


Fuzzy Private Set Intersection with Large Hyperballs

Aron van Baarsen^{*1,2} and Sihang Pu³ 
aronvanbaarsen@gmail.com, sihang.pu@gmail.com

¹ CWI, Cryptology Group, Amsterdam, The Netherlands

² Leiden University, Mathematical Institute, Leiden, The Netherlands


³ CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract. Traditional private set intersection (PSI) involves a receiver and a sender holding sets X and Y , respectively, with the receiver learning only the intersection $X \cap Y$. We turn our attention to its fuzzy variant, where the receiver holds $|X|$ hyperballs of radius δ in a metric space and the sender has $|Y|$ points. Representing the hyperballs by their center, the receiver learns the points $x \in X$ for which there exists $y \in Y$ such that $\text{dist}(x, y) \leq \delta$ with respect to some distance metric. Previous approaches either require general-purpose multi-party computation (MPC) techniques like garbled circuits or fully homomorphic encryption (FHE), leak details about the sender's precise inputs, support limited distance metrics, or scale poorly with the hyperballs' volume.

This work presents the first black-box construction for fuzzy PSI (including other variants such as PSI cardinality, labeled PSI, and circuit PSI), which can handle polynomially large radius and dimension (i.e., a potentially exponentially large volume) in two interaction messages, supporting general $L_{p \in [1, \infty]}$ distance, without relying on garbled circuits or FHE. The protocol excels in both asymptotic and concrete efficiency compared to existing works. For security, we solely rely on the assumption that the Decisional Diffie-Hellman (DDH) holds in the random oracle model.

* Research partially funded by NWO/TKI Grant 628.009.014.

Table of Contents

Fuzzy Private Set Intersection with Large Hyperballs	1
<i>Aron van Baarsen and Sihang Pu</i>  aronvanbaarsen@gmail.com , sihang.pu@gmail.com	
1 Introduction	3
1.1 Our Contributions	3
1.2 Related Work	5
1.3 Applications	6
2 Technical Overview	7
2.1 Recap: Apple’s PSI Protocol	7
2.2 Fuzzy Matching for Infinity Distance	7
2.3 Generalized Distance Functions	8
2.4 Fuzzy PSI in Low Dimensions	10
2.5 Extending to High Dimensions	11
3 Preliminaries	13
3.1 Oblivious Key-Value Store (OKVS)	14
3.2 Random Self-Reductions of DDH Tuples	15
3.3 Locality-Sensitive Hashing	15
3.4 Oblivious Programmable PRF	16
3.5 Partially Homomorphic Encryption	16
4 Definitions and Functionalities	18
4.1 Definition of Fuzzy Matching	18
4.2 Definition of Fuzzy (Circuit) Private Set Intersection	18
5 Fuzzy Matching	19
5.1 Fuzzy Matching for Infinity Distance	20
5.2 Fuzzy Matching for Minkowski Distance	22
6 Fuzzy PSI in Low-Dimension Space	24
6.1 Spatial Hashing Techniques	24
6.2 Fuzzy PSI-CA for Infinity Distance	26
6.3 Fuzzy PSI-CA for Minkowski Distance	28
7 Fuzzy PSI in High-Dimension Space	29
7.1 Infinity Distance	30
7.2 Minkowski Distance	33
8 Extending to Broader Functionalities	35
8.1 Labeled PSI	35
8.2 Standard PSI	36
8.3 Standard PSI with Sender Privacy (PSI-SP)	36
8.4 Circuit PSI	38
8.5 Reducing Comparisons for PSI-SP and Circuit PSI	39
9 Performance Evaluation	39
9.1 Concrete Performance	40
10 Conclusion	40

1 Introduction

Private set intersection (PSI) is a cryptographic primitive that allows two parties to compute the intersection $X \cap Y$ of their private datasets X and Y , without revealing any information about items not in the intersection. The first PSI protocol is often dated back to Meadows [Mea86] and many modern protocols still have the same structure using an oblivious pseudorandom function (OPRF) [KKRT16,RS21,RR22]. Recent PSI protocols are very practical and can for example compute the intersection of sets of 2^{20} elements in ≈ 0.37 seconds [RR22]. Many richer PSI functionalities have also been explored, such as: *PSI cardinality* [FNP04,IKN⁺20,DPT20], where only the cardinality of the intersection is revealed; *labeled PSI* [CGN98,CHLR18,CMdG⁺21], which allows the parties to learn labels associated to the items in the intersection; *circuit PSI* [HEK12,PSTY19,RS21], which only reveals secret shares of the intersection and allows the parties to securely evaluate any function on the intersection.

