence, T is computationally indistinguishable from an OKVS encoded from uniformly random key-value pairs.

Meanwhile, for each $i \in [n_2]$, let $z_i = \mathsf{Decode}(T, y_i) \in \mathbb{Z}_p$, set $b'_i = z_i \bmod \beta \in \mathbb{Z}_\beta$, and $b_i = \tau^{-1}(b'_i)$. If $y_i = x_j$ for some $j \in [n_1]$, OKVS correctness gives $z_i = t_j + r_j\beta$ with $t_j = \tau(\mathsf{Enc}_{pk}(u_j))$, so $b'_i = t_j$ and $b_i = \mathsf{Enc}_{pk}(u_j)$. If $y_i \notin \{x_1, \ldots, x_{n_1}\}$, the random decoding property yields z_i is uniform on $\mathbb{Z}_p$, hence b'_i is uniform on $\mathbb{Z}_\beta$. By the $\mathbb{Z}_\beta$-uniform ciphertext encoding, $\tau^{-1}(b'_i)$ is (except with negligible probability) computationally indistinguishable from a uniform ciphertext in C. □

C.2 Security Proof of Thm. 4.1

Theorem 4.1. *Assume that the additively homomorphic encryption scheme is IND-CPA secure, the protocol in Fig. 8 securely realizes the functionality $\mathcal{F}_{\mathrm{IMWS}}$ in the $\mathcal{F}_{\text{mq-RPMT}}$, $\mathcal{F}_{\mathrm{OEWT}}$, $\mathcal{F}_{\mathrm{OT}}$ hybrid model in the presence of semi-honest adversaries.*

Proof. The proof consists of two parts: correctness and privacy.

Correctness. For each $i \in [n_2]$, if Bob's item $y_i = x_j$ for some $j \in [n_1]$, then by the functionality of $\mathcal{F}_{\text{mq-RPMT}}$ Alice receives $a_i = 1$ in Step 2. By the functionality of $\mathcal{F}_{\mathrm{OEWT}}$, in Step 3 Bob obtains b_i as an additively homomorphic encryption of u_j, so in Step 4 he forms m_i as an encryption of $u_j + v_i$. Via OT, Alice receives m_i, decrypts it, and outputs $w_i = u_j + v_i$.

If $y_i \notin X$, then $a_i = 0$ in Step 2, and in Step 5 Alice receives $\perp$ from the OT and outputs $w_i = \perp$. Thus, correctness follows.

Privacy. We construct simulators Sim_A, Sim_B for simulating the view of corrupt Alice and Bob, respectively, and argue the indistinguishability of the produced transcript from the real execution.

Security against corrupt Alice. We construct Sim_A as follows. It is given Alice's input set $\{(x_i, u_i)\}_{i\in[n_1]}$ and the output vector $(w_1, \ldots, w_{n_2})$. Sim_A runs the honest Alice's protocol to generate its view with the following exceptions: In Step 2, for each $i \in [n_2]$, if $w_i \neq \perp$ then Sim_A sets $a_i = 1$, otherwise it sets $a_i = 0$, and Sim_A sends Alice the vector $\overrightarrow{a}$. In the OT of Step 5, for each $i \in [n_2]$, if $w_i \neq \perp$ then Sim_A sets $m_i \leftarrow \mathsf{AEnc}_{pk}(w_i)$; otherwise, it sets m_i to a random ciphertext. We argue the indistinguishability via the following hybrid argument:

- $\mathbf{Hyb}_0$: The Alice's view in the real protocol.
- $\mathbf{Hyb}_1$: Same as Hyb_0 except that in Step 2, Sim_A sends Alice the vector $\overrightarrow{a}$ as defined above.
  By the functionality of $\mathcal{F}_{\text{mq-RPMT}}$, this hybrid is identical to Hyb_0.
- $\mathbf{Hyb}_2$: Same as Hyb_1 except that in Step 5, Sim_A sets m_i as defined above.
  By the functionality of $\mathcal{F}_{\mathrm{OT}}$, this hybrid is statistically identical to Hyb_1.

