Efficient Fuzzy PSI Based on Prefix Representation

Chengrui Dang[1,2,3], Xv Zhou[4,1] and Bei Liang[a,1]

[1] Beijing Institute of Mathematical Sciences and Applications, Beijing, China
[2] State Key Laboratory of Mathematical Sciences, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] School of Cyber Science and Technology, Beihang University, Beijing, China

Abstract. Fuzzy PSI is a variant of PSI, which on input a set of points from the receiver and sender respectively, allows the receiver to learn which of the sender's points lie within a threshold distance δ under a specific distance metric.
Baarsen and Pu (EUROCRYPT'24) first proposed efficient fuzzy PSI protocols for general L_p distances (where $p \in [1, \infty]$) in d-dimensional space, achieving communication complexity linear in the input size, δ, and $2^d d$. However, they leave open the question of whether the prefix technique of Chakraborti et al. (USENIX Security'23) can further reduce the communication complexity of their fuzzy PSI protocols in both low and high dimensions.
In this work, we thoroughly explore using the prefix technique to reduce the complexity of fuzzy PSI. First, we propose fuzzy matching protocols for L_∞ and L_p distances, where the communication complexity is improved from $O(\delta d)$ to $O(\log \delta\, d)$ for L_∞, and from $O(\delta^p)$ to $O((\log \delta)^d p)$ for L_p distance. By applying our fuzzy matching protocol in conjunction with spatial hashing, we propose fuzzy PSI protocols for low-dimensional space. For high-dimensional space, we present the first fuzzy PSI protocols achieving communication and computation complexity that scales logarithmically in δ and linearly in dimension d and input set sizes.
We implement our fuzzy PSI protocols and compare them with state-of-the-art protocols. Experimental results demonstrate that our protocols achieve superior performance for large δ: for input size $N = 2^8$, $d = 5$, and $\delta = 256$, our protocol requires 10–36× less running time and 3–4.5× lower communication than existing protocols.

1 Introduction

Private Set Intersection (PSI) is a widely used cryptographic primitive that enables two parties to compute the intersection of their private sets without disclosing elements outside the intersection. Numerous highly efficient PSI protocols have been proposed over the last decade [PSZ14, KKRT16, PRTY19, CM20, RS21, RR22], and have been applied in various real-world scenarios, including biometric data identification [UCK+21], image data matching [KM21], and online advertising measurement [IKN+20, MPR+20], among others.

Structure-aware PSI, introduced by Garimella et al. [GRS22] as a variant of PSI, considers scenarios where the receiver holds a structured set (e.g., single-dimensional intervals, high-dimensional balls, or disjoint unions of balls) with publicly known structure,

E-mail: dangchengrui@bimsa.cn (Chengrui Dang), zhouxv@bimsa.cn (Xv Zhou), lbei@bimsa.cn (Bei Liang)
[a]Corresponding author

while the sender has an unstructured set of points. Fuzzy PSI is a specialized form of structure-aware PSI where, given N_r points from the receiver and N_s points from the sender (both in $\mathbb{Z}^d$), the receiver learns which sender points lie within threshold distance δ of any receiver point under metrics like Minkowski (L_∞, L_p) or Hamming distance. Garimella et al. [GRS22] constructed the first semi-honest fuzzy PSI protocol for L_∞ distance using spatial hashing and function secret sharing, later extending it to achieve malicious security [GRS23]. While their protocol's communication cost is independent of the structured input set's total volume $N_r\delta^d$, the receiver's computation cost scales linearly with $N_r\delta^d$, resulting in poor performance in high dimensions.

To circumvent this blowup of the computational cost on the receiver's side, Garimella et al. [GGM24] present a semi-honest fuzzy PSI protocol for d-dimensional L_∞-balls, which achieves both communication and computation complexity of $O(N_r(\log\delta)^d)$, instead of scaling with $N_r\delta^d$. The main building block underlying their construction is a primitive called Incremental Boolean Function Secret Sharing (ibFSS), which extends Boolean FSS (from prior work [GRS22]) by enabling evaluation on both inputs and all their prefixes. Their key observation is that a d-dimensional L_∞-ball can be represented by a distinct and bounded set of critical prefixes, where every point in the ball shares at least one critical prefix. The computational reduction relies on the fact that the number of critical prefixes is much smaller than the volume of the ball. Independently, Chakraborti et al. [CFR23] proposed a distance-aware PSI protocol for 1-dimensional L_1 distance over integers using prefix comparisons, achieving $O(\log\delta)$ communication complexity.

Concurrently, with noticing the drawback of the receiver's computation cost in [GRS22], Baarsen and Pu [vBP24] proposed fuzzy PSI protocols for L_∞ and L_p distances based on fuzzy matching, where both the communication and computation costs in high-dimensional cases scale linearly or quadratically with the dimension d. Fuzzy matching, introduced by [vBP24], is a core building block of fuzzy PSI, which, on input a point $\mathbf{x} \in \mathbb{Z}^d$ from the receiver and a point $\mathbf{y} \in \mathbb{Z}^d$ from the sender, outputs 1 to the receiver if $\|\mathbf{x}-\mathbf{y}\|_p \leq \delta$ and 0 otherwise. By using spatial hashing to identify points that are potentially close, and then applying fuzzy matching on those pairs, Baarsen and Pu [vBP24] proposed fuzzy PSI protocols for both low- and high-dimensional cases, and extended their protocols to broader fuzzy PSI functionalities.

Note that Baarsen and Pu [vBP24] were aware of the possibility of achieving an even lower communication complexity by applying the prefix technique to their fuzzy matching protocol for L_∞ (refer to Remark 2 in [vBP24]). However, they left this as future work to explore. In fact, there are two main issues that need to be addressed. First, applying the prefix representation strategy [CFR23] directly might increase the sender's communication and computational complexity by a factor of $(\log\delta)^d$, and it is only applicable to low-dimensional cases. Moreover, we need to figure out how to extend it to high-dimensional cases without increasing the complexity by a factor of $O(2^d)$, which occurs when extending 1-to-1 fuzzy matching comparisons to N_s-to-N_r comparisons in fuzzy PSI using spatial hashing.

Another issue is that the prefix representation technique [CFR23] cannot be applied to the fuzzy matching protocol for L_p distance. The main reason is as follows. In the fuzzy matching protocol for L_p distance proposed by Baarsen and Pu [vBP24], for each dimension the receiver encodes all possible values $x_i + j$ (for $j \in [-\delta, \delta]$) in the interval into $r_{i,j}||r^s_{i,j}g^{|j|^p}$ where $r_{i,j} \leftarrow_\$ \mathbb{G}$. The sender uses its point y_i to decode the OKVS and then homomorphically sums up $|x_i - y_i|^p$ for every $i \in [d]$. Finally, the sender obtains an encryption of $(\|\mathbf{x}-\mathbf{y}\|_p)^p$. However, if the receiver wants to encode the prefixes representing the interval $[x_i - \delta, x_i + \delta]$ in an OKVS in a similar way, it must provide a solution to let the sender implicitly evaluate the distance $|x_i - y_i|^p$. This is because it is no longer feasible to set the value $g^{|j|^p}$, as there is no way to define the distance between a prefix representing an interval and a prefix of a point.

As is known, very recently, Gao et al. [GQL+25] studied the problem of how to overcome the $O(2^d)$ factor of complexity caused by spatial hashing in high-dimensional cases. Gao et al. [GQL+25] introduced fuzzy mapping to replace the use of spatial hash in constructing fuzzy PSI. Their fuzzy mapping construction achieves communication and computation complexity linear in δ, d, N_r and N_s. We consider whether it is possible to construct a fuzzy mapping protocol tailored to handle the prefix representation of intervals, leading to a fuzzy PSI with complexity sub-linear in δ.

In this work, we thoroughly explore the problem, left open by [vBP24], of how to use the prefix representation technique to further reduce the complexity of fuzzy PSI based on the paradigms proposed by Baarsen and Pu [vBP24]. We present fuzzy PSI protocols in the semi-honest setting for L_∞ and L_p distances with communication and computation complexity linear in $\log \delta$, dimension d, and the sizes of point sets N_r and N_s.

1.1 Our Contributions

Our main contributions are summarized as follows.

Fuzzy matching. We propose fuzzy matching protocols for L_∞ and L_p distances and compare the asymptotic complexity with [vBP24] in Table 1. For the L_∞ case, we reduce the communication from $O(\delta d)$ to $O(\log \delta\, d)$ at the cost of slightly increased computation by the receiver. In the L_p case, we reduce the communication cost from exponential to linear in p, namely from $O(\delta^p)$ to $O((\log \delta)^d p)$.

Fuzzy PSI in low dimension. By applying our fuzzy matching in conjunction with the spatial hash technique developed by Baarsen and Pu [vBP24], we propose fuzzy PSI-CA protocols (i.e., protocols that only let the receiver learn the cardinality of the intersection) for L_p and L_∞ distances, where the receiver's points are at least 4δ and $4\delta(d^{1/p}+1)$ apart from each other respectively. The asymptotic complexities are given in Table 1. Experimental results show that, in comparison with state-of-the-art protocols [vBP24, GQL+25], our low-dimensional (e.g., $d = 2$) fuzzy PSI protocol becomes the fastest when the threshold distance $\delta \geq 64$, and achieves the lowest communication among all protocols when $\delta \geq 32$ for L_∞ distance, $\delta \geq 64$ for L_1 distance, and $\delta \geq 128$ for L_2 distance.

Fuzzy PSI in high dimension. For the high-dimensional case, our main goal is to circumvent not only the $O((\log \delta)^d)$ complexity factor caused by enumerating all possible prefixes, but also the $O(2^d)$ complexity factor caused by spatial hashing. To address the $O((\log \delta)^d)$ factor, we introduce a new building block called *Private Index Search* (PIS), which takes as input $b \in \mathbb{Z}$ from the sender and a set $S = \{a_i\}_{i\in[0,n-1]} \subset \mathbb{Z}$ from the receiver, and outputs an index i to the receiver if $b = a_i \in S$, or a random index otherwise. Based on a secret-shared private equality test (ssPET) protocol, we construct a PIS protocol with complexity linear in $\log \delta$. To overcome the $O(2^d)$ factor, we develop a new semi-honest secure fuzzy mapping protocol by encoding the prefixes of the interval $[x_i - \delta, x_i + \delta]$ in each projection, instead of all points in the interval, into an OKVS in conjunction with the spatial additive sharing technique.

By applying our fuzzy mapping and private index search protocols, we construct fuzzy PSI-CA protocols for high-dimensional settings with communication and computation complexity linear in $\log \delta$, d, and the sizes of the point sets. See Table 1 for detailed comparisons of communication and computational complexities. Experimental results show that our high-dimensional fuzzy PSI protocols achieve superior performance for large δ values. For example, with $N_r = N_s = 2^8$, $d = 5$ and $\delta = 64$, our protocol requires 6–13× less running time and 1–2× lower communication than [GQL+25]; for $\delta = 256$, it requires 10-36× less running time and 3-4.5× lower communication.

Further Optimization. Let ω_δ denote the maximum number of prefixes representing all intervals of fixed size $2\delta + 1$, namely intervals $[x_i - \delta, x_i + \delta]$, and μ_δ the number of distinct prefixes of a point y_i. Both ω_δ and μ_δ are $O(\log \delta)$. We observe that reducing μ_δ

Table 1: Comparison of complexity for different fuzzy matching and fuzzy PSI-CA protocols. Receiver and Sender holds sets with size N_r and N_s respectively in $\mathbb{Z}^d$, d is the dimension, the distance threshold is δ. We define $(> c\delta)$ to be that the points in the Receiver's set are $c\delta$ apart, (disj.pro.) to be the L_∞ balls centered in points of Receiver's set have disjoint projection in some dimension with all other L_∞ balls, and (all disj.pro.) means the L_∞ balls centered in points of Receiver or Sender's set satisfies the above assumption. We omit the security parameter here. Cells in gray represents the best complexity.

	Metric	Protocols	Communication	Receiver Comp.	Sender Comp.
Fuzzy Matching	L_∞	[vBP24]	$O(\delta d)$	$O(\delta d)$	$O(d)$
		Ours	$O(\log \delta\, d)$	$O(\log \delta\, d + (\log \delta)^d)$	$O(\log \delta\, d)$
	L_p	[vBP24]	$O(\delta d + \delta^p)$	$O(\delta d)$	$O(d + \delta^p)$
		Ours	$O(\log \delta\, pd + \log \delta\, d + (\log \delta)^d p)$	$O(\log \delta\, pd + (2\log \delta)^d)$	$O(\log \delta\, pd + (2\log \delta)^d)$
Fuzzy PSI-CA	L_∞	conventional PSI[RR22]	$O((2\delta)^d N_r + N_s)$	$O((2\delta)^d N_r)$	$O((2\delta)^d N_r + N_s)$
		[GGM24]	$O(\log \delta\, dN_r + (2\log \delta)^d N_s)$	$O((\log \delta\, d + (\log \delta)^d)N_r + \log \delta\, dN_s)$	$O((2\log \delta)^d N_s)$
		[vBP24]$(> 2\delta)$	$O(\delta dN_r + 2^d N_s)$	$O(\delta dN_r + 2^d N_s)$	$O(2^d dN_s)$
		[vBP24]$(> 4\delta)$	$O(2^d \delta dN_r + N_s)$	$O(2^d \delta dN_r + N_s)$	$O(dN_s)$
		[GQL+25](all disj.pro.)	$O(\delta d(N_r + N_s))$	$O(\delta dN_r + N_s)$	$O(\delta dN_s + N_r)$
		Ours$(> 4\delta)$	$O(2^d \log \delta\, dN_r + \log \delta\, dN_s)$	$O(2^d \log \delta\, dN_r + (\log \delta)^d N_s)$	$O(\log \delta\, dN_s)$
		Ours(disj.pro.)	$O(\log \delta\, dN_r + \log \delta\, dN_s)$	$O(\log \delta\, dN_r + \log \delta\, dN_s)$	$O(\log \delta\, dN_s)$
	L_p	[vBP24]$(> 2\delta(d^{1/p} + 1))$	$O(\delta 2^d dN_r + \delta^p N_s)$	$O(\delta 2^d dN_r + N_s)$	$O((d + \delta^p)N_s)$
		[GQL+25](all disj.pro.)	$O(\delta dN_r + (\delta d + p \log \delta)N_s)$	$O(\delta dN_r + p \log \delta N_s)$	$O((\delta d + p \log \delta)N_s + N_r)$
		Ours $(> 2\delta(d^{1/p} + 1))$	$O(2^d p \log \delta\, dN_r + (\log \delta\, d + (2\log \delta)^d p)N_s)$	$O(2^d p \log \delta\, dN_r + (2\log \delta)^d N_s)$	$O(p \log \delta\, dN_s + (2\log \delta)^d N_s)$
		Ours(disj.pro.)	$O(\log \delta\, dpN_r + \log \delta\, dN_s)$	$O(\log \delta\, dpN_r + \log \delta\, dN_s)$	$O(\log \delta\, dpN_s)$

(which improves the sender's computational cost) leads to increasing ω_δ (which raises the receiver's communication cost).

To improve the overall efficiency of our fuzzy PSI-CA protocols in both low- and high-dimensional cases, we develop a method to trade off between ω_δ and μ_δ, enabling flexible balance between communication and computation. This optimization strategy may be of independent interest, as it can also be applied to recent work by Garimella et al. [GGM24].

1.2 Related Work

Freedman et al. [FNP04] informally introduced the problem of fuzzy private matching, which identifies whether two entries have a Hamming distance below a certain threshold. They presented a fuzzy matching protocol that was later proven insecure. Subsequent works [CH08, YSPW09] proposed fuzzy matching protocols for Hamming distance. Chakraborti et al. [CFR23] proposed a fuzzy PSI protocol (called distance-aware PSI) for Hamming distance. They also presented a protocol for one-dimensional L_1 distance over integers, constructed from any PSI protocol with input sets augmented by $O(\log \delta)$ prefix strings representing all integers in interval $[a - \delta, a + \delta]$ for each a in the input set.

Garimella et al. [GRS22] introduced structure-aware PSI (sa-PSI), which includes fuzzy PSI as a special case. They constructed a semi-honest sa-PSI from Boolean function secret sharing (bFSS). A subsequent work [GRS23] extended the protocol to achieve malicious security. Both constructions handle unions of L_∞-balls in d-dimensional space, where the receiver's computation cost scales linearly with her set's cardinality times δ^d. Garimella et al. [GGM24] presented a semi-honest sa-PSI with computation and communication scaling linearly with $(\log \delta)^d$, d, N_r and N_s. This was achieved through a new primitive called Incremental bFSS (ibFSS), which generalizes bFSS to allow evaluation on input prefixes.

Baarsen and Pu [vBP24] studied fuzzy PSI for general L_∞ and L_p distances in arbitrary dimension d. Based on the DDH assumption and public-key operations (particularly ElGamal encryption's homomorphism), they presented fuzzy PSI protocols for $L_{p\in[1,\infty]}$, with communication and computation costs scaling linearly with δ and exponentially in d. Their fuzzy matching protocols for $L_{p\in[1,\infty]}$ encode each point in the L_∞ ball's projection into DDH tuples verifiable using a secret key, using spatial hashing to map potentially close points to identical values. While [vBP24] enumerates all discrete points in each dimension's projected L_∞ ball interval, we reduce the interval's representation size by partitioning it

into prefix-representable sub-intervals, and use additive homomorphic encryption instead of DDH tuples for efficiency.

Gao et al. [GQL+25] introduced fuzzy mapping (Fmap), generalizing spatial hashing. Their Fmap-based fuzzy PSI protocol has communication and computation costs linear in δ, d, N_r and N_s. We construct a more efficient fuzzy mapping using prefix representations of intervals, achieving costs linear in $\log \delta$, d, N_r and N_s compared to Gao et al.'s linear scaling in δ.

2 High-Level Technical Overview

2.1 Fuzzy Matching for L_∞ Distance

We first briefly revisit the fuzzy matching protocol for L_∞ distance proposed by Baarsen and Pu [vBP24]. On input a point $\mathbf{x} \in \mathbb{Z}^d$ from the receiver and a point $\mathbf{y} \in \mathbb{Z}^d$ from the sender, fuzzy matching allows the receiver to learn whether $\|\mathbf{x}-\mathbf{y}\|_\infty \leq \delta$. Baarsen and Pu [vBP24] apply the idea of random self-reduction of DDH tuples in fuzzy matching, which enables homomorphic evaluation of the distance between private points. For simplicity, we present a simplified version of their protocol for L_∞ distance. Let $\mathbb{G}$ be a cyclic group of order q, $\delta \in \mathbb{Z}$, and $g, h = g^s \in \mathbb{G}$.

1. The receiver with an input $\mathbf{x} \in \mathbb{Z}^d$ and a secret $s \in \mathbb{Z}_q$ encodes every point in the interval $[x_i - \delta, x_i + \delta]$ into a DDH tuple via OKVS on each dimension, namely, $E_i = \mathsf{Encode}(\{(x_i + j, r_{i,j} || r_{i,j}^s : r_{i,j} \leftarrow_\$ \mathbb{G})\})$ for $j \in [-\delta, \delta]$.

2. The sender decodes on his point to retrieve values, namely, $\mathsf{Decode}(E_i, y_i) = u_i || v_i$ for each dimension $i \in [d]$. If $y_i \in [x_i - \delta, x_i + \delta]$, then $v_i = u_i^s$; otherwise, both u_i and v_i are random values. The sender then sends $u = g^a \prod_{i=1}^d u_i^b$ and $v = h^a \prod_{i=1}^d v_i^b$ to the receiver, where $a, b \leftarrow_\$ \mathbb{Z}_q$.

3. The receiver then checks if $u^s = v$.

The above approach yields a fuzzy matching protocol with communication complexity scaling linearly with the distance threshold δ. In this paper, we adopt the techniques of Chakraborti et al. [CFR23] to achieve communication complexity scaling linearly with $\log \delta$ by reducing the OKVS list size. Instead of encoding the entire interval $[x_i - \delta, x_i + \delta]$ in the OKVS for each dimension $i \in [d]$, the receiver encodes the prefixes representing $[x_i - \delta, x_i + \delta]$ into an OKVS where values are set as $\mathsf{Enc}_{\mathsf{pk}}(0)$. The sender must decode all possible prefixes of y_i for each dimension. Let μ denote the number of prefix representations of y_i. The sender retrieves values $\{u_{i,j}\}_{j \in [\mu]}$ for dimension i from OKVS decoding. The sender then sends $\{\boxplus_{\mathsf{pk}} {}_{i=1}^{d} u_{i,j_i}\}_{(j_i) \in [\mu]^d}$ to the receiver. If $\|\mathbf{x} - \mathbf{y}\|_\infty \leq \delta$, there exists a unique ciphertext encrypting 0 in this set. However, this method increases the sender's communication and computation complexity by a factor of $\mu^d = (\log \delta)^d$, which may be prohibitively large.

To address this, we modify the sender to transmit $\{u_{i,j} \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i), \mathsf{H}(\sum_{i=1}^d a_i)\}$ with random a_i. This optimization increases the sender's communication by only μ, shifting the computational burden to the receiver, who computes $\sum_{i=1}^d \mathsf{Dec}_{\mathsf{sk}}(u_{i,j_i} \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i))$ for all j_i combinations to verify $\|\mathbf{x} - \mathbf{y}\|_\infty \leq \delta$. Security holds because only one value in $\{u_{i,j} \boxplus_{\mathsf{pk}} a_i\}_{j \in [\mu]}$ encrypts a_i for each i, while others encrypt random values.

2.2 Fuzzy Matching for L_p Distance

In the fuzzy matching protocol for L_p distance proposed by Baarsen and Pu [vBP24], on each dimension the receiver encodes all possible values $x_i + j$ (for $j \in [-\delta, \delta]$) in the interval

into $r_{i,j}||r_{i,j}^s g^{|j|^p}$ where $r_{i,j} \leftarrow_\$ \mathbb{G}$. The sender uses its point y_i to decode the OKVS and then homomorphically sums up $|x_i - y_i|^p$ for every $i \in [d]$. In the end, the sender gets an encryption of $(\|\mathbf{x}-\mathbf{y}\|_p)^p$, then it can rerandomize both the ciphertext and plaintext, and send back the result. However, once the receiver encodes the prefix representations of the interval $[x_i - \delta, x_i + \delta]$ into an OKVS, Baarsen and Pu [vBP24]'s approach for L_p distance cannot be straightforwardly applied, since the distance between a prefix representing an interval and a point is undetermined.

Our key observation is that for each dimension it holds that $|y_i - x_i| = |y_i - y'| + |y' - x_i|$ if $y_i \geq y' \geq x_i$, and if y' is a value that can be computed by both parties, precisely the left or right endpoint of the interval generated by the prefix representation, then $|x_i - y_i|^p$ can be evaluated. More precisely, we divide the interval $[x_i - \delta, x_i + \delta]$ into two subintervals $[x_i - \delta, x_i]$ and $[x_i + 1, x_i + \delta]$, and consider these two cases separately.

Let us take the case $[x_i - \delta, x_i]$ for example. The receiver decomposes $[x_i - \delta, x_i]$ as prefix representations $\{x_{i,j}\}$ and encodes each prefix $x_{i,j}$ into the encryption of $|\mathsf{UpBound}(x_{i,j}) - x_i|^t$ for $t \in [p]$, where $\mathsf{UpBound}(x_{i,j})$ denotes the right endpoint of the interval $\mathsf{Interval}(x_{i,j})$. Suppose that x_{i,j_i} is one of the prefixes of y_i. Set $y' = \mathsf{UpBound}(x_{i,j_i})$, which means y' can be computed by both parties. Since the sender can locally compute the encryption of $|y_i - y'|^t$ for any $t \in [p]$, and decode x_{i,j_i} to retrieve $\mathsf{Enc}_{\mathsf{pk}}(|y' - x_i|^t)$ for $t \in [p]$, and given that $|y_i - x_i| = |y_i - y'| + |y' - x_i|$, the sender can homomorphically compute the encryption of $|y_i - x_i|^p$. That's because $|x_i - y_i|^p = \sum_{t=0}^{p} \binom{p}{t} |y_i - y'|^{p-t} |y' - x_i|^t$, and $\mathsf{Enc}_{\mathsf{pk}}\big(|x_i - y_i|^p\big) = \boxplus_{\mathsf{pk}}{}_{t=1}^{p} \big(\binom{p}{t} |y_i - y'|^{p-t} \boxtimes_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(|y' - x_i|^t)\big) \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(|y_i - y'|^p)$.

The sender sets $a_i = \mathsf{Enc}_{\mathsf{pk}}(|y' - x_i|^t)$ and decodes with all prefixes of y_i to obtain $\{u_{i,j}\}_{j \in [\mu]}$. If $|y_i - x_i| \leq \delta$, then $\exists j_i' \in [\mu]$, such that $|x_i - y_i|^p = \mathsf{Dec}_{\mathsf{sk}}(u_{i,j_i'})$. The sender sends $\{u_{i,j} \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i)\}_{j \in [\mu]}$ to the receiver, where $a_i \leftarrow_\$ \mathbb{Z}_q$.

Upon receiving $\{u_{i,j} \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i)\}$ from the sender, the receiver decrypts it and iterates over all possible $(j_1, \ldots, j_d) \in [\mu]^d$ to compute $\sum_{i=1}^{d} \big(\mathsf{Dec}_{\mathsf{sk}}(u_{i,j_i}) + a_i\big)$. If $\|\mathbf{x}-\mathbf{y}\|_p \leq \delta$, the receiver can obtain $(\|\mathbf{x}-\mathbf{y}\|_p)^p + \sum_{i=1}^{d} a_i$. The receiver can then verify whether $\|\mathbf{x}-\mathbf{y}\|_p \leq \delta$ by checking if $(\|\mathbf{x}-\mathbf{y}\|_p)^p + \sum_{i=1}^{d} a_i \in [\delta^p, \delta^p + \sum_{i=1}^{d} a_i]$, which in turn is a fuzzy matching protocol for L_∞ distance in dimension 1.

2.3 Fuzzy PSI-CA in Low Dimension

For the case where points are located in low-dimensional space, we use the spatial hash technique developed by Baarsen and Pu [vBP24] to construct fuzzy PSI-CA protocols (i.e., protocols that only let the receiver learn the cardinality of the intersection) for L_∞ and L_p distances, where the receiver's points are at least 4δ and $4\delta(d^{1/p} + 1)$ apart, respectively. However, the spatial hash introduces an $O(2^d)$ complexity factor, which becomes prohibitive for large dimensions d.

2.4 Fuzzy PSI-CA in High Dimension

For the high-dimensional case, our main goal is to circumvent both the $O((\log \delta)^d)$ complexity factor from the receiver enumerating all prefixes across dimensions, and the $O(2^d)$ complexity factor introduced by spatial hashing. We construct fuzzy PSI-CA protocols for large dimensions d with communication and computation complexity scaling linearly with $\log \delta$, dimension d, and the set sizes N_r and N_s of the receiver and sender. Our constructions are based on two main building blocks: Private Index Search (PIS) and Fuzzy Mapping.

2.4.1 Private Index Search.

Private index search is a two-party protocol where the sender's input is $b \in \mathbb{Z}$ and the receiver's input is a set $S = (a_i)_{i \in [0, n-1]} \subset \mathbb{Z}$. It allows the receiver to learn the index i if

$b = a_i \in S$; otherwise, receiver learns a random index.

In our fuzzy PSI-CA protocol for L_∞ distance in low dimension, the receiver's computation complexity scales linearly with $(\log \delta)^d$. This factor comes from the enumeration of j_i for $j_i = 1, \cdots, \mu$ on dimension i to check if there exists a $j_i' \in [\mu]$ such that $\mathsf{Dec_{sk}}(u_{i,j_i'}) + a_i = a_i$, where $\mu = \log \delta$. We aim to use PIS to overcome the factor $(\log \delta)^d$. On input $\{\mathsf{Dec_{sk}}(u_{i,j_i}) + a_i\}_{j_i \in [\mu]}$ from the receiver and a_i from the sender, our PIS functionality allows the receiver to learn the index j_i' such that $\mathsf{Dec_{sk}}(u_{i,j_i'}) + a_i = a_i$ on dimension i. By invoking PIS functionality on all d dimensions, the receiver then checks if $\sum_{i=1}^{d}(\mathsf{Dec_{sk}}(u_{i,j_i'}) + a_i) = \sum_{i=1}^{d} a_i$.