Recently Garimella et al. [GRS22,GRS23] introduced the concept of *structure-aware PSI*, where the receiver’s input set has some publicly known structure. For example, the receiver holds N balls of radius δ and dimension d and the sender holds a set of M points, and the sender learns which of the sender’s points lie within one of their balls. This special case is often referred to as *fuzzy PSI* and is the focus of our work. Using a standard PSI protocol for this task leads to a rather inefficient solution since the communication and computation complexity usually scale at least linearly in the cardinality of the input sets, i.e., the total volume of the balls $N \cdot \delta^d$. Garimella et al. can overcome this barrier in terms of communication in the semi-honest [GRS22] as well as in the malicious [GRS23] setting. However, the receiver’s computation is still proportional to the total volume of the input balls, which makes their protocols scale poorly with the dimension d . Moreover, their protocols are limited to the L_∞ and L_1 ⁴ distance and only realize a *standard PSI* functionality, where the receiver learns exactly which of the sender’s points lie in the intersection. Other works are either limited to the Hamming distance [UCK⁺21], Hamming and L_2 distance [IW06] or Hamming distance and one-dimensional L_1 distance [CFR23], and often require heavy machinery or yield non-negligible correctness error.

In this work, we present fuzzy PSI protocols in the semi-honest setting for general L_∞ and L_p distance with $p \in [1, \infty)$, and present several optimized variants for low as well as high dimensions. Notably, the communication as well as computation complexity of our high-dimension protocols scales linearly or quadratically with the dimension d . We moreover extend our protocols to various richer fuzzy PSI functionalities including PSI cardinality, labeled PSI, PSI with sender privacy, and circuit PSI. Our protocols have comparable performance to [GRS22] in the low-dimensional setting and significantly outperform other approaches when the dimension increases. Finally, our protocols rely only on the decisional Diffie-Hellman (DDH) assumption.

1.1 Our Contributions

Fuzzy Matching. The main building block for our fuzzy PSI constructions is a fuzzy matching protocol, which on input a point $\mathbf{w} \in \mathbb{Z}^d$ from the receiver and a point $\mathbf{q} \in \mathbb{Z}^d$ from the sender, outputs 1 to the receiver if $\text{dist}(\mathbf{w}, \mathbf{q}) \leq \delta$ and 0 otherwise. Here dist can either be L_∞ distance or L_p distance for $p \in [1, \infty)$. It results in a two-message protocol for L_∞ distance with communication complexity $O(\delta d)$, computation complexity $O(\delta d)$ for the receiver and $O(d)$ for the sender when $\text{dist} = L_\infty$; For L_p distance, it has communication complexity $O(\delta d + \delta^p)$, computation complexity $O(\delta d)$ for the receiver, and $O(d + \delta^p)$ for the sender.

⁴ The overhead of L_1 balls would be $\frac{2^d}{d}$ times larger than that of L_∞ balls in their protocols.

Table 1. Asymptotic complexities of fuzzy PSI protocols, where the receiver holds N hyperballs of radius δ and the sender holds M points in \mathbb{Z}^d . $\rho \leq 1/c$ is a parameter to the LSH scheme if the receiver’s points are distance $> c\delta$ apart. We ignore multiplicative factors of the computational security parameter λ and statistical parameter κ .

