Fast Fuzzy PSI from Symmetric-Key Techniques

Cong Zhang[1], Yu Chen[2], Yang Cao[2], Yujie Bai[2], Shuaishuai Li[3], Juntong Lin[2], Anyu Wang[1], and Xiaoyun Wang[1]

[1] Tsinghua University
{zhangcong,anyuwang,xiaoyunwang}@tsinghua.edu.cn,
[2] Shandong University
yuchen@sdu.edu.cn, {202437063,juntonglin}@mail.sdu.edu.cn, byj8348560@163.com,
[3] Zhongguancun Laboratory
liss@zgclab.edu.cn,

Abstract. Private set intersection (PSI) enables a sender holding a set Q and a receiver holding a set W to securely compute the intersection $Q \cap W$. Fuzzy PSI (FPSI) is a PSI variant where the receiver learns the items $q \in Q$ for which there exists $w \in W$ such that $\mathsf{dist}(q, w) \leq \delta$ with respect to some distance metric. Recently, Gao et al. (ASIACRYPT 2024) proposed the first FPSI protocols for L_∞ and $L_{p\in[1,\infty)}$ distance with linear complexity. They summarized their FPSI construction into two steps: fuzzy mapping and fuzzy matching. However, their realizations of the two steps heavily rely on public key operations, namely the DH-key exchange and additively homomorphic encryption, resulting in low efficiency.

In this work, we propose new FPSI protocols for L_∞ and $L_{p\in[1,\infty)}$ distances, primarily leveraging symmetric-key primitives. We revisit the definition of fuzzy mapping and rigorously redefine it as a cryptographic scheme. We further introduce consistency for fuzzy mapping scheme, which could simplify the fuzzy matching step into plain PSI. We then demonstrate how to execute fuzzy mapping step satisfying consistency. We also propose several new technologies to completely avoid the extensive use of computationally expensive public-key operations burden inherent in existing solutions.

We implement our FPSI protocols and compare them with the state-of-the-art FPSI protocols. Experiments show that our protocols perform better than state-of-the-art under all the parameters we tested. Specifically, our protocols achieve a $2.2 - 83.9\times$ speedup in running time and $1.5 - 11.5\times$ shrinking in communication cost, depending on set sizes, dimension and distance threshold.

1 Introduction

Private set intersection (PSI) allows two parties, the sender and the receiver, to compute the intersection of their private sets while revealing nothing more than the intersection itself. Owing to its extensive application scenarios, there has been a significant amount of work on PSI [PSZ14, KKRT16, PRTY19, CM20, PRTY20, GPR+21, GMR+21, RT21, RS21, RR22, BC23, CZZ+24] over the last decade. The current state-of-the-art PSI [RS21, RR22] is mainly based on symmetric-key operations and is truly practical for real-world privacy-preserving computations.

In numerous practical applications, identifying an exact intersection between the datasets of two parties can often be improbable or overly constraining. For instance, certain applications of PSI are hampered by the inherent noise present in data. Consider a scenario where two parties possess databases containing biometric data, such as fingerprints or facial features represented as vectors. They may seek to identify matches or near-matches due to the natural variations that arise in the sampling of biometric data, without disclosing the full extent of their databases. In such cases, what is desperately needed is fuzzy PSI (FPSI). Unlike plain PSI, FPSI allows the receiver to learn elements in the sender's set that are less than a certain threshold away from its own set elements. FPSI also serves as a key tool for genomic sequences matching [WHZ+15], similar images matching [KM21], compromised credentials checking [PIRC21], illegal content detection [vBP24], and vulnerable password detection [CFR23].

Despite many important applications, the efficiency of FPSI is still far from satisfactory. Most previous FPSI only considered Hamming distance [FNP04, IW06, CH08, YSPW09, UCK+21, CFR23, CLO24], while some works considered L_∞ distance [GRS22, GRS23, GGM24, vBP24, GQL+24]. It was not until recently that Baarsen and Pu [vBP24] proposed the first FPSI protocol for $L_{p\in[1,\infty)}$

distance. However, their protocols exhibit super-linear complexity with respect to both the dimension d and the distance threshold δ. More recently, Gao et al. [GQL+24] introduced the first FPSI protocols for $L_{p\in[1,\infty]}$ distance that exhibits linear complexity with respect to set size, dimension, and distance threshold. Although it signifies a notable theoretical progress, their protocol heavily relies on public-key operations, e.g., the DH-key exchange and additively homomorphic encryption (AHE), resulting in low performance in practice.

1.1 Motivation

In this work, we focus on the FPSI protocols for $L_{p\in[1,\infty]}$ distance. Most previous FPSI protocols were based on the simple idea of performing $m \cdot n$ fuzzy matching operations to obtain the final result, where m and n are the sizes of the sender and receiver sets, respectively. However, this approach results in communication and computational complexities that are at least proportional to $m \cdot n$. To circumvent this barrier, it is necessary to reduce the number of matching operations, which often requires making assumptions about the input set. Garimella et al. [GRS22] introduced a strong assumption called the *global disjoint assumption*, which posits that each element in the receiver's set is more than 2δ distant from every other element in *every* dimension. By introducing this strong assumption, they removed the $m \cdot n$ factor in their FPSI for L_∞ distance. However, as Garimella et al. [GRS22] themselves noted, this assumption is somewhat artificial. Baarsen and Pu [vBP24] weakened this assumption to *existing disjoint assumption*, which assumes that for each element in the receiver set, there *exists* a dimension where other elements are more than 2δ away from that element in that dimension. To demonstrate the reasonableness of this assumption, they further proved that when the elements of a set are uniformly randomly selected, the probability of the set satisfying this assumption is overwhelming. Gao et al. [GQL+24] further assumed that the set of senders also satisfies this condition and proposed the first FPSI protocols for $L_{p\in[1,\infty]}$ with linear complexity. However, both protocols in [vBP24] and [GQL+24] require a significant amount of public key operations, which renders their protocols inefficient in practice. This leads us to pose the following question:

Is it possible to devise a practical FPSI protocol for $L_{p\in[1,\infty]}$ distance primarily based on symmetric-key operations under reasonable assumptions?

1.2 Our Contribution

In this work, we make positive answers to the aforementioned questions. In summary, we have the following contributions:

Fuzzy Mapping Definition Revisit. We find the original definition of fuzzy mapping [GQL+24] is somewhat ad-hoc and not rigorous. They defined fuzzy mapping as a protocol that generates identifiers (IDs) for each participant's set, with specific properties for the ID function. In this work, we follow the definition paradigm of pseudorandom correlation generator (PCG) [BCG+19], rigorously defining the two ID generation algorithms (along with a setup algorithm) as fuzzy mapping scheme, and then separately defining the functionality of generating IDs, which we refer to as distributed ID generation. Moreover, we introduce two new properties for the fuzzy mapping schemes, which we term silence and consistency. The former allows that parties can generate IDs in a non-interactive manner, while the latter greatly simplifies the construction of FPSI to direct concatenation of distributed ID generation and plain PSI. Our refinement of the fuzzy mapping definition is not only conceptual, but also provides a clear and precise framework for understanding and implementing FPSI protocols.

Distributed ID Generation Protocol. To construct distributed ID generation protocol, we propose a generalized assumption, namely the s-separated set assumption, which implies both global disjoint assumption [GRS22] and existing disjoint assumption [vBP24]. We argue this assumption is mild as we have proved that, given randomly selected inputs, the set satisfies this assumption with overwhelming probability. Based on this assumption, we then construct a distributed ID generation protocol that satisfies consistency for L_∞ distance. We introduce several novel techniques, making our protocol rely entirely on oblivious transfer (OT) and symmetric-key operations. We present a variant of oblivious PRF (OPRF), which may be of independent interest, called OPRF-sum, to replace the expensive AHE operations of previous work. We also propose a new property of oblivious key-value store (OKVS) [GPR+21] called t-randomness, which is a generalization of randomness [ZCL+23] and allows us to

remove expensive DH key exchange steps from the protocol [GQL+24]. Along the way, we identify a subtle proof flaw in previous work [GRS22, ZCL+23, vBP24] when proving randomness.

Fuzzy PSI Protocols for $L_{p\in[1,\infty]}$ Distance. With the distributed ID generation protocol in hand, we construct FPSI protocols for both L_∞ distance and $L_{p\in[1,\infty)}$ distance. For L_∞ distance, the construction is straightforward: by integrating a distributed ID generation protocol that ensures consistency with the plain private set operation (PSO) framework [CZZ+24], we obtain not only the FPSI but also its variants, e.g., FPSI-CA, FPSI-SP, and labeled FPSI. For $L_{p\in[1,\infty)}$ distance, the first step is still to run the distributed ID generation protocol for L_∞ distance. Given that $L_\infty(q,w) \le L_p(q,w)$ holds for any q, w, consistency is no longer satisfied, so further filtering is required. We employ a combination of hashing and the permute + share [MS13, CGP20] to circumvent the need for the costly AHE operations required by previous approaches [GQL+24, vBP24]. In summary, we have proposed FPSI and its variants for both L_∞ and $L_{p\in[1,\infty)}$ distance mainly based on symmetric-key operations.

Experiment Evaluation. We implement our FPSI protocols for both L_∞ and $L_{p\in[1,\infty)}$ distance and compare them with the state-of-the-art FPSI protocols [GQL+24]. Experiments show that our protocols perform better than [GQL+24] under all the parameters we tested. Specifically, our protocols achieve a $2.2 - 83.9\times$ speedup in running time and $1.5 - 11.5\times$ shrinking in communication cost, depending on set sizes, dimension and distance threshold.

1.3 Overview of Our Techniques

We provide the high-level technical overview of our FPSI protocols. We use the sender $\mathcal{S}$ and the receiver $\mathcal{R}$ to denote the two parties engaged in the FPSI protocol, and use $Q = \{q_j\}_{j\in[m]} \subseteq \mathbb{U}^d$ and $W = \{w_i\}_{i\in[n]} \subseteq \mathbb{U}^d$ to denote their sets, where d is the dimension of the set elements and δ is the distance threshold. We use $q_{j,k}$ (resp. $w_{i,k}$) to denote the k-th component of q_j (resp. w_i) for $k \in [d]$.

Fuzzy mapping definition revisit. We revisit the definition of fuzzy mapping by Gao et al. [GQL+24]. They defined fuzzy mapping as a two-party protocol in which the sender $\mathcal{S}$ inputs a set Q and the receiver $\mathcal{R}$ inputs a set W. As a result, the sender outputs $\mathsf{ID}(Q) = \{\mathsf{ID}(q_j)\}_{j\in[m]}$ and the receiver outputs $\mathsf{ID}(W) = \{\mathsf{ID}(w_i)\}_{i\in[n]}$. Both $\mathsf{ID}(q_j)$ and $\mathsf{ID}(w_i)$ are sets, and the correctness requires that if $\mathsf{dist}(q_j, w_i) \le \delta$, then $\mathsf{ID}(q_j) \cap \mathsf{ID}(w_i) \neq \emptyset$. The security follows the standard simulation-based definition for secure two-party computation [Gol04, Lin17]. They also defined the expansion rate as $\mathsf{max}\{|\mathsf{ID}(q_j)|, |\mathsf{ID}(w_i)|\}$.

However, we find that their definition is somewhat ad-hoc and lacks rigor. Firstly, the ID mappings for the sender and the receiver are different, so using the same symbol, namely $\mathsf{ID}(q_j)$ and $\mathsf{ID}(w_i)$, is misleading. Secondly, correctness is a requirement for the ID mapping, while security is a requirement for the protocol execution; these are concepts at different levels and should be treated separately. Finally, the meaning of $\mathsf{ID}(q_j)$ is imprecise. In their fuzzy mapping protocol for Hamming distance, knowing q_j allows for the direct computation of $\mathsf{ID}(q_j)$, making the protocol non-interactive. In contrast, in their fuzzy mapping protocol for $L_{p\in[1,\infty]}$ distance, knowing q_j alone is insufficient to compute $\mathsf{ID}(q_j)$; additional information about W is required, rendering the protocol interactive.

We opt to define the ID mapping and the ID-generation protocol separately. This allows for a clearer and more modular framework, enabling a more precise understanding and implementation of fuzzy mapping in FPSI protocols. Following the definition paradigm of pseudorandom correlation generator (PCG) [BCG+19], we define the fuzzy mapping as a scheme, which consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{IDGen}_\mathcal{S}, \mathsf{IDGen}_\mathcal{R})$. Here, the Setup algorithm takes the sets Q and W of both parties as input and outputs a state st. The algorithm $\mathsf{IDGen}_\mathcal{S}$ (resp. $\mathsf{IDGen}_\mathcal{R}$) takes an element $q_j \in Q$ (resp. $w_i \in W$) and the state st as input, and outputs the ID set of q_j (resp. w_i), denoted as ID_{q_j} (resp. ID_{w_i}). In this way, we can distinguish between the ID generation algorithms of the sender and the receiver. Additionally, depend on whether the Setup algorithm outputs $\mathsf{st} = \perp$, we can ascertain if the ID generation algorithms $\mathsf{IDGen}_\mathcal{S}$ (resp. $\mathsf{IDGen}_\mathcal{R}$) can be computed from q_j (resp. w_i) without st. We define the case where Setup outputs $\perp$ as *silence*, which means that the parties can non-interactively compute their ID set. With the above new definition of fuzzy mapping, we can naturally define the ideal functionality for ID generation, which we call distributed ID generation functionality $\mathcal{F}_{\mathsf{didg}}$. It takes the sender's set Q and the receiver's set W as input, runs $\mathsf{Setup}(Q, W) \to \mathsf{st}$, and outputs $\{\mathsf{IDGen}_\mathcal{S}(q_j, \mathsf{st})\}_{q_j\in Q}$ to the sender and $\{\mathsf{IDGen}_\mathcal{R}(w_i, \mathsf{st})\}_{w_i\in W}$ to the receiver.

Recall that the correctness of the fuzzy mapping [GQL+24] stipulates that if $\mathsf{dist}(q_j, w_i) \le \delta$, then $\mathsf{ID}_{q_j} \cap \mathsf{ID}_{w_i} \neq \emptyset$. However, the converse may not hold, that is, if $\mathsf{dist}(q_j, w_i) > \delta$, it is still possible that

$\mathsf{ID}_{q_j} \cap \mathsf{ID}_{w_i} \neq \emptyset$. Therefore, after completing the distributed ID generation, the parties must perform a fuzzy matching step to verify whether those q_j and w_i satisfying $\mathsf{ID}_{q_j} \cap \mathsf{ID}_{w_i} \neq \emptyset$ are indeed close enough, a.k.a. $\mathsf{dist}(q_j, w_i) \leq \delta$. This paradigm is referred to as "coarse mapping" and "refined filtering" [vBP24, GQL+24]. Our key insight is that if we enhance coarse mapping to refined mapping, namely the converse is also true, i.e., $\mathsf{dist}(q_j, w_i) \leq \delta \iff \mathsf{ID}_{q_j} \cap \mathsf{ID}_{w_i} \neq \emptyset$, the "refined filtering" step could be eliminated. We define this enhanced correctness property of fuzzy mapping as *consistency*. Moreover, if the expansion rate of the fuzzy mapping is 1, then $\mathsf{ID}_{q_j} \cap \mathsf{ID}_{w_i} \neq \emptyset$ is equivalent to $\mathsf{ID}_{q_j} = \mathsf{ID}_{w_i}$. In this case, a plain PSI on the ID sets suffices to obtain the final result of FPSI.[1]

Distributed ID generation protocol for L_∞ distance. Previous discussion has reduced the task of FPSI to the construction of distributed ID generation protocol with consistency and expansion rate 1. Here we focus on L_∞ distance.

Our starting point is the private ID protocol [GMR+21, ZLDL23]. In this protocol, both parties input their respective sets, and then they learn a random ID for each element in their sets. Garimella et al. [GMR+21] proposed a private ID framework, in which the sender and receiver perform two OPRFs symmetrically. In the first OPRF, the sender learns the PRF key k_s, and the receiver learns the PRF value $\{f_{k_s}(w)\}_{w \in W}$ of its input W. In the second OPRF, the sender learns the PRF value $\{f_{k_r}(q)\}_{q \in Q}$ of its input Q, and the receiver learns the PRF key k_r. Then, the random ID for an element q is defined as

$$\mathsf{ID}_q := f_{k_s}(q) + f_{k_r}(q)$$

In our distributed ID generation protocol, we consider using a similar form for IDs. However, the attempt of using PRF to build fuzzy mapping fails because it does not satisfy the correctness requirement of fuzzy mapping, i.e., $\mathsf{dist}(q, w) \leq \delta \implies \mathsf{ID}_q = \mathsf{ID}_w$. Instead, we define the ID as

$$\mathsf{ID}_q := F_Q(q) + F_W(q)$$

where F_Q and F_W are functions defined w.r.t. Q, W, where Q and W could be viewed as a name of key. These functions satisfy the mutual requirement that for all $w \in W$, if there exists a $q \in Q$ such that $\mathsf{dist}(q, w) \leq \delta$, then $F_Q(w) = F_Q(q)$. Conversely, if for all $q \in Q$, $\mathsf{dist}(q, w) > \delta$, then $F_Q(w) \neq F_Q(q)$.

The problem now shifts to how to construct the function F_Q and how the receiver learns $\{F_Q(w)\}_{w \in W}$ (since the protocol is symmetric, the construction for F_W is similar). To construct F_Q, we draw inspiration from spatial additive sharing strategies [GQL+24]. We assign a random value $\mathsf{Rand}^{\mathcal{S}}_{j,k}$ to each dimension $q_{j,k}$ of the sender's elements and define $F_Q(q_j) := \sum_{k \in [d]} \mathsf{Rand}^{\mathcal{S}}_{j,k}$ for any $q_j \in Q$. To enable the receiver to obtain $F_Q(w_i)$ in an oblivious manner, the sender uses an oblivious key-value store (OKVS) [GPR+21] to encode the key-value pairs $A := \{(k||(q_{j,k} + t), \mathsf{Rand}^{\mathcal{S}}_{j,k})\}_{j \in [m], k \in [d], t \in [-\delta, \delta]}$, i.e., $E_S := \mathsf{Encode}(A)$. The receiver then obtains the decoding result, i.e., $F_Q(w_i) := \sum_{k \in [d]} \mathsf{Decode}(E_S, w_{i,k})$. However, this approach has two issues. Firstly, since the values in the OKVS are not all randomly selected (multiple keys may correspond to the same value), the obliviousness property is not satisfied. Directly sending E_S would leak information about the sender's set Q. Secondly, due to potential overlaps in the sender's elements, i.e., there exist $j_1, j_2 \in [m], k \in [d], t_1, t_2 \in [-\delta, \delta]$ such that $q_{j_1,k} + t_1 = q_{j_2,k} + t_2$, the OKVS cannot simultaneously encode both $(k||q_{j_1,k} + t_1, \mathsf{Rand}^{\mathcal{S}}_{j_1,k})$ and $(k||q_{j_2,k} + t_2, \mathsf{Rand}^{\mathcal{S}}_{j_2,k})$.