We construct a PIS protocol based on a secret-shared private equality test (ssPET) protocol with complexity linear in $\log \delta$. Here, we require $n = 2^l$ for some $l \in \mathbb{Z}$. Initially, an ssPET protocol is invoked with every element a_i in set S and the element b as inputs. As a result, two parties obtain shares s_i^* and t_i^* respectively, such that $s_i^* \oplus t_i^* = 1$ if $b = a_i$, otherwise $s_i^* \oplus t_i^* = 0$. By computing $s_0 = \oplus_{i=0}^{n-1} s_i^*$ and $t_0 = \oplus_{i=0}^{n-1} t_i^*$, we can obtain the secret share of whether $b \in S$. If $b = a_{\mathsf{idx}}$, we represent the index idx as a binary string and determine the shares of each bit of idx sequentially. Specifically, we set $S_k = \{i = \overline{x_l \cdots x_{k+1} 1 x_{k-1} \cdots x_1} : x_1, \ldots, x_l \in \{0,1\}\}$, and by computing $s_k = \oplus_{i \in S_k} s_i^*$ and $t_k = \oplus_{i \in S_k} t_i^*$, we can obtain the secret share of whether $\mathsf{idx} \in S_k$. This allows us to determine the k-th bit of idx, namely, $\mathsf{idx}_k = s_k \oplus t_k$. In the end, if $b \in S$, the receiver and sender obtain shares $s = (s_1, \cdots, s_l)$ and $t = (t_1, \cdots, t_l)$. If $b \notin S$, $s \oplus t = 0$. Finally, the receiver and sender run an OT protocol with inputs (q_0, q_1) from the sender and $c = s_0$ from the receiver, where $q_0 = (t_0 \oplus 1) \cdot r \oplus t$, $q_1 = r \oplus q_0$, and r is a random bit string of length l. The functionality $\mathcal{F}_{\mathsf{OT}}$ returns h to the receiver, who then outputs $\mathsf{idx} = h \oplus s$.

2.4.2 Fuzzy Mapping.

Fuzzy mapping is a notion introduced by Gao et al. [GQL+25] as a generalization of spatial hash. Gao et al. [GQL+25] construct a fuzzy mapping for L_∞ distance in high-dimensional cases, which incurs complexity scaling linearly with δ, dimension d, and the set size.

Their high-level intuition is that for each point $\mathbf{x}$, the receiver assigns a random value r_i to each projection of the L_∞ ball in every dimension and sums them to obtain the identifier $\mathsf{id}(\mathbf{x}) = \sum_{i=1}^{d} r_i$. If two projections intersect, they are assigned the same value. Their construction assumes the L_∞ ball $\{\mathbf{x}' : \|\mathbf{x}' - \mathbf{x}\|_\infty \le \delta\}$ has disjoint projections in some dimension for every $\mathbf{x}$. This assumption ensures each identifier $\mathsf{id}_{\mathbf{x}}$ can be viewed as independently sampled. The receiver sends an OKVS for each dimension to the sender, with keys being all points in $[x_i - \delta, x_i + \delta]$ and values being encryptions of r_i. When $|y_i - x_i| \le \delta$, the sender can homomorphically compute identifier encryptions and recover them using random values as one-time pads.

To reduce complexity to scale linearly with $\log \delta$, we propose a new semi-honest secure fuzzy mapping protocol that encodes interval $[x_i - \delta, x_i + \delta]$ prefixes (rather than all points) into an OKVS using spatial additive sharing.

2.5 Further Optimization

We observe that the complexity of all of our protocols are dependent on the parameters μ_δ and ω_δ, where ω_δ is the upper bound of the number of key-value pairs, and μ_δ is the number of prefixes that the Sender decodes on in each dimension. Once we reduce the number of prefixes of the sender's input y_i on each dimension i, we must increase the size of the OKVS to make sure that the prefix sets are intersected. We give a method to trade off between two parameters μ_δ and ω_δ.

Let the sender compute prefixes of a specific length in a set U. Let $\bar{\omega}(t, U)$ denote the size of the prefix set obtained by decomposing intervals with size t. According to our analysis, we list different choices of the parameters t, U and $\bar{\omega}(t, U)$. After applying this

optimization to our protocols, the complexities scales linearly in terms of $\bar{\omega}(t, U)$ and $|U|$ instead of $\log \delta$.

3 Preliminaries

Denote the computational security parameter as $\lambda \in \mathbb{N}$, the statistical security parameter as $\kappa \in \mathbb{N}$. We use $\approx_c$ to denote computational indistinguishability and $\approx_s$ to denote statistical indistinguishability. The notation $[n]$ indicates the set $\{1, \cdots, n\}$ and $[a : b]$ the set $\{a, a+1, \cdots, b-1, b\}$. Denote $\mathcal{U}$ as the space $\{0,1\}^u$ for some $u \in \mathbb{Z}$. In the following we give definitions of private set membership and fuzzy mapping. All protocols involved in this paper are secure against semi-honest and computationally bounded adversaries. And the definitions of security model, oblivious key-value store (OKVS), and additive homomorphic encryption (AHE) are given in Appendix A.

3.1 Prefixes of L_∞ Ball

For a binary string $x = \overline{x_l x_{l-1} \cdots x_1}$ of length l and $x_i \in \{0,1\}$, we define the following notions:

1. $\mathsf{Prefix}_l(x, j) := \overline{x_l x_{l-1} \cdots x_j}$
2. $\mathsf{Interval}_l(\overline{x_l x_{l-1} \cdots x_j}) := \{\overline{x_l x_{l-1} \cdots x_j x^*} : x^* \in \{0,1\}^{j-1}\}$
3. $\mathsf{UpBound}_l(\overline{x_l x_{l-1} \cdots x_j}) := \overline{x_l x_{l-1} \cdots x_j 1 \cdots 1}$
4. $\mathsf{LowBound}_l(\overline{x_l x_{l-1} \cdots x_j}) := \overline{x_l x_{l-1} \cdots x_j 0 \cdots 0}$

$\mathsf{Prefix}_l(x, j)$ denotes the left $l-j$ bits of the binary representation of x. Let $\mathsf{Interval}_l(\overline{x_l x_{l-1} \cdots x_j})$ be the integer interval generated by $\overline{x_l x_{l-1} \cdots x_j}$, which is all l-length bit string with prefix $\overline{x_l x_{l-1} \cdots x_j}$. $\mathsf{UpBound}_l(\overline{x_l x_{l-1} \cdots x_j})$ and $\mathsf{LowBound}_l(\overline{x_l x_{l-1} \cdots x_j})$ denote the right endpoint and left endpoint of the interval $\mathsf{Interval}_l(\overline{x_l x_{l-1} \cdots x_j})$ respectively.

In this paper, we will decompose any interval S into a disjoint union of prefix sets. We give the following definition:

Definition 1 ($\mathsf{Decompose}$)**.** We define $\mathsf{Decompose}(S) = \{b_j\}_{j \in \omega'}$, if it follows the following properties:

1. $S = \bigsqcup_{j \in \omega'} \mathsf{Interval}_l(b_j)$.
2. for any j, if bit string b' is a prefix of b_j, $\mathsf{Interval}_l(b') \nsubseteq S$.

In [GGM24] the decomposition of an interval is implicitly defined, but no concrete implementation is given. In Appendix 8 in Figure 21, we give an algorithm for $\mathsf{Decompose}()$, which achieves complexity logarithmic on interval size. For any interval S of a fixed size, the set size $|\mathsf{Decompose}(S)|$ has a upper bound, and the length of the prefixes has a lower bound. We give the following lemma.

Lemma 1 ([CFR23])**.** *For any interval $S \subset \{0,1\}^l$ of a fixed size less than h, define $\mathsf{Decompose}(S) = \{b_i\}$, then $|\{b_i\}|$ has a upper bound $2\lfloor \log h \rfloor$, and the length of all prefix is larger than $l - \lfloor \log h \rfloor$.*

3.2 Secret Shared Private Equality Test

Secret shared private equality test (ssPET) is a two party functionality that takes as inputs an element x from P_0 and an element y from P_1. The functionality returns $b_0, b_1 \in \{0,1\}$ to both parties, where $b_0 \oplus b_1 = 1$ if $x = y$; else $b_0 \oplus b_1 = 0$. The functionality is described in Figure 1.

Parameter: An integer u, denote $\mathcal{U} := \{0,1\}^u$.
Functionality: Upon receiving $x \in \mathcal{U}$ from P_0 and $y \in \mathcal{U}$ from P_1. $\mathcal{F}_{\mathsf{ssPET}}$ returns $b_0 \in \{0,1\}$ and $b_1 \in \{0,1\}$ to P_0 and P_1, such that $b_0 \oplus b_1 = 1$ if $x = y$, else $b_0 \oplus b_1 = 0$.

Figure 1: Functionality $\mathcal{F}_{\mathsf{ssPET}}$

3.3 Fuzzy Mapping

Fuzzy mapping (Fmap) defined by Gao et al. [GQL+25] enables two parties to map their points to a set of identifiers. If two points from the Sender and the Receiver are close enough, then they will share a same identifier. The formal definition is as follows.

Definition 2. A fuzzy mapping Π is a two party protocol with inputs $X = (\mathbf{x}_i)_{i\in[n]} \in \mathcal{U}^{d\times n}$ from the Receiver and $Y = (\mathbf{y}_j)_{j\in[m]} \in \mathcal{U}^{d\times m}$ from the Sender. Let the public parameters be threshold δ and distance function $\mathsf{dist}()$. The protocol outputs $\mathsf{ID}(X) = \{\mathsf{ID}(\mathbf{x}_i)\}_{i\in[n]}$ and $\mathsf{ID}(Y) = \{\mathsf{ID}(\mathbf{y}_j)\}_{j\in[m]}$ to the Receiver and the Sender respectively. The outputs satisfy the following properties:

- **Correctness.** For any two points $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, if $\mathsf{dist}(\mathbf{x},\mathbf{y}) \le \delta$, then $\mathsf{ID}(\mathbf{x}) \cap \mathsf{ID}(\mathbf{y}) \neq \emptyset$.
- **Distinctiveness.** For the Receiver, it holds that $Pr(\exists i, i' \in [n], \text{ s.t. } i \neq i' \text{ and } \mathsf{ID}(\mathbf{x}_i) \cap \mathsf{ID}(\mathbf{x}_{i'}) \neq \emptyset) = negl(\lambda)$.
- **Security.** For corrupted Sender $\mathcal{S}$, for any $X_1, X_2 \in \mathcal{U}^{d\times n}$, $Y \in \mathcal{U}^{d\times m}$, it hols that $\mathsf{Real}_{\Pi,\mathcal{S}}(\kappa,\lambda,X_1,Y) \approx_c \mathsf{Real}_{\Pi,\mathcal{S}}(\kappa,\lambda,X_2,Y)$.

  For corrupted Receiver $\mathcal{R}$, for any $X \in \mathcal{U}^{d\times n}$, $Y_1, Y_2 \in \mathcal{U}^{d\times m}$, it hols that $\mathsf{Real}_{\Pi,\mathcal{R}}(\kappa,\lambda,X,Y_1) \approx_c \mathsf{Real}_{\Pi,\mathcal{R}}(\kappa,\lambda,X,Y_2)$.

4 Functionalities

4.1 Functionality of Fuzzy Matching

We define the functionality of fuzzy matching, denoted as $\mathcal{F}_{\mathsf{FMat}}$, between two points with a distance function $\mathsf{dist}()$ in Figure 2.

Parameter: Dimension d, distance δ, a distance function $\mathsf{dist}()$.
Functionality: Upon receiving $\mathbf{x} \in \mathbb{Z}^d$ from Receiver and $\mathbf{y} \in \mathbb{Z}^d$ from Sender. If $\mathsf{dist}(\mathbf{x},\mathbf{y}) \le \delta$, $\mathcal{F}_{\mathsf{FMat}}$ returns 1 to Receiver, else returns 0 to Receiver.

Figure 2: Functionality $\mathcal{F}_{\mathsf{FMat}}$

In the rest of this paper, we focus on two distance metrics:

- The L_∞ distance between vectors $\mathbf{x}$ and $\mathbf{y}$ is given by: $\|\mathbf{x}-\mathbf{y}\|_\infty = \max_{i\in[d]} |x_i - y_i|$.
- The L_p distance between vectors $\mathbf{x}$ and $\mathbf{y}$ is given by: $\|\mathbf{x}-\mathbf{y}\|_p = (\sum_{i=1}^d |x_i - y_i|^p)^{\frac{1}{p}}$, where $\mathbf{x} = (x_1,\ldots,x_d)$ and $\mathbf{y} = (y_1,\ldots,y_d)$.

4.2 Functionality of Fuzzy PSI (-CA)

We describe the functionalities of fuzzy PSI and fuzzy PSI cardinality (PSI-CA), parameterized by the distance function $\mathsf{dist}()$, denoted as $\mathcal{F}^{\mathsf{dist}}_{\mathsf{FPSI}}$ and $\mathcal{F}^{\mathsf{dist}}_{\mathsf{FPSI-CA}}$ respectively in Figure 3. With the Receiver holding N_r balls of radius δ and dimension d and the Sender holding a set of N_s points as inputs, fuzzy PSI allows the receiver to learn which of the sender's points lie within one of their balls, while fuzzy PSI-CA allows the receiver only to learn how many sender's points lie inside the receiver's balls, but not the sender's exact points.

Parameter: Dimension d, distance δ, a distance function $\mathsf{dist}()$.
Functionality: Upon receiving $X \in \mathbb{Z}^{N_r \times d}$ from Receiver and $Y \in \mathbb{Z}^{N_s \times d}$ from Sender. The functionality proceeds as follows:
Return $\#\{\mathbf{y} \in Y : \mathbf{x} \in X, \mathsf{dist}(\mathbf{x}, \mathbf{y}) \leq \delta\}$ to Receiver.($\mathcal{F}^{\mathsf{dist}}_{\mathsf{FPSI-CA}}$)
Return $\{\mathbf{y} \in Y : \mathbf{x} \in X, \mathsf{dist}(\mathbf{x}, \mathbf{y}) \leq \delta\}$ to Receiver.($\mathcal{F}^{\mathsf{dist}}_{\mathsf{FPSI}}$)

Figure 3: Functionalities $\mathcal{F}^{\mathsf{dist}}_{\mathsf{FPSI}}$ and $\mathcal{F}^{\mathsf{dist}}_{\mathsf{FPSI-CA}}$

5 Our Constructions for Fuzzy Matching

5.1 Fuzzy Matching for L_∞ Distance

We start by presenting a fuzzy matching protocol for two points with L_∞ distance. Intuitively, the receiver encodes the prefixes representing the interval $[x_i - \delta, x_i + \delta]$ into an OKVS for each dimension $i \in [d]$, where the values are set as the ciphertexts of 0, i.e., $\mathsf{Enc}_{\mathsf{pk}}(0)$. The sender must decode all possible prefixes of y_i for each dimension. This guarantees that the sender will decode a ciphertext of 0 if its i-th dimension lies in the interval, and will obtain a ciphertext of a random plaintext otherwise. Our protocol achieves a complexity scaling logarithmically in the threshold distance δ.

Before providing the detailed fuzzy matching protocol, let us define two algorithms for L_∞ distance: $\mathsf{GetList}_\infty()$ and $\mathsf{GetValue}_\infty()$ as shown in Figure 4. Algorithm $\mathsf{GetList}_\infty()$ generates the list of key-value pairs with the hashes of prefix representations of interval $[x_i - \delta, x_i + \delta]$ as keys and the ciphertexts of 0 as values. Algorithm $\mathsf{GetValue}_\infty()$ enables a party to decode all possible prefixes of y_i and obtain messages that can be verified using the secret key sk to check if $|x_i - y_i| \leq \delta$.

Let μ_δ denote the number of prefixes the Sender needs to process in each dimension, and ω_δ denote the upper bound of $|\mathsf{Decompose}(\{x'_i : \|x'_i - x_i\|_\infty \leq \delta\})|$. For L_∞ distance, we set $\mu_\delta = \lfloor \log \delta \rfloor + 1$ and $\omega_\delta = 2\lfloor \log(2\delta + 1) \rfloor$ according to Lemma 1, since the size of interval $[x_i - \delta, x_i + \delta]$ is $2\delta + 1$.

We present our fuzzy matching protocol for L_∞ distance in Figure 5. In Figures 4 and 5, H_{γ_1} and H_{γ_2} are universal hash functions with $\gamma_1 = \kappa + 2\log(d \log \delta)$ and $\gamma_2 = \kappa + d \cdot \log \log \delta$.

The communication complexity of our protocol in Figure 5 is $O(\omega_\delta d\lambda + \gamma_2)$. The computational complexity is $O(\mu_\delta d + \mu_\delta^d)$ for the Receiver and $O(\mu_\delta d)$ for the Sender. Here $\mu_\delta = \lfloor \log \delta \rfloor + 1$, $\omega_\delta = 2\lfloor \log(2\delta + 1) \rfloor$. The proofs of correctness and security are presented in Appendix C.1.

5.2 Fuzzy Matching for L_p Distance

5.2.1 Fuzzy Matching for Intervals.

Here, we discuss a specific case of fuzzy matching, namely, fuzzy matching in dimension 1 for the L_∞ distance, which is referred to the Fuzzy Matching for Intervals (IFMat). For

$\mathsf{GetList}_\infty(\mathsf{pk}, \delta, \mathbf{x}, \Delta)$

1. For each $i \in [d]$:
   - Initiate $\mathsf{list}_i = \emptyset$. Denote $\{x_{i,j}\}_{j\in[0,\omega_i]} = \mathsf{Decompose}\ (\{x' : ||x' - x_i||_\infty \leq \delta\})$. For each $j' \in [0, \omega_i]$: set $\mathsf{list}_i \leftarrow \mathsf{list}_i \cup \{(\mathbf{H}_{\gamma_1}(\Delta||x_{i,j'}), \mathsf{Enc}_{\mathsf{pk}}(0))\}$.
2. Pad list_i with random items to size ω_δ and output.

$\mathsf{GetValue}_\infty(\mathsf{pk}, \delta, \mathbf{y}, \Delta, \mathbf{E})$

1. Denote $\mathbf{E} = \{E_i\}_{i\in[d]}$. Compute $y^*_{i,j} = \mathsf{Prefix}(y_i, j)$, $j \in [0, \mu_\delta]$.
2. For each $i \in [d]$:
   - Sample $a_i \leftarrow_\$ \mathcal{P}$. For each $j \in [0, \mu_\delta]$:
     - Set $u_{i,j} \leftarrow \mathsf{Decode}(E_i, \mathsf{H}_{\gamma_1}(\Delta||y^*_{i,j}))$.
     - Set $u^*_{i,j} = u_{i,j} \boxplus_{pk} \mathsf{Enc}_{pk}(a_i)$.
3. Output $(\{u^*_{i,j}\}_{j\in[0,\mu_\delta],i\in[d]}, \{a_i\}_{i\in[d]})$.

Figure 4: Algorithm $\mathsf{GetList}_\infty()$ and $\mathsf{GetValue}_\infty()$

completeness, we describe the functionality of IFMat, denoted as $\mathcal{F}_{\mathsf{IFMat}}$, in Figure 6.

There are multiple ways to achieve the functionality $\mathcal{F}_{\mathsf{IFMat}}$. We apply the method proposed by Chakraborti et al. [CFR23], which involves using a Private Set Intersection with Cardinality (PSI-CA) protocol on the collections of representative bit strings corresponding to their binary representations. This results in an IFMat protocol with linear complexity in $\log \delta$.

More precisely, the Receiver decomposes the interval as $X = \mathsf{Decompose}([x - \delta, x + \delta])$, while the Sender computes $Y = \{\mathsf{Prefix}(y, j) : j \in [0, \lfloor \log \delta \rfloor + 1]\}$. The two parties then execute a PSI-CA protocol with inputs X from the Receiver and Y from the Sender. If $y \in [x - \delta, x + \delta]$, then $|\mathsf{Decompose}([x - \delta, x + \delta]) \cap Y| = 1$. The Receiver outputs 1; otherwise, outputs 0. The communication complexity of this IFMat is $O(\log \delta \cdot \lambda)$, and the computation complexity is $O(\log \delta)$ for the Receiver and $O(\log \delta)$ for the Sender.

5.2.2 Construction.

To achieve a fuzzy matching for L_p distance, we aim for the Sender to retrieve the encryption of $|y_i - x_i|^p$ on each dimension when decoding y_i if $y_i \in [x_i - \delta, x_i + \delta]$. We observe that for each dimension it holds that $|y_i - x_i| = |y_i - y'| + |y' - x_i|$ if $y_i \geq y' \geq x_i$. We set y' to be a value that can be computed by both parties, specifically the left endpoint of the interval generated by the prefix in $\mathsf{Decompose}([x_i - \delta, x_i + \delta])$. For the case $y_i \leq x_i$, y' is set to be the right endpoint. Therefore, the Receiver partitions the interval $[x_i - \delta, x_i + \delta]$ into two parts $[x_i - \delta, x_i]$ and $[x_i + 1, x_i + \delta]$, and handles them separately.

For the sub-interval $[x_i - \delta, x_i]$, the Receiver decomposes it into intervals presented by prefixes and encodes each prefix $x_{i,j}$ into the encryption of $|\mathsf{UpBound}(x_{i,j}) - x_i|^t$ for $t \in [p]$, as shown in algorithm $\mathsf{GetList}_p$ in Figure 7, where $\mathsf{UpBound}(x_{i,j})$ denotes the right endpoint of $\mathsf{Interval}(x_{i,j})$. Suppose that $x_{i,j_i} \in \mathsf{Decompose}([x_i - \delta, x_i])$ is a prefix of y_i. Setting $y' = \mathsf{UpBound}(x_{i,j_i})$, which implies y' can be computed by both parties. Given that the Sender can locally compute the encryption of $|y_i - y'|^t$ for any $t \in [p]$, and decode x_{i,j_i} to obtain $\mathsf{Enc}_{\mathsf{pk}}(|y' - x_i|^t)$ for $t \in [p]$, the Sender can homomorphically compute the

Parameter setting: Receiver $\mathcal{R}$ holds a point $\mathbf{x} \in \mathbb{Z}^d$. Sender $\mathcal{S}$ holds a point $\mathbf{y} \in \mathbb{Z}^d$. A distance threshold δ. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1. $\mathcal{R}$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $\mathcal{R}$ sets $\{\mathsf{list}_i\}_{i\in[d]} \leftarrow \mathsf{GetList}_\infty(\mathsf{pk}, \delta, \mathbf{x}, 0^\kappa)$. For each $i \in [d]$, $\mathcal{R}$ sets $\mathbf{E} = \{E_i\}_{i\in[d]}$, where $E_i \leftarrow \mathsf{Encode}(\mathsf{list}_i)$.
3. $\mathcal{R}$ sends $(\mathsf{pk}, \mathbf{E})$ to $\mathcal{S}$.
4. $\mathcal{S}$ computes $(\{u_{i,j}\}, \{a_i\}) \leftarrow \mathsf{GetValue}_\infty(\mathsf{pk}, \delta, \mathbf{y}, 0^\kappa, \mathbf{E})$.
5. $\mathcal{S}$ shuffles $\{u_{i,j}\}$ with j, and sends to $\mathcal{R}$ the shuffled set and $r = \mathsf{H}_{\gamma_2}(\sum_{i=1}^d a_i)$.
6. If $\exists (j_i)_{i\in[d]} \in [0, \mu_\delta]^d$, s.t. $\mathbf{H}_{\gamma_2}(\sum_{i=1}^d \mathsf{Dec}_{\mathsf{sk}}(u_{i,j_i})) = r$, then $\mathcal{R}$ outputs 1, else outputs 0.

Figure 5: Fuzzy matching for L_∞ distance

Parameter: Threshold distance δ.
Functionality: Upon receiving $x \in \mathbb{Z}$ from Receiver and $y \in \mathbb{Z}$ from Sender. $\mathcal{F}_{\mathsf{IFMat}}$ returns 1 to Receiver if $|x - y| \le \delta$, else 0.

Figure 6: Functionality $\mathcal{F}_{\mathsf{IFMat}}$

encryption of $|y_i - x_i|^p = \sum_{t=0}^p \binom{p}{t} |y_i - y'|^{p-t} |y' - x_i|^t$, as shown in algorithm $\mathsf{GetValue}_p$ in Figure 7.

After the Sender returns the encryption of $|y_i - x_i|^p + \mathsf{mask}_i$ to the Receiver, the two parties run the IFMat protocol with the inputs $\sum_{i=1}^d (|y_i - x_i|^p + \mathsf{mask}_i)$ from the Receiver and $\sum_{i=1}^d \mathsf{mask}_i + \frac{1}{2}\delta^p$ from the Sender, where the threshold is $\frac{1}{2}\delta^p$, thereby allowing the Receiver to determine whether $(\|\mathbf{x} - \mathbf{y}\|_p)^p \le \delta^p$.

Our fuzzy matching protocol for L_p distance is shown in Figure 8. Here $\mu_\delta = \lfloor \log \delta \rfloor$ and $\omega_\delta = 2\lfloor \log(\delta + 1) \rfloor$ according to Lemma 1, since we deal with two sub-intervals separately, i.e., $[x_i - \delta, x_i]$ and $[x_i + 1, x_i + \delta]$, and the sizes of two sub-intervals are $\delta + 1$ and δ respectively. H_{γ_1} and H_{γ_2} are universal hash functions, where $\gamma_1 = \kappa + 2\log(\log \delta \cdot d)$ and $\gamma_2 = \kappa + p \log \delta$.

In our construction of Figure 8, the functionality $\mathcal{F}_{\mathsf{IFMat}}$ is invoked. We use the protocol shown in Section 5.2.1 to achieve it. Therefore, the communication complexity of our protocol in Figure 8 is $O(2\omega_\delta pd\lambda + 2\mu_\delta\, d\lambda + (2\mu_\delta)^d \log \delta\, p\lambda)$. The computational complexity is $O(\omega_\delta pd + (2\mu_\delta)^d \log \delta\, p)$ for the Receiver, and $O(\mu_\delta pd + (2\mu_\delta)^d \log \delta\, p)$ for the Sender. Here $\mu_\delta = \lfloor \log \delta \rfloor$ and $\omega_\delta = 2\lfloor \log(\delta + 1) \rfloor$. The proofs of correctness and security are given in Appendix C.2.

6 Fuzzy PSI (-CA) Protocol in Low Dimension

We first cope with the case that points are located in a low-dimension space. We use the spatial hash technique developed by Baarsen et al. [vBP24] to get around the quadratic increase in computational and communicative overheads, which incurs only a $O(2^d)$ factor to the overall communication and receiver's computational complexity. In this section, we

$\mathsf{GetList}_p(\mathsf{pk}, \delta, \mathbf{x}, \Delta)$

1. For each $i \in [d]$: Set $\{x_{0,i,j}\}_{j\in[0,\omega_{0,i}]} = \mathsf{Decompose}(\{x'_i : x'_i \le x_i, |x'_i - x_i| \le \delta\})$, $\{x_{1,i,j}\}_{j\in[0,\omega_{1,i}]} = \mathsf{Decompose}(\{x'_i : x'_i > x_i, |x'_i - x_i| \le \delta\})$.
2. For each $\sigma \in \{0,1\}$, $i \in [d]$:
   - Initiate $\mathsf{list}_{\sigma,i} = \emptyset$. For each $j' \in [0, \omega_{\sigma,i}]$:
     * If $\sigma = 0$, $\mathbf{x}^* = \mathsf{UpBound}(x_{\sigma,i,j'})$. Else If $\sigma = 1$, $x^* = \mathsf{LowBound}(x_{\sigma,i,j'})$.
     * Set $\mathsf{list}_{\sigma,i} \leftarrow \mathsf{list}_{\sigma,i} \cup (\mathbf{H}_{\gamma_1}(\Delta||\sigma||x_{\sigma,i,j'}), \mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|)||\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^2)||\cdots||\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^p))$.
3. Pad $\mathsf{list}_{\sigma,i}$ with dummy random items to size ω_δ, and output.