Setting [protocol]	Communication	Comp.(receiver)	Comp. (sender)	
L_∞	trad. PSI [RR22]	$O((2\delta)^d N + M)$	$O((2\delta)^d N)$	$O((2\delta)^d N + M)$
	$> 2\delta$ [ours]	$\mathbf{O}(\delta \mathbf{d} \mathbf{N} + \mathbf{2}^{\mathbf{d}} \mathbf{M})$	$\mathbf{O}(\delta \mathbf{d} \mathbf{N} + \mathbf{2}^{\mathbf{d}} \mathbf{M})$	$\mathbf{O}(\mathbf{2}^{\mathbf{d}} \mathbf{d} \mathbf{M})$
	$> 2\delta$ [GRS22]	$O((4 \log \delta)^d N + M)$	$O((2\delta)^d N)$	$O((2 \log \delta)^d M)$
	$> 4\delta$ [ours]	$O(\delta 2^d d N + M)$	$\mathbf{O}(\delta \mathbf{2}^{\mathbf{d}} \mathbf{d} \mathbf{N} + \mathbf{M})$	$\mathbf{O}(\mathbf{d} \mathbf{M})$
	$> 4\delta$ [GRS22]	$O(2^d d N \log \delta + M)$	$O((2\delta)^d N)$	$O(d M \log \delta)$
	$> 8\delta$ [GRS23]	$O(d N \log \delta + M)$	$O((2\delta)^d N)$	$O(d M \log \delta)$
	\exists disj. proj. [ours]	$O((\delta d)^2 N + M)$	$O((\delta d)^2 N + M)$	$O(d^2 M)$
\forall disj. proj. [GRS22]	$O(d N \log \delta + M)$	$O((2\delta)^d N)$	$O(d M \log \delta)$	
L_p	$> 2\delta(d^{1/p} + 1)$ [ours]	$O(\delta 2^d d N + \delta^p M)$	$O(\delta 2^d d N + M)$	$O((d + \delta^p) M)$
	LSH [ours]	$O(\delta d N^{1+\rho} + \delta^p M N^\rho \log N)$	$O(\delta d N^{1+\rho} + M N^\rho \log N)$	$O((d + \delta^p) M N^\rho \log N)$

Fuzzy PSI in Low-Dimensions. Using a fuzzy matching protocol we can trivially obtain a fuzzy PSI protocol by letting the sender and receiver run the fuzzy matching protocol for every combination of inputs, but this leads to an undesirable $N \cdot M$ blowup in communication and computation complexity. To circumvent this blowup for a low dimension d , we develop a new spatial hashing technique for disjoint L_∞ balls which incurs only a $O(2^d)$ factor to the receiver’s communication and sender’s computational complexity. To support L_p balls, we extend the “shattering” idea from [GRS22] to generalized L_p setting. The asymptotic complexities are given in Table 1. It is worth noting that, unlike to [GRS22, GRS23], the computation complexity of our protocols scale *sublinearly* to the volume of balls.

Fuzzy PSI in High-Dimensions. Unfortunately, the above spatial hashing approaches still yield a 2^d factor in the communication and computation complexities, which becomes prohibitive for large dimensions d . The earlier work [GRS22] proposes a protocol that can overcome this factor, for communication costs, in the L_∞ setting under the *globally disjoint* assumption that the projections $[w_{k,i} - \delta, w_{k,i} + \delta]$ of the sender’s balls $k \in [N]$ are disjoint *for all* dimensions $i \in [d]$, which the authors themselves mention is a somewhat artificial assumption. We present a two-message protocol with comparable communication and much lower computation complexity under a milder assumption that for each $k \in [N]$ *there exists* a dimension $i \in [d]$ where the projection is disjoint from all other $k' \in [N]$, namely, not necessary to be globally disjoint. We argue that this is a more realistic assumption since points in high dimensions tend to be sparser and show that it is satisfied with a high probability if the points \mathbf{w}_k are uniformly distributed.

We moreover present a two-message protocol in the L_p setting which can circumvent this exponential factor in the dimension d , while achieving sub-quadratic complexity in the number of inputs. The key idea of this protocol is to use locality-sensitive hashing (LSH) to perform a coarse mapping such that points close to each other end up in the same bucket with high probability, and

subsequently use our fuzzy matching protocol for L_p distance to compare the items in each bucket. See Table 1 for the asymptotic complexities of our protocols.