Security against corrupt Bob. We construct Sim_B as follows. It is given Bob's input set $\{(y_j, v_j)\}_{j\in[n_2]}$ and the output item y^*_j. Sim_B runs the honest Bob's protocol to generate its view with the following exceptions: In Step 1, Sim_B generates and sends the public key pk to Bob. In Step 3 of $\mathcal{F}_{\mathrm{OEWT}}$, Sim_B randomly samples Alice's input $\{(x_i, u_i)\}_{i\in[n_1]}$. In Step 7, Sim_B sends j^* to Bob. We argue the indistinguishability via the following hybrid argument:

- $\mathbf{Hyb}_0$: The Bob's view in the real protocol.
- $\mathbf{Hyb}_1$: Same as Hyb_0 except that in Step 1, Sim_B generates and sends the public key pk to Bob.
  This hybrid is statistically identical to Hyb_1.
- $\mathbf{Hyb}_2$: Same as Hyb_1 except that in Step 3 of $\mathcal{F}_{\mathrm{OEWT}}$, Sim_B randomly samples Alice's input $\{(x_i, u_i)\}_{i\in[n_1]}$.
  Since the underlying additively homomorphic encryption scheme is IND-CPA secure, each b_i is computationally indistinguishable from a random ciphertext. Hence, the hybrids Hyb_1 and Hyb_2 are computationally indistinguishable.
- $\mathbf{Hyb}_3$: Same as Hyb_2 except that in Step 7, Sim_B sends j^* to Bob.
  This hybrid is identical to Hyb_2.

□

C.3 Security Proof of Thm. 5.1

Theorem 5.1. *Assume that the additively homomorphic encryption scheme is IND-CPA secure, the protocol in Fig. 10 securely realizes the functionality $\mathcal{F}_{\mathrm{SISA}}$ in the $\mathcal{F}_{\mathrm{OT}}$ hybrid model in the presence of semi-honest adversaries.*

Proof. Correctness has already been established above. Thus, it remains to prove privacy.

Privacy. We construct simulators Sim_A, Sim_B for simulating the view of corrupt Alice and Bob, respectively, and argue the indistinguishability of the produced transcript from the real execution.

Security against corrupt Alice. We construct Sim_A as follows. It is given Alice's input vector $\overrightarrow{a}$ and the output value S. Sim_A runs the honest Alice's protocol to generate its view with the following exceptions: In Step 1, for each $i \in [n_2]$, Sim_A samples $r_i \leftarrow \mathbb{Z}_\alpha$. For all indices i with $a_i = 1$, it samples random values $v_i \in \mathbb{Z}_\alpha$ such that $\sum_{\substack{i\in[n_2]\\ a_i=1}} v_i = S$, and sets $m_{i,0} = r_i$ and $m_{i,1} = \mathsf{AEnc}_{pk}(v_i + r_i)$. In Step 2, Sim_A invoke the $\mathcal{F}_{\mathrm{OT}}$ simulator to simulate Alice's view. In Step 3, Sim_A sends $R = \sum_{i=1}^{n_2} r_i$ to Alice. We argue the indistinguishability via the following hybrid argument:

- $\mathbf{Hyb}_0$: The Alice's view in the real protocol.

- **Hyb**$_1$: Same as Hyb$_0$ except that in Step 1, Sim$_A$ samples $(m_{i,0}, m_{i,1})$ as defined above.
  Since the additively homomorphic encryption scheme is IND-CPA secure, for each i, the distribution of $(m_{i,0}, m_{i,1})$ is indistinguishable to that in Hyb$_0$. Thus, the hybrids Hyb$_0$ and Hyb$_1$ are computationally indistinguishable.
- **Hyb**$_2$: Same as Hyb$_1$ except that in Step 2, Sim$_A$ invoke the $\mathcal{F}_{\text{OT}}$ simulator to simulate Alice's view using the inputs described above.
  By the security of the $\mathcal{F}_{\text{OT}}$ simulator, the hybrids Hyb$_1$ and Hyb$_2$ are computationally indistinguishable.
- **Hyb**$_3$: Same as Hyb$_2$ except that in Step 3, Sim$_A$ sends $R = \sum_{i=1}^{n_2} r_i$ to Alice.
  This hybrid is statistically identical to Hyb$_2$.

Security against corrupt Bob. Bob receives no messages in the protocol. Hence, the security is guaranteed. □

C.4 Security Proof of Thm. 5.2

Theorem 5.2. *Assume that the additively homomorphic encryption scheme is IND-CPA secure. Then the protocol in Fig. 11 securely realizes $\mathcal{F}_{\text{PFA-WSI}}$ with aggregated output, for $f \in \{\text{IP}, \text{D}_{L_2}, \text{D}_{L_1}\}$, in the hybrid model with $\mathcal{F}_{\text{mq-RPMT}}$, $\mathcal{F}_{\text{OEWT}}$, $\mathcal{F}_{\text{SIC}}$, $\mathcal{F}_{\text{OT}}$ and $\mathcal{F}_{\text{SISA}}$ against semi-honest adversaries.*

Proof. The proof consists of two parts: correctness and privacy.