$\mathsf{GetValue}_p(\mathsf{pk}, \delta, \mathbf{y}, \Delta, \mathbf{E})$

1. Denote $y_{i,j} = \mathsf{Prefix}(y_i, j)$, $j \in [0, \mu_\delta]$.
2. For each $i \in [d]$:
   - Sample $a_i \leftarrow_\$ \mathcal{P}$. For each $\sigma \in \{0,1\}$, $j \in [0, \mu_\delta]$:
     * Set $(u^*_{\sigma,i,j,t})_{t\in[p]} \leftarrow \mathsf{Decode}(E_{\sigma,i}, \mathsf{H}_{\gamma_1}(\Delta||\sigma||y_{i,j}))$.
     * Set $e = |y_i - \mathsf{UpBound}(y_{i,j})|$ if $\sigma = 0$, else $e = |y_i - \mathsf{LowBound}(y_{i,j})|$.
     * Compute $u_{\sigma,i,j} = \boxplus_{\mathsf{pk}}{}_{t=1}^{p}(\binom{p}{t} e^{p-t} \boxtimes_{\mathsf{pk}} u^*_{\sigma,i,j,t}) \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i + e^p)$.
3. Output $(\{u_{\sigma,i,j}\}_{i\in[d],\sigma\in\{0,1\},j\in[0,\mu_\delta]}, \{a_i\}_{i\in[d]})$.

Figure 7: Algorithm $\mathsf{GetList}_p$ and $\mathsf{GetValue}_p$

provide fuzzy PSI-CA protocols for L_∞ and L_p distance respectively, then show how to extend PSI-CA to PSI functionality.

6.1 Fuzzy PSI-CA Protocol for L_∞ Distance

The high-level intuition is that the Receiver encodes each of its balls in at most 2^d different cells that it intersects with, , and then the Sender checks the unique cell where its point is located, according to Lemma 4. Since for each cell, there is only one ball intersecting with it, thus any point inside the ball must lie in one of these cells and result in distinct key.

We provide the detailed protocol in Figure 17 in Appendix B.1, where the Receiver's points are at least 4δ apart from each other.

The communication complexity of our protocol in Figure 17 is $O(\omega_\delta 2^d dN_r\lambda + \mu_\delta dN_s\lambda + N_s\kappa)$. The computational complexity is $O(\omega_\delta 2^d dN_r + (\mu_\delta)^d N_s)$ for the Receiver and $O(\mu_\delta dN_s)$ for the Sender. Here, $\mu_\delta = \lfloor \log \delta \rfloor + 1$ and $\omega_\delta = 2\lfloor \log(2\delta + 1) \rfloor$.

6.2 Fuzzy PSI-CA Protocol for L_p Distance

Analogously, to realize fuzzy PSI-CA for L_p distance, we apply spatial hash to our fuzzy matching for L_p distance. We provide the detailed protocol in Figure 18 in Appendix B.2, where the Receiver's points are at least $4\delta(d^{1/p} + 1)$ apart from each other.

The communication complexity of our protocol in Figure 18 is $O(2p\omega_\delta 2^d dN_r\lambda + 2\mu_\delta dN_s\lambda + (\mu_\delta)^d \log \delta\, pN_s\lambda)$. The computational complexity is $O(p\omega_\delta 2^d dN_r + (2\mu_\delta)^d \log \delta\, pN_s)$

Parameter setting: $p \in \mathbb{N}$. Receiver $\mathcal{R}$ holds $\mathbf{x}$, Sender $\mathcal{S}$ holds $\mathbf{y}$, $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^d$. A distance threshold δ. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1. $\mathcal{R}$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $\mathcal{R}$ sets $\{\mathsf{list}'_{\sigma,i}\}_{\sigma\in\{0,1\},i\in[d]} \leftarrow \mathsf{GetList}_p(\mathsf{pk}, \delta, \mathbf{x}, 0^\kappa)$. For each $i \in [d]$, $\sigma \in \{0,1\}$, sets $\mathbf{E} = \{E_{\sigma,i}\}_{\sigma\in\{0,1\},i\in[d]}$, where $E_{\sigma,i} \leftarrow \mathsf{Encode}(\mathsf{list}_{\sigma,i})$.
3. $\mathcal{R}$ sends $(\mathsf{pk}, \mathbf{E})$ to $\mathcal{S}$.
4. $\mathcal{S}$ computes $(\{u_{\sigma,i,j}\}_{i\in[d],\sigma\in\{0,1\},j\in[0,\mu_\delta]}, \{a_i\}_{i\in[d]}) \leftarrow \mathsf{GetValue}_p(\mathsf{pk}, \delta, \mathbf{y}, 0^\kappa, \mathbf{E})$, and sets $a = \sum_{i=1}^d a_i$
5. $\mathcal{S}$ shuffles $\{u_{\sigma,i,j}\}$ by j and σ, sends to $\mathcal{R}$, where $\sigma \in \{0,1\}$,$i \in [d]$, $j \in [0, \mu_\delta]$.
6. Set $c = 0$, for $\forall(j_i)_{i\in[d]} \in [0, \mu_\delta]^d, (\sigma_i)_{i\in[d]} \in \{0,1\}^d$:
   - $\mathcal{R}$ and $\mathcal{S}$ runs $\mathcal{F}_{\mathsf{IFMat}}$ with inputs $\sum_{i=1}^d(\mathsf{Dec}_{\mathsf{sk}}(u_{\sigma_i,i,j_i}))$ and $\{a + \frac{1}{2}\delta^p\}$ respectively, the threshold is $\frac{1}{2}\delta^p$.
   - $\mathcal{R}$ gets output e, if $e = 1$, set $c = 1$.
7. Output c.

Figure 8: Fuzzy matching for L_p distance with finite p

for the Receiver and $O(\mu_\delta dpN_s + (2\mu_\delta)^d N_s)$ for the Sender. Here, $\mu_\delta = \lfloor \log \delta \rfloor$ and $\omega_\delta = 2\lfloor \log(\delta + 1) \rfloor$.

Remark 1 (Extending to PSI). Notice that in the penultimate step of both protocols in Figure 18, the Receiver obtains $(e_1, e_2, \ldots, e_{N_s})$. Then, the Receiver and Sender run N_s oblivious transfers (OT) with inputs e_i and $q_0 = \perp, q_1 = \mathbf{x}_i$ for each $i \in [N_s]$. The correctness and security follow from the properties of PSI-CA. The same extension applies to the PSI-CA protocols in the rest of this paper.

7 Fuzzy PSI(-CA) in High Dimension

We begin by introducing the definition and construction of Private Index Search (PIS). PIS serves as a fundamental building block for our fuzzy mapping and fuzzy PSI-CA protocols, enabling the elimination of the $(\log \delta)^d$ factor in both communication and computation complexity. Then, we present our efficient fuzzy mapping protocol. Finally, we provide our fuzzy PSI-CA protocol for both L_∞ and L_p distances.

7.1 Private Index Search

Let the Receiver hold a set $S \subset \mathcal{U}$ consisting of n elements, and the Sender hold an element $b \in \mathcal{U}$. A private index search (PIS) protocol allows the Receiver to learn index of b if $b \in S$, else obtain a random index of an element inside S. We give the definition as follows.

Definition 3 (Private Index Search)**.** Let public parameters $\mathcal{U}$ be a space and n be an integer. A private index search protocol Π_{PIS} is a two party protocol with inputs $b \in \mathcal{U}$ from the Sender and a set $S = (a_i)_{i\in[0,n-1]} \in \mathcal{U}^n$ from the Receiver. After of the execution of protocol Π_{PIS}, Receiver outputs $\mathsf{idx} \in [0, n-1]$, which satisfies following properties:

- **Correctness.** If $b = a_i \in S$, then $\mathsf{Pr}(\mathsf{idx} = i) \geq 1 - nelg(\lambda)$.
- **Randomness.** If $b \notin S$, then $\mathsf{idx} \leftarrow_\$ [0, n-1]$.
- **Security.** For the semi-honest corrupted Receiver $\mathcal{R}$, for any $a_i \in S$, and $a' \leftarrow_\$ \mathcal{U}$, it holds that $\mathsf{Real}_{\Pi,\mathcal{R}}(\kappa, \lambda, a_i, S) \approx_c \mathsf{Real}_{\Pi,\mathcal{R}}(\kappa, \lambda, a', S)$. For the semi-honest corrupted Sender $\mathcal{S}$, for any $S' \in \mathcal{U}^n$, it holds that $\mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, b, S') \approx_c \mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, b, S)$.

Let $n = 2^l$ for some integer l. We construct a PIS protocol shown in Figure 9, which is based on an ssPET protocol, leading to a complexity scaling linearly in $\log \delta$.

Parameter setting: A space $\mathcal{U} = \{0,1\}^u$, an integer $n = 2^l$, where $l \in \mathbb{N}^+$. Sender $\mathcal{S}$ holds $b \in \mathcal{U}$. Receiver $\mathcal{R}$ holds a set $S = (a_i)_{i \in [0,n-1]} \in \mathcal{U}^n$.
Protocol:

1. For each $i \in [0, n-1]$: $\mathcal{R}$ and $\mathcal{S}$ invoke $\mathcal{F}_{\mathsf{ssPET}}$ with a_i and b as inputs. $\mathcal{F}_{\mathsf{ssPET}}$ outputs s_i^* and t_i^* to $\mathcal{R}$ and $\mathcal{S}$ respectively. $\mathcal{R}$ and $\mathcal{S}$ set $s_0 = \oplus_{i=0}^{n-1} s_i^*$ and $t_0 = \oplus_{i=0}^{n-1} t_i^*$.
2. Let $S_k = \{\overline{x_l \cdots x_{k+1} 1 x_{k-1} \cdots x_1} : x_1, \cdots, x_l \in \{0,1\}\}$ denote the set consisting of bit strings of l length with the k-th bit being 1. For each $k \in [1, l]$: $\mathcal{R}$ and $\mathcal{S}$ set $s_k = \oplus_{i \in S_k} s_i^*$ and $t_k = \oplus_{i \in S_k} t_i^*$ respectively. Denote $s = \overline{s_l s_{l-1} \cdots s_1}$, $t = \overline{t_l t_{l-1} \cdots t_1}$.
3. $\mathcal{R}$ samples $r \leftarrow_\$ \{0,1\}^l$. $\mathcal{S}$ acting as sender with inputs (q_0, q_1) and $\mathcal{R}$ acting as receiver with input $c = s_0$ invoke $\mathcal{F}_{\mathsf{OT}}$, where $q_0 = (t_0 \oplus 1) \cdot r \oplus t, q_1 = r \oplus q_0$. $\mathcal{F}_{\mathsf{OT}}$ returns h back to $\mathcal{R}$. $\mathcal{R}$ outputs $\mathsf{idx} = h \oplus s$.

Figure 9: Private index search protocol Π_{PIS}

In our protocol in Figure 9, n instances of ssPET are invoked. The communication and computational complexity of ssPET are both $O(\lambda)$. Thus, the communication complexity of our PIS protocol is $O(n\lambda)$, while the computational complexity are both $O(n)$ for the Receiver and the Sender. The proofs of the properties are given in Appendix D.1.

7.2 Our Construction of Fuzzy Mapping

Notice that in the low dimension case, the Receiver uses spatial hash to map their balls into disjoint cells. If $\|\mathbf{x} - \mathbf{y}\|_\infty \leq \delta$, then the cell containing the Sender's point $\mathbf{y}$ intersects with the cells mapped by the Receiver's ball centered at $\mathbf{x}$. However, this spatial hashing approach introduces a 2^d factor in the Receiver's communication complexity, which becomes prohibitive for large dimensions d. To overcome the 2^d complexity factor in high-dimensional settings, we propose a new mapping approach.

We utilize fuzzy mapping to assign identifiers to points in $\mathbb{Z}^d$. Fuzzy mapping (Fmap), introduced by Gao et al. [GQL+25] as a generalization of spatial hashing, constructs mappings for L_∞ distance in high dimensions. Their core insight is that for each point $\mathbf{x}$, the Receiver assigns a random value r_i to each dimension's L_∞-ball projection and computes the identifier $\mathsf{id}(\mathbf{x}) = \sum_{i=1}^d r_i$, where intersecting projections share identical r_i values.

The protocol proceeds as follows: The Receiver sends an Oblivious OKVS to the Sender, where keys are all points in each projection interval $[x_i - \delta, x_i + \delta]$ and values are encryptions $\mathsf{Enc}_{\mathsf{pk}}(r_i)$. The Sender decodes the OKVS using y_i to retrieve $\mathsf{Enc}_{\mathsf{pk}}(r_i')$, homomorphically computes $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{mask} + \sum_{i=1}^d r_i')$, and sends this ciphertext to the Receiver.

After decryption, the Receiver returns $\sum_{i=1}^{d} r'_i + \mathsf{mask}$ to the Sender, who subtracts the mask to obtain $\mathsf{id}(\mathbf{y})$. This construction achieves complexity linear in δ.

To further reduce the complexity to scale logarithmically with δ, we propose a new semi-honest secure Fmap protocol that encodes interval prefixes $[x_i - \delta, x_i + \delta]$ (instead of all points) into the OKVS, combined with spatial additive sharing techniques.

Before showing our Fmap construction, we present an algorithm IDGen to generate identifiers for the Receiver, as shown in Figure 10.

$\mathsf{GetID}(X, \mathsf{pk})$: input $X \subset \mathcal{U}^d$ is a set of size N. $\mathcal{U} = \{0,1\}^u$.

1. For each $i \in [d]$:
   - For each $k \in [N]$:
     * Initiate $\mathsf{interval}_i = \emptyset$ and $r_{k,i} \leftarrow_{\$} \mathcal{U}$.
     * If $\forall (U, r) \in \mathsf{interval}_i$, $[x_{k,i} - \delta, x_{k,i} + \delta] \cap U = \emptyset$: set $\mathsf{interval}_i \leftarrow \mathsf{interval}_i \cup ([x_{k,i} - \delta, x_{k,i} + \delta], r_{k,i})$.
       Else: denote $\{(U_i, r_i)\}_{i \in I} \subset \mathsf{interval}_i$ be s.t. $[x_{k,i} - \delta, x_{k,i} + \delta] \cap U_i \neq \emptyset$, then remove $\{(U_i, r_i)\}_{i \in I}$ from $\mathsf{interval}_i$, and set $\mathsf{interval}_i \leftarrow \mathsf{interval}_i \cup ([x_{k,i} - \delta, x_{k,i} + \delta] \bigcup (\cup_{i \in I} U_i), r_{k,j})$.
2. For each $k \in [N]$: set $\mathsf{id}_{\mathbf{x}_k} = \sum_{i=1}^{d} r_i$, where r_i satisfies that $x_{k,i} \in U_i$ and $\{(U_i, r_i)\} \in \mathsf{interval}_i$.
3. For each $i \in [d]$:
   - Initiate $\mathsf{list}_i = \emptyset$. For each $(U, r) \in \mathsf{interval}_i$:
     * Denote $\{x_j\}_{j \in [0, \omega']} = \mathsf{Decompose}(U)$, where x_j has length no less than $u - \mu_\delta$. For each $j \in [0, \omega']$: set $\mathsf{list}_i \leftarrow \mathsf{list}_i \cup (x_j, \mathsf{Enc}_{\mathsf{pk}}(r) || \mathsf{Enc}_{\mathsf{pk}}(0))$.
   - Pad list_i with dummy random items to get list_i of size $N_r \omega_\delta$.
4. Output $(\mathsf{list}_i)_{i \in [d]}$ and $(\mathsf{id}_{\mathbf{x}})_{\mathbf{x} \in X}$.

Figure 10: Algorithm $\mathsf{GetID}()$

The detailed fuzzy mapping protocol Π_{PFmap} is described in Figure 11. The communication complexity of our protocol in Figure 11 is $O(\mu_\delta d \lambda N_s + \omega_\delta d \lambda N_r)$. The computational complexity is $O(\mu_\delta \, d N_s)$ for the Sender and $O(\mu_\delta \, d N_s + \omega_\delta d N_r)$ for the Receiver. Here $\mu_\delta = \lfloor \log \delta \rfloor + 1$, $\omega_\delta = 2 \lfloor \log(2\delta + 1) \rfloor$.

Note that our protocol Π_{PFmap} invokes Π_{PIS} protocol which requires the Receiver's set size to be power of 2. Thus, the Sender needs to pad the number of $\{(u_{k,i,j} || v_{k,i,j})\}_j$ to a power of 2, namely, $2^{\lceil \cdot \rceil}$ items, as described in step 3 of Figure 11.

Lemma 2 ([vBP24]). *If X satisfies that $\forall i \in [N], \exists j \in [d], \forall i' \neq i$, s.t. $\mathsf{Ball}_{1,\delta}(x_{i,j}) \cap \mathsf{Ball}_{1,\delta}(x_{i,j'}) = \emptyset$, we call it a separated set. Then if points in $X = \{\boldsymbol{x}_i\}_{i \in N} \subset \mathcal{U}^d$ are uniformly distributed, $\mathcal{U} = \{0,1\}^u$, we have $Pr(X$ is a separated set $) \geq 1 - negl(d)$.*

Assumption: *The L_∞-balls associated with the Receiver must have disjoint projections in at least one dimension. All our subsequent protocols in this section adhere to this assumption.*

The distinctiveness of our fuzzy mapping protocol stems from this assumption. By Lemma 2, if the Receiver's points are uniformly sampled, this assumption holds with overwhelming probability for sufficiently large dimensions d. Therefore, the protocols

Parameter setting: Receiver $\mathcal{R}$ holds a set $X = \{\mathbf{x}\}$ of size N_r and Sender $\mathcal{S}$ holds a set $Y = \{\mathbf{y}\}$ of size N_s, where $\mathbf{x}, \mathbf{y} \in \mathcal{U}^d$ and $\mathcal{U} = \{0,1\}^u$. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1. $\mathcal{R}$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $\mathcal{R}$ runs $\mathsf{GetID}(X, \mathsf{pk})$ and obtains outputs $(\mathsf{list}_i)_{i\in[d]}$ and $(\mathsf{id}_\mathbf{x})_{\mathbf{x}\in X}$. $\mathcal{R}$ encodes $\mathsf{E}_i = \mathsf{Encode}(\mathsf{list}_i)$ and sends $(\mathsf{E}_i, \mathsf{pk})$ to $\mathcal{S}$.
3. For each $k \in [N_s]$, $i \in [d]$:
   - $\mathcal{S}$ computes $y_{k,i,j} = \mathsf{Prefix}(y_{k,i}, j), j \in [0, \mu_\delta]$, then computes $(u_{k,i,j}||v_{k,i,j}) = \mathsf{Decode}(\mathsf{E}_i, y_{k,i,j})$, and pads the number of items to $2^{\lceil \log \mu_\delta \rceil}$ with randomness.
   - $\mathcal{S}$ sample $\mathsf{mask}_{k,i}, \mathsf{mask}'_{k,i} \leftarrow_\$ \mathcal{P}$ for $j \in [0, 2^{\lceil \log \mu_\delta \rceil} - 1]$, sets $(u'_{k,i,j}, v'_{k,i,j}) = (u_{k,i,j} \boxplus_\mathsf{pk} \mathsf{Enc}_\mathsf{pk}(\mathsf{mask}_{k,i}), v_{k,i,j} \boxplus_\mathsf{pk} \mathsf{Enc}_\mathsf{pk}(\mathsf{mask}'_{k,i}))$. $\mathcal{S}$ shuffles $(u'_{k,i,j}||v'_{k,i,j})$ by k, then by j, and sends it to $\mathcal{R}$.
4. Upon receiving $(u'_{k,i,j}||v'_{k,i,j})$ from $\mathcal{S}$, $\mathcal{R}$ shuffles $(u'_{k,i,j}||v'_{k,i,j})$ by j. For each $k \in [N_s]$:
   - For each $i \in [d]$: $\mathcal{R}$ acting as Receiver with input $(\mathsf{Dec}_\mathsf{sk}(v'_{k,i,j}))_{j\in[0,2^{\lceil \log \mu_\delta \rceil}-1]}$, and $\mathcal{S}$ acting as Sender with input $\mathsf{mask}'_{k,i}$ invoke Π_{PIS}. $\mathcal{R}$ obtains output $j_{k,i}$.
   - $\mathcal{R}$ sends $w_k = \sum_{i=1}^d \mathsf{Dec}_\mathsf{sk}(u'_{k,i,j_{k,i}})$ to $\mathcal{S}$.
5. For each $k \in [N_s]$: $\mathcal{S}$ computes $\mathsf{id}_{\mathbf{y}_k} = w_k - \sum_{i=1}^d \mathsf{mask}_{k,i}$.

Figure 11: Fuzzy mapping protocol Π_{PFmap} in high dimension

proposed in this section are feasible in high-dimensional settings. The proofs of correctness, distinctiveness and security are given in Appendix D.2.

7.3 Fuzzy PSI-CA for L_∞ Distance

We construct fuzzy PSI-CA protocols by applying Fmap to avoid the exponential factor $(\log \delta)^d$ in complexity.

Initially, the Receiver and Sender, holding sets X and Y respectively, execute Fmap with a threshold δ to obtain identifiers $\{\mathsf{id}_\mathbf{x}\}_{\mathbf{x}\in X}$ and $\{\mathsf{id}_\mathbf{y}\}_{\mathbf{y}\in Y}$. If $\|\mathbf{x} - \mathbf{y}\|_\infty \le \delta$, $\mathsf{id}_\mathbf{x}$ equals $\mathsf{id}_\mathbf{y}$. Then, the Receiver encodes $(\mathsf{Decompose}([x_i - \delta, x_i + \delta]), \mathsf{id}_\mathbf{x})$ as 0 in an OKVS. The Sender decodes on prefixes of y_i concatenated with $\mathsf{id}_\mathbf{y}$ to get $\mathsf{Enc}_\mathsf{pk}(0)$ if there exists $\mathbf{x}$ such that $\|\mathbf{x} - \mathbf{y}\|_\infty \le \delta$. The Sender sends all ciphertexts homomoraphically added with masks mask_i back to the Receiver, who decrypts them to get plaintexts U_i. Then two parties engage in ssPET protocols to check if $\mathsf{mask}_i \in U_i$ on each dimension. If yes, they compute sharing bits b_i and b'_i of 1; otherwise 0. It holds that $(b_i \oplus 1) - b'_i$ is 0 if $\|\mathbf{x} - \mathbf{y}\|_\infty \le \delta$, else it is ± 1. The Receiver sends $\mathsf{Enc}(b_i \oplus 1)$ to the Sender, who computes $\mathsf{Enc}(\sum_{i=1}^d r_i(b_i \oplus 1 - b'_i))$. The Receiver confirms $\|\mathbf{x} - \mathbf{y}\|_\infty \le \delta$ if this ciphertext is an encryption of 0.

Our fuzzy PSI-CA protocol for L_∞ distance is shown in Figure 12, where the $\mathsf{GetList}_\infty$ algorithm and the $\mathsf{GetValue}_\infty$ algorithm are defined as Figure 4 with the parameter of universal hash function H_{γ_1} chosen as $\gamma_1 = \kappa + 2\log(d \log \delta)$. The communication complexity of our protocol in Figure 12 is $O(\omega_\delta d\lambda N_r + \mu_\delta\, d\lambda N_s)$. The computational

complexity is $O(\omega_\delta dN_r + \mu_\delta\, dN_s)$ for the Receiver, and $O(\mu_\delta\, dN_s)$ for the Sender. Here $\mu_\delta = \lfloor \log \delta \rfloor + 1$, $\omega_\delta = 2\lfloor \log(2\delta + 1) \rfloor$. The proofs of correctness and security are deferred to Appendix D.3.

Parameter setting: Receiver $\mathcal{R}$ holds a set $X = \{\mathbf{x}\}$ of size N_r, and Sender $\mathcal{S}$ holds a set $Y = \{\mathbf{y}\}$ of size N_s, where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^d$. A distance threshold δ. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1. $\mathcal{R}$ and $\mathcal{S}$ invoke Π_{PFmap}, $\mathcal{R}$ acts as Receiver with inputs X, and $\mathcal{S}$ acts as Sender with inputs Y. $\mathcal{R}$ receives $\mathsf{ID}(X) = \{\mathsf{id}_{\mathbf{x}_i}\}_{i\in[N_r]}$, and $\mathcal{S}$ receives $\mathsf{ID}(Y) = \{\mathsf{id}_{\mathbf{y}_j}\}_{j\in[N_s]}$.
2. $\mathcal{R}$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
3. $\mathcal{R}$ initiates $\mathbf{list_i} = \emptyset$. For each $k \in [N_r]$: $\mathcal{R}$ computes $\{\mathsf{list}'_i\}_{i\in[d]} \leftarrow \mathsf{GetList}_\infty(\mathsf{pk}, \delta, \mathbf{x}_k, \mathsf{id}_{\mathbf{x}_k})$, and for each $i \in [d]$, sets $\mathsf{list}_i \leftarrow \mathsf{list}_i \cup \mathsf{list}'_i$.
4. $\mathcal{R}$ sets $\mathbf{E} = \{E_i\}_{i\in[d]}$, where $E_i \leftarrow \mathsf{Encode}(\mathsf{list}_i)$, and sends $(\mathsf{pk}, \mathbf{E})$ to $\mathcal{S}$.
5. For each $k \in [N_s]$: $\mathcal{S}$ computes $(u_{k,i,j}, a_{k,i}) \leftarrow \mathsf{GetValue}_\infty(\mathsf{pk}, \delta, \mathbf{y}_k, \mathsf{id}_{\mathbf{y}_k}, \mathbf{E})$, and sets $a_k = \sum_{i=1}^{d} a_{k,i}$.
6. $\mathcal{S}$ shuffles $\{u_{k,i,j}\}$ by j, then by k, and sends the shuffled set to $\mathcal{R}$, where $i \in [d]$, $j \in [0, \mu_\delta]$, $k \in [N_s]$.
7. For each $k \in [N_s]$:
   - For each $i \in [d]$: $\mathcal{S}$ and $\mathcal{R}$ invoke $\mathcal{F}_{\mathsf{ssPET}}$ for $j \in [0, \mu_\delta]$ with input $\{\mathsf{Dec}(u_{k,i,j})\}$ and $a_{k,i}$. $\mathcal{R}$ receives $b^*_{k,i.j} \in \{0, 1\}$ and sets $b_{k,i} = \oplus_{j=0}^{\mu_\delta} b^*_{k,i.j}$; $\mathcal{S}$ receives ${b'_{k,i,j}}^* \in \{0, 1\}$ and sets $b'_{k,i} = \oplus_{j=0}^{\mu_\delta} {b'_{k,i.j}}^*$.
   - $\mathcal{R}$ sends $\mathsf{ct}_{k,i} = \mathsf{Enc}_{\mathsf{pk}}(b_{k,i} \oplus 1)$ to $\mathcal{S}$. Then $\mathcal{S}$ samples $r_{k,i} \leftarrow_\$ \mathcal{P}$, and sends $\mathsf{ct}'_k = \boxplus_{\mathsf{pk}}{}_{i=1}^{d}\big(r_{k,i} \boxtimes_{\mathsf{pk}} (\mathsf{Enc}_{\mathsf{pk}}(-b'_{k,i}) \boxplus_{\mathsf{pk}} \mathsf{ct}_{k,i})\big)$ to $\mathcal{R}$. $\mathcal{R}$ sets $e_k = 1$ if $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}'_k) = 0$ else $e_k = 0$.
8. $\mathcal{R}$ outputs $c = \sum_{i=1}^{N_s} e_i$.