To address the first issue, Gao et al. [GQL+24] suggested encrypting the OKVS with additive homomorphic encryption (AHE), i.e., $\tilde{E}_S := \mathsf{Enc}_{pk_S}(E_S)$, and send it to the receiver. The receiver then selects a random $mask_i$ and computes $mask_i \cdot \sum_{k \in [d]} \mathsf{Decode}(\tilde{E}_S, k||w_{i,k})$, sends the ciphertext back to the sender, who decrypts it and sends the result back to the receiver. The receiver then removes the $mask_i$, i.e., $F_Q(w_i) := mask_i^{-1} \cdot m_i$, where m_i is the decrypted message from the sender. To address the second issue, they used later random values to overwrite earlier ones, as illustrated in Figure 1. However, this approach introduces false positives, i.e., in some dimension k, even if $|w_{i,k} - q_{j,k}| > \delta$, it is still possible that $\mathsf{Decode}(E_S, w_{i,k}) = \mathsf{Rand}^{\mathcal{S}}_{j,k}$. This may result in two points w_{i_1}, w_{i_2} satisfying $F_Q(w_{i_1}) = F_Q(w_{i_2})$. If the receiver observes this, it can infer that some element of the sender is near w_{i_1} or w_{i_2}, thereby leaking information about the sender. [2]

[1] Since the plain PSI only reveals the elements in the receiver's set, the protocol obtained is FPSI with sender privacy (FPSI-SP). The standard FPSI could be obtained by a small tweak, see Section 4.1 for details.

[2] Even if the receiver eventually obtains this element, $F_Q(w_{i_1}) = F_Q(w_{i_2})$ still tells the receiver that the sender has another element that overlaps with this element in a certain dimension, which is not allowed.

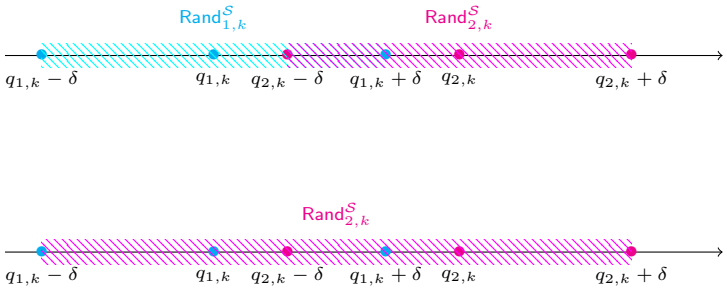

Fig. 1. Illustration of Gao et al.'s solution of overlapping. As shown in the upper part, in the k-th dimension, the value corresponding to the key in interval $[q_{1,k}-\delta, q_{1,k}+\delta]$ is $\mathsf{Rand}^{\mathcal{S}}_{1,k}$ (represented in cyan), and the value corresponding to the key in interval $[q_{2,k}-\delta, q_{2,k}+\delta]$ is $\mathsf{Rand}^{\mathcal{S}}_{2,k}$ (represented in magenta). To determine the values corresponding to the keys in the overlapping interval $[q_{2,k}-\delta, q_{1,k}+\delta]$, they used the later values to overwrite the former values, as shown in the lower part of the figure. At this point, all the values corresponding to the keys in the entire interval $[q_{1,k}-\delta, q_{2,k}+\delta]$ are modified to $\mathsf{Rand}^{\mathcal{S}}_{2,k}$.

To mitigate this, Gao et al. [GQL+24] employed a Diffie-Hellman key exchange protocol to prevent the receiver from directly obtaining $F_Q(w_i)$. Instead, the receiver obtains $(F_Q(w_i)+F_W(w_i))^{sk_S \cdot sk_R}$, where sk_S and sk_R are the DH keys of $\mathcal{S}$ and $\mathcal{R}$, respectively. This ensures that even if $F_Q(w_{i_1}) = F_Q(w_{i_2})$, the receiver cannot infer the sender's information because $F_W(w_{i_1}) \neq F_W(w_{i_2})$.[3]

In this work, we take an approach departing from that of Gao et al. to address the aforementioned issues. The advantage of our approach is that it does not require public-key operations such as AHE and DH, and it eliminates false positives (thus satisfying consistency), thereby obviating the need for a fuzzy matching step. To solve the first problem, we borrow the idea from oblivious programmable PRF [KMP+17], which involves using an OPRF to generate one-time pads for encrypting the values. Specifically, the sender generates a PRF key sk and defines a new set of key-value pairs as $A := \left\{(k||(q_{j,k}+t),\ \mathsf{Rand}^{\mathcal{S}}_{j,k} + F_{sk}(k||(q_{j,k}+t))\right\}_{j\in[m],k\in[d],t\in[-\delta,\delta]}$. The sender then computes the OKVS as $E_S := \mathsf{Encode}(A)$ and sends it to the receiver. The receiver computes and outputs $F_Q(w) := \mathsf{Decode}(E_S, w) - f_{sk}(w)$, where $f_{sk}(w)$ is the receiver's output from the OPRF. However, this idea cannot be directly applied to our scenario because we need the receiver to obtain the *sum* of some decoded values. If the receiver directly obtains each component and sums them locally, it may be insecure and could leak information about the sender. To address this issue, we propose a new variant of OPRF, which we call OPRF-sum. In OPRF-sum, the sender still generates a PRF key sk, and the receiver inputs a set of queries $\{w_{i,k}\}_{k\in[d]}$ and obtains the *sum* of their PRF values, $f_i = \sum_{k\in[d]} f_{sk}(w_{i,k})$. Based on OPRF-sum, the sender still computes the OKVS as $E_S := \mathsf{Encode}(A)$ and sends it to the receiver. The receiver then computes $F_Q(w_i) := \sum_{k\in[d]} \mathsf{Decode}(E_S, w_{i,k}) - f_i$. Since the receiver does not know the PRF value for each $w_{i,k}$, no additional information is leaked.

To address the second issue, namely the overlap between different elements of the sender, our initial idea is to encode all dimensions simultaneously instead of encoding each dimension separately. That is, instead of encoding $A = \{(k||(q_{j,k}+t), \mathsf{Rand}^{\mathcal{S}}_{j,k})\}_{j\in[m],k\in[d],t\in[-\delta,\delta]}$, we encode $B = \{(q_j + t, \mathsf{Rand}^{\mathcal{S}}_j)\}_{j\in[m],t\in[-\delta,\delta]^d}$. The existing disjoint assumption [vBP24] ensures that for any $j \in [m]$, the keys in B do not overlap (since each element has at least one dimension where its distance from all other elements is greater than 2δ). However, this approach causes the number of encoded key-value pairs to

[3] It could be easily verified from the definition of F_W.

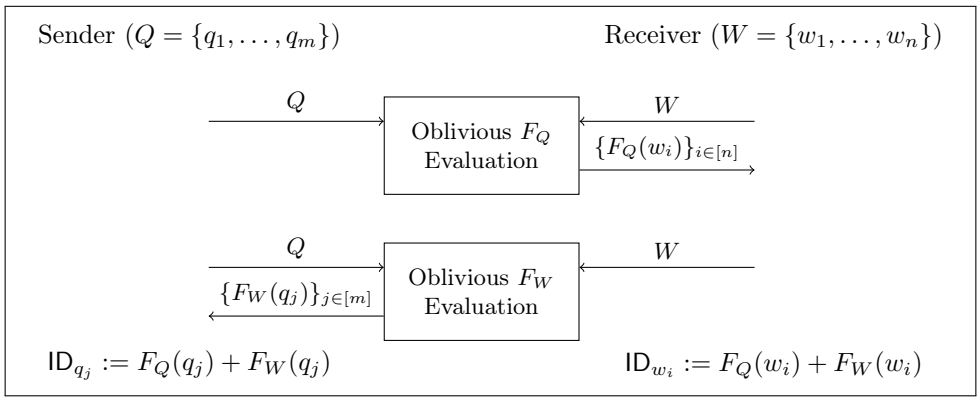

Fig. 2. Core idea of our distributed ID generation protocol. The parties symmetrically execute two instances of oblivious evaluation protocols for function F. The function F_Q satisfies the condition that for all $w \in W$, if there exists a $q \in Q$ such that $\mathsf{dist}(q, w) \leq \delta$, then $F_Q(w) = F_Q(q)$. The same property also holds for F_W.

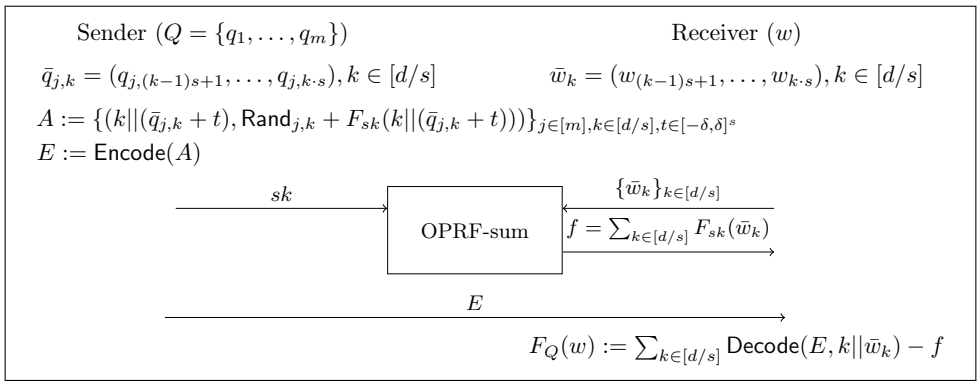

Fig. 3. Core idea of our oblivious evaluation protocol of F_Q.

expand from $(2\delta + 1)dm$ to $(2\delta + 1)^d m$. Another idea is to strengthen the assumption to the global disjoint assumption [GRS22], which assumes that every element has a distance greater than 2δ from all other elements in every dimension, thereby avoiding overlap issues. However, this assumption is too strong and may be difficult to satisfy in practice. Therefore, we propose an intermediate assumption, which we call the s-separated set assumption. This assumption states that for every s dimensions of each element, at least one dimension has a distance greater than 2δ from all other elements. When $s = 1$, this reduces to the global disjoint assumption [GRS22], and when $s = d$, it degenerates to the existing disjoint assumption [vBP24]. Based on this assumption, we can encode the set elements for every s dimensions, resulting in $(2\delta+1)^s \cdot d/s \cdot m$ encoded pairs. Similar to [vBP24], we also demonstrate that when $s = \Omega(d^{1/c})$, where c is an arbitrary constant, the probability that a randomly selected set satisfies the s-separated assumption is overwhelming. Note that once the overlap issue is resolved, the problem of false positives is eliminated. Therefore, our protocol satisfies consistency. Furthermore, we find that if the OKVS satisfies *separate sum randomness*, which is a new property we defined and can be further guaranteed by our proposed t-randomness, then the generated $F_Q(w)$ is indistinguishable from a uniform random distribution. As a result, the probability of $\exists w_{i_1} \neq w_{i_2}$, s.t., $F_Q(w_{i_1}) = F_Q(w_{i_2})$ is negligible, allowing us to remove the DH-key exchange step without compromising security. Along the way, we also find the prior proof [GRS22, vBP24, ZCL+23] for randomness of OKVS has a subtle flaw.

In summary, by employing the techniques mentioned above, we can implement a distributed ID generation protocol that satisfies consistency using only symmetric-key operations and OT. The high-level idea of our distributed ID generation protocol is shown in Figure 2 and 3.

Fuzzy PSI for $L_{p\in[1,\infty]}$ distance. Now we discuss how to construct FPSI protocol and its variants for the L_∞ and $L_{p\in[1,\infty)}$ distances, respectively. For the L_∞ distance, the construction of the FPSI protocol is relatively simple. Since the distributed ID generation protocol has already transformed $\mathsf{dist}(q_j, w_i) \le \delta$ into $\mathsf{ID}_{q_j} = \mathsf{ID}_{w_i}$, we can directly use plain PSI to implement FPSI. We remark that there is a subtle consideration here: since the receiver only outputs elements from its own set, plain PSI actually implements FPSI with sender privacy (FPSI-SP) [vBP24]. FPSI requires the receiver to obtain elements from the sender's set, so we actually need to perform multi-query reverse private membership test (mq-RPMT) [ZCL+23] to test whether $\mathsf{ID}_{w_i} \in \mathsf{ID}_Q$ and then use OT to retrieve the sender's elements. More details can be found in Section 4.1.

For the $L_{p\in[1,\infty)}$ distance, since $\mathsf{dist}_p(q, w) \ge \mathsf{dist}_\infty(q, w)$ for any q, w, the distributed ID generation protocol no longer satisfies consistency for $L_{p\in[1,\infty)}$ distance. Therefore, we need to further test whether elements q and w sharing the same identifier $\mathsf{ID}_w = \mathsf{ID}_q$ also satisfy $\mathsf{dist}_p(q, w) \le \delta$. Our starting point is the method proposed by Gao et al. [GQL+24]. The idea is to first have the receiver encode $\{(id_{w_i}||k||(w_{i,k}+t), |t|^p)\}_{i\in[n],k\in[d],t\in[-\delta,\delta]}$ using OKVS. For any $q_j \in Q$, if $\mathsf{dist}_p(q_j, w_i) \le \delta$, then let $d_j := \sum_{k\in[d]} \mathsf{Decode}(E_R, \mathsf{ID}_{q_j}||k||q_{j,k}) = \sum_{k\in[d]} |q_{j,k} - w_{i,k}|^p = \mathsf{dist}_p(w_i, q_j)^p$. Since the sender cannot directly obtain the distance information, they use AHE to encrypt the OKVS, i.e., $\tilde{E}_R := \mathsf{Enc}_{pk_R}(E_R)$, and have the sender choose a random r_j and compute $ct_j := \mathsf{Decode}(\tilde{E}_R, q_j) + r_j$ to return to the receiver. The receiver decrypts to obtain $v_j := \mathsf{Dec}(ct_j) = \mathsf{dist}_p(q_j, w_i)^p + r_j$. Subsequently, the parties invoke the fuzzy matching for interval [CFR23] functionality $\mathcal{F}_{\mathsf{fmat}}$ to test whether $v_j \in [r_j, r_j + \delta^p]$, which is equivalent to $\mathsf{dist}_p(q_j, w_i)^p \in [0, \delta^p]$.

To avoid using AHE, we still consider using OPRF to generate a one-time pad to encrypt the value. However, the method we used in the distributed ID generation protocol is no longer applicable here. The reason is, in our aforementioned distributed ID generation protocol for L_∞ distance, the party encoding the OKVS and the party obtaining the decoding result are different, i.e., the receiver encodes, and the sender obtains the decoding result. In this case, however, the party encoding and the party obtaining the decoding result are the same, i.e., the receiver encodes and obtains the decoding result, while the sender only provides the decoding key and the mask r but cannot know the decoding result.

To address this issue, our key observation is that the result of the above scheme is essentially a random additive secret sharing of d_j between the parties. The sender selects r_j as its own share, and the receiver's share is $r_j + d_j$. Therefore, we consider letting the receiver choose its own random share r_j, encoding r_j into the OKVS, and then having the sender use the decoding result as its own share. As a result, this scenario becomes the same as in the distributed ID generation protocol, i.e., the receiver encodes OKVS, and the sender obtains the decoding result. However, since the receiver does not know the order in which the sender decodes, it is challenging for the receiver to encode r_j into the OKVS. To enable the two parties to share d_j, we need the receiver to align the sender's set elements in some way. To solve this problem, we consider having the sender use cuckoo hashing to hash its set into a cuckoo hashing table, while the receiver selects a random share r_j for each bin and includes the hash index in the encoding key. This ensures that the sender's decoding order follows the bin index. We note that introducing cuckoo hashing also leads to a new problem: the receiver may eventually know whether the elements in the i-th cuckoo hash bin of the sender are outputs of FPSI, which leaks information about the sender's entire set. To address this, we consider having the sender choose a random permutation π, and then the two parties execute a permute + share protocol [MS13, CGP20], ensuring that the receiver does not obtain information about the sender's cuckoo hashing index.

Combining all the above techniques, we obtain FPSI protocols for L_∞ and $L_{p\in[1,\infty)}$ distances based solely on OT and symmetric-key operations. We refer to Section 4 for more details.

1.4 Related Work

We briefly review previous FPSI protocols. The asymptotic complexities are given in Table 1.

FPSI for Hamming distance. Freedman et al. [FNP04] formally defined the problem of secure fuzzy matching, which involves determining whether the Hamming distance between two given tuples falls below a predefined threshold. Their protocol relies on additively homomorphic encryption (AHE) and polynomial interpolation, however, it was later proven to be insecure [CH08]. Subsequent works [IW06, YSPW09, UCK+21, CFR23] focused on the Hamming distance as well. These approaches require

traversing all possible element pairs for fuzzy matching, resulting in communication and computational overhead that scales at least proportionally to the product of the set sizes, i.e., $O(mn)$. Chongchitmate et al. [CLO24] introduced an assumption regarding the spatial distribution of participant set elements: for any $q \in Q, w \in W$, either $\mathsf{dist}(q,w) < \delta$ or $\mathsf{dist}(q,w) > t\delta$, where $t = O(\log n)$. This assumption effectively eliminates the mn factor in both communication and computation complexity. However, their protocol implementation relies on garbled circuits and requires multiple invocations of PSI, which significantly compromises its practical efficiency. Gao et al. [GQL+24] proposed the UniqC assumption and designed a novel FPSI protocol for Hamming distance that similarly eliminates the mn factor. Despite this improvement, their protocol still depends on computationally expensive AHE operations. Most recently, Blass and Noubir [BN25] presented an FPSI protocol for Hamming distance utilizing predicate encryption. While their approach eliminates the need for specific assumptions and achieves linear communication complexity, it still maintains a computational complexity of $O(mn)$.