Correctness. If Bob's item y_j equals some x_i with $i \in [n_1]$, it suffices to show that, after the function-specific homomorphic evaluation phase, Bob obtains an additively homomorphic encryption of $f(u_i, v_j)$. Then, by the aggregation functionality $\mathcal{F}_{\text{SISA}}$, Alice obtains the final output $S^{(f)}$. We now prove this separately for each $f \in \{\text{D}_{L_2}, \text{D}_{L_1}\}$.

- **Case** $f = \text{D}_{L_2}$**:**
  In Step (a), we have

  $$d_j = \text{A`}\big(\{b_j,\ \text{AEnc}_{pk}(-v_j + t_j)\}\big) = \text{AEnc}_{pk}(u_i - v_j + t_j),$$

  and thus $e_j = \text{AEnc}_{pk}\big((\text{ADec}_{sk}(d'_j))^2\big) = \text{AEnc}_{pk}\big((u_i - v_j + t_j)^2\big)$.
  By the identity

  $$(u_i - v_j)^2 = (u_i - v_j + t_j)^2 + t_j^2 - 2t_j(u_i - v_j + t_j),$$

  we obtain $c_j^{(\text{D}_{L_2})} = \text{AEnc}_{pk}\big((u_i - v_j)^2\big)$.
- **Case** $f = \text{D}_{L_1}$**:**
  For each $j \in [n_2]$, let $\Delta_j = u_i - v_j$. Consider first the case $\Delta_j < 0$ and $\tau_j = +1$. Then in Step (c) we have $\eta_j = \mathbf{1}_{\tau_j\Delta_j<0} = 1$, and in the OT of Step (d) Alice receives $m_{\eta_j} = \text{AEnc}_{pk}(-\Delta_j + s_j)$. After Bob removes the mask s_j in Step (e), he obtains $c_j^{(\text{D}_{L_1})} = \text{AEnc}_{pk}(|\Delta_j|)$.
  The remaining three cases $\Delta_j < 0$, $\tau_j = -1$; $\Delta_j > 0$, $\tau_j = +1$; and $\Delta_j > 0$, $\tau_j = -1$ are analogous: in each case, the choice bit η_j and the ordering of (D_j^+, D_j^-) ensure that $m_{\eta_j} = \text{AEnc}_{pk}(|\Delta_j| + s_j)$, so after removing s_j we always obtain $c_j^{(\text{D}_{L_1})} = \text{AEnc}_{pk}(|\Delta_j|)$.

Privacy. The security of the mq-RPMT and encrypted intersection weight transfer phases follows similarly to Thm. 4.1. The only minor difference is in simulating the vector $\overrightarrow{d}$: since Bob may arbitrarily shuffle his input set before the protocol, the output $\overrightarrow{d}$ of $\mathcal{F}_{\text{mq-RPMT}}$ can be simulated as a random binary vector of length n_2 with Hamming weight $|X \cap Y|$. The secure intersection aggregation phase can be simulated solely from the final output. Therefore, it remains to prove the security of the function-specific homomorphic evaluation phase. We do so separately for each $f \in \{\text{D}_{L_2}, \text{D}_{L_1}\}$.

- **Case** $f = \text{D}_{L_2}$**:**
  Security against corrupt Bob follows from the IND-CPA security of the encryption scheme, which hides all plaintext values in the homomorphic evaluations. Security against corrupt Alice follows from the randomness of t_j sampled by Bob in Step (a): Alice only decrypts masked values that are (from her perspective) uniformly random, so her view can be simulated.
- **Case** $f = \text{D}_{L_1}$**:**
  Security against corrupt Alice. Alice's view consists of the ciphertexts $e_j = \text{AEnc}_{pk}(\tau_j\Delta_j + t_j)$ from Step (b), the bits $\eta_j = \mathbf{1}_{\tau_j\Delta_j<0}$ from Step (c), and the ciphertexts $m_{\eta_j} = \text{AEnc}_{pk}(|\Delta_j| + s_j)$ obtained via OT in Step (d). Here, e_j is masked by the random t_j, η_j is hidden by the random sign τ_j, and m_{η_j} is masked by the random s_j. Thus each component of Alice's view can be simulated and does not leak additional information.
  Security against corrupt Bob. Bob's view contains only additively homomorphic ciphertexts, and its security is guaranteed by the IND-CPA security of the encryption scheme.

□