Figure 12: Fuzzy PSI-CA protocol for L_∞ distance in high dimension

7.4 Fuzzy PSI-CA for L_p Distance

Before showing fuzzy PSI protocol for L_p distance, we provide our modified algorithms $\mathsf{GetList}'_p$ and $\mathsf{GetValue}'_p$ in Figure 19 in Appendix D.4. Differing from the $\mathsf{GetList}_p$ and $\mathsf{GetValue}_p$ algorithm in Figure 7, we allow the Receiver to additionally encode an encryption of 0. Sender homomorphically adds a mask s_i to the values retrieved by decoding on prefixes and sends them to Receiver. Receiver decrypts them to get $\{s'_{j_i}\}_{j_i\in[\mu_\delta]}$. By running PIS with Sender on input s_i and Receiver on input $\{s'_{j_i}\}_{j_i\in[\mu_\delta]}$, Receiver can obtain the index of intersected prefix j_i, s.t. $s_i = s'_{j_i}$. Once knowing the index j_i, Receiver can locate the value of $\mathsf{mask}_i + |x_i - y_i|^p$. Then by using IFMat we can check if $(||\mathbf{x} - \mathbf{y}||_p)^p \le \delta^p$.

Our detailed fuzzy PSI-CA protocol for L_p is shown in Figure 13. In this construction, we assume that any L_∞ balls of Receiver must has disjoint projection in some dimension. The communication complexity of our protocol in Figure 13 is $O(\omega_\delta\, pd\lambda N_r + \mu_\delta\, d\lambda N_s)$.

The computation complexity is $O(\omega_\delta\, pdN_r + \mu_\delta\, dN_s)$ for Receiver, and $O(\mu_\delta\, pdN_s)$ for Sender. Here $\mu_\delta = \lfloor \log \delta \rfloor$, $\omega_\delta = 2\lfloor \log(\delta+1) \rfloor$. The proofs of correctness and security are given in Appendix D.4.

Parameter setting: $p \in \mathbb{N}$. Receiver $\mathcal{R}$ holds a set $X = \{\mathbf{x}\}$ of size N_r. Sender $\mathcal{S}$ holds a set $Y = \{\mathbf{y}\}$ of size N_s. $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^d$. A distance threshold δ. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1-4. Step 1-4 are the same as protocol in Figure 12, except that $\mathsf{GetList}_p$ is replaced with $\mathsf{GetList}'_p$.

5. For each $k \in [N_s]$: $\mathcal{S}$ computes $(u_{k,\sigma,i,j}, v_{k,\sigma,i,j}, a_{k,i}, s_{k,i}) \leftarrow \mathsf{GetValue}'_p(\mathsf{pk}, \delta, \mathbf{y}_k, \mathsf{id}_{\mathbf{y}_k}, \mathbf{E})$, and sets $a_k = \sum_{i=1}^{d} a_{k,i}$, where $k \in [N_s], \sigma \in \{0,1\}, i \in [d], j \in [\mu_\delta]$. $\mathcal{S}$ pads the items with random values to size $2^{\lceil \log \mu_\delta \rceil}$.

6. $\mathcal{S}$ shuffles $(u_{k,\sigma,i,j}, v_{k,\sigma,i,j})$ by σ, j, then by k, sends to $\mathcal{R}$, where $i \in [d]$, $\sigma \in \{0,1\}$, $j \in [0, 2^{\lceil \log \mu_\delta \rceil} - 1]$, $k \in [N_s]$.

7. $\mathcal{R}$ shuffles $(u_{k,\sigma,i,j}, v_{k,\sigma,i,j})$ by σ, j. For each $k \in [N_s]$, $i \in [d]$:

   – $\mathcal{S}$ and $\mathcal{R}$ invoke Π_{PIS}, where $\mathcal{R}$ acts as the Receiver with input $(\mathsf{Dec}_{\mathsf{sk}}(v_{k,\sigma,i,j}))_{\sigma\in\{0,1\}, j\in[0,\mu_\delta]}$, and $\mathcal{S}$ acts as the Sender with input $s_{k,i}$. $\mathcal{R}$ receives outputs $(j_i)_{i\in[d]} \in [0, 2^{\lceil \log \mu_\delta \rceil} - 1]^d$ and $(\sigma_i)_{i\in[d]} \in \{0,1\}^d$.

8. For each $k \in [N_s]$: $\mathcal{R}$ and $\mathcal{S}$ invoke $\mathcal{F}_{\mathsf{IFMat}}$ with inputs $\sum_{i=1}^{d} \mathsf{Dec}_{\mathsf{sk}}(u_{k,\sigma_i,i,j_i}))$ and $a_k + \frac{1}{2}\delta^p$ respectively, where the threshold is $\frac{1}{2}\delta^p$. $\mathcal{R}$ gets output e_k.

9. $\mathcal{R}$ outputs $c = \sum_{i=1}^{N_s} e_i$.

Figure 13: Fuzzy PSI-CA protocol for L_p distance in high dimension

8 Further Optimization

We observe that the complexity of all of our protocols are dependent on the parameters μ_δ and ω_δ. ω_δ is the upper bound of the number of key-value pairs, which is $2\lfloor 2\delta + 1 \rfloor$ or $2\lfloor \delta + 1 \rfloor$, and μ_δ is the number of prefixes that the Sender decodes on in each dimension, where $\mu_\delta = \lfloor \log \delta \rfloor + 1$ or $\lfloor \log \delta \rfloor$. Once we reduce the number of prefixes of the Sender's input y_i on each dimension i, we must increase the size of the OKVS to make sure that the prefix sets are intersected. In this section, we give methods to trade off between two parameters μ_δ and ω_δ, thereby improving the overall efficiency of all our protocols in Figures 17, 18, 12, and 13. This method may be of independent interest, since it can also be applied into [GGM24].

Here we provide an simplified example to illustrate our basic idea. Consider the one-dimensional case. Assume the Receiver and Sender have points $x = 7$ and $y = 5$ respectively, with $\delta = 7$. Receiver decomposes $[x - \delta, x + \delta] = [0, 14] = [0000, 1110]$ as a prefix set of $\{0***, 10**, 110*, 1110\}$, while Sender computes his prefix set as$\{\mathsf{Prefix}(y = 0101, i)\}_{i\in[0,3]} = \{0101, 010*, 01**, 0***\}$.

If the Sender intends to reduce the number of prefix set of $\{\mathsf{Prefix}(y = 0101, i)\}_{i\in[0,3]}$ from 4 to 3, namely, he removes the prefix $0***$ from the set. For correctness and security, the Receiver must encode key-value pairs corresponding to $01**$ and $00**$ instead of $0***$.

This implies the Receiver has to encode one more key-value pair. Thus, the reduction on the number of μ_δ, which improves the computational cost of Sender, leads to growing on ω_δ which however increases the communication cost of Receiver.

Let t denote the size of interval, which is $2\delta+1$ for L_∞ distance and $\delta+1$ for L_p distance. Let $U \subset [0, \lfloor \log t \rfloor]$ denote the set of indexes of prefixes that Sender decodes on, which are $\mathsf{Prefix}(y_i, j)$ for $j \in U$ on dimension i. We denote $\bar{\omega}(t, U)$ as the maximum size of the prefix set obtained by decomposing intervals with size t. Notice $\omega_\delta = \bar{\omega}(2\delta+1, [0, \lfloor \log(2\delta+1) \rfloor])$ in the L_∞ distance setting. Please refer to the Appendix E for formal definitions, theoretic analysis and our detailed results on $\bar{\omega}(t, U)$. In Table 2, we list different choices of the parameters t, U and $\bar{\omega}(t, U)$ according to our analysis. We note that here Receiver needs to decompose the interval in a different manner, denoted as $\mathsf{SteDec}(*, U)$. See Appendix E.4 for details.

Table 2: Relationship between $\bar{\omega}(t, U)$ and U

	$t=17, U \subset [0,4]$			$t=33, U \subset [0,5]$		
U	$\{0,2\}$	$[0,2]$	**[0,3]**	$\{0,2\}$	$[0,3]$	**[0,4]**
$\bar{\omega}(t,U)$	12	6	**5**	9	7	**6**
	$t=65, U \subset [0,6]$			$t=129, U \subset [0,7]$		
U	$\{0,3\}$	$\{0,1,3,4\}$	**[0,5]**	$\{0,3\}$	$\{0,1,3,5\}$	**[0,6]**
$\bar{\omega}(t,U)$	16	9	**7**	24	11	**8**
	$t=257, U \subset [0,8]$			$t=513, U \subset [0,9]$		
U	$\{0,4\}$	$\{0,2,4,6\}$	**[0,7]**	$\{0,4\}$	$\{0,2,4,6\}$	**[0,6]**
$\bar{\omega}(t,U)$	32	14	**9**	48	18	**10**

Modification to protocols. As we optimize the parameters of ω_δ and μ_δ, we adjust our protocols accordingly. In fuzzy matching protocols (Figures 5, 8) and fuzzy PSI-CA protocols (Figures 12, 13,17, 18), along with the $\mathsf{GetList}$ algorithms (Figures 4, 7), ω_δ can be adjusted to $\bar{\omega}(t, U)$, where $t = 2\delta + 1$ for $p = \infty$ and $t = \delta + 1$ for $p < \infty$, while μ_δ is adjusted to $|U| - 1$ for index counting, and $\mathsf{Decompose}(*)$ is replaced by $\mathsf{setDec}(*, U)$. In the $\mathsf{GetValue}$ algorithms (Figures 4, 7), let $y^*_{i,j} = \mathsf{Prefix}(y_i, j)$ for $j \in U$ in Step 1. All other steps in the algorithm and protocols remain the same.

Note that after applying this optimization to our protocols, the complexities scale linearly in terms of $\bar{\omega}(t, U)$ and $|U|$ instead of $\log \delta$. In our experiment, we choose parameters as the Table 2 in bold type. For $t = 17$ we set $\bar{\omega} = 5$ and $U = [0, 3]$, while for $t = 33$ we set $\bar{\omega} = 6$ and $U = [0, 4]$, as they achieve nearly optimal communication among other choices and allows for optimal computation.

9 Performance and Evaluation

9.1 Implementation Details

We implement our PSI-CA protocols and use method in Remark 1 to convert them to fuzzy PSI protocols. We perform all experiments under following parameter setting: the metrics L_∞, L_1 and L_2, $d = 2$ for low dimension, $d = \{5, 10\}$ for high dimension, threshold distance $\delta = 2^{\{4,5,6,7,8\}}$, and set size $N_r = N_s = 2^{\{4,8,12\}}$. For protocols in low dimension, we choose parameters ω_δ and μ_δ as Table 2 in bold type, i.e., $U = [0, \eta - 1]$ and $\bar{\omega}(t, U) = \eta + 1$ for interval size $t = 2^\eta + 1$. For high dimension case, we set the size of prefix set U to be 2 for $\delta = 2^{\{4,5\}}$, and 4 for $\delta = 2^{\{6,7,8\}}$. We set statistical and computational security parameters as $\kappa = 40$ and $\lambda = 128$.

Our benchmarks are implemented on a PC with two Intel Xeon Gold 6338 CPUs (128 cores, 256 threads), 503 GiB of RAM, and a virtualization-enabled x86-64 architecture. We measure the time of online phase in a local network setting with network latency of 0.03 ms and bandwidth of 10 Gbps. All of our experiments use a single process, where each

party in the protocol uses a separate thread. Within the protocol, we employ a number of threads corresponding to the number of dimensions to accelerate RBOKVS encoding.

Table 3: Communication (MB) and computation (s) of fuzzy PSI in low ($d = 2$) and high ($d = 5, 8$) dimension case in LAN network. We set $N_r = N_s = 2^8$ and $\delta = 2^{\{4,6,8\}}$. The best results in communication and computation cost are marked in blue and red, respectively. [Ours] in [Low dim] and [High dim] are our low dimension PSI and high dimension PSI respectively.

metric	Protocol	(d,δ) [Low dim]						(d,δ) [High dim]						(d,δ) [High dim]					
		$(2,16)$		$(2,64)$		$(2,256)$		$(5,16)$		$(5,64)$		$(5,256)$		$(8,16)$		$(8,64)$		$(8,256)$	
		Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time	Comm.	Time
L_∞	[vBP24]	**6.203**	7.287	24.403	27.711	96.203	110.826	-	-	-	-	-	-	-	-	-	-	-	-
	[GQL+25]	11.695	3.884	45.107	12.576	178.757	47.465	**28.927**	9.084	112.459	30.515	446.584	118.027	**46.163**	14.675	179.813	49.738	714.413	196.305
	[Ours]	6.820	**0.967**	**9.130**	**1.248**	**11.38**	**1.622**	54.299	**2.024**	**62.737**	**2.334**	**100.83**	**3.265**	86.810	**2.803**	**100.275**	**3.060**	**161.281**	**4.863**
L_1	[vBP24]	**6.221**	7.162	24.468	27.74	96.456	110.626	-	-	-	-	-	-	-	-	-	-	-	-
	[GQL+25]	11.792	**4.223**	45.252	12.993	178.949	48.250	**29.025**	9.325	112.603	31.444	446.775	118.748	**46.261**	14.958	179.958	50.319	714.605	197.166
	[Ours]	15.514	5.145	**24.174**	**10.776**	**33.415**	**18.59**	76.512	**2.839**	**97.404**	**3.506**	**144.473**	**4.799**	122.183	**3.777**	**155.816**	**5.151**	**231.615**	**7.071**
L_2	[vBP24]	**6.456**	7.169	**28.206**	28.268	160.205	110.282	-	-	-	-	-	-	-	-	-	-	-	-
	[GQL+25]	11.886	**4.469**	45.392	13.335	179.136	48.477	**29.119**	9.596	112.744	31.664	446.963	119.058	**46.355**	15.266	180.099	50.718	714.792	197.871
	[Ours]	26.764	5.227	40.141	**10.893**	**56.964**	**18.873**	87.797	**3.161**	**110.499**	**5.510**	**167.471**	**12.807**	139.965	**4.238**	**176.162**	**7.275**	**266.165**	**15.107**

Our protocols are implemented in C++, utilizing the following instantiations. We implement RBOKVS [BPSY23] in C++, where each key is mapped to values encrypted using the Paillier cryptosystem, represented as BigInt ciphertexts. For oblivious transfer (OT), we leverage the implementation provided by libOTe[1]. Additionally, we incorporate an efficient C++ library for the Paillier partially homomorphic encryption scheme[2] to support computations involving the $L_{p\in[1,\infty]}$ distance. To ensure both high performance and cryptographic security, we instantiate universal hash functions using BLAKE3. Our implementation is available online on GitHub[3].

9.2 Low Dimension Case

We report the performance of our fuzzy PSI protocols for L_∞ and L_p distances in comparison with the state-of-the-art proposed in [vBP24] and [GQL+25]. A detailed benchmark for set size $N_s = N_r = 2^8$, dimension $d = 2$ and threshold distance $\delta = 2^{4,6,8}$ are given in Table 3.

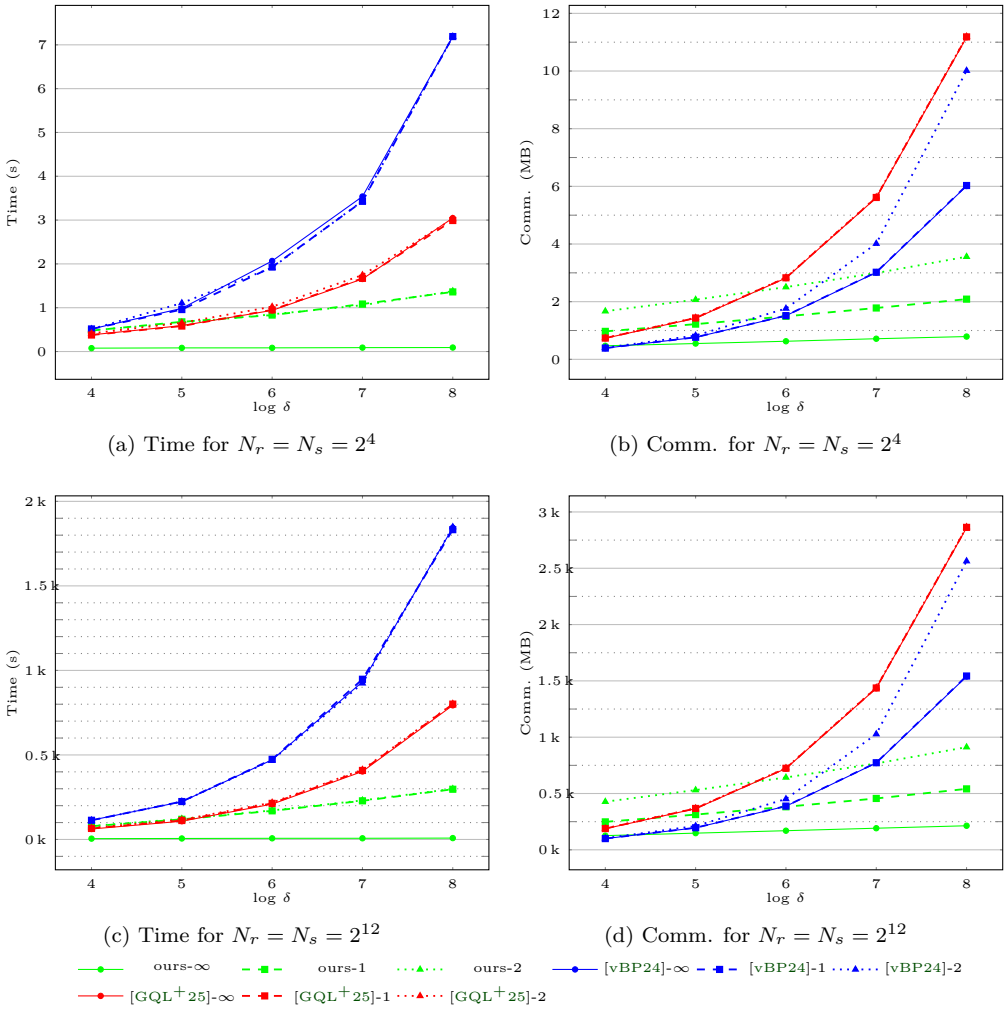
We compare the running time and communication of the fuzzy PSI protocols on different threshold distance δ. As shown in Table 3, when $N_r = N_s = 2^8$, $d = 2$ and $\delta = 64$, our protocols outperform other protocols in term of running time, which is $1.2 - 20\times$ faster than [GQL+25] and $2.5 - 50\times$ faster than [vBP24]. With the increase in δ, the advantage of our protocol in running time becomes more significant. In terms of communication, when $\delta = 256$, our protocols have the lowest communication cost, which is $2.8 - 7.5\times$ and $3 - 14\times$ lower than that of [vBP24] and [GQL+25], respectively.

In order to identify the break-even point where our fuzzy PSI protocol outperforms other protocols, in Figure 14, we illustrate the variation of communication/running time with threshold distance $\delta = 2^{\{4,5,6,7,8\}}$, given dimension $d = 2$ and set sizes $N_s = N_r = 2^{\{4,12\}}$ for L_∞, L_1, L_2 distances. As shown in Figure 14, our protocol overtakes other protocols as the fastest fuzzy PSI when threshold distance $\delta \geq 64$. Our protocol achieves the lowest communication cost when $\delta \geq 32$ for L_∞ distance, $\delta \geq 64$ for L_1 distance, and $\delta \geq 128$ for L_2 distance.

9.3 High Dimension Case

A detailed benchmark for set size $N_r = N_s = 2^8$, $d = 5, 8$, and threshold distances $2^4 - 2^8$ are presented in Table 3. Since for high dimension fuzzy PSI, the state-of-the-art is the one in [GQL+25], and [vBP24] does not implement their high dimension protocol,

[1] https://github.com/osu-crypto/libOTe
[2] https://github.com/intel/pailliercryptolib
[3] https://github.com/zhouxv/ourFuzzyPSI-C



(a) Time for $N_r = N_s = 2^4$ (b) Comm. for $N_r = N_s = 2^4$

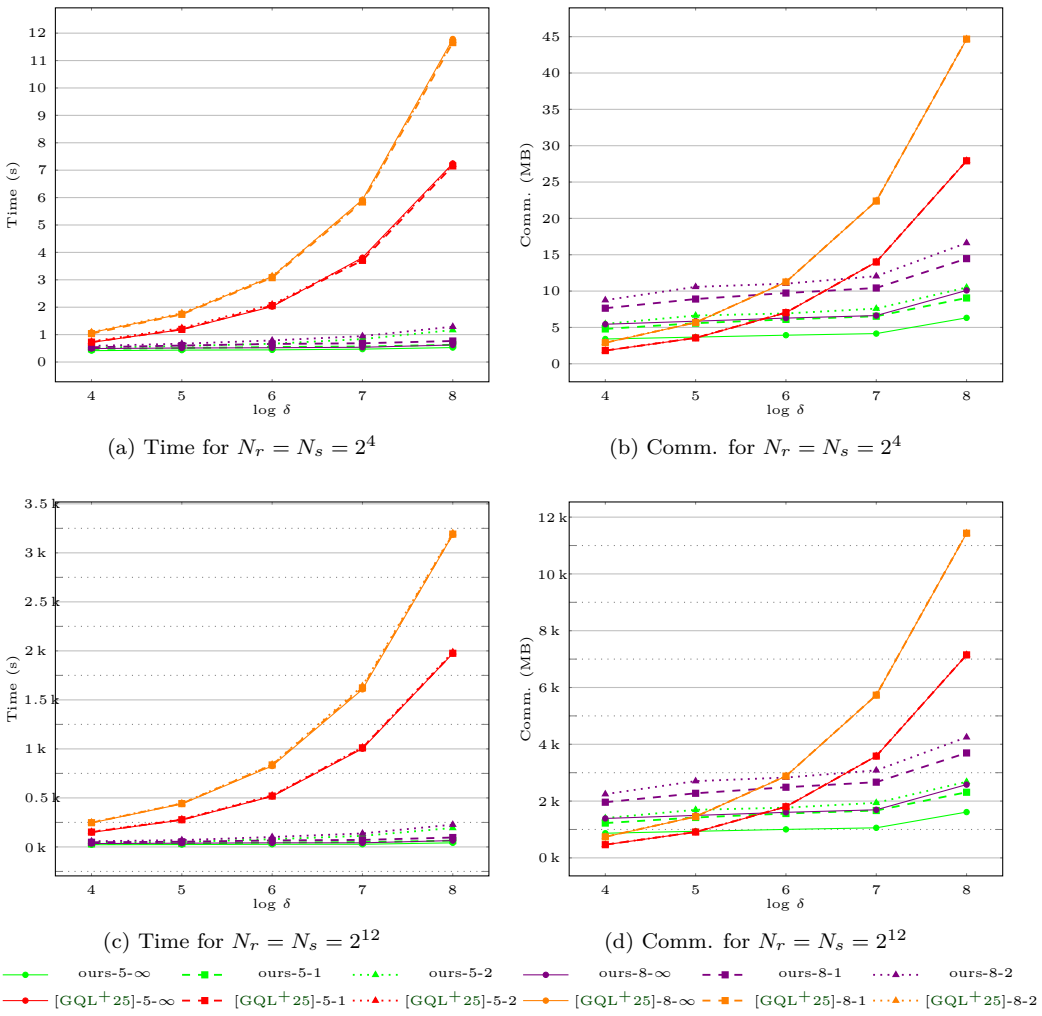
(c) Time for $N_r = N_s = 2^{12}$ (d) Comm. for $N_r = N_s = 2^{12}$

Figure 14: Communication (MB) and running time (s) of fuzzy PSI for dimension $d = 2$, $N_r = N_s = 2^{\{4,12\}}$, $\delta = 2^{\{4,5,6,7,8\}}$. [Protocol]-$p$ is protocol for L_p distance.

here we mainly compare with [GQL+25]. In Figure 15, we illustrate the variation of communication/running time for our protocols and and [GQL+25] with threshold distance $2^4 - 2^8$, dimension $d = 5, 8$ and set sizes $N_s = N_r = 2^{\{4,12\}}$ for L_∞, L_1, L_2 distances.

As shown in Table 3 and Figure 15, our protocols requires the least running time across all configurations. With the growth of δ, the advantage of our protocol in running time becomes more significant. For example, for $N_r = N_s = 2^8$, $d = 5$ and $\delta = 64$, the running time of our protocol achieves a $6 - 13\times$ improvement over [GQL+25], while for $\delta = 256$, it achieves a factor of $10 - 36\times$ improvement.

In terms of communication, our protocol achieves $1 - 4.5\times$ lower than [GQL+25]. As shown in Figure 15, the larger the threshold distances δ are, the larger the communication cost ratios are. In particular, for dimension $d = 8$ and L_∞ distance, our protocol surpasses [GQL+25] and achieves lower communication cost at the point of $\delta \geq 64$, yielding at most $4.5\times$ bandwidth reduction at $\delta = 256$, which is 161.281 MB. For L_1 and L_2 distance, the break-even point is also on $\delta = 64$, our protocol achieves $3\times$ improvement over [GQL+25] when $\delta = 256$.



(a) Time for $N_r = N_s = 2^4$

(b) Comm. for $N_r = N_s = 2^4$

(c) Time for $N_r = N_s = 2^{12}$

(d) Comm. for $N_r = N_s = 2^{12}$

Figure 15: Communication (MB) and running time (s) of fuzzy PSI for dimension $d = 5, 8$, $N_r = N_s = 2^{\{4,12\}}$, $\delta = 2^{\{4,5,6,7,8\}}$. [Protocol]-$d$-$p$ is protocol for L_p distance in dimension d.

We also conduct experiments in WAN setting. Please see Appendix F for detailed discussion.

10 Conclusion

In this work, we present fuzzy PSI protocols for general L- p distance with complexity logarithmical on distance threshold δ, and linear on d and input size. We implement the protocols and evaluate their performance. Experimental results show that our constructions outperform the state-of-art.

References

[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 *USENIX Security 23*, pages 301–318, 2023.

[CFR23] Anrin Chakraborti, Giulia Fanti, and Michael K Reiter. Distance-aware private set intersection. In *USENIX Security 23*, pages 319–336, 2023.

[CH08] Lukasz Chmielewski and Jaap-Henk Hoepman. Fuzzy private matching. In *ARES 2008*, pages 327–334. IEEE, 2008.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious prf. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020*, pages 34–63, Cham, 2020. Springer International Publishing.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. doi:10.1109/TIT.1985.1057074.

[FNP04] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Eurocrypt 2004*, pages 1–19. Springer, 2004.

[GGM24] Gayathri Garimella, Benjamin Goff, and Peihan Miao. Computation efficient structure aware PSI from incremental function secret sharing. Cryptology ePrint Archive, Paper 2024/842, 2024.

[GM19] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pages 173–201. 2019.

[GQL+25] Ying Gao, Lin Qi, Xiang Liu, Yuanchao Luo, and Longxin Wang. Efficient fuzzy private set intersection from fuzzy mapping. In *ASIACRYPT 2025*, pages 36–68. Springer, 2025.

[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, *CRYPTO 2022*, pages 323–352, Cham, 2022. Springer Nature Switzerland.

[GRS23] Gayathri Garimella, Mike Rosulek, and Jaspa Singh. Malicious secure, structure-aware private set intersection. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023*, pages 577–610, Cham, 2023. Springer Nature Switzerland.

[IKN+20] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE EuroS&P*, pages 370–389. IEEE, 2020.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *ACM CCS 2016*, pages 818–829, 2016.

[KM21] Anunay Kulshrestha and Jonathan Mayer. Identifying harmful media in {End-to-End} encrypted communication: Efficient private membership computation. In *USENIX Security 21*, pages 893–910, 2021.

[MPR+20] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In *CRYPTO 2020*, pages 3–33. Springer, 2020.

[Ode09] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 1st edition, 2009.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt 1999*, pages 223–238. Springer, 1999.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: lightweight private set intersection from sparse ot extension. In *CRYPTO 2019*, pages 401–431. Springer, 2019.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. Cryptology ePrint Archive, Paper 2014/447, 2014.

[RR22] Srinivasan Raghuraman and Peter Rindal. Blazing fast psi from improved okvs and subfield vole. In *ACM CCS 2022*, pages 2505–2517, 2022.

[RS21] Peter Rindal and Phillipp Schoppmann. Vole-psi: fast oprf and circuit-psi from vector-ole. In *Eurocrypt 2021*, pages 901–930. Springer, 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 *USENIX Security 21*, pages 911–928, 2021.