FPSI for $L_{p\in[1,\infty]}$ distance. Garimella et al. [GRS22] introduced the notion of structure-aware PSI (sa-PSI), where the receiver holds a set with a publicly known structure and the sender holds an unstructured set of points. They focused on the case where the input set is the union of n balls of radius δ with respect to the L_∞ norm in d-dimensional space. Consequently, performing FPSI with the receiver's input $W = \{w_i\}_{i\in[n]}$ is equivalent to executing sa-PSI with the receiver's input $W' = \{w' | \mathsf{dist}(w', w) \leq \delta, w \in W\}$. Their subsequent works further considered malicious secure sa-PSI [GRS23] and reducing the computational cost of sa-PSI [GGM24], both focusing on the L_∞ distance. Baarsen and Pu [vBP24] proposed the first FPSI protocol for general $L_{p\in[1,\infty]}$ distance. Their construction is based on a novel usage of random self-reduction of DDH tuples and a two-layer OKVS encoding strategy. Gao et al. [GQL+24] first proposed the FPSI protocol for $L_{p\in[1,\infty]}$ distance that exhibits linear complexity with respect to set size m and n, dimension d, and distance threshold δ by introducing interactive fuzzy mapping. However, their construction incurs significant computational costs owing to the extensive use of AHE operations. Recently, Richardson et al. [RRX24] introduced a FPSI framework that supports arbitrary distance metrics. Their protocol exhibits a mere logarithmic dependency on the distance threshold δ. However, its heavy reliance on the general 2PC technique, e.g., garbled circuits, results in inefficiencies.

Metric	Protocol	Assumption	Communication	Computation	
				Sender	Receiver
L_∞	[GRS22]	$\mathcal{R}$. $\mathsf{min} > 2\delta$	$O((4\log\delta)^d n + m)$	$O((2\log\delta)^d m)$	$O((2\delta)^d n)$
		$\mathcal{R}$. $\mathsf{min} > 4\delta$	$O(2^d dn\log\delta + m)$	$O(dm\log\delta)$	$O((2\delta)^d n)$
		$\mathcal{R}$. 1-Separate	$O(dn\log\delta + m)$	$O(dm\log\delta)$	$O((2\delta)^d n)$
	[vBP24]	$\mathcal{R}$. $\mathsf{min} > 2\delta$	$O(\delta dn + 2^d m)$	$O(2^d dm)$	$O(\delta dn + 2^d m)$
		$\mathcal{R}$. $\mathsf{min} > 4\delta$	$O(\delta 2^d dn + m)$	$O(dm)$	$O(\delta 2^d dn + m)$
		$\mathcal{R}$. d-Separate	$O((\delta d)^2 n + m)$	$O((\delta d)^2)n + m$	$O(d^2 m)$
	[GGM24]	-	$O(dn\log\delta + m(2\log\delta)^d)$	$O(m2^d((\log\delta)^d + d\log\delta))$	$O((\log\delta)^d m + d\log\delta(m+n))$
	[GQL+24]	$\mathcal{R}\wedge\mathcal{S}$. d-Separate	$O(\delta d(m+n))$	$O(\delta dm + n)$	$O(\delta dn + m)$
	[RRX24]	$\mathcal{R}$. Disjoint Hash	$O(d\log\delta(n2^s + m2^{d-s}))$	$O(2^s nd\log\delta)$	$O(2^{d-s} md\log\delta)$
	Ours	$\mathcal{R}\wedge\mathcal{S}$. s-Separate	$O(\delta^s \frac{d}{s}(m+n))$	$O(\delta^s \frac{d}{s} m + n)$	$O(\delta^s \frac{d}{s} n + m)$
$L_{p\in[1,\infty)}$	[vBP24]	$\mathcal{R}$. $\mathsf{min} > 2\delta(d^{\frac{1}{p}} + 1)$	$O(\delta 2^d dn + \delta^p m)$	$O(\delta 2^d dn + m)$	$O((d+\delta^p)m)$
		LSH	$O(\delta dn^{1+\rho} + \delta^p mn^\rho \log n)$	$O(\delta dn^{1+\rho} + mn^\rho \log n)$	$O((d+\delta^p)mn^\rho \log n)$
	[GQL+24]	$\mathcal{R}\wedge\mathcal{S}$. d-Separate	$O(\delta d(m+n) + pm\log\delta)$	$O((\delta d + p\log\delta)m + n)$	$O(\delta dn + p\log\delta m)$
	Ours	$\mathcal{R}\wedge\mathcal{S}$. s-Separate	$O(\delta^s \frac{d}{s}(m+n) + pm\log\delta)$	$O((\delta^s \frac{d}{s} + p\log\delta)m + n)$	$O(\delta^s \frac{d}{s} n + p\log\delta m)$

Table 1. Asymptotic complexities of existing FPSI protocols for $L_{p\in[1,\infty]}$ distance, where sender holds m elements and receiver holds n elements in a d-dimensional space. δ is the threshold of FPSI. $0 < \rho < 1$ is a parameter in LSH scheme. $1 \leq s \leq d$ is a parameter of the parties' choice. We ignore multiplicative factors of the computational security parameter κ and statistical parameter λ. $\mathsf{min} > l$ means that the minimum distance between points of the set is greater than l. $\mathcal{R}$. denotes that the set of receivers satisfies the assumption, $\mathcal{R}\wedge\mathcal{S}$ indicates that both the sets of senders and receivers satisfy the assumption.

2 Preliminaries

2.1 Notation

We use κ to denote the computational security parameter. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a bit string b we let b_i denote the ith bit. We use the abbreviation PPT to denote probabilistic polynomial-time. We say that a function f is negligible in κ if it vanishes faster than the inverse of any polynomial in κ, and write it as $f(\kappa) = \mathsf{negl}(\kappa)$. $a := b$ denotes that a is assigned by b. We use $\mathbb{U}$ to denote the universe of item components, and $Q = \{q_1, \ldots, q_m\} \subseteq \mathbb{U}^d$ to denote a set of m d-dimensional elements. In this work, we consider L_∞ and $L_{p\in[1,\infty)}$ distance metrics, i.e., $\mathsf{dist}_\infty(q, w) = \mathsf{max}_{k\in[d]}|q_k - w_k|$ and $\mathsf{dist}_p(q, w) = (\sum_{k\in[d]} |q_k - w_k|^p)^{\frac{1}{p}}$, where $q = (q_1, \ldots, q_d), w = (w_1, \ldots, w_d)$.

2.2 Oblivious Key-Value Stores

The Oblivious Key-Value Store (OKVS) [PRTY20, GPR+21, ZCL+23] is a data structure that maps a set of keys to corresponding values. The obliviousness means that when the values are randomly selected, the distribution of the data structure is independent from the key set. Polynomials are a typical example of OKVS, where encoding corresponds to interpolation. The formal definition is as follows:

Definition 1 (Oblivious Key-Value Store). *An OKVS is parameterized by a set $\mathcal{K}$ of keys, a set $\mathcal{V}$ of values, and consists of two algorithms:*

- $\mathsf{Encode}(\{(x_1, y_1), \ldots, (x_n, y_n)\})$*: on input key-value pairs $\{(x_i, y_i)\}_{i\in[n]} \subseteq \mathcal{K} \times \mathcal{V}$, outputs an object D (or, probability $2^{-\lambda}$, an error $\perp$).*
- $\mathsf{Decode}(D, x)$ *: on input D and a key x, outputs a value $y \in \mathcal{V}$.*

Correctness. For all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys:

$$(x, y) \in A \text{ and } \perp \neq D \leftarrow \mathsf{Encode}(A) \Longrightarrow \mathsf{Decode}(D, x) = y$$

Obliviousness. For all distinct $\{x_1^0, \ldots, x_n^0\}$ and $\{x_1^1, \ldots, x_n^1\}$, if Encode does not output $\perp$ for $\{x_1^0, \ldots, x_n^0\}$ or $\{x_1^1, \ldots, x_n^1\}$, then the following distributions are computationally indistinguishable:

$$\{D | y_i \leftarrow \mathcal{V}, i \in [n], \mathsf{Encode}((x_1^0, y_1), \ldots, (x_n^0, y_n))\}$$
$$\{D | y_i \leftarrow \mathcal{V}, i \in [n], \mathsf{Encode}((x_1^1, y_1), \ldots, (x_n^1, y_n))\}$$

Linearity [GPR+21]. An OKVS is *linear* (over a field $\mathbb{F}$) if $\mathcal{V} = \mathbb{F}$ (i.e., "values" are elements of $\mathbb{F}$), the output of Encode is a vector D in $\mathbb{F}^m$, and the Decode function is defined as: $\mathsf{Decode}(D, x) = \langle \mathsf{row}(x), D\rangle := \sum_{j=1}^m \mathsf{row}(x)_j D_j$ for some function $\mathsf{row} : \mathcal{K} \to \mathbb{F}^m$. Hence Decode is a linear map from $\mathbb{F}^m$ to $\mathbb{F}$. Specifically, if $\mathsf{row} : \mathcal{K} \to \{0, 1\}^m \subseteq \mathbb{F}^m$, the OKVS is called binary OKVS.

The ratio $n/m \leq 1$ is called the rate of OKVS. Polynomials achieve the optimal rate 1 (i.e., $m = n$) but with a super-linear encoding complexity. Recent advancements of OKVS [GPR+21, RR22, BPSY23] achieve linear encoding with a near optimal rate. Concretely, these schemes achieves $m = (1 + \epsilon)n$ for some small ϵ, i.e., $0 < \epsilon < 0.3$.

In this work, we propose a new property of OKVS called *t-Randomness*, which is a generalization of randomness [ZCL+23]. We first recall the definition of randomness and then propose our t-randomness as follows.

Randomness [ZCL+23]. For any $A = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and for any $x^* \notin \{x_1, \ldots, x_n\}$, the distribution of $\mathsf{Decode}(D, x^*)$ is statistically indistinguishable to that of uniform distribution over $\mathcal{V}$, where $D \leftarrow \mathsf{Encode}(A)$.

t-Randomness. For any $A = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and for any $x_i^* \notin \{x_1, \ldots, x_n\}, i \in [t]$, the distribution of $\{\mathsf{Decode}(D, x_i^*)\}_{i\in[t]}$ is statistically indistinguishable to that of uniform distribution over $\mathcal{V}^t$, where $D \leftarrow \mathsf{Encode}(A)$.

Previous works only consider 1-randomness [GRS22, ZCL+23, BPSY23, vBP24]. Somewhat surprisingly, we find that previous works implicitly require a new property of OKVS when proving 1-randomness, which we call $(n + 1)$-independence. This property cannot be derived from the correctness of OKVS and needs to be clarified. In Appendix A.1, we provide a generalized definition of

t-independence and prove that $(n+t)$-independence implies t-randomness. We also give a detailed discussion of the issues in the proofs of prior works.

In this work, we also introduce a customized property of OKVS called *separate sum randomness* to better showcase our security proof idea. See section 3.4 for more detail.

2.3 Oblivious Transfer

Oblivious transfer (OT) [Rab05] is an important cryptographic primitive used in various multiparty computation protocols. The most commonly used variant is the 1-out-of-2 OT. We give the formal definition of 1-out-of-2 OT functionality in Figure 4.

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, message length κ
Functionality:

- Wait for input $b \in \{0,1\}$ from the receiver $\mathcal{R}$.
- Wait for input (x_0, x_1) from the sender $\mathcal{S}$.
- Give x_b to the receiver $\mathcal{R}$.

Fig. 4. 1-out-of-2 Oblivious Transfer Functionality $\mathcal{F}_{\mathsf{ot}}$

2.4 Oblivious PRF with Shared-Output

OPRF [FIPR05] allows the receiver to input x and learn $F_k(x)$, where F is a PRF, and k is known to the sender. OPRF with shared-output [APRR24, vBS24] is a variant of OPRF, in which it returns the secret sharing result of the PRF value to the parties. The ideal functionality for OPRF with shared-output is shown in Figure 5.

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, a PRF F, query sizes n
Functionality:

- Wait for input $\{x_1, \ldots, x_n\}$ from the receiver $\mathcal{R}$.
- Wait for input k from the sender $\mathcal{S}$.
- Pick random a_i from the range of PRF and compute $b_i := F_k(x_i) - a_i, i \in [n]$.
- Give $\{a_i\}_{i\in[n]}$ to the sender $\mathcal{S}$ and give $\{b_i\}_{i\in[n]}$ to the receiver $\mathcal{R}$.

Fig. 5. OPRF with Shared-Output Functionality $\mathcal{F}_{\mathsf{oprf\text{-}so}}$

2.5 Multi-Query (Reverse) Private Membership Test

In multi-query private membership test (mq-PMT), the sender inputs a set $Q = \{q_j\}_{j\in[m]}$ and the receiver inputs a set $W = \{w_i\}_{i\in[n]}$. As a result, the receiver learns an indication bit string $b \in \{0,1\}^n$, where $b_i = 1$ if and only if $w_i \in Q$. Note that mq-PMT is actually equivalent to plain PSI. The multi-query reverse private membership test (mq-RPMT) [ZCL+23, CZZ+24] is different from mq-PMT in the definition of b, where $b \in \{0,1\}^m$ and $b_j = 1$ if and only if $q_j \in W$. Chen et al. [CZZ+24] demonstrated that mq-RPMT can be used to construct various private set operation (PSO) protocols. The formal definition of mq-(R)PMT is given in Figure 6.

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, set sizes m and n
Functionality:

- Wait for input $Q = \{q_1, \ldots, q_m\}$ from the sender $\mathcal{S}$.
- Wait for input $W = \{w_1, \ldots, w_n\}$ from the receiver $\mathcal{R}$.
-
  - For mq-PMT: Define $b \in \{0,1\}^n$, where $b_i = 1$ if and only if $w_i \in Q, i \in [n]$.
  - For mq-RPMT: Define $b \in \{0,1\}^m$, where $b_j = 1$ if and only if $q_j \in W, j \in [m]$.
- Give output b to the receiver $\mathcal{R}$.

Fig. 6. Multi-Query (Reverse) Private Membership Test Functionality $\mathcal{F}_{\mathsf{mq\text{-}pmt}}/\mathcal{F}_{\mathsf{mq\text{-}rpmt}}$

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, threshold δ.
Functionality:

- Wait for input $x \in \mathbb{Z}$ from the sender $\mathcal{S}$.
- Wait for input $y \in \mathbb{Z}$ from the receiver $\mathcal{R}$.
- Define $b = 1$ if $y \in [x - \delta, x + \delta]$, otherwise $b = 0$.
- Give b to the receiver $\mathcal{R}$.

Fig. 7. Fuzzy Matching for Interval Functionality $\mathcal{F}_{\mathsf{fmat}}$

2.6 Fuzzy Matching for Interval

In this section, we define the functionality of fuzzy matching for interval [CFR23, GQL+24]. The formal definition of this functionalities is given in Figure 7.

2.7 Permute+ Share

The Permute + Share [MS13, CGP20] functionality receives a permutation from the sender and a vector from the receiver. As a result, the parties learn additive shares of the permuted vector. The formal definition of Permute + Share functionality is given in Figure 8.

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, a finite field $\mathbb{F}$, vector length n.
Functionality:

- Wait for input π from the receiver $\mathcal{R}$.
- Wait for input $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}^n$ from the sender $\mathcal{S}$.
- Pick random $a_i \leftarrow \mathbb{F}$ and compute $b_i = x_{\pi(i)} - a_i$ for $i \in [n]$.
- Give $\boldsymbol{a} = (a_1, \ldots, a_n)$ to the sender $\mathcal{S}$ and give $\boldsymbol{b} = (b_1, \ldots, b_n)$ to the receiver $\mathcal{R}$.

Fig. 8. Permute + Share Functionality $\mathcal{F}_{\mathsf{ps}}$

2.8 Fuzzy Private Set Intersection and Its Variants

In this section, we define the functionality of fuzzy PSI and its variants, including fuzzy PSI (FPSI), fuzzy PSI-cardinality (FPSI-CA), labeled fuzzy PSI (LFPSI), and fuzzy PSI with sender privacy (FPSI-SP). The formal definition of these functionalities is given in Figure 9.

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, set sizes m and n
Functionality:

- Wait for input $Q = \{q_j\}_{j\in[m]} \subseteq \mathbb{U}^d$ from the sender $\mathcal{S}$.
  - For LFPSI, wait another input $\mathsf{Label}_Q = \{\mathsf{label}_j\} \subseteq \{0,1\}^l$ from $\mathcal{S}$.
- Wait for input $W = \{w_i\}_{i\in[n]} \subseteq \mathbb{U}^d$ from the receiver $\mathcal{R}$.
  - For FPSI: Give output $I := \{q_j | \exists i \in [n], \mathsf{dist}(q_j, w_i) \leq \delta\}$ to the receiver $\mathcal{R}$.
  - For FPSI-SP: Give output $I := \{w_i | \exists j \in [m], \mathsf{dist}(q_j, w_i) \leq \delta\}$ to the receiver $\mathcal{R}$.
  - For FPSI-CA: Give output $I := \sum_{i\in[n],j\in[m]} \mathbf{1}(\mathsf{dist}(q_j, w_i) \leq \delta)$ to the receiver $\mathcal{R}$.
  - For LFPSI: Give output $I := \{\mathsf{label}_j | \exists i \in [n], \mathsf{dist}(q_j, w_i) \leq \delta\}$ to the receiver $\mathcal{R}$.

Fig. 9. Fuzzy PSI and Its Variants Functionalities $\mathcal{F}_{\mathsf{fpsi}}, \mathcal{F}_{\mathsf{fpsi\text{-}sp}}, \mathcal{F}_{\mathsf{fpsi\text{-}ca}}, \mathcal{F}_{\mathsf{lfpsi}}$

3 Fuzzy Mapping

3.1 Definition of Fuzzy Mapping

In this section, we define fuzzy mapping from the perspective of cryptographic primitives. The ideal functionality for generating fuzzy mapping between two parties will be given in Section 3.4. The formal definition of fuzzy mapping is as follows.

Definition 2 (Fuzzy Mapping). *A $(\mathsf{r}_\mathcal{S}, \mathsf{r}_\mathcal{R})$-Fuzzy Mapping scheme for distance metric* dist *consists a tuple of algorithms* $\mathsf{Fmap} = (\mathsf{Setup}, \mathsf{IDGen}_\mathcal{S}, \mathsf{IDGen}_\mathcal{R})$*:*

- $\mathsf{Setup}(Q, W)$*: on input two sets* $Q = \{q_j\}_{j\in[m]} \subseteq \mathbb{U}^d, W = \{w_i\}_{i\in[n]} \subseteq \mathbb{U}^d$*, outputs a state* st*.*
- $\mathsf{IDGen}_\mathcal{S}(q, \mathsf{st})$*: on input an item* $q \in \mathbb{U}^d$ *and a state* st*, outputs a size-*$\mathsf{r}_\mathcal{S}$ *ID set* $\mathsf{ID}_q = \{id_1^\mathcal{S}, \ldots, id_{\mathsf{r}_\mathcal{S}}^\mathcal{S}\}$*.*
- $\mathsf{IDGen}_\mathcal{R}(w, \mathsf{st})$*: on input an item* $w \in \mathbb{U}^d$ *and a state* st*, outputs a size-*$\mathsf{r}_\mathcal{R}$ *ID set* $\mathsf{ID}_w = \{id_1^\mathcal{R}, \ldots, id_{\mathsf{r}_\mathcal{R}}^\mathcal{R}\}$*.*

$\mathsf{r}_\mathcal{S}$ *and* $\mathsf{r}_\mathcal{R}$ *are the expansion rates of the sender and receiver, respectively. The expansion rate of the Fuzzy mapping scheme is* $\mathsf{r} := \mathsf{max}(\mathsf{r}_\mathcal{S}, \mathsf{r}_\mathcal{R})$*.*

Correctness. For any $q \in Q$ and $w \in W$, if $\mathsf{dist}(q, w) \leq \delta$, then $\mathsf{ID}_q \cap \mathsf{ID}_w \neq \emptyset$, where $\mathsf{st} \leftarrow \mathsf{Setup}(Q, W), \mathsf{ID}_q \leftarrow \mathsf{IDGen}_\mathcal{S}(q, \mathsf{st}), \mathsf{ID}_w \leftarrow \mathsf{IDGen}_\mathcal{R}(w, \mathsf{st})$.
Consistency. In addition to correctness, its opposite also holds. That is, for any $q \in Q$ and $w \in W$, if $\mathsf{dist}(q, w) \leq \delta$, then $\mathsf{ID}_q \cap \mathsf{ID}_w \neq \emptyset$. Conversely, if $\mathsf{dist}(q, w) > \delta$, then the probability of $\mathsf{ID}_q \cap \mathsf{ID}_w \neq \emptyset$ is negligible.
Distinctiveness. $\forall i, i' \in [n], i \neq i'$, the probability of $\mathsf{ID}_{w_i} \cap \mathsf{ID}_{w_{i'}} \neq \emptyset$ is negligible, where $\mathsf{st} \leftarrow \mathsf{Setup}(Q, W), \mathsf{ID}_{w_i} \leftarrow \mathsf{IDGen}_\mathcal{R}(\mathsf{st}, w_i), \mathsf{ID}_{w_{i'}} \leftarrow \mathsf{IDGen}_\mathcal{R}(\mathsf{st}, w_{i'})$.
Randomness. For any $Q = \{q_j\}_{j\in[m]} \subseteq \mathbb{U}^d, W = \{w_i\}_{i\in[n]} \subseteq \mathbb{U}^d$, the distribution of $\{\mathsf{ID}_q\}_{q\in Q}$ and $\{\mathsf{ID}_w\}_{w\in W}$ is statistically indistinguishable to that of uniform distribution, where $\mathsf{st} \leftarrow \mathsf{Setup}(Q, W), \mathsf{ID}_q \leftarrow \mathsf{IDGen}_\mathcal{S}(q, \mathsf{st}), \mathsf{ID}_w \leftarrow \mathsf{IDGen}_\mathcal{R}(w, \mathsf{st}), q \in Q, w \in W$.
Silence. The output of algorithm Setup is $\perp$. In other words, the algorithm $\mathsf{IDGen}_\mathcal{S}(q, \mathsf{st})$ and $\mathsf{IDGen}_\mathcal{R}(w, \mathsf{st})$ do not require a common input and work independently.

A fuzzy mapping scheme only needs to satisfy the correctness and distinctiveness. Consistency, randomness, and silence are considered optional, subject to the constraint that if the fuzzy mapping is not silent, it must satisfy randomness. This is because the ID generation function takes st as input, which itself depends on both parties' inputs, and the ID generation process may leak additional information if randomness is not satisfied.

Note that due to correctness requirements, the output of $\mathsf{IDGen}_\mathcal{S}$ and $\mathsf{IDGen}_\mathcal{R}$ must contain at least one element, so the expansion rate of fuzzy mapping is at least 1. When the expansion rate equals 1, the intersection test degenerates to an equality test and is thus optimal.

Most previous fuzzy PSIs [FNP04, GRS22, vBP24] implicitly used fuzzy mapping that satisfies the silence property, which enables participants to locally compute their ID sets. However, this property necessitates more complex encoding methods to ensure correctness without global pre-processing, consequently leading to high expansion rates in these schemes. There is no known fuzzy mapping

scheme that satisfies silence to achieve rate 1. Gao et al. [GQL+24] first used a fuzzy mapping that does not satisfy silence and constructed a fuzzy mapping with an optimal expansion rate of 1. However, Gao et al.'s construction only satisfies correctness, but not consistency. We find that there is a very simple way to construct $(1,1)-$fuzzy mapping with consistency if silence is not required. We give it as follows:

- $\mathsf{Setup}(Q, W)$:
  1. For $q_j \in Q$: pick random $id_{q_j} \leftarrow \{0,1\}^{\lambda+\log m+\log n}$.
  2. For $w_i \in W$: if $\exists q_j \in Q$, s.t., $\mathsf{dist}(w_i, q_j) \leq \delta$, then define $id_{w_i} := id_{q_j}$. Otherwise, pick random $id_{w_i} \leftarrow \{0,1\}^{\lambda+\log m+\log n}$
  3. Let $\mathsf{st} := \{(q_j, id_{q_j})\}_{j\in[m]} \cup \{(w_i, id_{w_i})\}_{i\in[n]}$. Then, pad st with dummy random elements to size $m+n$.
- $\mathsf{IDGen}_{\mathcal{S}}(q, \mathsf{st})$:
  1. Parse $\mathsf{st} := \{(e_i, id_{e_i})\}_{i\in[m+n]}$.
  2. Find e_i s.t., $q = e_i$, and output $\mathsf{ID}_q := \{id_{e_i}\}$.
- $\mathsf{IDGen}_{\mathcal{R}}(w, \mathsf{st})$:
  1. Parse $\mathsf{st} := \{(e_i, id_{e_i})\}_{i\in[m+n]}$.
  2. Find e_i s.t., $w = e_i$, and output $\mathsf{ID}_w := \{id_{e_i}\}$.

The correctness, randomness, and distinctiveness of the above construction are easy to verify. For consistency, if $\mathsf{dist}(q_j, w_i) > \delta$, both id_{q_j} and id_{w_i} are randomly selected from $\{0,1\}^{\lambda+\log m+\log n}$. The union bound guarantees the probability of $\exists q_j \in Q, w_i \in W$, s.t., $\mathsf{ID}_q = \mathsf{ID}_w$[4] is negligible $2^{-\lambda}$.

Although the above fuzzy mapping construction is simple, it does not meet the silence. To construct the fuzzy PSI protocol using the above fuzzy mapping, we need to design a two-party protocol to generate a set of IDs for both parties, which is challenging. We will discuss in more detail in Section 3.4.

3.2 s-Separated Set Assumption

In this section, we introduce s-separated set assumption. We note that $1 \leq s \leq d$, and the smaller s, the stronger the assumption. The s-separated set assumption is a general assumption that captures both the global disjoint assumption [GRS22] and the existing disjoint assumption [vBP24, GQL+24] used in recent FPSI works. Specifically, the global disjoint assumption [GRS22] and the existing disjoint assumption [vBP24, GQL+24] correspond to the cases of $s = 1$ and $s = d$ in s-separated set assumption, respectively.

The s-separated set assumption means that each element has disjoint projections (i.e., separated) from others on at least one dimension over every s dimensions[5]. van Baarsen and Pu [vBP24] proved that if the set is uniformly distributed, then it satisfies $d-$separated set assumption with overwhelming probability. In our construction, we use a smaller s, i.e., $s < d$, resulting in a stronger assumption. Nevertheless, we can still prove that if the set is uniformly distributed, then it satisfies s-separated set assumption with overwhelming probability when $s = \Omega(d^{1/c})$ for any constant c. The formal definition and proof is as follows.

Definition 3 (s-Separated Set). *A set $Q \subseteq \mathbb{U}^d$ of size m is an s-separated set if $\forall q_j \in Q, \forall k \in [d/s], \exists i_k^* \in [(k-1)s, ks]$ such that:*

$$\forall t \in [-\delta, \delta], q_{j,i_k^*} + t \notin \{q_{j',i_k^*+t'}\}_{j\in[m]\setminus\{j\},t'\in[-\delta,\delta]}$$

Theorem 1. *If all elements in set Q are uniformly distributed, i.e., $q_j \leftarrow \mathbb{U}^d, j \in [m]$, then $Q = \{q_j\}_{j\in[m]}$ is an s-separated set with probability $1 - \mathsf{negl}(d)$, where $s = \Omega(d^{1/c})$ for any constant c.*

Proof. Let $\log|\mathbb{U}| = u$. We define the event A_k as $\forall q_j \in Q, \exists i_k^* \in [(k-1)s, ks]$ s.t. $\forall t \in [-\delta, \delta], q_{j,i_k^*}+t \notin \{q_{j',i_k^*+t'}\}_{j\in[m]\setminus\{j\},t'\in[-\delta,\delta]}$. Then Q is an s-separated set if and only if $\bigwedge_{k\in[d/s]} A_k$ occurs.

Now we analyze the probability of $\bar{A}_k$. Fix $q_j \in Q$ and a dimension $i_k \in [(k-1)s, ks]$, the probability of q_j is not separate in dimension i_k (i.e., $\exists q_{j'} \in Q, \exists t \in [-\delta, \delta]$, s.t. $q_{j,i_k} + t \in \{q_{j',i_k+t'}\}_{t'\in[-\delta,\delta]}$) is

[4] Since the expansion rate is 1, $\mathsf{ID}_q \cap \mathsf{ID}_w \neq \emptyset$ means $\mathsf{ID}_q = \mathsf{ID}_w$.
[5] We assume s is divided by d for convenience.

smaller than $\frac{2\cdot(2\delta+1)\cdot(m-1)}{2^u}$. Therefore, the probability of q_j is not separate in each dimension is smaller than $(\frac{2\cdot(2\delta+1)\cdot(m-1)}{2^u})^s$. According to the union bound, $\forall q_j \in Q$,

$$\Pr[\bar{A_k}] \leq (\frac{2\cdot m^{1+\frac{1}{s}}(2\delta+1)\cdot}{2^u})^s$$

Note that the above probability is $\mathsf{negl}(s)$ when $\frac{\log m+\log\delta+2}{u}$ is a constant smaller than 1. As a result,

$$\begin{aligned}\Pr[Q \text{ is not } s\text{-separate set}] &= 1-\Pr[\bigwedge_{k\in[d/s]} A_k] = 1-\Pr[A_k]^{\frac{d}{s}} \\ &= 1-(1-\Pr[\bar{A_k}])^{\frac{d}{s}} \leq \frac{d}{s}\cdot\Pr[\bar{A_k}]\end{aligned}$$

When $s=\Omega(d^{1/c})$ for some constant c, we have $\frac{d}{s}\cdot\Pr[\bar{A_k}] = \frac{d}{d^{1/c}}\cdot\mathsf{negl}(d^{1/c}) = \mathsf{negl}(d)$.

3.3 Oblivious PRF Sum

We introduce a variant of OPRF, namely, OPRF-sum. Unlike OPRF, OPRF-sum requires the receiver to obtain the sum of queried PRF values. This necessitates that the PRF's range forms an additive group. The ideal functionality for OPRF-sum is shown in Figure 10.

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, a PRF F, query sizes n
Functionality:

- Wait for input $\{x_1,\ldots,x_n\}$ from the receiver $\mathcal{R}$.
- Sample a random PRF key k and give it to the sender $\mathcal{S}$.
- Give $\sum_{i\in[n]} F_k(x_i)$ to the receiver $\mathcal{R}$.

Fig. 10. OPRF-sum Functionality $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$

We can use OPRF with shared-output specified in Figure 5 to construct OPRF-sum. The formal description of OPRF-sum protocol is given in Figure 11.

Parameters:

- Two parties: sender $\mathcal{S}$ and receiver $\mathcal{R}$, set size n.
- Ideal $\mathcal{F}_{\mathsf{oprf\text{-}so}}$ primitive specified in Figure 5.

Protocol:

1. $\mathcal{S}$ and $\mathcal{R}$ invoke the OPRF with shared-output functionality $\mathcal{F}_{\mathsf{oprf\text{-}so}}$. The receiver $\mathcal{R}$ acts as the receiver in sum OPRF with input $\{x_i\}_{i\in[n]}$, and learns $\{b_i\}_{i\in[n]}$. The sender $\mathcal{S}$ acts as the sender in sum OPRF with no input, and receives a PRF key k and $\{a_i\}_{i\in[n]}$.
2. $\mathcal{S}$ computes and sends $c := \sum_{i\in[n]} a_i$ to $\mathcal{R}$.
3. $\mathcal{R}$ outputs $f := \sum_{i\in[n]} b_i + c$.

Fig. 11. OPRF-sum Protocol $\Pi_{\mathsf{oprf\text{-}sum}}$

Parameters: Sender $\mathcal{S}$, Receiver $\mathcal{R}$, a finite field $\mathbb{F}$, set size m, n, a $(\mathsf{r}_\mathcal{S}, \mathsf{r}_\mathcal{R})-$fuzzy mapping scheme $\mathsf{Fmap} = (\mathsf{Setup}, \mathsf{IDGen}_\mathcal{S}, \mathsf{IDGen}_\mathcal{R})$.
Functionality:

- Wait for input $Q = \{q_j\}_{j\in[m]} \subseteq \mathbb{U}^d$ from the sender $\mathcal{S}$.
- Wait for input $W = \{w_i\}_{i\in[n]} \subseteq \mathbb{U}^d$ from the receiver $\mathcal{R}$.
- Compute $\mathsf{st} \leftarrow \mathsf{Setup}(Q, W)$. Define $\mathsf{ID}_Q := \{\mathsf{IDGen}_\mathcal{S}(q_j, \mathsf{st})\}_{j\in[m]}$ and $\mathsf{ID}_W := \{\mathsf{IDGen}_\mathcal{R}(w_i, \mathsf{st})\}_{i\in[n]}$.
- Give ID_Q to the sender $\mathcal{S}$ and give ID_W to the receiver $\mathcal{R}$.

Fig. 12. Distributed ID Generation Functionality $\mathcal{F}_{\mathsf{didg}}$

3.4 Distributed ID Generation

The ideal functionality for Distributed ID Generation is given in Figure 12.

Note that if the fuzzy mapping scheme satisfies silence, the protocol for realizing $\mathcal{F}_{\mathsf{didg}}$ is trivial, i.e., the parties compute the ID_Q and ID_W locally. However, if the fuzzy mapping scheme does not satisfy silence, it remains challenging to design a distributed ID generation protocol since there is no party can run the Setup algorithm alone. We show our distributed ID generation protocol for L_∞ in Figure 13.

Parameters:

- Two parties: sender $\mathcal{S}$ and receiver $\mathcal{R}$, set size m, n, item universe $\mathbb{U}^d$, dimension d, distance threshold δ, partition parameter s, an OKVS scheme $(\mathsf{Encode}, \mathsf{Decode})$.
- Ideal $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$ primitive specified in Figure 10.

Protocol:

1. $\mathcal{S}$ picks random $\mathsf{Rand}^\mathcal{S}_{j,k} \leftarrow \mathbb{F}$ for $j \in [m], k \in [d/s]$ and computes $\mathsf{pid}^\mathcal{S}_j := \sum_{k\in[d/s]} \mathsf{Rand}^\mathcal{S}_{j,k}, j \in [m]$.
2. For $i \in [n]$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OPRF-sum functionality $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$. The receiver $\mathcal{R}$ acts as the receiver in OPRF-sum with input $\{k||\bar{w}_{i,k}\}_{k\in[d/s]}$, and learns $f_{\mathcal{R},i} = \sum_{k\in[d/s]} F_{sk_\mathcal{S}}(k||\bar{w}_{i,k})$, where $\bar{w}_{i,k} = (w_{i,(k-1)s+1}, \ldots, w_{i,k\cdot s})$. The sender $\mathcal{S}$ acts as the sender in OPRF-sum with a randomly selected PRF key $sk_\mathcal{S}$ as input, and has no output.
3. $\mathcal{S}$ computes $E_\mathcal{S} := \mathsf{Encode}(\{(k||(\bar{q}_{j,k}+t), \mathsf{Rand}^\mathcal{S}_{j,k} + F_{sk_\mathcal{S}}(k||(\bar{q}_{j,k}+t)))\}_{j\in[m],k\in[d/s],t\in[-\delta,\delta]^s})$, where $\bar{q}_{j,k} = (q_{j,(k-1)s+1}, \ldots, q_{j,k\cdot s})$. Then, $\mathcal{S}$ sends $E_\mathcal{S}$ to $\mathcal{R}$.
4. $\mathcal{R}$ computes $\sigma_{\mathcal{R},i} := \sum_{k\in[d/s]} \mathsf{Decode}(E_\mathcal{S}, k||\bar{w}_{i,k}) - f_{\mathcal{R},i}, i \in [n]$.
5. Symmetrically, $\mathcal{R}$ picks random $\mathsf{Rand}^\mathcal{R}_{i,k} \leftarrow \mathbb{F}$ for $i \in [n], k \in [d/s]$, computes $\mathsf{pid}^\mathcal{R}_i := \sum_{k\in[d/s]} \mathsf{Rand}^\mathcal{R}_{i,k}, i \in [n]$.
6. For $j \in [m]$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OPRF-sum functionality $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$. The sender $\mathcal{S}$ acts as the receiver in OPRF-sum with input $\{k||\bar{q}_{j,k}\}_{k\in[d/s]}$, and learns $f_{\mathcal{S},j} = \sum_{k\in[d/s]} F_{sk_\mathcal{R}}(k||\bar{q}_{j,k})$, where $\bar{q}_{j,k} = (q_{j,(k-1)s+1}, \ldots, q_{j,k\cdot s})$. The receiver $\mathcal{R}$ acts as the sender in OPRF-sum with a randomly selected PRF key $sk_\mathcal{R}$ as input, and has no output.
7. $\mathcal{R}$ computes $E_\mathcal{R} := \mathsf{Encode}(\{(k||(\bar{w}_{i,k}+t), \mathsf{Rand}^\mathcal{R}_{i,k} + F_{sk_\mathcal{R}}(k||(\bar{w}_{i,k}+t)))\}_{i\in[n],k\in[d/s],t\in[-\delta,\delta]^s})$, where $\bar{w}_{i,k} = (w_{i,(k-1)s+1}, \ldots, w_{i,k\cdot s})$. Then, $\mathcal{R}$ sends $E_\mathcal{R}$ to $\mathcal{S}$.
8. $\mathcal{S}$ computes $\sigma_{\mathcal{S},j} := \sum_{k\in[d/s]} \mathsf{Decode}(E_\mathcal{R}, k||\bar{q}_{j,k}) - f_{\mathcal{S},j}, j \in [m]$.
9. The sender $\mathcal{S}$ defines $id_{q_j} := \sigma_{\mathcal{S},j} + \mathsf{pid}^\mathcal{S}_j$ and outputs $\mathsf{ID}_{q_j} := \{id_{q_j}\}, j \in [m]$.
10. The receiver $\mathcal{R}$ defines $id_{w_i} := \sigma_{\mathcal{R},i} + \mathsf{pid}^\mathcal{R}_i$ and outputs $\mathsf{ID}_{w_i} := \{id_{w_i}\}, i \in [n]$.