[vBP24] Aron van Baarsen and Sihang Pu. Fuzzy private set intersection with large hyperballs. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024*, pages 340–369, Cham, 2024. Springer Nature Switzerland.

[YSPW09] Qingsong Ye, Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. Efficient fuzzy matching and intersection on private datasets. In *Inscrypt 2009*, pages 211–228. Springer, 2009.

A Additional Preliminaries

A.1 Security model

We define the security of a two-party protocol by following the definition of [Ode09]. In a nutshell, security is defined by comparing two distributions of the outputs of all parties in the real execution and the ideal model, respectively.

In the real world, the corrupted party $\mathcal{A}$ runs the protocol by following some strategies, and the honest party runs the protocol honestly. We denote by $\mathsf{Real}_{\Pi,\mathcal{A}}(x,y)$ the view of adversary in the real execution, which is , where inputs x and y are inputs of two parties.

In the ideal world, we need to construct an algorithm $\mathcal{S}$ called a simulator for $\mathcal{A}$, $\mathcal{S}$ executes with input from the corrupted party and the output of the corrupted party in the ideal world. We denote by $\mathsf{Ideal}_{f,\mathcal{S}}(x,y)$ the joint execution of task f in the ideal world functionality, where inputs x and y are inputs of two parties. The functionality is a secure computation task, it is defined as a trusted third party in the ideal world.

Definition 4 (Security)**.** A protocol is secure if for every adversary $\mathcal{A}$, there exists a probabilistic polynomial time simulator $\mathcal{S}$, such that $\mathsf{Real}_{\Pi,\mathcal{A}}(x,y) \approx_c \mathsf{Ideal}_{f,\mathcal{S}}(x,y)$. Hence, if a protocol is secure, the corrupted party cannot obtain more information about the real execution than in the ideal world.

semi-honest model. A protocol is said to be secure under the semi-honest model if the adversary $\mathcal{A}$ is a passive adversary. $\mathcal{A}$ is not allowed to change the message during the execution, $\mathcal{A}$ runs the protocol honestly but tries to extract extra information after the execution of the protocol.

A.2 Oblivious Key Value Store (OKVS)

Definition 5 (Oblivious Key Value Store)**.** An OKVS is equipped with a key space $\mathcal{K}$, a value space $\mathcal{V}$, a statistical security parameter κ, and consists of two algorithms:

- Encode: Takes as input a set of $A \in (\mathcal{K} \times \mathcal{V})^n$, and outputs a vector $\mathbf{r} \in \mathcal{V}^m$ (or a failure indicator $\perp$ with negligible probability on κ).
- Decode: Takes as input a vector $\mathbf{r} \in \mathcal{V}^m$ and a key $k \in \mathcal{K}$, and outputs a value $v \in \mathcal{V}$.

An OKVS satisfies the following properties:

- Correctness: For all $A \in (\mathcal{K} \times \mathcal{V})^n$ for which $\mathsf{Encode}(L) \neq \perp$, it holds for all $(k, v) \in A$ that $\mathsf{Decode}(\mathsf{Encode}(A), k) = v$.
- Obliviousness: For all distinct $\{k_1, k_2, \cdots, k_n\}, \{k'_1, k'_2, \cdots, k'_n\} \subset \mathsf{K}$, if encode doesn't outputs $\perp$, then for $v_1, v_2, \cdots, v_n \leftarrow_\$ \mathcal{V}$: $\mathsf{Encode}(\{(k_i, v_i)_{i\in[n]}\}) \approx_c \mathsf{Encode}(\{(k'_i, v_i)_{i\in[n]}\})$

Definition 6 (Independence)**.** An OKVS satisfies Independence property if for any key $A \in (\mathcal{K} \times \mathcal{V})^n$ with distinct keys and uniformly random values, if k^* is not in the first component of A, it holds that $\mathsf{Decode}(\mathsf{Encode}(A), k^*)$ is indistinguishable from random value in $\mathcal{V}$.

A.3 Additive Homomorphic Encryption

An additive homomorphic encryption scheme is an encryption scheme that enables anyone to compute the encryption of the sum of two plaintexts by performing operations on their ciphertexts, and compute the encryption of the scalar multiplication by performing operations on a ciphertext and a plaintext. An AHE scheme with plaintext space $\mathcal{P}$ and ciphertext space space $\mathcal{C}$ contains the following algorithms:

- $\mathsf{Gen}(1^\lambda) \rightarrow_\$ (\mathsf{pk}, \mathsf{sk})$: takes as input the computation security parameter and outputs a pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}_{\mathsf{pk}}(m) \rightarrow_\$ c$: takes as input the public key pk and a plaintext $m \in \mathcal{P}$, and outputs a ciphertext $c \in \mathcal{C}$.
- $\mathsf{Dec}_{\mathsf{sk}}(c) \rightarrow_\$ m$: takes as input the secret key sk and a ciphertext $c \in \mathcal{C}$, and outputs a plaintext $m \in \mathcal{P}$.
- $c' \boxplus_{\mathsf{pk}} c'' \rightarrow_\$ c$: takes as input the public key pk and ciphertext $c', c'' \in \mathcal{C}$, and outputs a ciphertext c. For all $m', m'' \in \mathcal{P}$, let $c' = \mathsf{Enc}_{\mathsf{pk}}(m')$, $c'' = \mathsf{Enc}_{\mathsf{pk}}(m'')$, then for $c \leftarrow_\$ c' \boxplus_{\mathsf{pk}} c''$, it has $\mathsf{Dec}_{\mathsf{sk}}(c) = m' + m''$.
- $s \boxtimes_{\mathsf{pk}} c' \rightarrow_\$ c$: takes as input the public key pk and ciphertext $c' \in \mathcal{C}$, a scalar $s \in \mathcal{P}$, and outputs a ciphertext c. For all $m' \in \mathcal{P}$, let $c' = \mathsf{Enc}_{\mathsf{pk}}(m')$, then for $c \leftarrow_\$ s \boxtimes_{\mathsf{pk}} c'$, it has $\mathsf{Dec}_{\mathsf{sk}}(c) = sm'$.

We require the AHE to be IND-CPA secure, which means that for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(1^\lambda)$, $m_0, m_1 \in \mathcal{P}$, $(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_0))$ and $(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_1))$ are computationally indistinguishable. ElGamal cryptosystem [Elg85] is an AHE that is IND-CPA secure under the DDH assumption. Moreover, Paillier cryptosystem [Pai99] and Goldwasser-Micali cryptosystem [GM19] are both additive homomorphic.

A.4 Oblivious Transfer

Oblivious Transfer (OT) is a foundational cryptographic primitive that is used widely in MPC protocols. The functionality is given in Figure. 16.

Parameter: An integer l.
Functionality: Upon receiving $c \in \{0,1\}$ from Receiver and $x_1, x_2 \in \{0,1\}^l$ from Sender. $\mathcal{F}_{\mathsf{OT}}$ returns x_c to Receiver.

Figure 16: Functionality $\mathcal{F}_{\mathsf{OT}}$

A.5 Spatial Hash

Consider the case that points are located in a low-dimension space $\mathcal{U}^d$ where $\mathcal{U}$ is the universe for each dimension. We use spatial hash technique from [vBP24, GRS22] to separate the entire space into hyper-cubes with side length l, each hyper-cubes is called a cell. For $\mathbf{x} \in \mathcal{U}^d$, we define:

$$\mathsf{Cell}_l(\mathbf{x}) := (l \cdot \lfloor \frac{x_1}{l} \rfloor, l \cdot \lfloor \frac{x_2}{l} \rfloor, \cdots, l \cdot \lfloor \frac{x_d}{l} \rfloor)$$

And L_p ball:

$$\mathsf{Ball}_{p,l}(\mathbf{x}) = \{\mathbf{y} : |\mathbf{x} - \mathbf{y}|_p \leq l\}$$

We have the following lemmas from [vBP24].

Lemma 3. *Suppose there are multiple L_p balls ($p < \infty$) with radius δ lying in a d-dimension space, which is separated into cells with side length 2δ. If these balls' centers are at least $2\delta d^{1/p}$ (or 2δ for $p = \infty$) apart, then for each cell, there is at most one center of the balls lying in this cell.*

Lemma 4. *Suppose there are multiple δ-radius L_p balls distributed in a d-dimension space which is separated into cells with side length 2δ. If these balls' centers are at least $2\delta(d^{1/p} + 1)$ (or 4δ for $p = \infty$) apart from each other, then there exists at most one ball intersecting with the same cell.*

Lemma 5. *Any L_∞ ball with radius δ will intersect with exactly 2^d cells in a d-dimension space. Moreover, if we denote such 2^d cells together as a block (which is a hypercube with side length 4δ), then each block is unique for each disjoint ball. In other words, any two disjoint balls must be associated with different blocks.*

B Constructions of Fuzzy PSI-CA in Low Dimension

B.1 Fuzzy PSI-CA Protocol for L_∞ Distance

We show our construction of fuzzy PSI-CA for L_∞ distance in Figure 17.

In our protocol of Figure 17, H_{γ_1} and H_{γ_2} are universal hash functions, where $\gamma_1 = \kappa + d + 2\log(\log\delta \cdot d) + \log(N_s \cdot N_r)$ and $\gamma_2 = \kappa + \log N_s$.

Parameter setting: Receiver $\mathcal{R}$ holds a set $X = \{\mathbf{x}\}$ of size N_r. Sender $\mathcal{S}$ holds a set $Y = \{\mathbf{y}\}$ of size N_s. $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^d$. A distance threshold δ. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1. $\mathcal{R}$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $\mathcal{R}$ Initiates $\mathsf{list}_{\sigma,i} = \emptyset$. For each $k \in [N_r]$:
   - For each cell $\mathcal{C}_{k'}$ intersecting $\mathsf{Ball}_{p,\delta}(\mathbf{x}_k)$, $\mathcal{R}$ sets:
     * $\{\mathsf{list}'_{\sigma,i}\}_{\sigma\in\{0,1\},i\in[d]} \leftarrow \mathsf{GetList}_\infty(\mathsf{pk}, \delta, \mathbf{x}_k, \mathcal{C}_{k'})$.
     * For each $i \in [d]$, $\sigma \in \{0,1\}$, $\mathsf{list}_{\sigma,i} \leftarrow \mathsf{list}_{\sigma,i} \cup \mathsf{list}'_{\sigma,i}$.

   Pads $\mathsf{list}_{\sigma,i}$ with dummy pairs to size $2^d N_r$.
3. $\mathcal{R}$ sets $\mathbf{E} = \{E_{\sigma,i}\}_{\sigma\in\{0,1\},i\in[d]}$, where $E_{\sigma,i} \leftarrow \mathsf{Encode}(\mathsf{list}_{\sigma,i})$.
4. $\mathcal{R}$ sends $(\mathsf{pk}, \mathbf{E})$ to $\mathcal{S}$.
5. For each $k \in [N_s]$: $\mathcal{S}$ computes $(u_{k,\sigma,i,j}, a_{k,i}) \leftarrow \mathsf{GetValue}_\infty(\mathsf{pk}, \mathbf{y}_k, \mathcal{C}_k = \mathsf{Cell}_{2\delta+1}(\mathbf{y}_k), \mathbf{E})$, sets $a_k = \sum_{i=1}^d a_{k,i}$.
6. $\mathcal{S}$ shuffles $\{u_{k,\sigma,i,j}\}$ by σ, j, then by k, and computes $A = \{\mathsf{Hash}_{\gamma_2}(a_k)\}_{k\in[N_s]}$, sends $\{u_{k,\sigma,i,j}\}$ and A to $\mathcal{R}$, where $\sigma \in \{0,1\}$,$i \in [d]$, $j \in [0, \mu_\delta]$, $k \in [N_s]$.
7. For each $k \in [N_s]$: Receiver checks if there exists $(j_i)_{i\in[d]} \in [0, \mu_\delta]^d$ such that $\mathbf{H}_{\gamma_2}\left(\sum_{i=1}^d \mathsf{Dec}_{\mathsf{sk}}(u_{i,j_i})\right) \in A$. If so, the Receiver sets $e_k = 1$; otherwise, the Receiver sets $e_k = 0$.
8. $\mathcal{R}$ output $c = \sum_{k=1}^{N_s} e_k$.

Figure 17: Fuzzy PSI-CA protocol for L_∞ distance, where the Receiver's points are 4δ apart

Theorem 1 (Correctness)**.** *If the OKVS satisfies the perfect correctness property and the independence property, then the protocol in Figure 17 outputs the correct result with overwhelming probability.*

Proof. According to Lemma 4, if the points in X are separated by at least 4δ, then there does not exist a cell $\mathcal{C}$ such that $\mathcal{C} \cap \mathsf{Ball}_{p,\delta}(\mathbf{x}) \neq \emptyset$ and $\mathcal{C} \cap \mathsf{Ball}_{p,\delta}(\mathbf{x}') \neq \emptyset$ for any $\mathbf{x}, \mathbf{x}' \in X$.

If $\|\mathbf{x} - \mathbf{y}\|_\infty \leq \delta$, then $\mathsf{Cell}_\delta(\mathbf{y}) \cap \mathsf{Ball}_{p,\delta}(\mathbf{x}) \neq \emptyset$, and there exists no other $\mathbf{x}' \in X$ such that $\mathsf{Cell}_\delta(\mathbf{y}) \cap \mathsf{Ball}_{p,\delta}(\mathbf{x}') \neq \emptyset$. Consequently, the messages related to $\mathbf{x}$ and $\mathbf{y}$ are identical to those in the fuzzy matching protocol. The correctness of the protocol therefore follows directly from Theorem 3. □

Theorem 2 (Security)**.** *The protocol in Figure 5 is secure against semi-honest adversaries, provided that the OKVS satisfies the obliviousness and independence properties, and the IND-CPA security of ADH holds.*

Proof. **Security against a corrupted Sender:** Since the size of list_i is fixed at $2^d N_r$, the simulator must encode the same number of random dummy key-value pairs for each dimension i. The remainder of the proof follows similarly to Theorem 4.

Security against a corrupted Receiver: The simulator takes $\mathbf{x}$ and c as input and randomly selects a subset $K \subseteq [N_s]$ with $|K| = c$. For each $k \in K$, the simulator sets the

message as described in the case $c = 1$ of the corrupted Receiver scenario in Theorem 4. For each $k \notin K$, the message is set as described in the case $c = 0$ of Theorem 6. The indistinguishability is derived from Theorem 4. □

B.2 Fuzzy PSI-CA Protocol for L_p Distance

We show our construction of fuzzy PSI-CA for L_p distance in Figure 18. In our protocol in Figure 18, H_{γ_1} and H_{γ_2} are universal hash functions, where $\gamma_1 = \kappa + d + 2\log(\log\delta \cdot d) + \log(N_s \cdot N_r)$ and $\gamma_2 = \kappa + \log(N_s \cdot \delta^p)$. The proof for correctness and security are similar to the L_∞ case, we omit them here.

Parameter setting: $p, q \in \mathbb{N}$. Receiver $\mathcal{R}$ holds a set $X = \{\mathbf{x}\}$ of size N_r. Sender $\mathcal{S}$ holds a set $Y = \{\mathbf{y}\}$ of size N_s. $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^d$. A distance threshold δ. An AHE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \boxplus)$.
Protocol:

1. $\mathcal{R}$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $\mathcal{R}$ Initiates $\mathsf{list}_{\sigma,i} = \emptyset$. For each $k \in [N_r]$:
   - For each cell $\mathcal{C}_{k'}$ intersecting $\mathsf{Ball}_{p,\delta}(\mathbf{x}_k)$, $\mathcal{R}$ sets:
     * $\{\mathsf{list}'_{\sigma,i}\}_{\sigma\in\{0,1\},i\in[d]} \leftarrow \mathsf{GetList}_p(\mathsf{pk}, \delta, \mathbf{x}_k, \mathcal{C}_{k'})$.
     * For each $i \in [d]$, $\sigma \in \{0,1\}$, $\mathsf{list}_{\sigma,i} \leftarrow \mathsf{list}_{\sigma,i} \cup \mathsf{list}'_{\sigma,i}$.

   Pads $\mathsf{list}_{\sigma,i}$ with dummy pairs to size $2^d N_r$.
3. $\mathcal{R}$ sets $\mathbf{E} = \{E_{\sigma,i}\}_{\sigma\in\{0,1\},i\in[d]}$, where $E_{\sigma,i} \leftarrow \mathsf{Encode}(\mathsf{list}_{\sigma,i})$.
4. $\mathcal{R}$ sends $(\mathsf{pk}, \mathbf{E})$ to $\mathcal{S}$.
5. For each $k \in [N_s]$:
   - $\mathcal{S}$ computes $(u_{k,\sigma,i,j}, a_{k,i}) \leftarrow \mathsf{GetValue}_p(\mathsf{pk}, \mathbf{y}_k, \mathcal{C}_k = \mathsf{Cell}_{2\delta+1}(\mathbf{y}_k), \mathbf{E})$, and sets $a_k = \sum_{i=1}^d a_{k,i}$.
6. $\mathcal{S}$ shuffles $\{u_{k,\sigma,i,j}\}$ by σ, j, then by k, and sends to $\mathcal{R}$, where $\sigma \in \{0,1\}$,$i \in [d]$, $j \in [0, \mu_\delta]$, $k \in [N_s]$.
7. For each $k \in [N_s]$:

   For $\forall(j_1, j_2, \cdots, j_d) \in [0, \mu_\delta]^d, (\sigma_1, \sigma_2, \cdots, \sigma_d) \in \{0,1\}^d$:
   * $\mathcal{R}$ and $\mathcal{S}$ run $\mathcal{F}_{\mathsf{IFMat}}$ with inputs $\sum_{i=1}^d(\mathsf{Dec}_{\mathsf{sk}}(u_{k,\sigma_i,i,j_i}))$ and $a_k + \frac{1}{2}\delta^p$ respectively, the threshold is $\frac{1}{2}\delta^p$.
   * $\mathcal{R}$ gets output e_i.
8. $\mathcal{R}$ output $c = \sum_{i=1}^d e_i$.

Figure 18: Fuzzy PSI-CA protocol for L_p distance for finite p where the Receiver's points are $4\delta(d^{1/p} + 1)$ apart

Our two protocols presented in Figure 17 and Figure 18 require that the points of the Receiver are at least $4\delta(d^{1/p} + 1)$ for $p < \infty$ (or 4δ for $p = \infty$) apart from each other. We can also consider the case of disjoint balls where the points of the Receiver are at least $2\delta(d^{1/p} + 1)$ for $p < \infty$ (or 2δ for $p = \infty$) apart, a different spatial hash can be utilized

as in [vBP24] . More precisely, the Receiver identifies each of its balls as a unique block of side length $4\delta + 2$ that the ball lies in and uses it as the input Δ for $\mathsf{GetList}()$. Then the Sender iterates all possible 2^d blocks containing the cell where its point is located to decode the OKVS. The rest of the protocol is adapted accordingly. In this case, the communication cost for the Receiver is reduced, but the communication cost for the Sender is increased. Moreover, it increases the computation cost for the Receiver by a factor of 2^d.

C Proofs of Fuzzy Matching Protocols

C.1 Fuzzy Matching for L_∞ Distance

Theorem 3 (Correctness)**.** *If the OKVS satisfies the perfect correctness and independence properties, then the protocol described in Figure 5 outputs the correct value with overwhelming probability.*

Proof. **Case 1:** $||\mathbf{x} - \mathbf{y}||_\infty \leq \delta$**.** In this case, for every i, there exists $j_i \in [0, \mu_\delta]$ such that $\mathsf{Prefix}(y_i, j_i) \in \mathsf{Decompose}(\{x' : ||x' - x_i||_\infty \leq \delta\})$. The value $u_{i,j_i} = \mathsf{Decode}(\mathsf{E}_i, \mathsf{H}_{\gamma_1}(\Delta || y^*_{i,j_i}))$ can be expressed as $\mathsf{Enc}_{\mathsf{pk}}(0)$, and $u^*_{i,j_i} = \mathsf{Enc}_{\mathsf{pk}}(a_i)$. The receiver computes $\sum_{i=1}^d \mathsf{Dec}_{\mathsf{sk}}(u^*_{i,j_i}) = \sum_{i=1}^d a_i$ and outputs the correct result.

Case 2: $||\mathbf{x} - \mathbf{y}||_\infty > \delta$**.** In this case, there exists an index $i' \in [d]$ such that for all $j \in [0, \mu_\delta]$, $\mathsf{Prefix}(y_{i'}, j) \notin \mathsf{Decompose}(\{x' : ||x' - x_{i'}||_\infty \leq \delta\})$. By the independence property of OKVS, the values $(u_{i',j})$ are random for all $j \in [0, \mu_\delta]$. Consequently, for any $(j_1, j_2, \ldots, j_d) \in [0, \mu_\delta]^d$, the sum $\sum_{i=1}^d \mathsf{Dec}_{\mathsf{sk}}(u^*_{i,j_i})$ is random in $\mathcal{P}$. Therefore, the receiver outputs 1 with probability less than $\mathsf{negl}(\kappa)$, since $\gamma_2 = \kappa + d \cdot \log\log\delta$ and the inputs to the hash function $\mathbf{H}_{\gamma_2}$ are random. □

Theorem 4 (Security)**.** *The protocol in Figure 5 is secure against semi-honest adversaries if the OKVS satisfies the obliviousness and independence properties, and the IND-CPA of ADH holds.*

Proof. We analyze two cases and construct corresponding simulators for each.

Security against a Corrupted Sender. We describe a simulator for the case where the Sender is corrupted. In this scenario, the simulator only needs to simulate the message sent by the Receiver to the Sender in Step 3.

Simulator:

1*. The simulator begins by receiving the Sender's input $\mathbf{y}$.

2*. The simulator encodes lists with ω_δ random dummy items for all $i \in [d]$, where $key_{i,j} \leftarrow_\$ \{0,1\}^{\gamma_1}$ and $val_{i,j} = c \leftarrow_\$ \mathcal{C}$.

We now argue that the adversary's view in the real execution of the protocol is indistinguishable from the simulated view. The proof proceeds via the following hybrid games, where the simulator in $\mathcal{H}_2$ corresponds to the one we constructed.

$\mathcal{H}_0$: The real-world execution of the protocol, where the simulator acts exactly as an honest Receiver.

$\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except that the simulator encodes lists with random dummy values in Step 2, i.e., $value \leftarrow_\$ \mathcal{V}$.

$\mathcal{H}_2$: Identical to $\mathcal{H}_1$, except that the simulator encodes lists with random dummy keys in Step 2, i.e., $key \leftarrow_\$ \mathcal{K}$.

$\mathcal{H}_0 - \mathcal{H}_1$**:** By the IND-CPA security of our AHE, the value c is indistinguishable from a uniformly random pair for the Sender. Hence, the hybrid games $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable.

$\mathcal{H}_1 - \mathcal{H}_2$**:** By the obliviousness property of OKVS, the vectors encoded with distinct keys and the same values are indistinguishable. The only difference between $\mathcal{H}_1$ and $\mathcal{H}_2$ is the selection of keys. Therefore, $\mathcal{H}_1$ and $\mathcal{H}_2$ are indistinguishable.

In conclusion, the distribution of the simulated view is computationally indistinguishable from that of the real execution.

Security against a Corrupted Receiver. We now describe a simulator for the case where the Receiver is corrupted. In this scenario, the simulator needs to simulate the message sent by the Sender to the Receiver in Step 5.

Simulator:

1′. The simulator begins by receiving the Receiver's input $\mathbf{x}$ and the output of the functionality $c = \mathcal{F}_{\mathsf{FMat}}(\mathbf{x}, \mathbf{y})$.

2′. If $c = 1$ (i.e., $\mathsf{dist}(\mathbf{x}, \mathbf{y}) \leq \delta$), the simulator selects $j_i \leftarrow_{\$} [0, \mu_\delta]$ and sends $\{(u'_{i,j}, r')\}$ to the Receiver, where:

$$u'_{i,j} = \begin{cases} \mathsf{Enc}_{\mathsf{pk}}(a_i) & \text{if } j = j_i \ (r \leftarrow_{\$} \mathbb{Z}_q), \\ \text{random} & \text{if } j \neq j_i, \end{cases}$$

and

$$r' = \mathsf{H}_{\gamma_2}\left(\sum_{i=1}^{d} a_i\right).$$

3′. If $c = 0$ (i.e., $\mathsf{dist}(\mathbf{x}, \mathbf{y}) > \delta$), the simulator sends random items $\{(u'_{i,j}, R')\}$ to the Receiver, where $u'_{i,j} \leftarrow_{\$} \mathcal{C}$ and $r' \leftarrow_{\$} \{0,1\}^{\gamma_2}$.

To prove security, we present the following hybrid games. The simulator in $\mathcal{H}_2$ corresponds to the one we constructed.

$\mathcal{H}_0$: The real-world execution of the protocol, where the simulator acts exactly as an honest Sender.

$\mathcal{H}'_0$: Identical to $\mathcal{H}_0$, except that when $c = 1$, the simulator modifies the message sent in Step 5. For an honest Sender running $\mathsf{GetValue}_\infty(\mathsf{pk}, \mathbf{y}, 0^\kappa, \mathbf{E})$ in Step 4, let:

$$y''_{i,j} = \mathsf{Prefix}(y_i, j), \quad j \in [0, \mu_\delta],$$

and

$$\mathsf{X}_i = \mathsf{Decompose}(\{x' : ||x' - x_i||_\infty \leq \delta\}).$$

Since $c = 1$, there exists $j_i \in [0, \mu_\delta]$ for all i such that $y''_{i,j} \in \mathsf{X}_i$ if $j = j_i$, and $y''_{i,j} \notin \mathsf{X}_i$ if $j \neq j_i$. For all i, j, the simulator sets the message as:

- $u''_{i,j} = \mathsf{Enc}_{\mathsf{pk}}(a'_i)$, where $a'_i \leftarrow_{\$} \mathcal{P}$, if $j = j_i$,
- $u''_{i,j} \leftarrow_{\$} \mathcal{C}$, if $j \neq j_i$,
- $r' = \mathsf{H}_{\gamma_2}\left(\sum_{i=1}^{d} a'_i\right)$.

$\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except that when $c = 1$, the simulator chooses j_i randomly as in Step 2′.

$\mathcal{H}'_1$: Identical to $\mathcal{H}_1$, except that when $c = 0$, the simulator modifies the message sent in Step 5. The definitions of $y''_{i,j}$ and X_i follow $\mathcal{H}'_0$. Since $c = 0$, there exists an index set $J = \{j_i : j_i \in [0, \mu_\delta], i \in I \subsetneq [d]\}$ such that $|J| < d$, $y''_{i,j} \in \mathsf{X}_i$ if $j = j_i$, and $y''_{i,j} \notin \mathsf{X}_i$ if $j \neq j_i$. For all i, j, the simulator sets the message as:

- $u''_{i,j} = \mathsf{Enc}_{\mathsf{pk}}(a'_i)$, where $a'_i \leftarrow_{\$} \mathcal{P}$, if $i \in I$ and $j = j_i$,
- $u''_{i,j} \leftarrow_{\$} \mathcal{C}$, if $j \neq j_i$ or $i \notin I$,
- $r' = \mathsf{H}_{\gamma_2}\left(\sum_{i=1}^{d} a'_i\right)$, where $a_i \leftarrow_{\$} \mathcal{P}$ for $i \notin I$.

$\mathcal{H}_2$: Identical to $\mathcal{H}_1$, except that when $c = 0$, the simulator acts as in Step $3'$.

In the real-world execution, $u_{i,j} \leftarrow \mathsf{Decode}(\mathsf{E}_i, \mathsf{H}_{\gamma_1}(\Delta||y^*_{i,j}))$, and the message consists of $u^*_{i,j} = \mathsf{Enc}_{\mathsf{pk}}(a_i)$ and $r = \mathsf{H}_{\gamma_2}\left(\sum_{i=1}^{d} a_i\right)$.