Fig. 13. Distributed ID Generation Protocol Π_{didg} for L_∞

Correctness. If $\exists q_j \in Q, w_i \in W$, s.t. $\mathsf{dist}(q_j, w_i) \leq \delta$, we have $\forall k \in [d/s], k||\bar{q}_{j,k} = k||\bar{w}_{i,k}$ and

$$\begin{aligned} id_{q_j} &= \sigma_{\mathcal{S},j} + \mathsf{pid}_j^{\mathcal{S}} \\ &= \sum_{k\in[d/s]} \mathsf{Decode}(E_{\mathcal{R}}, k||\bar{q}_{j,k}) - f_{\mathcal{S},j} + \mathsf{pid}_j^{\mathcal{S}} \\ &= \sum_{k\in[d/s]} (\mathsf{Rand}_{i,k}^{\mathcal{R}} + F_{sk_{\mathcal{R}}}(k||(\bar{w}_{i,k}+t)) - f_{\mathcal{S},j} + \mathsf{pid}_j^{\mathcal{S}} \\ &= \mathsf{pid}_i^{\mathcal{R}} + \mathsf{pid}_j^{\mathcal{S}} \end{aligned}$$

Similarly, $id_{w_i} = \mathsf{pid}_i^{\mathcal{R}} + \mathsf{pid}_j^{\mathcal{S}} = id_{q_j}$. Conversely, if $\forall i \in [n], j \in [m], \mathsf{dist}(q_j, w_i) > \delta$, then $\exists k \in [d/s]$, s.t. $\mathsf{Decode}(E_{\mathcal{R}}, k||\bar{q}_{j,k}) \neq \mathsf{Rand}_{i,k}^{\mathcal{R}}$. The s-separate property of Q guarantees $\forall j \in [m], k \in [d/s], \mathsf{Decode}(E_{\mathcal{R}}, k||\bar{q}_{j,k})$ are distinct. This means there is at least a fresh random value $\mathsf{Rand}_{i,k}^{\mathcal{R}}$ is added to ID_{w_i}. By setting $l = \log|\mathbb{F}| = \lambda + \log m + \log n$, a union bound shows the probability of $\exists i \in [n], j \in [m], \mathsf{ID}_{q_j} = \mathsf{ID}_{w_i}$ is negligible $2^{-\lambda}$.

Next, we prove the randomness of generated fuzzy mapping. We prove this through a new ad-hoc property for OKVS which we call the *separate sum randomness*. The formal definition is as follows.

Definition 4 (Separate Sum Randomness). *Let* $(\mathsf{Encode}, \mathsf{Decode})$ *be an OKVS scheme. Considering the following interactive machine* $\mathcal{C}$ *who interacts with a "caller"* $\mathcal{A}$*:*

1. $\mathcal{C}$ *receives* $W \in \mathbb{U}^{d\times n}$ *from* $\mathcal{A}$.
2. $\mathcal{C}$ *selects random* $\mathsf{Rand}_{i,k} \leftarrow \mathcal{V}, i \in [n], k \in [d/s]$, *and computes* $E := \mathsf{Encode}(\{(k||\bar{w}_{i,k}+t, \mathsf{Rand}_{i,k})\}_{i\in[n],k\in[d/s],t\in[-\delta,\delta]^s})$
3. $\mathcal{C}$ *receives* $Q \in \mathbb{U}^{d\times m}$ *from* $\mathcal{A}$.
4. $\mathcal{C}$ *computes* $\sigma_j := \sum_{k\in[d/s]} \mathsf{Decode}(E, k||\bar{q}_{j,k})$ *and sends* $\{\sigma_j\}_{j\in[m]}$ *to* $\mathcal{A}$.
5. $\mathcal{C}$ *halts.*

We say an OKVS scheme satisfies (s, m, n)*-separate sum randomness if for every* s*-separate sets* W, Q *input by a PPT* $\mathcal{A}$*, the output of* $\mathcal{C}$ *is statistically indistinguishable from the uniformly random values from* $\mathcal{V}^m$.

Theorem 2. *A binary OKVS* $(\mathsf{Encode}, \mathsf{Decode})$ *satisfying* m*-randomness also satisfies* (s, m, n)*-separate sum randomness.*

We give the formal proof of Theorem 2 in Appendix A.2.

Theorem 3. *The protocol in Figure 13 securely computes* $\mathcal{F}_{\mathsf{didg}}$ *against semi-honest adversaries in the* $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$*-hybrid model.*

Due to space limitations, the security proof is deferred to Appendix A.3.

4 Fuzzy PSI

4.1 Fuzzy PSI for L_∞ Distance

In this section, we give the construction of FPSI and its variants for L_∞ distance. The formal description is given in Figure 14.

We discuss the correctness of fuzzy PSI in Figure 14. The correctness for its variants is similar.

Correctness. For $j \in [m]$, if $\exists w_i \in W$, s.t. $\mathsf{dist}(q_j, w_i) \leq \delta$, then the correctness of fuzzy mapping ensures $id_{q_j} = id_{w_i}$. Then, by the correctness of mq-RPMT, we have $b_j = 1$. The correctness of OT ensures the receiver learns q_j. If $\forall i \in [n], j \in [m], \mathsf{dist}(q_j, w_i) > \delta$, the correctness of fuzzy mapping ensures $id_{q_j} \neq id_{w_i}$ for any $w_i \in W$ with overwhelming probability. Then, by the correctness of mq-RPMT, we have $b_j = 0$. The correctness of OT ensures the receiver learns nothing about q_j.

We prove the security of the protocol in Figure 14 by the case of FPSI. The security proof of its variants is similar.

Theorem 4. *The protocol in Figure 14 securely computes* $\mathcal{F}_{\mathsf{fpsi}}$ *for* L_∞ *against semi-honest adversaries in the* $(\mathcal{F}_{\mathsf{didg}}, \mathcal{F}_{\mathsf{mq\text{-}rpmt}}, \mathcal{F}_{\mathsf{ot}})$*-hybrid model.*

Due to space limitations, the security proof is deferred to Appendix A.4.

Parameters:

- Two parties: sender $\mathcal{S}$ and receiver $\mathcal{R}$, set size m, n.
- Ideal $\mathcal{F}_{\mathsf{didg}}$, $\mathcal{F}_{\mathsf{mq\text{-}pmt}}$, $\mathcal{F}_{\mathsf{mq\text{-}rpmt}}$ and $\mathcal{F}_{\mathsf{ot}}$ primitives specified in Figure 12, Figure 6 and Figure 4, respectively.

Protocol:

1. $\mathcal{S}$ and $\mathcal{R}$ invoke the distributed ID generation functionality $\mathcal{F}_{\mathsf{didg}}$ with input Q and W, respectively. As a result, the sender $\mathcal{S}$ learns $\mathsf{ID}_{q_j} = \{id_{q_j}\}$ for $j \in [m]$. The receiver $\mathcal{R}$ learns $\mathsf{ID}_{w_i} = \{id_{w_i}\}$ for $i \in [n]$.
2. For FPSI-SP: $\mathcal{S}$ and $\mathcal{R}$ invoke the mq-PMT functionality $\mathcal{F}_{\mathsf{mq\text{-}pmt}}$ with input $\mathsf{ID}_Q = \{id_{q_j}\}_{j\in[m]}$ and $\mathsf{ID}_W = \{id_{w_i}\}_{i\in[n]}$, respectively. As a result, the sender $\mathcal{S}$ learns nothing and the receiver $\mathcal{R}$ learns $b \in \{0,1\}^n$, where $b_i = 1$ if and only if $id_{w_i} \in \mathsf{ID}_Q$. Then, the receiver $\mathcal{R}$ outputs $I := \{w_i | i \in [n], b_i = 1\}$.
3. For others: $\mathcal{S}$ and $\mathcal{R}$ invoke the mq-RPMT functionality $\mathcal{F}_{\mathsf{mq\text{-}rpmt}}$ with input $\mathsf{ID}_Q = \{id_{q_j}\}_{j\in[m]}$ and $\mathsf{ID}_W = \{id_{w_i}\}_{i\in[n]}$, respectively. As a result, the sender $\mathcal{S}$ learns nothing and the receiver $\mathcal{R}$ learns $b \in \{0,1\}^m$, where $b_j = 1$ if and only if $id_{q_j} \in \mathsf{ID}_W$.
   - For FPSI: The receiver $\mathcal{R}$ initialize set $I := \{\}$. Then, for $i \in [m]$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OT functionality $\mathcal{F}_{\mathsf{ot}}$. $\mathcal{S}$ acts as sender with input $(\perp, q_j)$ and receives nothing. $\mathcal{R}$ acts as receiver with input b_j and receives z_j. $\mathcal{R}$ sets $I := I \cup \{z_j\}$ if $z_j \neq \perp$. Finally, $\mathcal{R}$ outputs I.
   - For LFPSI: The receiver $\mathcal{R}$ initialize set $I := \{\}$. Then, for $i \in [m]$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OT functionality $\mathcal{F}_{\mathsf{ot}}$. $\mathcal{S}$ acts as sender with input $(\perp, \mathsf{label}_j)$ and receives nothing. $\mathcal{R}$ acts as receiver with input b_j and receives z_j. $\mathcal{R}$ sets $I := I \cup \{z_j\}$ if $z_j \neq \perp$. Finally, $\mathcal{R}$ outputs I.
   - For FPSI-CA: The receiver $\mathcal{R}$ outputs the Hamming weight of b.

Fig. 14. Fuzzy PSI and its variants Protocols $\Pi_{\mathsf{fpsi}}, \Pi_{\mathsf{fpsi\text{-}sp}}, \Pi_{\mathsf{fpsi\text{-}ca}}, \Pi_{\mathsf{lfpsi}}$ for L_∞

4.2 Fuzzy PSI for $L_{p\in[1,\infty)}$ Distance

In this section, we give the construction of fuzzy PSI and its variants for $L_{p\in[1,\infty)}$ distance. The formal description is given in Figure 15.

We discuss the correctness of fuzzy PSI in Figure 15. The correctness for its variants is similar.

Correctness. For $u \in [m']$, if $\exists w_i \in W$, s.t. $\mathsf{dist}(\mathsf{CH}_Q[\pi(u)], w_i) \leq \delta$, then the correctness of fuzzy mapping ensures $\mathsf{CH}_{\mathsf{ID}}[\pi(u)] = id_{w_i}$, thus $\forall k \in [d], \mathsf{CH}_{\mathsf{ID}}[\pi(u)]||k||\alpha_{\pi(u)} = id_{w_i}||k||\alpha_{\pi(u)}$ and $|\mathsf{CH}_Q[\pi(u)]_k - w_{i,k}| \leq \delta$, we have

$$
\begin{aligned}
&\mathsf{Sh}_u^{\mathcal{S}} + \mathsf{Sh}_u^{\mathcal{R}} = \mathsf{Rand}_{\pi(u)}^{\mathcal{S}} - r_u^{\mathcal{S}} - r_u^{\mathcal{R}} \\
=&\sum_{k\in[d]} \mathsf{Decode}(E_{\mathcal{R}}, \mathsf{CH}_{\mathsf{ID}}[\pi(u)]||k||\mathsf{CH}_Q[\pi(u)]_k||\alpha_{\pi(u)}) - f_{\mathcal{S},\pi(u)} - \mathsf{Rand}_{\pi(u)}^{\mathcal{R}} \\
=&\sum_{k\in[d]} (|\mathsf{CH}_Q[\pi(u)]_k - w_{i,k}|^p + r_{\pi(u),k} + F_{sk_{\mathcal{R}}}(id_{w_i}||k||\mathsf{CH}_Q[\pi(u)]_k||\alpha_{\pi(u)})) - f_{\mathcal{S},\pi(u)} - \mathsf{Rand}_{\pi(u)}^{\mathcal{R}} \\
=&\mathsf{dist}(\mathsf{CH}_Q[\pi(u)], w_i)^p
\end{aligned}
$$

By the correctness of $\mathcal{F}_{\mathsf{fmat}}$, $b_u = 1$ if and only if $\mathsf{Sh}_u^{\mathcal{S}} + \mathsf{Sh}_u^{\mathcal{R}} = \mathsf{dist}(\mathsf{CH}_Q[\pi(u)], w_i)^p \in [0, \delta^p]$. The correctness of OT ensures the receiver learns $\mathsf{CH}_Q[\pi(u)]$ if $b_u = 1$, i.e., $0 < \mathsf{dist}(\mathsf{CH}_Q[\pi(u)], w_i) < \delta$. If $\forall i \in [n], u \in [m'], \mathsf{dist}(\mathsf{CH}_Q[\pi(u)], w_i) > \delta$, the correctness of fuzzy mapping ensures $\mathsf{CH}_{\mathsf{ID}}[\pi(u)] \neq id_{w_i}$ for any $w_i \in W$ with overwhelming probability. The pseudorandomness of $f_{\mathcal{S},\pi(u)}$ guarantees $\mathsf{Rand}_{\pi(u)}^{\mathcal{S}}$ is also pseudorandom. We have $\mathsf{Sh}_u^{\mathcal{S}} + \mathsf{Sh}_u^{\mathcal{R}}$ is a pseudorandom value. By setting $l = \log|\mathbb{F}| = \lambda + \log m + \log n + p\log\delta$, a union bound shows the probability of $\exists i \in [n], u \in [m'], \mathsf{Sh}_u^{\mathcal{S}} + \mathsf{Sh}_u^{\mathcal{R}} \in [0, \delta^p]$ is negligible $2^{-\lambda}$.

We prove the security of the protocol in Figure 15 by the case of FPSI. The security proof of its variants is similar.

Parameters:

- Two parties: sender $\mathcal{S}$ and receiver $\mathcal{R}$, set size m, n.
- An OKVS scheme $(\mathsf{Encode}, \mathsf{Decode})$.
- Ideal $\mathcal{F}_{\mathsf{didg}}$, $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$, $\mathcal{F}_{\mathsf{ps}}$ $\mathcal{F}_{\mathsf{fmat}}$ and $\mathcal{F}_{\mathsf{ot}}$ primitives specified in Figure 12, Figure 10, Figure 8, Figure 7 and Figure 4, respectively.

Protocol:

1. $\mathcal{S}$ and $\mathcal{R}$ invoke the distributed ID generation functionality $\mathcal{F}_{\mathsf{didg}}$ with input Q and W, respectively. As a result, the sender $\mathcal{S}$ learns $\mathsf{ID}_{q_j} = \{id_{q_j}\}$ for $j \in [m]$. The receiver $\mathcal{R}$ learns $\mathsf{ID}_{w_i} = \{id_{w_i}\}$ for $i \in [n]$.
2. The sender $\mathcal{S}$ inserts set $\mathsf{ID}_Q = \{id_{q_j}\}_{j\in[m]}$ into the Cuckoo hash table $\mathsf{CH}_{\mathsf{ID}} := \mathsf{CuckooHash}_H(\mathsf{ID}_Q)$. Let $m' = O(m)$ denote the length of the Cuckoo hash table and $\mathsf{CH}_{\mathsf{ID}}[u]$ the item in u-th bin, $u \in [m']$. The sender $\mathcal{S}$ also defines the cuckoo table for items CH_Q, where $\mathsf{CH}_Q[u] = q_{j'}$ satisfying $id_{q_{j'}} = \mathsf{CH}_{\mathsf{ID}}[u], u \in [m']$. Let $\mathsf{CH}_Q[u]_k := q_{j',k}$ for $k \in [d]$. Let $\sigma : [m] \to [m']$ denote the injective function such that $\sigma(1), \ldots, \sigma(m)$ are the non-dummy item bins of $\mathsf{CH}_{\mathsf{ID}}$ and CH_Q.
3. For $j \in [m]$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OPRF-sum functionality $\mathcal{F}_{\mathsf{oprf\text{-}sum}}$. The sender $\mathcal{S}$ acts as the receiver in OPRF-sum with input $\{\mathsf{CH}_{\mathsf{ID}}[\sigma(j)]||k||\mathsf{CH}_Q[\sigma(j)]_k||\alpha_{\sigma(j)}\}_{k\in[d]}$, and learns $f_{\mathcal{S},\sigma(j)} = \sum_{k\in[d]} F_{sk_{\mathcal{R}}}(\mathsf{CH}_{\mathsf{ID}}[\sigma(j)]||k||\mathsf{CH}_Q[\sigma(j)]_k||\alpha_{\sigma(j)})$, where $\alpha_{\sigma(j)} \in [3]$ is the index of hash function used to insert $\mathsf{CH}_{\mathsf{ID}}[\sigma(j)]$, i.e., $h_{\alpha_{\sigma(j)}}(\mathsf{CH}_{\mathsf{ID}}[\sigma(j)]) = \sigma(j)$. The receiver $\mathcal{R}$ acts as the sender in OPRF-sum with a randomly selected PRF key $sk_{\mathcal{R}}$ as input, and has no output.
4. $\mathcal{R}$ picks random $r_{u,k} \leftarrow \mathbb{F}$ for $u \in [m'], k \in [d]$, computes $\mathsf{Rand}^{\mathcal{R}}_u := \sum_{k\in[d]} r_{u,k}, u \in [m']$.
5. $\mathcal{R}$ computes $E_{\mathcal{R}} := \mathsf{Encode}(\{(id_{w_i}||k||(w_{i,k}+t)||\alpha, |t|^p + r_{h_\alpha(id_{w_i}),k} + F_{sk_{\mathcal{R}}}(id_{w_i}||k||(w_{i,k}+t)||\alpha)\}_{i\in[n],k\in[d],t\in[-\delta,\delta],\alpha\in[3]})$. Then, $\mathcal{R}$ sends $E_{\mathcal{R}}$ to $\mathcal{S}$.
6. For $j \in [m]$, $\mathcal{S}$ computes $\mathsf{Rand}^{\mathcal{S}}_{\sigma(j)} := \sum_{k\in[d]} \mathsf{Decode}(E_{\mathcal{R}}, \mathsf{CH}_{\mathsf{ID}}[\sigma(j)]||k||\mathsf{CH}_Q[\sigma(j)]_k||\alpha_{\sigma(j)}) - f_{\mathcal{S},\sigma(j)}$. $\mathcal{S}$ also picks random $\mathsf{Rand}^{\mathcal{S}}_u \leftarrow \mathbb{F}$ for $u \in [m'] \setminus \{\sigma(j)\}_{j\in[m]}$.
7. The sender $\mathcal{S}$ selects a random permutation π over $[m']$. Then $\mathcal{S}$ and $\mathcal{R}$ invoke the Permute + Share functionality $\mathcal{F}_{\mathsf{ps}}$. The receiver $\mathcal{R}$ acts as the sender in Permute + Share with input $\{\mathsf{Rand}^{\mathcal{R}}_u\}_{u\in[m']}$, and learns $\{r^{\mathcal{R}}_u\}_{u\in[m']}$. The sender $\mathcal{S}$ acts as the receiver in Permute + Share with input π and receives $\{r^{\mathcal{S}}_u\}_{u\in[m']}$, where $r^{\mathcal{S}}_u + r^{\mathcal{R}}_u = \mathsf{Rand}^{\mathcal{R}}_{\pi(u)}, u \in [m']$. $\mathcal{S}$ defines $\mathsf{Sh}^{\mathcal{S}}_u := \mathsf{Rand}^{\mathcal{S}}_{\pi(u)} - r^{\mathcal{S}}_u$ and $\mathcal{R}$ defines $\mathsf{Sh}^{\mathcal{R}}_u := -r^{\mathcal{R}}_u$.
8. For $u \in [m']$, $\mathcal{S}$ and $\mathcal{R}$ invoke the fuzzy matching for interval functionality $\mathcal{F}_{\mathsf{fmat}}$ with threshold of $\frac{1}{2}\delta^p$. The sender $\mathcal{S}$ acts as the sender in $\mathcal{F}_{\mathsf{fmat}}$ with input $-\mathsf{Sh}^{\mathcal{S}}_u + \frac{1}{2}\delta^p$. The receiver $\mathcal{R}$ acts as the receiver in $\mathcal{F}_{\mathsf{fmat}}$ with input $\mathsf{Sh}^{\mathcal{R}}_u$, and learns b_u, where $b_u = 1$ if and only if $\mathsf{Sh}^{\mathcal{R}}_u \in [-\mathsf{Sh}^{\mathcal{S}}_u, -\mathsf{Sh}^{\mathcal{S}}_u + \delta^p]$.
9. - For FPSI: The receiver $\mathcal{R}$ initialize set $I := \{\}$. Then, for $u \in [m']$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OT functionality $\mathcal{F}_{\mathsf{ot}}$. $\mathcal{S}$ acts as sender with input $(\perp, \mathsf{CH}_Q[\pi(u)])$ and receives nothing. $\mathcal{R}$ acts as receiver with input b_u and receives z_u. $\mathcal{R}$ sets $I := I \cup \{z_u\}$ if $z_u \neq \perp$. Finally, $\mathcal{R}$ outputs I.
   - For LFPSI: The receiver $\mathcal{R}$ initialize set $I := \{\}$. Then, for $u \in [m']$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OT functionality $\mathcal{F}_{\mathsf{ot}}$. $\mathcal{S}$ acts as sender with input $(\perp, \mathsf{label}_{\pi(u)})$ and receives nothing. $\mathcal{R}$ acts as receiver with input b_u and receives z_u. $\mathcal{R}$ sets $I := I \cup \{z_u\}$ if $z_u \neq \perp$. Finally, $\mathcal{R}$ outputs I.
   - For FPSI-CA: The receiver $\mathcal{R}$ outputs the Hamming weight of b.
   - For FPSI-SP: The receiver $\mathcal{R}$ initialize set $I := \{\}$. Then, for $u \in [m']$, $\mathcal{S}$ and $\mathcal{R}$ invoke the OT functionality $\mathcal{F}_{\mathsf{ot}}$. $\mathcal{S}$ acts as sender with input $(\perp, \mathsf{CH}_{\mathsf{ID}}[\pi(u)])$ and receives nothing. $\mathcal{R}$ acts as receiver with input b_u and receives id'_u. $\mathcal{R}$ sets $I := I \cup \{w_i\}$ if $\exists id_{w_i} = id'_u$. Finally, $\mathcal{R}$ outputs I.

Fig. 15. Fuzzy PSI and its variants Protocols $\Pi_{\mathsf{fpsi}}, \Pi_{\mathsf{fpsi\text{-}sp}}, \Pi_{\mathsf{fpsi\text{-}ca}}, \Pi_{\mathsf{lfpsi}}$ for $L_{p\in[1,\infty)}$

Theorem 5. *The protocol in Figure 15 securely computes $\mathcal{F}_{\mathsf{fpsi}}$ for $L_{p\in[1,\infty)}$ against semi-honest adversaries in the $(\mathcal{F}_{\mathsf{didg}}, \mathcal{F}_{\mathsf{oprf\text{-}sum}}, \mathcal{F}_{\mathsf{ps}}, \mathcal{F}_{\mathsf{fmat}}, \mathcal{F}_{\mathsf{ot}})$-hybrid model.*

Due to space limitations, the security proof is deferred to Appendix A.5.

5 Implementation and Performance

We experimentally evaluate our FPSI protocols and compare them with the state-of-the-art counterpart [GQL+24]. We have implemented the FPSI and its variant functionalities that we defined

in Figure 9 for L_∞ and $L_{p\in[1,2]}$ distance. Since [GQL+24] only implements FPSI, we compare the performance of the FPSI protocol. Two versions of our protocol are implemented, corresponding to the s-separated set assumption with $s = 1$ and $s = 2$, respectively. While the $s = 1$ variant delivers superior performance, it also relies on a stronger assumptions, i.e., global disjoint assumption [GRS22].

(d, δ)	Protocol	Set size $m = n$							
		2^4		2^8		2^{12}		2^{16}	
		Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time
$(2, 5)$	[GQL+24]	0.253	0.097	4.038	1.047	64.542	16.597	1032.604	287.696
	Ours ($s = 1$)	0.044	0.028	0.436	0.044	7.068	0.376	113.328	5.681
	Ours ($s = 2$)	0.102	0.035	1.531	0.054	23.370	0.535	375.045	8.173
$(2, 8)$	[GQL+24]	0.383	0.109	6.126	1.301	97.954	21.105	1567.204	366.139
	Ours ($s = 1$)	0.052	0.043	0.899	0.059	9.201	0.401	147.423	6.135
	Ours ($s = 2$)	0.206	0.046	3.350	0.092	53.329	0.922	855.434	14.173
$(2, 10)$	[GQL+24]	0.470	0.128	7.518	1.518	120.229	24.194	1923.604	417.467
	Ours ($s = 1$)	0.057	0.051	0.899	0.066	10.624	0.426	170.371	6.345
	Ours ($s = 2$)	0.300	0.057	4.999	0.109	80.336	1.251	1289.098	19.662
$(6, 5)$	[GQL+24]	0.731	0.188	11.703	2.588	187.179	43.190	2994.804	729.562
	Ours ($s = 1$)	0.094	0.030	1.296	0.087	20.375	0.723	326.504	11.222
	Ours ($s = 2$)	0.264	0.041	4.345	0.101	69.542	1.189	1115.597	18.779
$(6, 8)$	[GQL+24]	1.123	0.262	17.968	3.473	287.417	57.196	4598.604	958.916
	Ours ($s = 1$)	0.116	0.048	1.952	0.109	26.775	0.812	429.433	12.540
	Ours ($s = 2$)	0.721	0.056	9.931	0.176	159.217	2.354	2554.782	37.638
$(6, 10)$	[GQL+24]	1.384	0.265	22.144	3.940	354.242	66.219	5667.804	1115.373
	Ours ($s = 1$)	0.132	0.056	1.952	0.090	30.975	0.857	496.966	13.294
	Ours ($s = 2$)	0.942	0.061	15.017	0.236	240.755	3.326	3862.912	53.807
$(10, 5)$	[GQL+24]	1.211	0.276	19.368	4.082	309.817	69.142	4957.005	1170.141
	Ours ($s = 1$)	0.143	0.035	2.295	0.107	33.677	1.070	539.964	16.734
	Ours ($s = 2$)	0.733	0.045	7.211	0.139	115.544	1.843	1855.231	29.577
$(10, 8)$	[GQL+24]	1.863	0.356	29.809	5.515	476.880	92.384	7630.005	1557.095
	Ours ($s = 1$)	0.181	0.051	2.783	0.126	44.393	1.213	711.782	19.008
	Ours ($s = 2$)	1.030	0.063	16.543	0.283	265.528	3.771	4260.551	61.399
$(10, 10)$	[GQL+24]	2.299	0.431	36.770	6.396	588.255	107.385	9408.000	-
	Ours ($s = 1$)	0.206	0.060	3.210	0.117	51.475	1.291	825.593	20.295
	Ours ($s = 2$)	1.578	0.084	24.970	0.370	400.751	5.361	6430.024	88.752

Table 2. Communication cost (in MB) and running time (in seconds) comparing our protocols to [GQL+24] for L_∞ distance. Cells with - denotes trials that ran out of memory. The best result is marked in blue, and the second best result is marked in cyan.

5.1 Experimental Setup

We conduct our experiments on Ubuntu 22.04 with a single Intel i7-13700 2.10 GHz CPU (16 physical cores) and 64 GB RAM. All experiments are executed with a single thread. We set the computational security parameter $\kappa = 128$ and the statistical security parameter $\lambda = 40$. We simulate the network connection using the Linux command `tc`. We benchmark the full protocol runtime in a low-latency LAN environment (10Gbps bandwidth with 0.02ms RTT latency).

5.2 Implementation Details

Our protocols are written in C++, and we use the following libraries in our implementation. Our complete implementation will be freely available on GitHub.

– OPRF with shared-output. We use OPRF with shared-output in [APRR24] and the underlining PRF is their $(\mathbb{F}_3, \mathbb{F}_2)$-wPRF construction. However, as their implementation is currently not pub-

licly available, we use the data from their paper (see Figure 7 in [APRR24]) as an estimate [6], i.e., 7.5 μs and 1151 bits per OPRF-so evaluation.
- OKVS. We use the optimized OKVS in [RR22] as our OKVS instantiation, and re-use the implementation of OKVS by the authors of in [RR22][7].
- mq-RPMT. We implement the mq-RPMT based on [CZZ+24], drawing on the authors' implementation[8].
- Permute + Share. We use the implementation of [JSZ+22][9] to implement Permute + Share functionality.
- Fuzzy matching for interval. We use the implementation of previous FPSI work [GQL+24][10] to implement fuzzy matching for interval functionality [CFR23].
- OT. We use SoftSpokenOT [Roy22] implemented in libOTe, and set field bits to 5 to balance computation and communication.
- For computing hash functions and PRG calls, we employ the cryptoTools[11] library, while for network communication, we utilize Coproto[12].

5.3 Performance Evaluation

We compare our protocols with the state-of-the-art fuzzy PSI protocol [GQL+24]. Specifically, we consider the set size $m = n \in \{2^4, 2^8, 2^{12}, 2^{16}\}$, dimension $d \in \{2, 6, 10\}$, and distance threshold $\delta \in \{5, 8, 10\}$. Detailed comparisons of protocol performance for L_∞ distance are shown in Table 2. We also compared our FPSI protocol for L_1 and L_2 distances with [GQL+24]. Due to space limitations, we have included these comparisons in Appendix B.

Comparison. As shown in Table 2, both our protocols with $s = 1$ and $s = 2$ perform better than Gao's protocol [GQL+24] under all the parameters we tested. Specifically, our $s = 1$ protocol performs the best among all protocols, which is consistent with our theoretical analysis in Table 1. The improvement factor of both of our protocols increases with the increase of set size and dimension, for example, when $n = 2^{16}, d = 10, \delta = 8$, our protocol with $s = 1$ requires 19.008 seconds, while [GQL+24] requires 1557.095 seconds, achieving a 81.9× improvement, our protocol with $s = 2$ requires 61.394 seconds, also achieving a 25.4× improvement. Our protocols also achieves remarkable communication reductions: for $n = 2^{16}, d = 10, \delta = 5$, the $s = 1$ variant requires only 539.965 MB, representing a 9.2× improvement over Gao et al.'s [GQL+24] 4957.005 MB, while the $s = 2$ variant uses 1855.232 MB, still achieving a 2.7× reduction.

As δ increases, the improvement factor of our $s = 1$ variant remains almost unchanged, while the improvement factor of our $s = 2$ variant decreases. This is because the asymptotic cost of our protocol scales linearly with δ^s, while Gao's protocol [GQL+24] scales linearly with δ.

6 Conclusion

In this work, we propose new FPSI protocols for both L_∞ and $L_{p\in[1,\infty)}$ distance. Our construction primarily relies on OT and symmetric-key cryptographic primitives, achieving significant efficiency gains — up to 80× faster runtime and 5× lower communication overhead compared to prior schemes. While our protocol requires a slightly stronger assumption than previous work, we rigorously prove that this assumption holds with overwhelming probability when inputs are randomly sampled. A promising direction for future research is to relax this assumption while retaining high efficiency. Additionally, extending the protocol to achieve malicious security is another promising direction.

[6] Note that the running time in [APRR24] was measured on a Core i7 consumer grade laptop with 16GB RAM, while our testing environment was a desktop with Intel i7-13700 2.10 GHz CPU (16 physical cores) and 64 GB RAM. We believe that the actual running time in our machine will only be faster.
[7] https://github.com/Visa-Research/volepsi.git
[8] https://github.com/yuchen1024/Kunlun.git
[9] https://github.com/dujiajun/PSU.git
[10] https://github.com/ql70ql70/Fuzzy-Private-Set-Intersection-from-Fuzzy-Mapping.git
[11] https://github.com/ladnir/cryptoTools.git
[12] https://github.com/Visa-Research/coproto.git

References

[APRR24] Navid Alamati, Guru-Vamsi Policharla, Srinivasan Raghuraman, and Peter Rindal. Improved alternating-moduli prfs and post-quantum signatures. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VIII*, volume 14927 of *Lecture Notes in Computer Science*, pages 274–308. Springer, 2024.

[BC23] Dung Bui and Geoffroy Couteau. Improved private set intersection for sets with small entries. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 190–220. Springer, 2023.

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, pages 489–518, 2019.

[BN25] Erik-Oliver Blass and Guevara Noubir. Assumption-free fuzzy PSI via predicate encryption. Cryptology ePrint Archive, Paper 2025/217, 2025.

[BPSY23] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Near-optimal oblivious key-value stores for efficient psi, PSU and volume-hiding multi-maps. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 301–318. USENIX Association, 2023.

[CFR23] Anrin Chakraborti, Giulia Fanti, and Michael K. Reiter. Distance-aware private set intersection. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 319–336. USENIX Association, 2023.

[CGP20] Melissa Chase, Esha Ghosh, and Oxana Poburinnaya. Secret-shared shuffle. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 342–372. Springer, 2020.

[CH08] Lukasz Chmielewski and Jaap-Henk Hoepman. Fuzzy private matching (extended abstract). In *Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008, March 4-7, 2008, Technical University of Catalonia, Barcelona , Spain*, pages 327–334. IEEE Computer Society, 2008.

[CLO24] Wutichai Chongchitmate, Steve Lu, and Rafail Ostrovsky. Approximate PSI with near-linear communication. *IACR Cryptol. ePrint Arch.*, page 682, 2024.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In *CRYPTO 2020*, 2020.

[CZZ+24] Yu Chen, Min Zhang, Cong Zhang, Minglang Dong, and Weiran Liu. Private set operations from multi-query reverse private membership test. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part III*, volume 14603 of *Lecture Notes in Computer Science*, pages 387–416. Springer, 2024.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004*, 2004.

[GGM24] Gayathri Garimella, Benjamin Goff, and Peihan Miao. Computation efficient structure-aware PSI from incremental function secret sharing. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology*

Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VIII, volume 14927 of *Lecture Notes in Computer Science*, pages 309–345. Springer, 2024.

[GMR+21] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In *PKC 2021*, 2021.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *CRYPTO 2021*, 2021.

[GQL+24] Ying Gao, Lin Qi, Xiang Liu, Yuanchao Luo, and Longxin Wang. Efficient fuzzy private set intersection from fuzzy mapping. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VI*, volume 15489 of *Lecture Notes in Computer Science*, pages 36–68. Springer, 2024.

[GRS22] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 323–352. Springer, 2022.

[GRS23] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Malicious secure, structure-aware private set intersection. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 577–610. Springer, 2023.

[IW06] Piotr Indyk and David P. Woodruff. Polylogarithmic private approximations and efficient matching. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 245–264. Springer, 2006.

[JSZ+22] Yanxue Jia, Shi-Feng Sun, Hong-Sheng Zhou, Jiajun Du, and Dawu Gu. Shuffle-based private set union: Faster and more secure. In *USENIX Security 22*, 2022.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *CCS 2016*, 2016.

[KM21] Anunay Kulshrestha and Jonathan R. Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 893–910. USENIX Association, 2021.

[KMP+17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, pages 1257–1272. ACM, 2017.

[Lin17] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017.

[MS13] Payman Mohassel and Seyed Saeed Sadeghian. How to hide circuits in MPC an efficient framework for private function evaluation. In *EUROCRYPT 2013*, 2013.

[PIRC21] Bijeeta Pal, Mazharul Islam, Thomas Ristenpart, and Rahul Chatterjee. Might I get pwned: A second generation password breach alerting service. *CoRR*, abs/2109.14490, 2021.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse OT extension. In *CRYPTO 2019*, 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from paxos: Fast, malicious private set intersection. In *EUROCRYPT 2020*, 2020.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *USENIX Security*, 2014.

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2005:187, 2005.

[Roy22] Lawrence Roy. Softspokenot: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 657–687. Springer, 2022.

[RR22] Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 2505–2517. ACM, 2022.

[RRX24] David Richardson, Mike Rosulek, and Jiayu Xu. Fuzzy PSI via oblivious protocol routing. Cryptology ePrint Archive, Paper 2024/1642, 2024.