$\mathcal{H}_0 - \mathcal{H}'_0$: For $j = j_i$, we have $u_{i,j} = \mathsf{Enc}_{\mathsf{pk}}(0)$, and $\mathsf{Enc}_{\mathsf{pk}}(r) \approx_c g \leftarrow_{\$} \mathcal{C}$ for all $r \leftarrow_{\$} \mathcal{P}$ by the IND-CPA security of AHE. Thus, $u''_{i,j} = \mathsf{Enc}_{\mathsf{pk}}(a'_i) \approx_s u^*_{i,j}$, which matches the message in the real world. For $j \neq j_i$, $(u_{i,j}, v_{i,j}) \approx_c (g_1, g_2) \approx_s (u^*_{i,j}, v^*_{i,j})$ by the independence property of OKVS. Hence, $\mathcal{H}_0$ and $\mathcal{H}'_0$ are indistinguishable.

$\mathcal{H}'_0 - \mathcal{H}_1$: The only difference between $\mathcal{H}'_0$ and $\mathcal{H}_1$ is the random selection of j_i in $\mathcal{H}_1$. Since $(u_{i,j}, v_{i,j})$ are shuffled with respect to j in Step 5, $\mathcal{H}'_0$ and $\mathcal{H}_1$ are identical.

$\mathcal{H}_1 - \mathcal{H}'_1$: For $j = j_i$, we have $u_{i,j} = \mathsf{Enc}_{\mathsf{pk}}(0)$. The indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}'_1$ follows from the same analysis as $\mathcal{H}_0 - \mathcal{H}'_0$.

$\mathcal{H}'_1 - \mathcal{H}_2$: The differences between $\mathcal{H}'_1$ and $\mathcal{H}_2$ lie in the selection of $(u''_{i,j}, v''_{i,j})$ when $j = j_i$ and R''. Since $|I| < d$, $(\sum_{i=1}^{d} a'_i, \mathsf{Enc}_{\mathsf{pk}}(a'_l))_{l \in I} \approx_s$ (uniformly random items). Thus, $\mathcal{H}'_1$ and $\mathcal{H}_2$ are indistinguishable.

In conclusion, the distribution of the simulated view is computationally indistinguishable from that of the real execution. □

C.2 Fuzzy Matching for L_p Distance

Theorem 5 (Correctness)**.** *If OKVS satisfies the perfect correctness property and the independence property, then the protocol in Figure 8 outputs correctly with overwhelming probability.*

Proof. **For the case $||\mathbf{x} - \mathbf{y}||_p \leq \delta$:** in this case, since $||\mathbf{x} - \mathbf{y}||_\infty \leq ||\mathbf{x} - \mathbf{y}||_p$, there exists $j_i \in [0, \mu_\delta]$ and $\sigma_i \in \{0, 1\}$ for every i, such that $\mathsf{Prefix}(y_i, j_i) \in \mathsf{Decompose}(\{x'_i : x'_i \leq x_i, |x'_i - x_i| \leq \delta\})$ if $\sigma_i = 0$, or $\mathsf{Prefix}(y_i, j_i) \in \mathsf{Decompose}(\{x'_i : x'_i > x_i, |x'_i - x_i| \leq \delta\})$ if $\sigma_i = 1$.

In the case $\sigma_i = 0$, then $y_i \leq x_i$, and $u_{i,j} = \mathsf{Decode}(\mathsf{E}_i, \mathsf{H}_{\gamma_1}(\Delta||\sigma_i||y_{i,j_i}))$ can be represented as $(\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|)||\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^2)|| \cdots ||\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^p)$, where $x^* = \mathsf{UpBound}(x_{\sigma,i,j_i})$, $y_{i,j_i} = \mathsf{Prefix}(y_i, j_i)$. We can see that $|x_i - y_i| = |y_i - x^*| + |x^* - x_i|$. Then for $u^*_{\sigma_i,i,j_i}$ computed in step 2 in algorithm 7, we have

$$
\begin{aligned}
u^*_{\sigma,i,j_i} &= \boxplus^{p}_{\mathsf{pk}\,t=1}\left(\tbinom{p}{t}|y_i - x^*|^{p-t} \boxtimes_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^t)\right) \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i) \\
&\quad \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(e^p) \\
&= \mathsf{Enc}_{\mathsf{pk}}\left(\sum_{t=0}^{p} \tbinom{p}{t}|y_i - x^*|^{p-t}|x^* - x_i|^t\right) \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i) \\
&= \mathsf{Enc}_{\mathsf{pk}}(|x_i - y_i|^p + a_i)
\end{aligned}
$$

Then if $||\mathbf{x} - \mathbf{y}||_p \leq \delta$, the Sender computes $\sum_{i=1}^{d}(\mathsf{Dec}_{\mathsf{sk}}(u^*_{\sigma_i,i,j_i})) = ||\mathbf{x} - \mathbf{y}||^p + \sum_{i=1}^{d} a_i = ||\mathbf{x} - \mathbf{y}||^p + a$ in step 7. The inputs to IFMat are $||\mathbf{x} - \mathbf{y}||^p + a$ and a. Since $||\mathbf{x} - \mathbf{y}||^p + a \in [a, a + \delta^p]$, the Receiver outputs 1.

For the case $||\mathbf{x} - \mathbf{y}||_p > \delta$: If $||\mathbf{x} - \mathbf{y}||_\infty > \delta$, there exists $i' \in [d]$ such that $\mathsf{Prefix}(y_i, j) \notin \mathsf{Decompose}\{x'_i : x'_i \leq x'_i, |x'_i - x_{i'}| \leq \delta\} \cup \mathsf{Decompose}\{x'_i : x'_i > x_{i'}, |x'_i - x_{i'}| \leq \delta\}$ for all $j \in [0, \mu_\delta]$. Then, according to the independence property of OKVS, $u_{\sigma,i',j}$

are random items for all j, σ. Therefore, for all $(j_1, j_2, \cdots, j_d) \in [0, \mu_\delta]^d$ and for all $(\sigma_1, \sigma_2, \cdots, \sigma_d) \in \{0,1\}^d$, $\sum_{i=1}^{d}(\mathsf{Dec}_{\mathsf{sk}}(u^*_{\sigma_i,i,j_i}))$ is random in $\mathcal{P}$. Hence, the Receiver outputs 1 with probability less than $negl(\lambda)$ according to $\mathcal{F}_{\mathsf{IFMat}}$.

If $||\mathbf{x}-\mathbf{y}||_\infty \leq \delta$, similar to the case $||\mathbf{x}-\mathbf{y}||_p \leq \delta$, the inputs to IFMat are $||\mathbf{x}-\mathbf{y}||^p + a$ and a. Receiver outputs 1 with probability less than $negl(\lambda)$. □

Theorem 6 (Security)**.** *The protocol in Figure 8 is secure against semi-honest adversaries if the OKVS satisfies the obliviousness and independence properties, and the IND-CPA of ADH holds.*

Proof. We analyze two cases and construct simulators accordingly.

Security against a corrupted Sender: We describe a simulator for a corrupted Sender. In this case, the simulator only needs to simulate the message that the Receiver sends to the Sender in step 3.

Simulator:

1*. The simulator starts by receiving the Sender's input $\mathbf{y}$.

2*. The simulator encodes lists with random dummy items for all $i \in [d]$ as follows: $key_{i,j} \leftarrow_\$ \{0,1\}^{\gamma_1}$ and $val_{i,j} = (g_1, g_2, \ldots, g_p) \leftarrow_\$ \mathcal{C}^p$.

The analysis is similar to the security proof for L_∞ distance fuzzy matching in Theorem 4 against a corrupted Sender, so we omit the proof here. Under the IND-CPA of ADH and the obliviousness property of OKVS, the distribution of the simulated view is computationally indistinguishable from that of the real execution.

Security against a corrupted Receiver: We describe a simulator for a corrupted Receiver. In this case, the simulator needs to simulate the message that the Sender sends to the Receiver in step 5 and the inputs to $\mathcal{F}_{\mathsf{IFMat}}$.

Simulator:

1′. The simulator starts by receiving the Receiver's input $\mathbf{x}$ and the output of the functionality $c = \mathcal{F}_{\mathsf{FMat}}(\mathbf{x}, \mathbf{y})$. And generates $(\mathsf{pk}', \mathsf{sk}') \leftarrow_\$ \mathsf{Gen}(1^\lambda)$, adds to view of adversary.

2′. If $c = 1$, which means $\mathsf{dist}(\mathbf{x}, \mathbf{y}) \leq \delta$, the simulator selects $\sigma_i \leftarrow_\$ \{0,1\}$ and $j_i \leftarrow_\$ [0, \mu_\delta]$ for all i, and sends $\{u'_{\sigma,i,j}\}$ to the Receiver, where:

$$u'_{\sigma,i,j} = \begin{cases} \mathsf{Enc}_{\mathsf{pk}'}(a_i) & \text{if } j = j_i \text{ and } \sigma = \sigma_i \quad (a_i \leftarrow_\$ \mathcal{P}) \\ \text{random} & \text{if } j \neq j_i \text{ or } \sigma \neq \sigma_i \end{cases}$$

3′. If $c = 0$, which means $\mathsf{dist}(\mathbf{x}, \mathbf{y}) > \delta$, the simulator sends random items $u'_{\sigma,i,j}$ to the Receiver and selects $a_i \leftarrow_\$ \mathcal{P}$.

4′. The simulator invokes simulator for $\mathcal{F}_{\mathsf{IFMat}}$ with input

$$(\{\sum_{i=1}^{d} \mathsf{Dec}_{\mathsf{sk}}(u'_{\sigma_i,i,j_i})\}_{\sigma_i \in \{0,1\}, j_i \in [0,\mu_\delta]}, c)$$

Adds the output to the view of adversary.

We present the following hybrid games to prove the security. The simulator in $\mathcal{H}_2$ is the one we constructed.

$\mathcal{H}_0$: The real-world execution of the protocol, where the simulator acts exactly as an honest Sender.

$\mathcal{H}_0'$: Identical to $\mathcal{H}_0$, except that when $c = 1$, the simulator changes the message sent in step 5. For an honest Sender who runs $\mathsf{GetValue}_p(\mathsf{pk}, \mathbf{y}, 0^\kappa, \mathbf{E})$ in step 4, we denote:

$$y''_{i,j} = \mathsf{Prefix}(y_i, j), \quad j \in [0, \mu_\delta]$$

$$\mathsf{X}_{i,0} = \bigcup_{i\in[d]} (\mathsf{Decompose}(\{x'_i : x'_i \le x_i, ||x'_i - x_i||_p \le \delta\}))$$

$$\mathsf{X}_{i,1} = \bigcup_{i\in[d]} (\mathsf{Decompose}(\{x'_i : x'_i > x_i, ||x'_i - x_i||_p \le \delta\}))$$

Clearly, $\mathsf{X}_0 \cap \mathsf{X}_1 = \emptyset$. Since $c = 1$, there exist $j_i \in [0, \mu_\delta]$ and $\sigma_i \leftarrow_\$ \{0, 1\}$ for all i, such that $y''_{i,j} \in \mathsf{X}_{i,\sigma_i}$ if $j = j_i$, and $y''_{i,j} \notin \mathsf{X}_{i,0} \cup \mathsf{X}_{i,1}$ if $j \neq j_i$.

For all i, j, the simulator sets the message:

- $u''_{\sigma,i,j} = \mathsf{Enc}_{\mathsf{pk}}(|y_i - x_i|^p + a_i)$, where $a_i \leftarrow_\$ \mathcal{P}$, if $j = j_i$ and $\sigma = \sigma_i$.
- $u''_{\sigma,i,j} \leftarrow_\$ \mathcal{C}$, if $j \neq j_i$ or $\sigma \neq \sigma_i$.

The simulator invokes simulator for $\mathcal{F}_{\mathsf{IFMat}}$ as in step $4'$.

$\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except that when $c = 1$, the simulator acts as in step $2'$ and runs $\mathcal{F}_{\mathsf{IFMat}}$ as in step $4'$.

$\mathcal{H}_1'$: Identical to $\mathcal{H}_1$, except that when $c = 0$, the simulator changes the message sent in step 5. The notation $y''_{i,j}$ and X follows $\mathcal{H}_0'$.

Since $c = 0$, in the case $||\mathbf{y} - \mathbf{x}||_\infty \le \delta$, we define the message as in $\mathcal{H}_0'$.

In the case $||\mathbf{y} - \mathbf{x}||_\infty > \delta$, there exists an index set $J = \{(j_i, \sigma_i) : j_i \in [0, \mu_\delta], \sigma_i \in \{0, 1\}, i \in I \subsetneq [d]\}$, such that $|I| < d$, and $y''_{i,j} \in \mathsf{X}_{\sigma_i}$ if $(j, \sigma) \in J$; $y''_{i,j} \notin \mathsf{X}$ if $(j, \sigma) \notin J$.

For all i, j, the simulator sets the message:

- $u''_{\sigma,i,j} = \mathsf{Enc}_{\mathsf{pk}}(|y_i - x_i|^p + a_i)$, where $a_i \leftarrow_\$ \mathcal{P}$, if $(j, \sigma) \in J$.
- $u''_{\sigma,i,j} \leftarrow_\$ \mathcal{C}$, if $(j, \sigma) \notin J$.

The simulator invokes simulator for $\mathcal{F}_{\mathsf{IFMat}}$ as in step $4'$.

$\mathcal{H}_2$: Identical to $\mathcal{H}_1$, except that when $c = 0$, the simulator acts as in step $3'$ and runs $\mathcal{F}_{\mathsf{IFMat}}$ as in step $4'$.

The indistinguishability between the hybrid games is similar to the security proof for L_∞ distance fuzzy matching in Theorem 4 against a corrupted Receiver.

In conclusion, the distribution of the simulated view is computationally indistinguishable from that of the real execution. □

D Proofs of Fuzzy PSI-CA in High Dimension

D.1 Private Index Search

Theorem 7 (Correctness)**.** *The private index search protocol* Π_{PIS} *in Figure 9 satisfies the correctness requirement specified in Definition 3.*

Proof. Suppose $b = a_j \in S$. Let j be represented in binary string as $\overline{j_l \cdots j_2 j_1}$. By the functionality of the secret shared private equation test (ssPET) protocol, if $b = a_i$, then $s_i^* \oplus t_i^* = 1$, otherwise 0, which means $s_i^* \oplus t_i^* = 0$ for $i = j$ and otherwise $s_i^* \oplus t_i^* = 0$. Notice that if $j \in S_k$, $j_i = 1$, else $j_i = 0$, then:

- $s_0 \oplus t_0 = \oplus_{i=0}^{n-1}(s_i^* \oplus t_i^*) = 1$ (verification bit)
- $s_i \oplus t_i = \oplus_{i \in S_k}(s_i^* \oplus t_i^*) = j_i$ for all $i \in [l]$ (bit-wise index sharing)

This implies $s \oplus t = j$. Through the Oblivious Transfer (OT) mechanism, $h = s_0 \cdot r \oplus q_0 = s_0 \cdot r \oplus (t_0 \oplus 1) \cdot r \oplus t = t$. Therefore, $\mathsf{idx} = t \oplus s = j$, guaranteeing that $\mathcal{R}$ outputs the correct index. □

Theorem 8 (Randomness)**.** *The private index search protocol* Π_{PIS} *in Figure 9 satisfies the randomness requirement defined in Definition 3.*

Proof. If $b \notin S$, the ssPET protocol ensures:

- $s_0 \oplus t_0 = 0$ (negative verification)
- $s_i \oplus t_i = 0$ for all $i \in [l]$ (null index sharing)

Thus $s \oplus t = 0$. Through OT computation, $h = s_0 \cdot r \oplus q_0 = s_0 \cdot r \oplus (t_0 \oplus 1) \cdot r \oplus t = r \oplus t$. This results in $\mathsf{idx} = t \oplus s = r$, where r is a uniformly random bit string. Consequently, $\mathcal{R}$ outputs a random value indistinguishable from genuine indices. □

Theorem 9 (Security)**.** *The private index search protocol* Π_{PIS} *in Figure 9 satisfies the security definition in Definition 3 in the* $\mathcal{F}_{\mathsf{ssPET}}$ *and* $\mathcal{F}_{\mathsf{OT}}$ *hybrid model.*

Proof. **Security against a corrupted Sender:** We construct a simulator for a corrupted Sender.

Simulator $\mathsf{SIM}_{\mathsf{PIS}}^{\mathcal{S}}$**:**

1′. The simulator starts with an element $b \in \mathcal{U}$.

2′. The simulator selects n random bits $t_i^{*\prime} \leftarrow_{\$} \{0,1\}$, and invokes the ssPET simulator $\mathsf{SIM}_{\mathsf{ssPET}}^{\mathcal{P}_1}$ for P_1. The simulator calls $\mathsf{SIM}_{\mathsf{ssPET}}^{\mathcal{P}_1}(b, t_i^{*\prime})$ for each of the n ssPET invocations. The outputs form part of the Sender's view.

3′. The simulator samples $r' \leftarrow_{\$} \{0,1\}^l$, sets $q_0 = (t_0' \oplus 1) \cdot r' \oplus t'$ and $q_1 = r' \oplus q_0$, and invokes the OT simulator $\mathsf{SIM}_{\mathsf{OT}}^{\mathcal{S}}$ for the Sender. The simulator calls $\mathsf{SIM}_{\mathsf{OT}}^{\mathcal{S}}(q_0, q_1)$. The outputs form part of the Sender's view.

We define the following hybrid games, where hybrid game $\mathcal{H}_3$ outputs the view in the ideal world, denoted as $\mathsf{SIM}_{\mathsf{PIS}}^{\mathcal{S}}(b)$:

$\mathcal{H}_0$: The real-world execution of the protocol, where the simulator acts as an honest Receiver.

$\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except the simulator invokes the ssPET simulator $\mathsf{SIM}_{\mathsf{ssPET}}^{\mathcal{P}_1}(b, t_i)$ for the n ssPET invocations.

$\mathcal{H}_2$: Identical to $\mathcal{H}_1$, except the simulator acts as in step 2′.

$\mathcal{H}_3$: Identical to $\mathcal{H}_2$, except the simulator acts as in step 3′.

By the security of ssPET and OT, to prove all games are indistinguishable, it suffices to show $\mathcal{H}_1$ is indistinguishable from $\mathcal{H}_2$. According to the ssPET functionality, the adversary cannot distinguish t_i from t_i', making them indistinguishable.

Thus, we have shown $\mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, b, S) \approx_c \mathsf{SIM}_{\mathsf{PIS}}^{\mathcal{S}}(b)$, and similarly, $\mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, b, S') \approx_c \mathsf{SIM}_{\mathsf{PIS}}^{\mathcal{S}}(b)$. Therefore, $\mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, b, S') \approx_c \mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, b, S)$, proving security against a corrupted Sender.

Security against a corrupted Receiver: We construct a simulator for a corrupted Receiver.

Simulator $\mathsf{SIM}_{\mathsf{PIS}}^{\mathcal{R}}$**:**

1*. The simulator starts with the Receiver's input $S = (a_i)_{i\in[0,n-1]}$ and output idx.

2*. The simulator selects n random bits $s_i^{*\prime} \leftarrow_{\$} \{0,1\}$ for $i \in [0, n-1]$, and invokes the ssPET simulator $\mathsf{SIM}^{\mathcal{P}_0}_{\mathsf{ssPET}}$ for P_0. The simulator calls $\mathsf{SIM}^{\mathcal{P}_1}_{\mathsf{ssPET}}(S_i, s_i^{*\prime})$ for each of the n ssPET invocations. The outputs form part of the Receiver's view.

3′. The simulator sets $s_0 = \oplus_{i\in[0,n-1]} s_i^{*\prime}$, invokes the OT simulator $\mathsf{SIM}^{\mathcal{R}}_{\mathsf{OT}}$ for the Receiver, calling $\mathsf{SIM}^{\mathcal{R}}_{\mathsf{OT}}(s_0', \mathsf{idx})$. The outputs form part of the Receiver's view.

We define the following hybrid games, where hybrid game $\mathcal{H}_3$ outputs the view in the ideal world:

$\mathcal{H}_0$: The real-world execution of the protocol, where the simulator acts as an honest Receiver.

$\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except the simulator invokes $\mathsf{SIM}^{\mathcal{P}_1}_{\mathsf{ssPET}}(S_i, s_i)$ for the n ssPET invocations.

$\mathcal{H}_2$: Identical to $\mathcal{H}_0$, except the simulator acts as in step 2*.

$\mathcal{H}_3$: Identical to $\mathcal{H}_0$, except the simulator acts as in step 3′.

The proof follows similarly to the case for a corrupted Sender and is therefore omitted here. □

D.2 Fuzzy Mapping

Theorem 10 (Correctness)**.** *The fuzzy mapping protocol in Figure 11 satisfies the correctness property defined in Definition 2.*

Proof. We need to prove that for any $\mathbf{x} = (x_1, \ldots, x_d) \in X$ and any $\mathbf{y} = (y_1, \ldots, y_d) \in Y$, if $|\mathbf{x} - \mathbf{y}|_\infty \le \delta$, then $\mathsf{id}_{\mathbf{x}} = \mathsf{id}_{\mathbf{y}}$.

In the GetID algorithm, all prefixes representing the interval $[x_i - \delta, x_i + \delta]$ are encoded to the same value. Assume they are encoded to the value $\mathsf{Enc}_{\mathsf{pk}}(r_i)||\mathsf{Enc}_{\mathsf{pk}}(0)$. If $|y_i - x_i| \le \delta$, there exists $j_i \in [\mu_\delta]$ for all i such that the Sender $\mathcal{S}$ will obtain $(u_{i,j_i}||v_{i,j_i}) = (\mathsf{Enc}_{\mathsf{pk}}(r_i)||\mathsf{Enc}_{\mathsf{pk}}(0))$. Assuming, without loss of generality, that the Sender $\mathcal{S}$ and Receiver $\mathcal{R}$ do not shuffle the set, by the correctness of the private index search protocol Π_{PIS}, the Receiver $\mathcal{R}$ obtains j_i from Π_{PIS} in step 4. Consequently, the value w computed by the Receiver is:

$$w = \sum_{i=1}^{d} \mathsf{Dec}_{\mathsf{sk}}(u'_{i,j_i}) = \sum_{i=1}^{d} (r_i + \mathsf{mask}_i) = \mathsf{id}_{\mathbf{x}} + \sum_{i=1}^{d} \mathsf{mask}_i$$

Thus, we have:

$$\mathsf{id}_{\mathbf{y}} = w - \sum_{i=1}^{d} \mathsf{mask}_i = \mathsf{id}_{\mathbf{x}}$$

This completes the proof. □

Theorem 11 (Distinctiveness)**.** *The fuzzy mapping protocol in Figure 11 satisfies the distinctiveness property defined in Definition 2.*

Proof. We need to prove that for any $\mathbf{x} = (x_1, \ldots, x_d) \in X$ and $\mathbf{x}' = (x'_1, \ldots, x'_d) \in X$, if $\mathbf{x} \neq \mathbf{x}'$, then $\mathsf{id}_{\mathbf{x}} \neq \mathsf{id}_{\mathbf{x}'}$.

Recall the assumption that any L_∞ balls of the Receiver must have disjoint projections in some dimension. This means there exists an index $i \in [d]$ such that the intervals $[x_i - \delta, x_i + \delta]$ and $[x'_i - \delta, x'_i + \delta]$ are disjoint for all other $\mathbf{x}' \in X$.

After running step 1 of the GetID algorithm, the Receiver has $([x_i - \delta, x_i + \delta], r_i) \in \mathsf{interval}_i$. Suppose $x'_i \in U'$ and $(U, r') \in \mathsf{interval}_i$. Since $[x_i - \delta, x_i + \delta] \cap U' = \emptyset$, the values r_i and r' are independently sampled from the universe $\mathcal{U}$.

Therefore, the identifiers $\mathsf{id}_{\mathbf{x}}$ and $\mathsf{id}_{\mathbf{x}'}$ can be considered as independently chosen from $\mathcal{U}$. The probability that $\mathsf{id}_{\mathbf{x}} = \mathsf{id}_{\mathbf{x}'}$ is:

$$\Pr(\mathsf{id}_{\mathbf{x}} = \mathsf{id}_{\mathbf{x}'}) = \frac{1}{2^u} \leq \mathrm{negl}(\lambda)$$

where $\mathrm{negl}(\lambda)$ denotes a negligible function in the security parameter λ. This completes the proof. □

Theorem 12 (Security)**.** *The fuzzy mapping protocol in Figure 11 satisfies the security property defined in Definition 2 in the $\mathcal{F}_{\mathsf{PIS}}$ hybrid model, and if the independence properties of OKVS and the IND-CPA of AHE holds.*

Proof. **Security for a corrupted Sender:** We need to prove that for any X_1, X_2, the views of the Sender in the real-world execution with X_1 and X_2 are computationally indistinguishable, i.e., $\mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, X_1, Y) \approx_c \mathsf{Real}_{\Pi,\mathcal{S}}(\kappa, \lambda, X_2, Y)$. We define the following hybrid games with a simulator acting as the Receiver:

1′. The simulator starts with a set $Y \subset \mathcal{U}^d$ and its corresponding identifiers, $\mathsf{ID}(Y)$.

2′. The simulator encodes $\hat{\mathsf{list}}_i$ with dummy key-value pairs in step 2 and sends it to the Sender.

3′. The simulator samples $\hat{\mathsf{mask}}_{k,i}, \hat{\mathsf{mask}}'_{k,i} \leftarrow_{\$} \mathcal{P}$ and sets them as the randomness of the Sender. The simulator invokes the PIS simulator $\mathsf{SIM}_{\Pi_{\mathsf{PIS}},\mathsf{S}}$ for the Sender, which is $\mathsf{SIM}_{\Pi_{\mathsf{PIS}},\mathsf{S}}(\hat{\mathsf{mask}}_{k,i})$ for every $k \in [N_s]$, $i \in [d]$. Adds to the view of Sender.

4′. The simulator sends $\hat{w}_k = \mathsf{id}_{\mathbf{y}_k} + \sum_{i=1}^{d} \hat{\mathsf{mask}}_{k,i}$ in step 5.

The hybrid games are as follows:

$\mathcal{H}'_0$: The real-world execution of the protocol, where the simulator acts exactly as an honest Receiver.

$\mathcal{H}'_1$: Identical to $\mathcal{H}'_0$, except the simulator acts as in step 2'. This is computationally indistinguishable from $\mathcal{H}'_0$ due to the IND-CPA security of AHE and the obliviousness of OKVS.

$\mathcal{H}'_2$: Identical to $\mathcal{H}'_1$, except the simulator acts as in step 3'. This is indistinguishable from $\mathcal{H}'_1$ because the masks are sampled randomly and are thus indistinguishable from the real masks. Additionally, by the security of the PIS simulator:

$$\mathsf{Real}_{\Pi_{\mathsf{PIS}},\mathcal{S}}(\kappa, \lambda, \hat{\mathsf{mask}}_{k,i}, (\mathsf{Dec}_{\mathsf{sk}}(\hat{v'_{k,i,j}}))_{j \in [0, 2^{\lceil \mu_\delta \rceil} - 1]}) \approx_c \mathsf{SIM}_{\Pi_{\mathsf{PIS}},\mathsf{S}}(\hat{\mathsf{mask}}_{k,i})$$

Where $\hat{v'_{k,i,j}}$ is computed in the same way as $v'_{k,i,j}$. Hence, $\mathcal{H}'_2$ is indistinguishable from $\mathcal{H}'_1$.

$\mathcal{H}'_3$: Identical to $\mathcal{H}'_1$, except the simulator acts as in step 4'. Since the Sender only has access to $\mathsf{Enc}_{\mathsf{pk}}()$, and by the IND-CPA security of AHE, $\mathcal{H}'_3$ is indistinguishable from $\mathcal{H}'_2$. The view of the Sender in this game is denoted as $\mathsf{SIM}_{\Pi_{\mathsf{PFmap}},\mathcal{R}}(Y)$.