[RS21] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In *EUROCRYPT 2021*, 2021.

[RT21] Mike Rosulek and Ni Trieu. Compact and malicious private set intersection for small sets. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1166–1181. ACM, 2021.

[UCK+21] Erkam Uzun, Simon P. Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy labeled private set intersection with applications to private real-time biometric search. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 911–928. USENIX Association, 2021.

[vBP24] Aron van Baarsen and Sihang Pu. Fuzzy private set intersection with large hyperballs. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 340–369. Springer, 2024.

[vBS24] Aron van Baarsen and Marc Stevens. Amortizing circuit-psi in the multiple sender/receiver setting. *IACR Commun. Cryptol.*, 1(3):2, 2024.

[WHZ+15] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 492–503. ACM, 2015.

[YSPW09] Qingsong Ye, Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. Efficient fuzzy matching and intersection on private datasets. In Dong Hoon Lee and Seokhie Hong, editors, *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, volume 5984 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2009.

[ZCL+23] Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, and Dongdai Lin. Linear private set union from Multi-Query reverse private membership test. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 337–354, Anaheim, CA, August 2023. USENIX Association.

[ZLDL23] Cong Zhang, Weiran Liu, Bolin Ding, and Dongdai Lin. Efficient private multiset ID protocols. In Ding Wang, Moti Yung, Zheli Liu, and Xiaofeng Chen, editors, *Information and Communications Security - 25th International Conference, ICICS 2023, Tianjin, China, November 18-20, 2023, Proceedings*, volume 14252 of *Lecture Notes in Computer Science*, pages 351–369. Springer, 2023.

A Missing Proofs

A.1 Proof of t-Randomness

To prove the t-randomness of OKVS, we additionally propose a useful property called t-independence for linear OKVS. The definition is as follows.

t-Independence. For a linear OKVS scheme, let row be the mapping used in OKVS. Then for any distinct $\{x_i\}_{i\in[t]} \subseteq \mathcal{K}^t$, the vectors $\{\mathsf{row}(x_i)\}_{i\in[t]}$ are linear independent with overwhelming probability.

Now, we prove the linear OKVS satisfying $(n+t)$-independence also satisfies t-randomness.

Theorem 6. *If* $(\mathsf{Encode}, \mathsf{Decode})$ *is a linear OKVS, and it satisfies* $(t+n)$*-independence. Then it satisfies the* t*-randomness.*

Proof. In a binary OKVS, one can view the Encode algorithm as generating a solution to the linear system of equations:

$$\begin{bmatrix} -\mathsf{row}(x_1)- \\ -\mathsf{row}(x_2)- \\ \vdots \\ -\mathsf{row}(x_n)- \end{bmatrix} D = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{1}$$

where $\mathsf{row} : \mathcal{K} \to \mathbb{F}^m$ is a m-bits vector. We use $RD = Y$ to represent equation 1 for short.

From the $(n+t)$-independence of OKVS, for any $x_j^* \notin \{x_i\}_{i\in[n]}, j \in [t]$, the vectors $\{\mathsf{row}(x_i)\}_{i\in[n]} \cup \{\mathsf{row}(x_j^*)\}_{j\in[t]}$ are linear independent with overwhelming probability. As a result, vectors $\{\mathsf{row}(x_j^*)\}_{j\in[t]}$ are linear independent of the vectors $\{\mathsf{row}(x_i)\}_{i\in[n]}$ with overwhelming probability.

From the obliviousness of OKVS, the Encode algorithm selects a random solution from the subspace of all solutions to the linear system 1. Therefore, w.l.o.g, we can divide the matrix $R^{n\times m} = (R_1^{n\times n}, R_2^{n\times(m-n)})$, where $R_1^{n\times n}$ is a full rank matrix. The vector $D^{m\times 1}$ can also be divided into $(D_1^{n\times 1}, D_2^{(m-n)\times 1})^T$. Solving the equation 1 corresponds to

$$R_1^{n\times n} D_1^{n\times 1} + R_2^{n\times(m-n)} D_2^{(m-n)\times 1} = Y$$

We have

$$D_1 = R_1^{-1}(Y - R_2 D_2) \tag{2}$$

We can view the variables in D_2 as freedom variables and the variable in D_1 is determined by D_2. The Encode output $D = (D_1, D_2)^T$ by first selects random $D_2 \leftarrow \mathbb{F}^{m-n}$ and computes the corresponding D_1 as equation 2.

Now, for any other keys $x_j^* \notin \{x_i\}_{i\in[n]}, j \in [t]$, we denote the matrix

$$R^* := \begin{bmatrix} -\mathsf{row}(x_1^*)- \\ -\mathsf{row}(x_2^*)- \\ \vdots \\ -\mathsf{row}(x_t^*)- \end{bmatrix}$$

Similarly, we divide $R^{*\ t\times m} = (R_1^{*\ t\times n}, R_2^{*\ t\times(m-n)})$. Now, we need to prove the decoding result

$$\begin{aligned} Y^* &= R^* D = R_1^* D_1 + R_2^* D_2 \\ &= R_1^* R_1^{-1}(Y - R_2 D_2) + R_2^* D_2 \\ &= R_1^* R_1^{-1} Y + (R_2^* - R_1^* R_1^{-1} R_2) D_2 \end{aligned}$$

is uniform random. Note that $D_2 \in \mathbb{F}^{m-n}$ is randomly selected, we only need to prove the rank of $R_2^* - R_1^* R_1^{-1} R_2 \in \mathbb{F}^{t\times(m-n)}$ is t.

Let partitioned matrix

$$\bar{R} := \begin{bmatrix} R_1 & R_2 \\ R_1^* & R_2^* \end{bmatrix}$$

we have $rank(\bar{R}) = n + t$. Since linear transformation does not change the rank of the matrix, we subtract the left product $R_1^* R_1^{-1}$ of the first row from the second row, that is,

$$\begin{bmatrix} R_1 & R_2 \\ R_1^* - R_1^* R_1^{-1} \cdot R_1 & R_2^* - R_1^* R_1^{-1} \cdot R_1 \end{bmatrix} = \begin{bmatrix} R_1 & R_2 \\ 0 & R_2^* - R_1^* R_1^{-1} \cdot R_1 \end{bmatrix}$$

Since the rank of R_1 is n and the rank of the whole matrix is $n+t$, we have the rank of $R_2^* - R_1^* R_1^{-1} R_2$ is t.

Remark. Garimella et al. [GRS22] also proposed a property called *independence* for OKVS. Their definition is actually our 1-randomness. Then, they used the $(n+1)$-independence property we defined to prove any binary OKVS satisfies 1-randomness. van Baarsen and Pu [vBP24] further extend this proof to the general linear OKVS setting. However, both two proofs claimed the $(n+1)$-independence is implied by the correctness of OKVS, i.e., the Encdoe algorithm outputs $\perp$ with negligible probability. However, we find that the correctness can only imply t-independence for any $t \leq n$. For $t > n$, it is not trivially derived from the correctness of OKVS. The intuition behind this is that as t increases, the probability of the vectors $\{\mathsf{row}(x_i)\}_{i\in[t]}$ being linearly independent decreases. Consider a simple instantiation of linear OKVS, i.e., random matrix. In this case, the $\mathsf{row}(x)$ is a random bit vector in $\{0,1\}^m$. A random matrix with t rows and $m \geq t$ columns has linearly dependent rows with probability at least

$$\frac{\Pi_{i\in[0,t-1]}(2^m - 2^i)}{2^{tm}} \geq 1 - 2^{m-t}$$

This probability decreases as t increases, and is only negligible when $t < m - \lambda$. Zhang et al. [ZCL+23] also defined the 1-randomness and proved the linear OKVS satisfies this property. Although they did not explicitly assume $(n+1)$-independence, their proof also implies this property. It seems that $(n+t)$-independence is inevitable when proving the t-randomness. Bienstock et al. [BPSY23] defined the *random decodings* property for OKVS. Their definition is weaker than 1-randomness as they additionally require values are also randomly selected, while the 1-randomness does not need this condition.

A.2 Proof of Theorem 2

Below we give details of the proof of Theorem 2.

Proof. We will prove for any s-separate set W, Q, the $\{\sigma_j\}_{j\in[m]}$ given to the adversary $\mathcal{A}$ in $\mathsf{Exp}^{\mathcal{A}}(W, Q, \kappa)$ are statistically indistinguishable from a uniform distribution over $\mathcal{V}^m$.

The s-separate property of W guarantees the input keys of Encode are all distinct. The s-separate property of Q guarantees that for any $j \in [m], k \in [d/s]$, there is at most one $i \in [n]$, such that $k||\bar{q}_{j,k} \in \{k||\bar{w}_{i,k} + t\}_{t\in[-\delta,\delta]^s}$. This means for any $j \in [m], k \in [d/s]$, $\mathsf{Decode}(k||\bar{q}_{j,k})$ are all distinct.

For $j \in [m]$, the computation of σ_j can be divided into the following two cases:

1. $\exists k_0 \in [d/s], i_0 \in [n]$, s.t. $k_0||\boldsymbol{q_{j,k_0}} \in \{k||\boldsymbol{w_{i_0,k}}+t\}_{t\in[-\delta,\delta]^s}$. In this case, we have $\mathsf{Decode}(E, k_0||\boldsymbol{q_{j,k_0}}) = \mathsf{Rand}_{i_0,k_0}$. Since Rand_{i_0,k_0} is randomly selected, $\sigma_j = \sum_{k\in[d/s]} \mathsf{Decode}(E, k||\bar{q}_{j,k})$ is also uniform random.
2. $\forall k \in [d/s], \forall i \in [n], k||\bar{q}_{j,k} \notin \{k||\bar{w}_{i,k} + t\}_{t\in[-\delta,\delta]^s}$. In this case, for any $k_0 \in [d/s]$, the 1-randomness of OKVS guarantees $\mathsf{Decode}(E, k_0||\boldsymbol{q_{j,k_0}})$ is uniform random in $\mathcal{V}$. Therefore, $\sigma_j = \sum_{k\in[d/s]} \mathsf{Decode}(E, k||\bar{q}_{j,k})$ is also uniform random.

Consider the joint distribution of $S = \{\sigma_j\}_{j\in[m]}$, we define S_1 as the set of σs satisfy the first case and S_2 as the set of σs satisfy the second case. We have $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. Note that $\mathsf{Decode}(k||\bar{q}_{j,k})$ are all distinct, this means for any $\sigma \in S_1$, there is at least one fresh $\mathsf{Rand}_{j,k}$ is the addend of σ. Thus the S_1 is uniform distributed in $\mathcal{V}^{|S_1|}$. For $\sigma \in S_2$, the m-randomness of OKVS guarantees S_2 is uniform distributed in $\mathcal{V}^{|S_2|}$ since $|S_2| \leq m$.

A.3 Proof of Theorem 3

Below we give details of the proof of Theorem 3.

Proof. Since the protocol is symmetric, we only exhibit the simulator $\mathsf{Sim}_{\mathcal{S}}$ for simulating corrupt $\mathcal{S}$ and argue the indistinguishability of the produced transcript from the real execution.

Corrupt Sender: $\mathsf{Sim}_{\mathcal{S}}(Q, \{\mathsf{ID}_{q_j}\}_{j\in[m]})$ simulates the view of corrupt semi-honest sender. It executes as follows:

1. In step 1, $\mathsf{Sim}_{\mathcal{S}}$ executes like an honest sender, and learns the $\mathsf{Rand}^{\mathcal{S}}_{j,k}, \mathsf{pid}^{\mathcal{S}}_{j}, j \in [m], k \in [d/s]$.
2. In step 2, $\mathsf{Sim}_{\mathcal{S}}$ selects random $sk_{\mathcal{S}}$. Then, it invokes OPRF-sum sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{oprf\text{-}sum}}(sk_{\mathcal{S}})$ and appends the output to the view.
3. In step 6, $\mathsf{Sim}_{\mathcal{S}}$ selects random $f_{\mathcal{S},j} \leftarrow \mathbb{F}, j \in [m]$. Then, for $j \in [m]$, it invokes the OPRF-sum receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{oprf\text{-}sum}}(\{\boldsymbol{q}_{j,k}\}_{k\in[d/s]}, f_{\mathcal{S},j})$ and appends the output to the view.
4. In step 7, $\mathsf{Sim}_{\mathcal{S}}$ computes $\rho_{\mathcal{S},j} := \mathsf{ID}_{q_j} - \mathsf{pid}^{\mathcal{S}}_{j} + f_{\mathcal{S},j}$ for $j \in [m]$. Then, $\mathsf{Sim}_{\mathcal{S}}$ generates random additive sharing $\{sh_{\mathcal{S},j,k}\}_{k\in[d/s]}$ of $\rho_{\mathcal{S},j}$, i.e., $\sum_{k\in[d/s]} sh_{\mathcal{S},j,k} = \rho_{\mathcal{S},j}$. Now $\mathsf{Sim}_{\mathcal{S}}$ picks $((2\delta+1)^s-1)\cdot d/s \cdot n$ random key-value pairs, denoted as A, and computes $E_{\mathcal{R}} := \mathsf{Encode}(A \cup \{(k||\boldsymbol{q}_{j,k}, sh_{\mathcal{S},j,k}\})$. Then, it appends $E_{\mathcal{R}}$ to the view.

Now we argue that the view output by $\mathsf{Sim}_{\mathcal{S}}$ is indistinguishable from the real one. We formally prove this by a standard hybrid argument method. We define five hybrid transcripts T_0, T_1, T_2, T_3, T_4 where T_0 is real view of $\mathcal{S}$, and T_4 is the output of $\mathsf{Sim}_{\mathcal{S}}$.

- Hybrid_0. The first hybrid is the real interaction described in Figure 13. Here, an honest $\mathcal{R}$ uses input W, honestly interacts with the corrupt $\mathcal{S}$. Let T_0 denote the real view of $\mathcal{S}$.
- Hybrid_1. Let T_1 be the same as T_0, except that all PRF values $F_{sk_{\mathcal{R}}}(\cdot)$ are replaced by randomly selected values. This hybrid is computationally indistinguishable from T_0 by the pseudorandomness of the PRF.
- Hybrid_2. Let T_2 be the same as T_1, except that the OKVS $E_{\mathcal{R}}$ is randomly selected in $\mathcal{V}^m$. Note that in the previous hybrid, the key-value pairs used to compute OKVS are $\{(k||(\bar{w}_{i,k} + t), \mathsf{Rand}^{\mathcal{R}}_{i,k} + f_{i,k,t})\}_{i\in[n],k\in[d/s],t\in[-\delta,\delta]^s}$, where $f_{i,k,t}$ is truly random value. For $j \in [m]$, if $\exists i \in [n]$, s.t. $\mathsf{dist}(q_j, w_i) \le \delta$, then the values $\mathsf{Rand}^{\mathcal{R}}_{i,k} + f_{i,k,t_{i,k}}$ are uniform and independent of the sender's view, where $t_{i,k} = q_{j,k} - w_{i,k}$. This is because the sender only knows $\sum_{k\in[d/s]} f_{i,k,t_{i,k}}$ and the value $\sum_{k\in[d/s]} \mathsf{Rand}^{\mathcal{R}}_{i,k}$ is also randomly selected. If $\forall i \in [n], \mathsf{dist}(q_j, w_i) > \delta$, then the values $\mathsf{Rand}^{\mathcal{R}}_{i,k} + f_{i,k,t}$ are uniform and independent of the sender's view because all $f_{i,k,t}$'s are randomly selected. In summary, the values encoded in $E_{\mathcal{R}}$ are uniform random. By the obliviousness property of OKVS, T_1 and T_2 are statistically indistinguishable.
- Hybrid_3. Let T_3 be the same as T_2, except that the generation of OKVS $E_{\mathcal{R}}$ is now the same as the simulation. In other words, instead of computing $\sigma_{\mathcal{S},j}$ as in step 8, where values corresponding to $k||\boldsymbol{q}_{j,k}$ are uniform random, we instead compute $\sigma_{\mathcal{S},j}$ randomly and then encode OKVS to contain the correct value (and be otherwise uniform). This change has no effect on sender's view distribution. T_2 and T_3 are identical.
- Hybrid_4. Let T_4 be the same as T_3, except that two OPRF-sum executions are replaced by simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{oprf\text{-}sum}}(sk_{\mathcal{S}})$ and $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{oprf\text{-}sum}}(\{\boldsymbol{q}_{j,k}\}_{k\in[d/s]}, f_{\mathcal{S},j})$. The security of OPRF-sum functionality guarantees this view is indistinguishable from T_3. This hybrid is exactly the view output by the simulator.

A.4 Proof of Theorem 4

Below we give details of the proof of Theorem 4.

Proof. We exhibit simulators $\mathsf{Sim}_{\mathcal{S}}$ and $\mathsf{Sim}_{\mathcal{R}}$ for simulating corrupt $\mathcal{S}$ and $\mathcal{R}$ respectively, and argue the indistinguishability of the produced transcript from the real execution.

Corrupt Sender: $\mathsf{Sim}_{\mathcal{S}}(Q = \{q_j\}_{j\in[m]})$ simulates the view of corrupt semi-honest sender. It executes as follows:

1. In step 1, $\mathsf{Sim}_{\mathcal{S}}$ selects random $id_{q_j} \leftarrow \mathbb{F}$ for $j \in [m]$. Let $\mathsf{ID}_Q := \{id_{q_j}\}_{j\in[m]}$ Then, it invokes fuzzy mapping generation sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{FmapG}}(Q, \mathsf{ID}_Q)$ and appends the output to the view.
2. In step 3, $\mathsf{Sim}_{\mathcal{S}}$ invokes the mq-RPMT sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{mq\text{-}rpmt}}(\mathsf{ID}_Q)$ and appends the output to the view. Then, for $j \in [m]$, $\mathsf{Sim}_{\mathcal{S}}$ invokes OT sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{ot}}(\perp, q_j)$ and appends the output to the view.

Now we argue that the view output by $\mathsf{Sim}_{\mathcal{S}}$ is indistinguishable from the real one. The view output by $\mathsf{Sim}_{\mathcal{S}}$ is indistinguishable from the real one by the underlying simulators' indistinguishability.
Corrupt Receiver: $\mathsf{Sim}_{\mathcal{R}}(W, I)$ simulates the view of corrupt semi-honest receiver. It executes as follows:

1. In step 1, $\mathsf{Sim}_{\mathcal{R}}$ selects random $id_{w_i} \leftarrow \mathbb{F}$ for $i \in [n]$. Let $\mathsf{ID}_W := \{id_{w_i}\}_{i\in[n]}$ Then, it invokes fuzzy mapping generation receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{FmapG}}(W, \mathsf{ID}_W)$ and appends the output to the view.
2. In step 3, $\mathsf{Sim}_{\mathcal{R}}$ uses $\perp$ to pad I to m elements and permutes these elements randomly. Let $I = \{z_1, \ldots, z_m\}$. For $j \in [m]$, $\mathsf{Sim}_{\mathcal{R}}$ sets $b_j = 1$ if and only if $z_j \notin W$, otherwise, $b_j = 0$. Then, $\mathsf{Sim}_{\mathcal{R}}$ invokes the mq-RPMT receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{Ifmat}}(\mathsf{ID}_W, b)$ and appends the output to the view. Then, for $j \in [m]$, $\mathsf{Sim}_{\mathcal{R}}$ invokes OT receiver's simulator $\mathsf{Sim}^{R}_{\mathsf{ot}}(b_j, z_j)$ and appends the output to the view.

Now we argue that the view output by $\mathsf{Sim}_{\mathcal{R}}$ is indistinguishable from the real one. In the simulation, the way $\mathcal{R}$ obtains the elements in I is identical to the real execution since the elements in I are randomly permuted. By the underlying simulators' indistinguishability, the simulated view is computationally indistinguishable from the real.

A.5 Proof of Theorem 5

Below we give details of the proof of Theorem 5.

Proof. We exhibit simulators $\mathsf{Sim}_{\mathcal{S}}$ and $\mathsf{Sim}_{\mathcal{R}}$ for simulating corrupt $\mathcal{S}$ and $\mathcal{R}$ respectively, and argue the indistinguishability of the produced transcript from the real execution.

Corrupt Sender: $\mathsf{Sim}_{\mathcal{S}}(Q = \{q_j\}_{j\in[m]})$ simulates the view of corrupt semi-honest sender. It executes as follows:

1. In step 1, $\mathsf{Sim}_{\mathcal{S}}$ selects random $id_{q_j} \leftarrow \mathbb{F}$ for $j \in [m]$. Let $\mathsf{ID}_Q := \{id_{q_j}\}_{j\in[m]}$ Then, it invokes fuzzy mapping generation sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{FmapG}}(Q, \mathsf{ID}_Q)$ and appends the output to the view.
2. In step 2, $\mathsf{Sim}_{\mathcal{S}}$ executes like an honest sender, and learns $\mathsf{CH}_{\mathsf{ID}}$ and CH_Q.
3. In step 3, $\mathsf{Sim}_{\mathcal{S}}$ selects random $f_{\mathcal{S},\sigma(j)} \leftarrow \mathbb{F}, j \in [m]$. Then, for $j \in [m]$, it invokes the OPRF-sum receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{oprf\text{-}sum}}(\{\mathsf{CH}_{\mathsf{ID}}[\sigma(j)]||k||\mathsf{CH}_Q[\sigma(j)]_k||\alpha_{\sigma(j)}\}_{k\in[d]}, f_{\mathcal{S},\sigma(j)})$ and appends the output to the view.
4. In step 5, $\mathsf{Sim}_{\mathcal{S}}$ selects $3nd \cdot (2\delta + 1)$ random key-value pairs and computes the corresponding OKVS $E_{\mathcal{R}}$. Then $\mathsf{Sim}_{\mathcal{S}}$ appends $E_{\mathcal{R}}$ to the view.
5. In step 7, $\mathsf{Sim}_{\mathcal{S}}$ selects a random permutation π over $[m']$, random values $r^{\mathcal{S}}_u \leftarrow \mathbb{F}, u \in [m']$. Then, it invokes Permute+Share sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{ps}}(\pi, \{r^{\mathcal{S}}_u\}_{u\in[m']})$ and appends the output to the view. Then, it computes $\{\mathsf{Sh}^{\mathcal{S}}_u\}_{u\in[m']}$ as an honest sender.
6. In step 8, for $u \in [m']$, $\mathsf{Sim}_{\mathcal{S}}$ invokes the Ifmat sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{Ifmat}}(-\mathsf{Sh}^{\mathcal{S}}_u + \frac{1}{2}\delta^p)$ and appends the output to the view.
7. In step 9, for $u \in [m']$, $\mathsf{Sim}_{\mathcal{S}}$ invokes OT sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{ot}}(\perp, \mathsf{CH}_Q[\pi(u)])$ and appends the output to the view.

Now we argue that the view output by $\mathsf{Sim}_{\mathcal{S}}$ is indistinguishable from the real one. We formally prove this by a standard hybrid argument method. We define five hybrid transcripts T_0, T_1, T_2, T_3 where T_0 is real view of $\mathcal{S}$, and T_3 is the output of $\mathsf{Sim}_{\mathcal{S}}$.

– Hybrid_0. The first hybrid is the real interaction described in Figure 15. Here, an honest $\mathcal{R}$ uses input W, honestly interacts with the corrupt $\mathcal{S}$. Let T_0 denote the real view of $\mathcal{S}$.

- Hybrid_1. Let T_1 be the same as T_0, except that all PRF values $F_{sk_{\mathcal{R}}}(\cdot)$ are replaced by randomly selected values. This hybrid is computationally indistinguishable from T_0 by the pseudorandomness of the PRF.
- Hybrid_2. Let T_2 be the same as T_1, except that the OKVS $E_{\mathcal{R}}$ is computed from random key-value pairs. Note that in the previous hybrid, the key-value pairs used to compute OKVS are $\{(id_{w_i}||k||(w_{i,k}+t))||\alpha, |t|^p+r_{h_\alpha(id_{w_i}),k}+f_{i,k,t,\alpha}\}_{i\in[n],k\in[d],t\in[-\delta,\delta],\alpha\in[3]}$, where $f_{i,k,t,\alpha}$ is truly random value. For $u \in [m']$, if $\exists i \in [n]$, s.t. $\mathsf{dist}(\mathsf{CH}_Q[u], w_i) \leq \delta$, then the values $|t_{i,k}|^p + r_{h_{\alpha_u}(id_{w_i}),k} + f_{i,k,t_{i,k},\alpha}$ are uniform and independent of the sender's view, where $t_{i,k} = \mathsf{CH}_Q[u]_k - w_{i,k}$. This is because the sender only knows $\sum_{k\in[d]} f_{i,k,t_{i,k},\alpha_u}$ and the value $\sum_{k\in[d]} r_{h_{\alpha_u}(id_{w_i}),k}$ is also randomly selected. If $\forall i \in [n], \mathsf{dist}(\mathsf{CH}_Q[u], w_i) > \delta$, then the values $r_{h_\alpha(id_{w_i}),k} + f_{i,k,t,\alpha}$ are uniform and independent of the sender's view because all $f_{i,k,t,\alpha}$'s are randomly selected. In summary, the values encoded in $E_{\mathcal{R}}$ are uniform random. By the obliviousness property of OKVS, T_1 and T_2 are statistically indistinguishable.
- Hybrid_3. Let T_3 be the same as T_2, except that the OPRF-sum, Permute +Share, IFmat and OT execution are replaced by simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{oprf\text{-}sum}}, \mathsf{Sim}^{\mathcal{R}}_{\mathsf{ps}}, \mathsf{Sim}^{\mathcal{S}}_{\mathsf{IFmat}}, \mathsf{Sim}^{\mathcal{S}}_{\mathsf{ot}}$. The security of OPRF-sum, Permute +Share, IFmat and OT functionality guarantee this view is indistinguishable from T_2. This hybrid is exactly the view output by the simulator.

Corrupt Receiver: $\mathsf{Sim}_{\mathcal{R}}(W, I)$ simulates the view of corrupt semi-honest receiver. It executes as follows:

1. In step 1, $\mathsf{Sim}_{\mathcal{R}}$ selects random $id_{w_i} \leftarrow \mathbb{F}$ for $i \in [n]$. Let $\mathsf{ID}_W := \{id_{w_i}\}_{i\in[n]}$ Then, it invokes fuzzy mapping generation receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{FmapG}}(W, \mathsf{ID}_W)$ and appends the output to the view.
2. In step 3, $\mathsf{Sim}_{\mathcal{R}}$ selects random $sk_{\mathcal{R}}$. Then, it invokes OPRF-sum sender's simulator $\mathsf{Sim}^{\mathcal{S}}_{\mathsf{oprf\text{-}sum}}(sk_{\mathcal{R}})$ and appends the output to the view.
3. In step 4 and 5, $\mathsf{Sim}_{\mathcal{R}}$ executes like an honest receiver, and learns $r_{u,k}, \mathsf{Rand}^{\mathcal{R}}_u, E_{\mathcal{R}}, u \in [m'], k \in [d]$.
4. In step 7, $\mathsf{Sim}_{\mathcal{R}}$ selects random values $r^{\mathcal{R}}_u \leftarrow \mathbb{F}, u \in [m']$. Then, it invokes Permute+Share receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{ps}}(\{\mathsf{Rand}^{\mathcal{R}}_u\}_{u\in[m']}, \{r^{\mathcal{R}}_u\}_{u\in[m']})$ and appends the output to the view. Then, it computes $\{\mathsf{Sh}^{\mathcal{R}}_u\}_{u\in[m']}$ as an honest receiver.
5. In step 8, $\mathsf{Sim}_{\mathcal{R}}$ uses $\perp$ to pad I to m' elements and permutes these elements randomly. Let $I = \{z_1, \ldots, z_{m'}\}$. For $u \in [m']$, $\mathsf{Sim}_{\mathcal{R}}$ sets $b_u = 1$ if and only if $z_u \notin W$, otherwise, $b_u = 0$. Then, $\mathsf{Sim}_{\mathcal{R}}$ invokes the Ifmat receiver's simulator $\mathsf{Sim}^{\mathcal{R}}_{\mathsf{IFmat}}(\mathsf{Sh}^{\mathcal{R}}_u, b_u)$ and appends the output to the view.
6. In step 9, for $u \in [m']$, $\mathsf{Sim}_{\mathcal{R}}$ invokes OT receiver's simulator $\mathsf{Sim}^{R}_{\mathsf{ot}}(b_u, z_u)$ and appends the output to the view.

Now we argue that the view output by $\mathsf{Sim}_{\mathcal{R}}$ is indistinguishable from the real one. In the simulation, the way $\mathcal{R}$ obtains the elements in I is identical to the real execution since the elements in I are randomly permuted. By the underlying simulators' indistinguishability, the simulated view is computationally indistinguishable from the real.

B Comparison Results of FPSI for L_1 and L_2 Distances

In this section, we compare our FPSI protocols with the state-of-the-art protocol [GQL+24] for L_1 and L_2 distance. Specifically, we consider the set size $m = n \in \{2^4, 2^8, 2^{12}, 2^{16}\}$, dimension $d \in \{2, 6, 10\}$, and distance threshold $\delta \in \{5, 8, 10\}$. Detailed comparisons of protocol performance for L_1 and L_2 distance are shown in Table 3 and Table 4, respectively.

Consistent with the results for L_∞ distance, our protocols (for both $s = 1$ and $s = 2$ cases) demonstrate superior performance to Gao et al.'s protocol [GQL+24] across all tested parameter configurations for both L_1 and L_2 distance metrics. For L_1 distance, we observe runtime improvements of $1.1 - 35.5\times$ coupled with communication reductions of $1.1 - 5\times$. Similarly, for L_2 distance, the protocol achieves $1.1 - 29.2\times$ faster execution while reducing communication overhead by $1.1 - 4.9\times$ across all parameter configurations.

While the performance gains for L_p distance are less pronounced than those for L_∞ distance, our protocol still achieves significant improvements. This difference stems from the additional fuzzy matching steps required in our L_p distance protocol. Nevertheless, through the strategic use of efficient symmetric-key operations, we maintain considerable performance enhancements for L_p distance computations.

(d, δ)	Protocol	Set size $m = n$							
		2^4		2^8		2^{12}		2^{16}	
		Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time
$(2, 5)$	[GQL+24]	0.256	0.116	4.097	1.141	65.479	17.986	1047.604	305.411
	Ours ($s = 1$)	0.178	0.096	1.152	0.170	18.631	1.319	314.131	19.714
	Ours ($s = 2$)	0.237	0.104	2.246	0.190	34.932	1.463	575.834	22.309
$(2, 8)$	[GQL+24]	0.388	0.130	6.200	1.460	99.142	22.874	1586.204	392.120
	Ours ($s = 1$)	0.200	0.097	1.959	0.194	24.203	1.380	403.597	20.866
	Ours ($s = 2$)	0.355	0.112	4.411	0.237	68.332	1.902	1111.604	29.230
$(2, 10)$	[GQL+24]	0.475	0.142	7.593	1.655	121.417	25.883	1942.604	444.910
	Ours ($s = 1$)	0.213	0.103	1.959	0.195	27.727	1.426	460.313	21.703
	Ours ($s = 2$)	0.456	0.117	6.059	0.246	97.438	2.275	1579.035	35.294
$(6, 5)$	[GQL+24]	0.735	0.234	11.761	2.772	188.117	45.248	3009.804	760.695
	Ours ($s = 1$)	0.284	0.103	2.906	0.218	46.321	1.914	758.232	29.761
	Ours ($s = 2$)	0.454	0.111	5.955	0.247	95.487	2.394	1547.288	37.476
$(6, 8)$	[GQL+24]	1.128	0.267	18.042	3.689	288.604	59.503	4617.604	997.566
	Ours ($s = 1$)	0.343	0.106	4.178	0.227	62.585	2.137	1019.412	33.027
	Ours ($s = 2$)	0.947	0.119	12.157	0.346	195.027	3.704	3144.764	58.260
$(6, 10)$	[GQL+24]	1.389	0.307	22.218	4.132	355.429	68.587	5686.804	1156.825
	Ours ($s = 1$)	0.380	0.108	4.571	0.257	73.198	2.270	1189.918	35.161
	Ours ($s = 2$)	1.190	0.120	17.637	0.404	282.978	4.766	4555.867	76.501
$(10, 5)$	[GQL+24]	1.215	0.353	19.426	4.368	310.755	72.023	4972.005	1209.224
	Ours ($s = 1$)	0.389	0.114	4.809	0.263	74.028	2.543	1202.836	39.717
	Ours ($s = 2$)	0.978	0.114	9.724	0.299	155.996	3.306	2518.104	52.957
$(10, 8)$	[GQL+24]	1.868	0.377	29.883	5.723	478.067	95.468	7649.005	1609.335
	Ours ($s = 1$)	0.484	0.124	6.306	0.275	100.863	2.892	1633.434	45.308
	Ours ($s = 2$)	1.333	0.123	20.086	0.437	321.997	5.472	5182.204	87.801
$(10, 10)$	[GQL+24]	2.303	0.444	36.844	6.642	589.442	110.568	9427.000	-
	Ours ($s = 1$)	0.693	0.129	7.402	0.304	118.800	3.121	1921.258	48.885
	Ours ($s = 2$)	2.064	0.130	29.170	0.568	468.076	7.289	7525.610	118.606

Table 3. Communication cost (in MB) and running time (in seconds) comparing our protocols to [GQL+24] for L_1 distance. Cells with - denotes trials that ran out of memory. The best result is marked in blue, and the second best result is marked in cyan.

(d, δ)	Protocol	Set size $m = n$							
		2^4		2^8		2^{12}		2^{16}	
		Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time
$(2, 5)$	[GQL+24]	0.260	0.112	4.151	1.239	66.354	19.257	1061.604	326.394
	Ours ($s = 1$)	0.188	0.093	1.217	0.230	19.593	1.966	329.730	30.493
	Ours ($s = 2$)	0.247	0.099	2.312	0.229	35.898	2.150	591.433	33.022
$(2, 8)$	[GQL+24]	0.392	0.144	6.271	1.538	100.267	24.556	1604.204	417.760
	Ours ($s = 1$)	0.212	0.112	2.041	0.236	25.406	2.051	423.097	31.631
	Ours ($s = 2$)	0.368	0.114	4.492	0.274	69.535	2.578	1131.104	39.782
$(2, 10)$	[GQL+24]	0.480	0.156	7.679	1.712	122.792	27.811	1964.604	477.439
	Ours ($s = 1$)	0.229	0.118	2.063	0.263	29.251	2.552	485.012	39.420
	Ours ($s = 2$)	0.472	0.118	6.162	0.322	98.962	3.364	1603.735	52.741
$(6, 5)$	[GQL+24]	0.739	0.220	11.816	2.802	188.992	46.191	3023.804	777.793
	Ours ($s = 1$)	0.294	0.097	2.971	0.266	47.283	2.616	773.832	40.415
	Ours ($s = 2$)	0.464	0.107	6.020	0.298	96.450	3.100	1562.888	48.259
$(6, 8)$	[GQL+24]	1.132	0.263	18.112	3.698	289.729	61.189	4635.604	1021.320
	Ours ($s = 1$)	0.355	0.114	4.260	0.286	63.788	2.804	1038.912	43.603
	Ours ($s = 2$)	0.959	0.121	12.239	0.373	196.230	4.389	3164.264	68.267
$(6, 10)$	[GQL+24]	1.394	0.301	22.304	4.244	356.804	70.423	5708.804	1220.301
	Ours ($s = 1$)	0.396	0.121	4.674	0.314	74.722	3.368	1214.618	52.869
	Ours ($s = 2$)	1.206	0.126	17.740	0.471	284.502	5.866	4580.567	93.551
$(10, 5)$	[GQL+24]	1.218	0.313	19.481	4.419	311.630	72.651	4986.005	1225.036
	Ours ($s = 1$)	0.399	0.101	4.874	0.293	74.990	3.215	1218.436	50.505
	Ours ($s = 2$)	0.988	0.113	9.789	0.350	156.958	4.017	2533.703	63.704
$(10, 8)$	[GQL+24]	1.872	0.404	29.954	5.843	479.192	97.445	7667.005	1627.172
	Ours ($s = 1$)	0.497	0.121	6.388	0.358	102.066	3.564	1652.934	55.734
	Ours ($s = 2$)	1.346	0.125	20.167	0.486	323.200	6.171	5201.704	97.173
$(10, 10)$	[GQL+24]	2.309	0.462	36.930	6.771	590.817	112.630	9449.000	-
	Ours ($s = 1$)	0.709	0.124	7.505	0.365	120.324	4.232	1945.958	66.451
	Ours ($s = 2$)	2.080	0.136	29.274	0.630	469.599	8.359	7550.309	135.196

Table 4. Communication cost (in MB) and running time (in seconds) comparing our protocols to [GQL+24] for L_2 distance. Cells with - denotes trials that ran out of memory. The best result is marked in blue, and the second best result is marked in cyan.