From the above, we have:

$$\mathsf{Real}_{\Pi_{\mathsf{PFmap}},\mathcal{S}}(\kappa,\lambda,X_1,Y) \approx_c \mathsf{SIM}_{\Pi_{\mathsf{PFmap}},\mathcal{R}}(Y) \approx_c \mathsf{Real}_{\Pi_{\mathsf{PFmap}},\mathcal{S}}(\kappa,\lambda,X_2,Y)$$

Security for a corrupted Receiver: We need to prove that for any Y_1, Y_2, the views of the Receiver in the real-world execution with Y_1 and Y_2 are computationally indistinguishable, i.e., $\mathsf{Real}_{\Pi,\mathcal{R}}(\kappa,\lambda,X,Y_1) \approx_c \mathsf{Real}_{\Pi,\mathcal{R}}(\kappa,\lambda,X,Y_2)$. We define the simulator acting as the Sender:

1*. The simulator starts with a set $X \subset \mathcal{U}^d$ and $\mathsf{ID}(X)$.

2*. The simulator runs the following algorithm to set $\hat{\mathsf{list}}_i$ as list_i in GetID:

   1). For each $k \in [N_r]$, find i_k such that $[x_{k,i_k}-\delta, x_{k,i_k}+\delta]$ is disjoint from other projections on dimension i.

   2). Run step 1 of $\mathsf{GetID}(X)$ to get $\mathsf{interval}_i$, and set the corresponding value of $[x_{k,i_k}-\delta, x_{k,i_k}+\delta]$ to be $\hat{r}_{k,i_k} = \mathsf{id}_{\mathbf{x}_k} - \sum_{i\neq i_k} \hat{r}_{k,i}$, where $\hat{r}_{k,i}$ satisfies $x_{k,i} \in \hat{U}_{k,i}$ and $\{(\hat{U}_{k,i}, \hat{r}_{k,i})\} \in \hat{\mathsf{interval}}_i$.

   3). Run the remaining steps of $\mathsf{GetID}(X)$ to get $\hat{\mathsf{list}}_i$.

3*. The simulator samples random $(\hat{u}'_{k,i,j}, \hat{v}'_{k,i,j}) \leftarrow_\$ \mathcal{C}^2$ to be the message sent to the Receiver.

4*. The simulator samples $\hat{\mathsf{id}}\mathsf{x}_{k,i,j} \leftarrow_\$ [0,\mu_\delta]$ and invokes the PIS simulator $\mathsf{SIM}_{\Pi_{\mathsf{PIS}},\mathsf{R}}$ for the Receiver.

The hybrid games are defined as follows:

$\mathcal{H}_0^*$: The real-world execution of the protocol, where the simulator acts exactly as an honest Sender.

$\mathcal{H}_1^*$: Identical to $\mathcal{H}_0^*$, except the simulator acts as in step 2*.

$\mathcal{H}_2^*$: Identical to $\mathcal{H}_1^*$, except the simulator acts as in step 3*.

$\mathcal{H}_3^*$: Identical to $\mathcal{H}_1^*$, except the simulator acts as in step 4*.

We show that $\mathcal{H}_0^*$ is indistinguishable from $\mathcal{H}_1^*$. After running the algorithm in step 2*, $\mathsf{id}_{\mathbf{x}_k}$ remains unchanged, and according to the assumption that any L_∞ balls of the Receiver must have disjoint projections in some dimension, there exists r such that $([x_{k,i_k}-\delta, x_{k,i_k}+\delta], r) \in \hat{\mathsf{list}}_i$, making step 2 valid. Since values except $\hat{r}_{k,i_k}$ are uniformly sampled, we have $(\alpha_1,\alpha_2,\ldots,\alpha_d : \alpha_i \leftarrow_\$ \mathcal{U}) \approx_s (\alpha'_1,\ldots,\alpha'_{d-1},\alpha'_d :$ for $i \in [d-1], \alpha'_i \leftarrow_\$ \mathcal{U}, \alpha'_d = \sum_{i=1}^{d}\alpha_i - \sum_{i=1}^{d-1}\alpha'_i)$. Thus, $\mathcal{H}_1^*$ is indistinguishable from $\mathcal{H}_0^*$. The rest of the proof is similar to the case of corrupted Receiver, we omit here. □

D.3 Fuzzy PSI-CA for L_∞ in High Dimension

Theorem 13 (Correctness)**.** *The L_∞ distance fuzzy PSI-CA protocol in Figure 12 outputs correctly with overwhelming probability.*

Proof. For any $\mathbf{y}_k \in Y$, if there exists $\mathbf{x} \in X$ such that $|\mathbf{x}-\mathbf{y}_k|_\infty \le \delta$, we need to prove that $e_k = 1$; otherwise, $e_k = 0$ with overwhelming probability.

Case 1: $|\mathbf{x}-\mathbf{y}_k|_\infty \le \delta$ for some $\mathbf{x} \in X$. By the correctness of the fuzzy mapping protocol, $\mathsf{id}_{\mathbf{x}} = \mathsf{id}_{\mathbf{y}_k}$. Therefore, for each $i \in [d]$, there exists $j_i \in [0,\mu_\delta]$ such that $\mathsf{Decode}(\mathsf{list}_i, \mathsf{Prefix}(y_{k,i}, j_i)) = \mathsf{Enc}_{\mathsf{pk}}(0)$.

This implies $u_{k,i,j_i} = \mathsf{Enc}_{\mathsf{pk}}(0 + a_{k,i}) \implies \mathsf{Dec}_{\mathsf{sk}}(u_{k,i,j_i}) = a_{k,i}$. By the correctness of secret shared private equation test (ssPET), the Receiver and Sender compute $(b_{k,i} \oplus 1) \oplus b'_{k,i} = 0 \implies (b_{k,i} \oplus 1) - b'_{k,i} = 0$. Thus, the ciphertext $\mathsf{ct}'_k = \mathsf{Enc}_{\mathsf{pk}}(0)$, resulting in $e_k = 1$.

Case 2: $|\mathbf{x} - \mathbf{y}_k|_\infty > \delta$ for all $\mathbf{x} \in X$. We have $\mathsf{id}_{\mathbf{x}} \neq \mathsf{id}_{\mathbf{y}_k}$ for all $\mathbf{x} \in X$, the Sender decodes all prefixes into random values. By the correctness of ssPET: $(b_{k,i} \oplus 1) \oplus b'_{k,i} = 1 \implies (b_{k,i} \oplus 1) - b'_{k,i} = 1$ or -1. This results in $\mathsf{ct}'_k = \mathsf{Enc}_{\mathsf{pk}}(\pm r_{k,i}) \neq \mathsf{Enc}_{\mathsf{pk}}(0)$, leading to $e_k = 0$.

The two cases are mutually exclusive and cover all possibilities, ensuring the protocol outputs correctly with overwhelming probability. □

Theorem 14 (Security)**.** *The L_∞ distance fuzzy PSI-CA protocol in Figure 12 is secure against semi-honest adversary in the $\mathcal{F}_{\mathsf{PFmap}}$ and $\mathcal{F}_{\mathsf{ssPET}}$ hybrid model, and the independence properties of OKVS and the IND-CPA of AHE holds.*

Proof. **Security for corrupted Sender.** We construct a simulator who acts as the Receiver.

1′. Simulator starts with Y, which is the input of Sender.

2′. Simulator invokes $\mathsf{SIM}_{\Pi_{\mathsf{PFmap}},\mathcal{S}}(Y, \hat{\mathsf{id}}_{\mathbf{y}_k})$, which is simulator of fuzzy mapping for Sender, $\hat{\mathsf{id}}_{\mathbf{y}_k}$ is sampled uniformly by simulator.

3′. Simulator encodes OKVS with dummy key-value pairs.

4′. Simulator samples $\hat{a}_{k,j} \leftarrow_\$ \mathcal{U}$ to replace $a_{k,j}$ in step 5. And invokes $\mathsf{SIM}_{\mathsf{ssPET},\mathsf{P}_1}(\hat{a}_{k,i})$ $\mu_\delta + 1$ times for each k, i.

5′. Simulator sets random ciphertexts $\hat{\mathsf{ct}}_{k,i} \leftarrow_\$ \mathcal{C}$ as message sent in step 7.

Now we show the hybrid games.

$\mathcal{H}'_0$: The real-world execution of the protocol, simulator acts exactly as honest Receiver.

$\mathcal{H}'_1$: Identical to $\mathcal{H}'_0$, except that simulator acts as in step 2′. This is indistinguishable from $\mathcal{H}'_0$ because of the security of our fuzzy mapping protocol.

$\mathcal{H}'_2$: Identical to $\mathcal{H}'_1$, except that simulator acts as in step 3′. This is indistinguishable from $\mathcal{H}'_2$ because of the obliviousness and IND-CPA of AHE.

$\mathcal{H}'_3$: Identical to $\mathcal{H}'_2$, except that simulator acts as in step 4′. This is indistinguishable from $\mathcal{H}'_3$ because of $a_{k,j}$ are uniformly sampled, which is indistinguishable from $a'_{k,j}$, and because of the security of ssPET.

$\mathcal{H}'_4$: Identical to $\mathcal{H}'_3$, except that simulator acts as in step 5′. This is indistinguishable from $\mathcal{H}'_3$ because of the IND-CPA of AHE.

By now we have proved security against semi-honest Sender.

Security for corrupted Receiver. We construct a simulator who acts as the Sender.

1*. Simulator starts with X and c, which is the input of Sender and the number of matching points.

2*. Simulator invokes $\mathsf{SIM}_{\Pi_{\mathsf{PFmap}},\mathcal{R}}(X, \hat{\mathsf{id}}_{\mathbf{x}_k})$, which is simulator of fuzzy mapping for Sender, $\hat{\mathsf{id}}_{\mathbf{x}_k}$ is sampled uniformly by simulator

3*. Simulator sets $K' \subset [N_s]$ s.t. $|K'| = c$. Without loss of generality, both party doesn't shuffle the sets. For $k \in K'$, Simulator sets the message $\hat{u}_{k,i,j_i} = \mathsf{Enc}_{\mathsf{pk}}(\hat{a}_{k,j})$, where $\hat{a}_{k,j} \leftarrow_\$ \mathcal{U}$, $j_i \in [0, \mu_\delta]$, and $\hat{u}_{k,i,j} \leftarrow_\$ \mathcal{C}$ for $j \neq j_i$. For $k \notin K'$, Simulator sets $\hat{u}_{k,i,j} \leftarrow_\$ \mathcal{C}$ for $j \in [\mu_\delta]$.

4*. Simulator invokes $\mathsf{SIM}_{\mathsf{ssPET},\mathsf{P}_0}(\mathsf{Dec}_{\mathsf{sk}}\hat{u}_{k,i,j}, j_i)$ for each $k, i, j \in [0, \mu_\delta]$. For $k \notin K'$, simulator invokes $\mathsf{SIM}_{\mathsf{ssPET},\mathsf{P}_0}(\mathsf{Dec}_{\mathsf{sk}}\ (\hat{u}_{k,i,j}), r)$ for random r.

5*. For $k \in K'$, simulator sets $\hat{\mathsf{ct}}'_k = \mathsf{Enc}_{\mathsf{pk}}(0)$. For $k \notin K'$, simulator sets $\hat{\mathsf{ct}}'_k \leftarrow_{\$} \mathcal{C}$.

The hybrid games are similar to proof for corrupted Sender and Throrem 4, we omit the rest of the proof here. □

D.4 Fuzzy PSI-CA for L_p in High Dimension

We show the modified algorithm $\mathsf{GetList}'_p$ and $\mathsf{GetValue}'_p$ here. For algorithm $\mathsf{GetList}'_p$ and $\mathsf{GetValue}'_p$, H_{γ_1} is universal hash functions where $\gamma_1 = \kappa + 2\log(\log\delta \cdot d)$.

$\mathsf{GetList}'_p(\mathsf{pk}, \delta, \mathbf{x}, \Delta)$

1. For each $i \in [d]$: Set $\{x_{0,i,j}\}_{j\in[0,\omega_{0,i}]} = \mathsf{Decompose}(\{x'_i : x'_i \le x_i, |x'_i - x_i| \le \delta\})$, $\{x_{1,i,j}\}_{j\in[0,\omega_{1,i}]} = \mathsf{Decompose}(\{x'_i : x'_i > x_i, |x'_i - x_i| \le \delta\})$.
2. For each $\sigma \in \{0,1\}$, $i \in [d]$:
   - Initiate $\mathsf{list}_{\sigma,i} = \emptyset$. For each $j' \in [0, \omega_{\sigma,i}]$:
     * If $\sigma = 0$, $\mathbf{x}^* = \mathsf{UpBound}(x_{\sigma,i,j'})$. Else If $\sigma = 1$, $x^* = \mathsf{LowBound}(x_{\sigma,i,j'})$.
     * Set $\mathsf{list}_{\sigma,i} \leftarrow \mathsf{list}_{\sigma,i} \cup (\mathbf{H}_{\gamma_1}(\Delta||\sigma||x_{\sigma,i,j'}), \mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|)||\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^2)||\cdots||\mathsf{Enc}_{\mathsf{pk}}(|x^* - x_i|^p)\ ||\mathsf{Enc}_{\mathsf{pk}}(0))$.
3. Pad $\mathsf{list}_{\sigma,i}$ with dummy random items to size ω_δ, and output.

$\mathsf{GetValue}'_p(\mathsf{pk}, \delta, \mathbf{y}, \Delta, \mathbf{E})$

1. Denote $y_{i,j} = \mathsf{Prefix}(y_i, j)$, $j \in [0, \mu_\delta]$.
2. For each $i \in [d]$:
   - Sample $a_i \leftarrow_{\$} \mathcal{P}$, $s_i \leftarrow_{\$} \mathcal{P}$. For each $\sigma \in \{0,1\}$, $j \in [0, \mu_\delta]$:
     * Set $(u^*_{\sigma,i,j,t}) \leftarrow \mathsf{Decode}(E_{\sigma,i}, \mathsf{H}_{\gamma_1}(\Delta||\sigma||y_{i,j}))$, where $t \in [p+1]$.
     * Set $e = |y_i - \mathsf{UpBound}(y_{i,j})|$ if $\sigma = 0$, else $e = |y_i - \mathsf{LowBound}(y_{i,j})|$.
     * Compute $u_{\sigma,i,j} = \boxplus_{\mathsf{pk}}{}_{t=1}^{p}(\binom{p}{t} e^{p-t} \boxtimes_{\mathsf{pk}} u^*_{\sigma,i,j,t}) \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(a_i + e^p)$.
     * Compute $v^*_{\sigma,i,j} = u_{\sigma,i,j,p+1} \boxplus_{\mathsf{pk}} \mathsf{Enc}_{\mathsf{pk}}(s_i)$.
3. Output $(\{u_{\sigma,i,j}\}, \{a_i\}, \{v^*_{\sigma,i,j}\}, \{s_i\})$.

Figure 19: Algorithm $\mathsf{GetList}'_p$ and $\mathsf{GetValue}'_p$

Theorem 15 (Correctness)**.** *The L_p distance fuzzy PSI-CA protocol in Figure 13 outputs the correct result with overwhelming probability.*

Proof. For any $\mathbf{y}_k \in Y$, if there exists $\mathbf{x} \in X$ such that $|\mathbf{x} - \mathbf{y}_k| \le \delta$, we must show that $e_k = 1$. Conversely, if $|\mathbf{x} - \mathbf{y}_k| > \delta$ for all $\mathbf{x} \in X$, we must show that $e_k = 0$ with overwhelming probability.

Case 1: $|\mathbf{x} - \mathbf{y}_k| \le \delta$ for some $\mathbf{x} \in X$. By the correctness of the fuzzy mapping, $\mathsf{id}_{\mathbf{x}} = \mathsf{id}_{\mathbf{y}_k}$. This implies there exist indices $j_i \in [0, \mu_\delta]$ and $\sigma_i \in \{0,1\}$ such that the

decoding condition holds:

$$\mathsf{Decode}(\mathsf{list}_{0,i},\mathsf{H}_{\gamma_1}(\Delta||\sigma_i||\mathsf{Prefix}(y_{k,i},j_i))) = \\ (\mathsf{Enc}_{\mathsf{pk}}(|x^*-x_i|),\ldots,\mathsf{Enc}_{\mathsf{pk}}(|x^*-x_i|^p),\mathsf{Enc}_{\mathsf{pk}}(0))$$

where $x^* = \mathsf{UpBound}(x_{\sigma,i,j_i})$. Using Theorem **??**, the Sender computes encrypted values $u_{k,\sigma_i,i,j_i} = \mathsf{Enc}_{\mathsf{pk}}(a_{k,i}+|\mathbf{x}-\mathbf{y}_k|^p)$ and $v_{k,\sigma_i,i,j_i} = \mathsf{Enc}_{\mathsf{pk}}(s_{k,i})$. The Receiver retrieves (j_i,σ_i) via PIS, ensuring $e_k = 1$ by the correctness of IFMat.

Case 2: $|\mathbf{x}-\mathbf{y}_k| > \delta$ for all $\mathbf{x} \in X$. If $\mathsf{id}_{\mathbf{x}} \neq \mathsf{id}_{\mathbf{y}_k}$ for all $\mathbf{x} \in X$, the Sender's prefixes are decoded into random values, leading to $e_k = 0$ with overwhelming probability by IFMat's correctness. If $\mathsf{id}_{\mathbf{x}} = \mathsf{id}_{\mathbf{y}_k}$ for some $\mathbf{x} \in X$, there exists a dimension $i \in [d]$ that $y_i \notin [x_i-\delta, x_i+\delta]$. Then all prefixes $\mathsf{Prefix}(y_{k,i},j)$ (for $j \in [0,\mu_\delta]$) are decoded into random values, again ensuring $e_k = 0$ with overwhelming probability by IFMat's correctness. □

Theorem 16 (Security). *The L_p distance fuzzy PSI-CA protocol in Figure 13 is secure against semi-honest adversary in the $\mathcal{F}_{\mathsf{PFmap}}$ and $\mathcal{F}_{\mathsf{PIS}}$ hybrid model, and the independence properties of OKVS and the IND-CPA of AHE holds.*

Proof. For security against corrupted Sender, the simulator is similar to that in Theorem 14, we construct it by encoding random key-value pairs, and invokes simulator for Fmap, PIS, and IFMat. The same hold for security against corrupted Receiver. □

E Study of Prefix

E.1 Formal Description of the Question

We give a formal description of the question mentioned in Section 8. We define μ as the number of prefixes Sender decodes in every dimension, assume S' is an interval of length $t = \overline{t_w t_{w-1}\cdots t_1}$, $w = \lfloor \log t \rfloor + 1$, we define the nature decompose $\mathsf{naDec}(S',\mu)$:

Definition 7 (naDec). $\mathsf{naDec}(S',\mu) = \{b_j\}_{j\in\omega'}$ satisfies the following properties:

1. $S' = \bigsqcup_{j\in\omega'} \mathsf{Interval}_l(b_j)$.
2. $\forall j$, length of b_j is no less than $l-\mu$.
3. $\forall j$, for any bit string b' that is a prefix of b_j, and with length no less than $l-\mu$, $\mathsf{Interval}_l(b') \nsubseteq S'$.

We prove that the decomposition is unique for any S'.

Theorem 17 (Uniqueness). *For any interval S', and any μ, $\mathsf{naDec}(S',\mu)$ is unique.*

Proof. We do reduction on length of the interval. For interval of length 1, the proof is obvious.

For interval S' of length $t \in \mathbb{N}$, we assume that for any interval of length less than t, the decomposition is unique. We consider $x' = \mathsf{LowBound}(S')$, according to property 2 and 3, there must be only one prefix b' of x' with length less than $l-\mu$ in $\mathsf{Decompose}(S',\mu)$, otherwise if $x' \in \mathsf{Interval}(b') \subset S'$ and $x' \in \mathsf{Interval}(b'') \subset S'$, then $\mathsf{Interval}(b') \subsetneq \mathsf{Interval}(b'') \subset S'$ or $\mathsf{Interval}(b'') \subsetneq \mathsf{Interval}(b') \subset S'$, contradicting to property 3, we are done. Then we consider the interval $S' \setminus \mathsf{Interval}(b')$, its decomposition is unique following the reduction. In conclusion, $\mathsf{Decompose}(S',\mu)$ is unique. □

We give the following definition for ω.

Definition 8. The function $\omega(\cdot,\cdot) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined as:

$$\omega(t,\mu) = \max_{\substack{\mathsf{interval}S' \subset \mathbb{Z} \\ |S'|=t}} \left(|\mathsf{naDec}(S',\mu)|\right),$$

where $\mathsf{naDec}(S',\mu)$ is defined in 7.

Our destination is to computes the value of $\omega(t,\mu)$. Notice that the number of key-value pairs ω_δ mentioned in all protocols of L_∞ and L_p settings are $\omega(2\delta+1, \lfloor \log(2\delta+1) \rfloor)$ and $\omega(\delta+1, \lfloor \log(\delta+1) \rfloor)$ respectively.

E.2 Study of Case $\mu \geq \lfloor \log t \rfloor$

In this section, we focus on the case $\mu \geq \lfloor \log t \rfloor$. To simplify, we define $f(S',\mu) = \mathsf{naDec}(S',\mu)$.

Lemma 6. *For $\mu \geq \lfloor \log t \rfloor$, $\omega(t,\mu) = \omega(t, \lfloor \log t \rfloor)$.*

Proof. We only need to prove $f(S',\mu) = f(S', \lfloor \log t \rfloor)$. Assume $f(S', \lfloor \log t \rfloor) = \{b_j\}_{j\in\omega'}$, $|S'| = t$, and $|b_j| = l - \mu_j$, we have $|\mathsf{Interval}_l(b_j)| = 2^{\mu_j}$, then $t = \sum_{j\in\omega'} 2^{\mu_j}$. We prove $f(S',\mu) = f(S', \lfloor \log t \rfloor)$ for $\mu \geq \lfloor \log t \rfloor$.

First, Since $l - \mu \leq l - (\lfloor \log t \rfloor + 1)$, property 2 of Definition 7 is satisfied. Second, we argue there is no prefix has length smaller than $l - \lfloor \log t \rfloor$ can generate an interval contained in S'. Otherwise the interval generated by this prefix has size $2^{\lfloor \log t \rfloor + 1} > t$, which is impossible, property 3 must be satisfied. The lemma has been proved. □

Notice that $\mathsf{naDec}(S',\mu) = \mathsf{Decompose}(S')$ defined in Definition 1 if $\mu \geq \lfloor \log t \rfloor$.

Lemma 7. *If $\mu = \lfloor \log t \rfloor$, for any interval S' with $|S'| = t$, define n'_i as the number of prefixes of length $l - i$ in $f(S',\mu)$, $i \in [0,\mu]$, we have $n'_i \in \{0,1,2\}$.*

Proof. We can see $t = \sum_{i\in\mu} n'_i 2^i$. First we conclude that there exists no $\mathsf{Interval}_l(b_j)$ such that the size of its two adjacent intervals is larger.

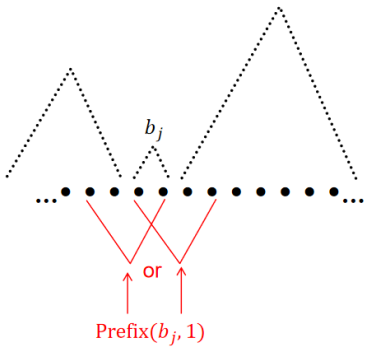


Figure 20: Adjacent Intervals

As shown in Figure 20, otherwise, $\mathsf{Interval}_l(\mathsf{Prefix}(\mathsf{b_j}, 1)) \subset S'$, contradict to property 3.

If $\exists i$, $n'_i > 2$, since $n'_i 2^\mu > 2 \times 2^{\lfloor \log t \rfloor} > t$, and $n'_i 2^i \leq t$, we have $i < \lfloor \log t \rfloor$, then there must be at least two adjacent intervals of the same size 2^i according to the above analysis, denote as b_j, b'_j. Notice that $\mathsf{Interval}_l(\mathsf{Prefix}(\mathsf{b_j}, 1)) \subset S'$ or $\mathsf{Interval}_l(\mathsf{Prefix}(\mathsf{b'_j}, 1)) \subset S'$, contradict to property 3. we have proved $n_i \in \{0,1,2\}$. □

Clearly, according to the definition of n'_i, we have $f(S',\mu) = \sum_{i\in[0,\mu]} n'_i$.

Lemma 8. *If $\mu = \lfloor \log t \rfloor$. For any $\{(i, n_i)\}_{i\in[0,\mu]}$ satisfying $t = \sum_{i\in[0,\mu]} n_i 2^i$, and $n_i \in \{0, 1, 2\}$, there exists an interval S' of length t, such that n_i is the number of prefixes of length $l - i$ in $f(S', \mu)$ for $i \in [0, \mu]$.*

Proof. Take $t_l = \sum_{i\in[0,\mu]:n_i=2} 2^i$, $t_r = \sum_{i\in[0,\mu]:n_i\neq 0} 2^i$, we have $t_l + t_r = t$, then for all $m \in \mathbb{Z}$, interval $[m \cdot 2^{\lfloor \log t\rfloor+1} - t_l, m \cdot 2^{\lfloor \log t\rfloor+1} + t_r - 1]$ meets the requirement. □

According to Lemma 7 and 8, we construct a map from S' to $\{(i, n'_i)\}$. this map is a bijection. If $\mu = \lfloor \log t \rfloor$, notice that:

$$\omega(t, \mu) = \max_{\substack{\text{interval} S'\subset\mathbb{Z} \\ |S'|=t}} (|f(S', \mu)|) = \max_{\substack{n_i\in\{0,1,2\} \\ \sum_{i\in[0,\mu]} n_i 2^i = t}} (\sum_{i\in[0,\mu]} n_i)$$

To continue, define $w = \lfloor \log t \rfloor + 1$, here we give some definitions:

1. We define α_2 to be the largest index number in the set $\{i \in [w] : t_i = 1, t_{i-1} = 1, \cdots, t_1 = 1\}$; and if $t_1 = 0$, then $\alpha_2 = 0$.
2. We define α_1 to be $\alpha_1 = |\{i \in [1, w] : t_i = 1\}| - \alpha_2$.

Now we have the following theorems.

Theorem 18. *For $\mu \geq \lfloor \log t \rfloor$, we have*

$$\omega(t, \mu) = \begin{cases} \alpha_1 + \lfloor \log t \rfloor & \text{if } \alpha_1 \neq 0, \\ \lfloor \log t \rfloor + 1 & \text{if } \alpha_1 = 0. \end{cases}$$

Proof. Denote $t = \overline{t_w t_{w-1} \cdots t_1}$, where $w = \lfloor \log t \rfloor + 1$. From Lemma 6, if $\mu \geq \lfloor \log t \rfloor$, then $\omega(t, \mu) = \omega(t, \lfloor \log t \rfloor)$. According to Lemmas 7 and 8, $\omega(t, w - 1)$ is the maximum of $\sum_{i=0}^{w-1} n_i$ over all $n_i \in \{0, 1, 2\}$ such that $\sum_{i=0}^{w-1} n_i 2^i = t$. We now analyze this.

Notice $t = \sum_{i:t_i=1} 2^{i-1}$. The set $M_t = \{(i, n'_i) : n'_i = 1 \text{ if } t_{i+1} = 1, n'_i = 0 \text{ if } t_{i+1} = 0\}$ satisfies $\sum_{i=0}^{w-1} n'_i 2^i = t$ and $n'_i \in \{0, 1, 2\}$. Now We present two operations:

1. If $n_i = 1$ and $n_{i+1} = 0$, switch them to $n_i = 0$ and $n_{i+1} = 2$.
2. If $n_i = 2$ and $n_{i+1} = 0$, switch them to $n_i = 1$ and $n_{i+1} = 2$.

The transformed set from M_t by these operations still satisfies $\sum_{i=0}^{w-1} n_i 2^i = t$ and $n_i \in \{0, 1, 2\}$. We prove the following lemma:

Lemma 9. *Any $\{(i, n_i)\}$ satisfying the conditions $\sum_{i=0}^{w-1} n_i 2^i = t$ and $n_i \in \{0, 1, 2\}$ can be transformed from M_t by repeating the two operations.*

Proof. We prove this lemma via reduction on $z = i \in [0, w - 1]$, which is the smallest index that $n_{i+1} \neq n'_{i+1}$.

For $z = w - 1$, it is trivial. Assume for $z \geq z_0 + 1$ the result holds. We have:

$$\sum_{i=0}^{w-1} n_i 2^i = \sum_{i=0}^{w-1} n'_i 2^i \text{ and } \sum_{i=0}^{z_0} n_i 2^i = \sum_{i=z_0}^{w-1} n'_i 2^i$$

$$\Longrightarrow \sum_{i=z_0+1}^{w-1} n_i 2^i = \sum_{i=z_0+1}^{w-1} n'_i 2^i$$

Since they are congruent modulo 2^{z_0+1}, and $n'_{z_0} \neq n_{z_0}$, then $n'_{z_0+1} = 0$ and $n_{z_0+1} = 2$. If $n'_{z_0+2} = 1$ and $n_{z_0+2} = 0$, we are done; else, $n'_{z_0+2} = 0$ and $n_{z_0+2} = 1$ modulo 2^{z_0+2}, because $n'_i \in \{0, 1\}$ and congruent modulo. Continuing, since $n'_{w-1} = 1$, there exists $z^* \leq w-1$ such that $(n'_{z^*}, n'_{z^*-1}, \ldots, n'_{z_0+1}) = (1, 0, \ldots, 0)$ and $(n_{z^*}, n_{z^*-1}, \ldots, n_{z_0+1}) = (0, 1, \ldots, 1, 2)$. Thus, any valid $\{(i, n_i)\}$ can be transformed from M_t by these operations. □

Observing that the both operations increases $\sum n'_i$ by 1. We need to find $\{(i, n^*_i)\}$ that is converted from M_t by performing the largest number of operations. If $\alpha_1 = 0$, then $\alpha_2 = w = \lfloor \log t \rfloor + 1$, and no operations can be performed, hence $\sum n^*_i = w = \lfloor \log t \rfloor + 1$. Otherwise, if $\alpha_1 \neq 0$, each 1 (except those in continuous 1s with smallest indexes) can be converted to 2, yielding exactly α_1 2s. And all other positions are 1, except $n^*_{w-1} = 0$. This results in a set where $\sum n_i = 2\alpha_1 + (w - \alpha_1 - 1) = \alpha_1 + w - 1 = \alpha_1 + \lfloor \log t \rfloor$. □

The construction of $\{(i, n^*_i)\}$ in the last paragraph will be presented in the next subsection.

According to Theorem 18, if $\delta = 32$, ω_δ mentioned in algorithms 4 and 7 are actually $\omega_\delta = 7$ and $\omega_\delta = 6$ respectively, and $\mu_\delta = 6$ and $\mu_\delta = 5$ respectively.

An algorithm for Decompose()**.** We give an simple way to find the prefixes that represents an interval according to the above analysis.

Decompose(S'): input $S' = [a, a + t - 1] \subset \mathcal{U}$, where $\mathcal{U} = \{0, 1\}^u$, $t \in \mathbb{Z}$.

1. Find a $a' \in [a, a + t - 1]$ such that $2^{w-1} | a'$, where $w = \lfloor \log t \rfloor + 1$.
2. Denote the binary representation of $x = a + t - a'$ and $y = a' - a$ as $\overline{x_w \cdots x_1}$ and $\overline{y_w \cdots y_1}$, where $x_i, y_i \in \{0, 1\}$.
3. Initiate $P = \emptyset$, $x' = a'$, $y' = a' - 1$, from $i = w$ to 1:
   - If $x_i = 1$, set $P = P \cup \{\mathsf{Prefix}(x', i - 1)\}$, then set $x' \leftarrow x' + 2^{i-1}$.
   - If $y_i = 1$, set $P = P \cup \{\mathsf{Prefix}(y', i - 1)\}$, then set $y' \leftarrow y' - 2^{i-1}$.
4. Output P.

Figure 21: Algorithm Decompose()

Theorem 19. *The output of the algorithm presented in Figure 21 satisfies properties of* Decompose(S') *defined in Definition 1.*

Proof. Since $2^{w-1} \leq t$, there exists at least one $a' \in [a, a + t - 1]$ such that $2^{w-1} | a'$. The first step of the algorithm is valid.

Then we prove the output P satisfies properties defined in Definition 1. Denote $x'_{(i)}$ as x' in loop i ,$y'_{(i)}$ as y' in loop i, notice that $2^{i-1} | x'_{(i)}$ and $2^{i-1} | y'_{(i)} + 1$ for $x'_{(i)}, y'_{(i)}$ in each loop, since $2^{w-1} | x'_{(i)}$ when $i = w$, and if $2^{i-1} | x'_{(i)}$ for some i, for all $k \leq i - 1$, we have $2^k | x'_{(i)} + 2^{i-1}$, which means $2^i | x'_{(i)}$ for $i \neq w$, the same holds for $y'_{(i)} + 1$, we conclude that $2^{i-1} | x'_{(i)}$ and $2^{i-1} | y'_{(i)} + 1$.

Then we have $\mathsf{Interval}(\mathsf{Prefix}(x'_{(i)}, i-1)) = [x'_{(i)}, x'_{(i)} + 2^{i-1} - 1]$ and $\mathsf{Interval}(\mathsf{Prefix}(y'_{(i)}, i - 1)) = [y'_{(i)} - 2^{i-1} + 1, y'_{(i)}]$. We have the following equation for $[a', a + t]$, where $j' = |\{i :$

$x'_i = 1\}|$.

$$\bigcup_{\substack{i_j:x_{i_j}=1,\\ i_1>i_2>\cdots>i_{j'}}} \mathsf{Interval}(\mathsf{Prefix}(x'_{(i)}, i-1)) = [a', a' + 2^{i_1-1} - 1] \cup$$

$$[a' + 2^{i_1-1}, a' + \sum_{j=1}^{2} 2^{i_j-1} - 1] \cup$$

$$\cdots$$

$$\cup [a' + \sum_{j=1}^{j'-1} 2^{i_j}, a' + \sum_{j=1}^{j'} 2^{i_j-1} - 1]$$

$$= [a', a' + a + (t - a') - 1]$$

$$= [a', a + t - 1]$$

The same holds for $[a, a' - 1]$:

$$\bigcup_{i:y_i=1} \mathsf{Interval}(\mathsf{Prefix}(y'_{(i)}, i-1)) = [a, a' - 1]$$

By now we have proved that the first property holds. For the second property, we only need to prove the interval generated by prefix $\mathsf{Prefix}(x'_{(i)}, i)$ and $\mathsf{Prefix}(y'_{(i)}, i)$ are not contained in $[a, a + t - 1]$ for each i, since we output $\mathsf{Prefix}(x'_{(i)}, i-1)$ and $\mathsf{Prefix}(y'_{(i)}, i-1)$. To be simple, we only prove for $\mathsf{Prefix}(x'_{(i)}, i)$.

In the case $i = w$, since $|\mathsf{Interval}(\mathsf{Prefix}(x'_{(w)}, w))| > t$, obviously $\mathsf{Interval}(\mathsf{Prefix}(x'_{(w)}, w)) \nsubseteq [a, a+t-1]$. In the case $i < w$, we know that $2^i | x'_{(i)}$ for $i \neq w$, then $\mathsf{Interval}(\mathsf{Prefix}(x'_{(i)}, i)) = [x'_{(i)}, x'_{(i)} + 2^i - 1]$. But we have $(a + t - 1) - x'_{(i)} + 1 \leq \sum_{l=1}^{i} 2^{l-1} < 2^i$, which means $[x'_{(i)}, x'_{(i)} + 2^i - 1] \nsubseteq [x'_{(i)}, a + t - 1]$, then $[x'_{(i)}, x'_{(i)} + 2^i - 1] \nsubseteq [a, a + t - 1]$. □

E.3 Further Study of Case $\mu < \lfloor \log t \rfloor$

In this section, we consider $\mu < \lfloor \log t \rfloor$. For this case, assume we have obtained $\mathsf{Decompose}(S')$, we only need to decompose the sub-interval of size larger than 2^μ in a simple way:

A simple decomposition: If we have a decomposition $\mathsf{Decompose}(S') = \{b_j\}_{j\in\omega'}$. For all $b^* \in \{b_j\}_{j\in\omega'}$ such that $|\mathsf{Interval}(b^*)| = 2^{\mu^*} > 2^\mu$, let $l - \mu^*$ be the length of b^*, then we decompose $\mathsf{Interval}(b^*)$ into $\bigcup_{\bar{b}^*\in\{0,1\}^{\mu^*-\mu}} \mathsf{Interval}(b^*||\bar{b}^*)$, containing $2^{\mu^*-\mu}$ intervals, each with length 2^μ.

This decomposition satisfies the property of Definition 7, and according to the uniqueness of decomposition in Theorem 17, we conclude that this natural decomposition is valid.

Now we show the following Lemma:

Lemma 10. *For $\mu < \lfloor \log t \rfloor$, we have*

$$\omega(t, \mu) = \max_{\substack{n_i\in\{0,1,2\}\\ \sum_{i\in[0,\lfloor\log t\rfloor]} n_i 2^i = t}} \Big(\sum_{i\in[0,\mu]} n_i + \sum_{i\in[\mu+1,\lfloor\log t\rfloor]} 2^{i-\mu} \cdot n_i \Big)$$

.

Proof. Since $\omega(t, \mu) = \max_{S'\subset\mathbb{Z}, |S'|=t}(|f(S', \mu)|)$, consider $f(S', \mu)$, n'_i is the number of prefixes of length $l - i$ in $f(S', \lfloor \log t \rfloor)$, $i \in [0, \lfloor \log t \rfloor]$, the uniqueness of Decompose and nature decomposition implies that $|f(S', \mu)| = \sum_{i\in[0,\mu]} n'_i + \sum_{i\in[\mu+1,\lfloor\log t\rfloor]} 2^{i-\mu} \cdot n'_i$, then we are done. □

Recall that $t = \overline{t_w t_{w-1} \cdots t_1}$, $w = \lfloor \log t \rfloor + 1$, α_2 is the largest index number in the set $\{i \in [w] : t_i = 1, t_{i-1} = 1, \cdots, t_1 = 1\}$, if $t_1 = 0$, $\alpha_2 = 0$. $\alpha_1 = |\{i \in [1, w] : t_i = 1\}| - \alpha_2$.

Define index set $I_t = \{i \in [\alpha_2 + 2, w - 1] : t_i = 1\} \cup \{\alpha_2 + 1\}$, and define a set $\hat{M}_t = \{(i, \hat{n}_{t,i})\}$.

$$\hat{n}_{t,i} = \begin{cases} 2 & \text{if } i+1 \in I_t \\ 0 & \text{if } i = \lfloor \log t \rfloor = w - 1 \\ 1 & \text{otherwise} \end{cases}$$

Lemma 11. *The set $\hat{M}_t$ defined above maximizes $\sum_{i\in[0,w-1]} n_i$ under the constraints $\sum_{i\in[0,w-1]} n_i 2^i = t$ and $n_i \in \{0, 1, 2\}$.*

Proof. By the final paragraph of the proof of Theorem 18, the composition of $\hat{M}_t$ includes α_1 instances of "2", one "0", and the remaining elements are "1". This structure ensures $\hat{M}_t$ maximizes $\sum_{i\in[0,w-1]} n_i$. To complete the proof, we verify that $\hat{M}_t$ satisfies $\sum_{i\in[0,w-1]} n_i 2^i = t$.

Let $I_t = \{i_1, i_2, \ldots, i_{\alpha_1}\}$ denote the indices in ascending order where $t_i = 1$. Then: $t_{i_2}, \ldots, t_{i_{\alpha_1}} = 1$, $i_1 = \alpha_2 + 1$, $t_{i_1} = 0$. And we have $t = 2^{w-1} + 2^{i_{\alpha_1 - 1}} + \cdots + 2^{i_2 - 1} + \sum_{i=0}^{\alpha_2 - 1} 2^i$.

Substituting these into the sum:

$$\begin{aligned} \sum_{i\in[0,w-1]} \hat{n}_{t,i} 2^i &= \sum_{i+1\in I_t} 2 \cdot 2^i + \sum_{i\in[0,w-2], i+1\notin I_t} 2^i \\ &= \sum_{i=0}^{\alpha_2 - 1} 2^i + (2 \cdot 2^{i_1 - 1} + 2^{i_1} + \cdots + 2^{i_2 - 2}) \\ &\quad + (2 \cdot 2^{i_2 - 1} + 2^{i_2} + \cdots + 2^{i_3 - 2}) + \cdots \\ &\quad + (2 \cdot 2^{i_{\alpha_1} - 1} + 2^{i_{\alpha_1}} + \cdots + 2^{w-2}) \\ &= \sum_{i=0}^{\alpha_2 - 1} 2^i + 2^{i_2 - 1} + 2^{i_3 - 1} + \cdots + 2^{w-1} = t. \end{aligned}$$

□

Theorem 20. *For $\mu < \lfloor \log t \rfloor$, we have*

$$\omega(t, \mu) = \begin{cases} \sum_{i\in[0,\mu]} \hat{n}_{t,i} + \sum_{i\in[\mu+1,\lfloor \log t \rfloor]} 2^{i-\mu} \cdot \hat{n}_{t,i} & \text{if } \alpha_1 \neq 0 \\ \mu + 2^{\lfloor \log t \rfloor - \mu + 1} - 1 & \text{if } \alpha_1 = 0 \end{cases}$$

Proof. Case $\alpha_2 = w$ is trivial, we focus on case $\alpha_2 \neq w$.According to Lemma 10, we have

$$\omega(t, \mu) = \max_{\substack{n_i \in \{0,1,2\} \\ \sum_{i\in[0,\lfloor \log t \rfloor]} n_i 2^i = t}} \left(\sum_{i\in[0,\mu]} n_i + \sum_{i\in[\mu+1,\lfloor \log t \rfloor]} 2^{i-\mu} \cdot n_i \right)$$

According to Theorem 18, any set $\{n_i\}_{i\in[0,\lfloor \log t \rfloor]}$ satisfying $\sum_{i\in[0,\lfloor \log t \rfloor]} n_i 2^i = t$ and $n_i \in \{0, 1, 2\}$ can be converted from M_t by two operations:

1. For some i, If $n_i = 1, n_{i+1} = 0$, switch them with $n_i = 0, n_{i+1} = 2$.
2. For some i, If $n_i = 2, n_{i+1} = 0$, switch them with $n_i = 1, n_{i+1} = 2$.

Notice that $\sum_{i\in[0,\mu]} n_i + \sum_{i\in[\mu+1,\lfloor \log t \rfloor]} 2^{i-\mu} \cdot n_i$ doesn't decrease after any operation, hence we do the same as in Theorem 18, we need to find the conversion that has performed the maximum number of operations. In Lemma 11, we know $\hat{M}_t$ satisfies our requirements. We are done. □

E.4 Set Decomposition

We have analyzed the value of $\omega(t,\mu)$ for any t and μ. The analysis above is based on the definition of $\mathsf{naDec}()$ provided in 7. By introducing a new definition for $\mathsf{naDec}()$, we can achieve smaller values of $\omega(t,\mu)$:

Definition 9 (setDec)**.** Let $U=\{u_1,\ldots,u_\mu\}\subseteq[0,\lfloor\log t\rfloor]$, where $\{0\}\in U$. The function $\mathsf{setDec}(S',U)$ produces a decomposition $\{b_j\}_{j\in\omega'}$ that satisfies the following properties:

1. $S'=\bigsqcup_{j\in\omega'}\mathsf{Interval}_l(b_j)$.
2. For all j, the length of b_j is in the set $\{l-i:i\in U\}$.
3. For all j, any bit string b' that is a prefix of b_j (with length in $\{l-i:i\in U\}$) must have $\mathsf{Interval}_l(b')\nsubseteq S'$.

It is clear that $\mathsf{setDec}(S',[0,\mu])=\mathsf{naDec}(S',\mu)$. The proof of uniqueness follows a similar approach to that in Theorem 17. We start by redefining the function $\bar{f}(S',U)=\mathsf{setDec}(S',U)$. Then, we define the function $\bar{\omega}:\mathbb{N}\times\{\mathbb{N}\}^*\to\mathbb{N}$ such that

$$\bar{\omega}(t,U)=\max_{\substack{\mathsf{inetrval}S'\subset\mathbb{Z}\\|S'|=t}}(|\bar{f}(S',U)|).$$

SetRound: We define an algorithm $\mathsf{SetRound}(\mu,U)=\mu^*$, where μ^* satisfies $\mu-\mu^*=\min_{i\in U,i<\mu}\{\mu-i\}$. This definition is valid because $0\in U$.

We give a simple way to get setDec from $\mathsf{Decompose}(S')$ in a manner similar to the method in Section E.3:

A simple decomposition: Given a decomposition $\mathsf{Decompose}(S')=\{b_j\}_{j\in\omega'}$, for every $b^*\in\{b_j\}_{j\in\omega'}$ such that $|\mathsf{Interval}(b^*)|=2^{\mu^*}$ and $\mu^*\notin U$, let $l-\mu^*$ be the length of b^*. We decompose $\mathsf{Interval}(b^*)$ into $2^{\mu^*-\mu^{**}}$ intervals of length $2^{\mu^{**}}$, where $\mu^{**}=\mathsf{SetRound}(\mu^*,U)$, resulting in $\cup_{\beta\in\{0,1\}^{\mu^*-\mu^{**}}}\mathsf{Interval}(b^*||\beta)$.

We have the following lemma:

Lemma 12. *For $|U|<\lfloor\log t\rfloor$, we have*

$$\bar{\omega}(t,U)=\max_{\sum_{i\in[0,\lfloor\log t\rfloor]}n_i2^i=t\ \ n_i\in\{0,1,2\}}(\sum_{i\in U}n_i+\sum_{i\notin U}2^{i-i^*}\cdot n_i)$$

Where $i^*=\mathsf{SetRound}(i,U)$.

Proof. Since $\bar{\omega}(t,U)=\max_{\forall S':|S'|=t}(|\bar{f}(S',U)|)$, consider $\bar{f}(S',U)$, n'_i is the number of prefixes of length $l-i$ in $\bar{f}(S',U)$, $i\in[0,\lfloor\log t\rfloor]$, the uniqueness of setDec and nature decomposition implies that $|\bar{f}(S',\mu)|=\sum_{i\in[U]}n'_i+\sum_{i\notin U}2^{i-i^*}\cdot n'_i$, where $i^*=\mathsf{SetRound}(i,U)$, then we are done. □

We observe that the two operations shown in Theorem 18 don't decrease $\bar{f}(S',U)$, then we can conclude that $\hat{M}_t=\{(i,\hat{n}_{t,i})\}$ constructed in Section E.3 can maximize $\bar{\omega}(t,U)$, we give the following theorem without proof.

Theorem 21. *For $U\subset[0,\lfloor\log t\rfloor],|U|=\mu\le\lfloor\log t\rfloor$, we have*

$$\bar{\omega}(t,U)=\begin{cases}\sum_{i\in U}\hat{n}_{t,i}+\sum_{i\notin U}2^{i-i^*}\cdot\hat{n}_{t,i} & \text{if } \alpha_1\neq 0\\ \mu+2^{\lfloor\log t\rfloor-\mu+1}-1 & \text{if } \alpha_1=0\end{cases}$$

where $i^*=\mathsf{SetRound}(i,U)$, α_2 is the largest index such that $t_{\alpha_2},\cdots,t_1=1$, if $t_1=0$, then $\alpha_2=0$.

E.5 Conclusion

To minimize both $\bar{\omega}(t, U)$ and $|U|$, recall that $\hat{n}_{t,i} = 0$ only for $i = \lfloor \log t \rfloor$. According to Theorem 21, we have:

$$\bar{\omega}(t, U) = \sum_{i \in U} \hat{n}_{t,i} + \sum_{i \notin U} 2^{i-i^*} \cdot \hat{n}_{t,i}$$

Thus, $\bar{\omega}(t, U)$ can be made small if U contains a small number of consecutive numbers. For example, if we set $U_t = \{2i \in [0, \lfloor \log t \rfloor] : i \in \mathbb{Z}\}$, then for all $i \notin U_t$, we have $i - \mathsf{SetRound}(i, U_t) = 1$. Consequently,

$$\begin{aligned} \bar{\omega}(t, U_t) &= \sum_{i \in U_t} \hat{n}_{t,i} + \sum_{i \notin U_t} 2\hat{n}_{t,i} \\ &= \sum_{i \text{ is even in } [0, \lfloor \log t \rfloor]} \hat{n}_{t,i} + 2 \sum_{i \text{ is odd in } [0, \lfloor \log t \rfloor]} \hat{n}_{t,i} \end{aligned}$$

We can efficiently compute $\bar{\omega}(t, U)$ for all settings of U based on the definition of $\hat{M}_t$ and Theorem 21. Some values of $\bar{\omega}(t, U)$ for $t = 17, 33, 65, 129, 257, 513$ and U with different sizes are presented in Table 4.

Table 4: Relationship between $\bar{\omega}(t, U)$ and U

$t = 17, U \subset [0, 4]$							
U	$\{0, 2\}$	$[0, 2]$	$[0, 3]$				
$\bar{\omega}(t, U)$	8	6	5				
$t = 33, U \subset [0, 5]$							
U	$\{0, 2\}$	$\{0, 2, 4\}$	$[0, 3]$	$[0, 4]$			
$\bar{\omega}(t, U)$	12	9	7	6			
$t = 65, U \subset [0, 6]$							
U	$\{0, 3\}$	$\{0, 2, 4\}$	$\{0, 1, 3, 4\}$	$[0, 4]$	$[0, 5]$		
$\bar{\omega}(t, U)$	16	11	9	8	7		
$t = 129, U \subset [0, 7]$							
U	$\{0, 3\}$	$\{0, 2, 4\}$	$\{0, 1, 3, 5\}$	$\{0, 1, 2, 3, 5\}$	$[0, 5]$	$[0, 6]$	
$\bar{\omega}(t, U)$	24	15	11	10	9	8	
$t = 257, U \subset [0, 8]$							
U	$\{0, 4\}$	$\{0, 2, 5\}$	$\{0, 2, 4, 6\}$	$\{0, 1, 2, 4, 6\}$	$\{0, 1, 2, 3, 4, 6\}$	$[0, 6]$	$[0, 7]$
$\bar{\omega}(t, U)$	32	17	14	12	11	10	9
$t = 513, U \subset [0, 9]$							
U	$\{0, 4\}$	$\{0, 3, 6\}$	$\{0, 2, 4, 6\}$	$\{0, 1, 3, 5, 7\}$	$\{0, 1, 2, 3, 5, 7\}$	$[0, 7] \setminus \{6\}$	$[0, 7]$	$[0, 8]$
$\bar{\omega}(t, U)$	48	23	18	14	13	12	11	10

F Performance and Evaluation in WAN Setting

We also do experiments in WAN setting with simulated network including 1Gbps with 40ms latency, and 100Mbps bandwidth with 80ms latency. The results are shown in Table 5 and 6. For our fuzzy PSI protocols in low-dimension (e.g., $d = 2$), the running time fluctuates slightly with different bandwidths because of the low communication cost. For example, for $N_r = N_s = 2^{12}$, $\delta = 256$, and L_2 distance, the running time only increases 4.8% when the bandwidth decreases from 1Gbps to 100Mbps. For our fuzzy PSI protocols in high-dimension (e.g., $d = 5$), when the set size is 2^{12}, the running time under 100Mbps bandwidth is approximately 2× slower than that under LAN configuration.

Table 5: Communication (MB) and computation (s) of our fuzzy PSI protocol in low ($d = 2$) dimension in LAN and WAN network. We set $N_r = N_s = 2^{\{4,8,12\}}$ and $\delta = 2^{\{4,5,6,7,8\}}$.

N	δ	L_∞				L_1				L_2			
		Comm.	Time			Comm.	Time			Comm.	Time		
			10Gbps	1Gbps	100Mbps		10Gbps	1Gbps	100Mbps		10Gbps	1Gbps	100Mbps
2^4	16	0.440	0.120	0.165	0.164	0.970	0.644	0.748	0.751	1.669	0.656	0.761	0.753
	32	0.520	0.128	0.175	0.173	1.221	0.962	0.981	1.060	2.071	0.855	1.011	1.055
	64	0.594	0.137	0.192	0.198	1.490	1.193	1.388	1.443	2.505	1.148	1.336	1.409
	128	0.672	0.149	0.202	0.207	1.781	1.505	1.763	1.792	2.994	1.518	1.758	1.712
	256	0.747	0.158	0.228	0.248	2.088	1.818	2.150	2.197	3.559	1.912	2.213	2.042
2^8	16	6.820	0.967	1.018	1.081	15.513	7.251	7.593	7.529	26.697	8.208	7.322	7.599
	32	7.989	1.130	1.057	1.085	19.552	10.469	11.342	11.731	33.157	11.404	10.676	11.046
	64	9.13	1.248	1.24	1.218	23.869	14.454	16.421	16.296	40.139	15.649	15.146	16.137
	128	10.240	1.408	1.379	1.428	28.612	19.264	20.260	20.652	47.916	20.919	19.887	23.679
	256	11.380	1.622	1.544	1.650	33.416	24.844	25.928	25.846	56.964	26.628	27.359	27.973
2^{12}	16	111.346	15.001	13.984	13.934	248.221	110.338	116.415	123.547	427.163	116.191	114.553	135.892
	32	129.381	17.624	15.847	16.284	312.844	164.382	180.834	182.395	530.489	165.26	172.729	195.904
	64	148.523	19.208	18.568	19.672	381.934	231.474	243.433	252.485	642.321	235.149	236.922	268.352
	128	165.835	21.545	22.624	21.176	456.058	306.930	334.630	320.626	766.632	309.994	315.972	369.883
	256	184.969	26.185	22.429	23.936	534.674	396.909	416.623	414.947	911.455	399.774	431.954	452.521

Table 6: Communication (MB) and computation (s) of our fuzzy PSI protocol in high ($d = 5$) dimension in LAN and WAN network. We set $N_r = N_s = 2^{\{4,8,12\}}$ and $\delta = 2^{\{4,5,6,7,8\}}$.

N	δ	L_∞				L_1				L_2			
		Comm.	Time			Comm.	Time			Comm.	Time		
			10Gbps	1Gbps	100Mbps		10Gbps	1Gbps	100Mbps		10Gbps	1Gbps	100Mbps
2^4	16	3.406	0.475	0.576	0.580	4.795	0.672	0.673	0.657	5.487	0.589	0.717	0.701
	32	3.669	0.494	0.612	0.610	5.581	0.748	0.742	0.736	6.638	0.658	0.868	0.798
	64	3.931	0.502	0.616	0.610	6.109	0.780	0.767	0.780	6.927	0.772	0.980	0.893
	128	4.151	0.523	0.646	0.645	6.542	0.803	0.818	0.807	7.589	0.997	1.163	1.064
	256	6.317	0.673	0.827	0.783	9.052	1.052	1.083	1.055	10.504	1.339	1.569	1.589
2^8	16	54.230	3.859	3.326	4.730	76.410	4.289	5.187	6.587	87.594	4.939	5.188	7.783
	32	58.410	4.159	3.812	5.176	88.921	4.875	4.852	7.597	105.826	5.931	6.076	9.980
	64	62.648	4.241	3.766	5.094	97.251	5.492	5.832	8.699	110.499	7.524	8.056	11.617
	128	66.131	4.384	3.828	5.471	104.242	5.889	6.471	9.231	121.177	9.944	10.843	14.601
	256	100.774	6.645	5.487	8.780	144.473	7.952	8.608	13.689	167.471	16.080	16.796	23.652
2^{12}	16	867.438	52.974	49.310	89.605	1222.339	63.807	67.038	129.593	1401.335	72.964	75.474	148.413
	32	934.878	56.539	53.042	95.254	1422.394	74.240	84.122	146.494	1692.893	88.737	93.701	182.842
	64	1002.178	54.520	59.747	102.369	1555.742	85.297	92.166	163.667	1767.740	118.481	124.372	219.734
	128	1057.915	57.283	63.270	110.726	1667.601	91.608	101.452	176.542	1938.598	157.057	164.746	289.706
	256	1612.166	86.925	94.767	167.077	2311.232	125.450	136.685	244.994	2679.230	271.537	271.360	436.095