Extensions to Broader Functionalities. By default, all our protocols except for the LSH-based protocol realize the stricter PSI functionality where the receiver only learns how many of the sender’s points lie close to any of the receiver’s points, which we call *PSI cardinality* (PSI-CA). Earlier works [GRS22,GRS23] realize the functionality where the receiver learns exactly which of the sender’s points are in the intersection. We refer to this functionality as *standard PSI*.

For all of our protocols discussed above, except for the LSH-based protocol, we show that we can extend them to realize the following functionalities: *standard PSI*; *PSI with sender privacy* (PSI-SP), where the receiver only learns which of the receiver’s balls are in the intersection; *labeled PSI*, where the receiver only learns some label associated to the sender’s points in the intersection; *circuit PSI*, where the parties only learn secret shares of the intersection and optional data associated to each input point, which they can use as the input to any secure follow-up computation. We can realize these extensions without increasing the asymptotic complexities of the protocols and without needing to introduce additional computational assumptions. With the only exception that for the circuit PSI extension we need a generic MPC functionality to compute a secure comparison circuit at the end of the protocol, which is common for traditional (non-fuzzy) circuit PSI protocols [PSTY19,RS21].

Performance. Our experimental results demonstrate that it requires only 1.2 GB bandwidth and 432 seconds in total to complete a standard fuzzy PSI protocol when parties have thousands of L_∞ balls and millions of points in a 5-dimensional space. As a comparison, prior works need $\gg 4300$ seconds (conservative estimate). We also explore the scenario where both parties have structured sets. For instance, when each party holds thousands of 64-radius L_∞ balls in a 5-dimensional space (representing 2^{46} points), our protocol takes only 240 MB and 97 seconds to finish. Please refer to Section 9 for details.

1.2 Related Work

Traditional PSI protocols have become very efficient [KKRT16,CM20,RR22], but are optimized for the setting where the parties’ input sets have approximately the same size, and their communication and computation costs scale linearly with the input size. This leads to an inefficient fuzzy PSI protocol since the receiver’s input size is $N \cdot \delta^d$ when the receiver holds N hyperballs of dimension d and radius δ . Asymmetric (or unbalanced) PSI protocols [CLR17,CHLR18,CMdG+21,ABD+21] target the setting where one party’s set is much larger than the other’s and can achieve communication complexity sublinear in the large set size, but $O(\sqrt{N} \cdot \delta^{d/2})$ computational complexity, using fully homomorphic encryption [CMdG+21]. For traditional PSI there exist many efficient protocols realizing richer functionalities such as PSI cardinality [IKN+20,DPT20], labeled PSI [CHLR18,CMdG+21] and circuit PSI [HEK12,PSTY19,RS21], but all of these suffer from the same limitations as discussed above when applied to the fuzzy PSI setting. There exists another line of works concerning threshold PSI [GS19,BMRR21,BDP21,GS23], where the fuzziness is measured by the number of *exact* matches between items.

Secure fuzzy matching was introduced by Freedman et al. [FNP04] as the problem of identifying when two tuples have a Hamming distance below a certain threshold. They propose a protocol based on additively homomorphic encryption and polynomial interpolation, which was later shown to be insecure [CH08]. Follow-up works focus on the Hamming distance as well and use similar oblivious

polynomial evaluation techniques [CH08, YSPW10]. Indyk and Woodruff [IW06] construct a fuzzy PSI protocol for L_2 and Hamming distance using garbled circuits. Uzun et al. [UCK⁺21] give a protocol for fuzzy labeled PSI for Hamming distance using garbled circuits and fully homomorphic encryption. Chakraborti et al. [CFR23] propose a fuzzy PSI protocol for Hamming distance based on additively homomorphic encryption and vector oblivious linear evaluation (VOLE), which has a non-negligible false positive rate. They moreover present a protocol for one-dimensional L_1 distance, which can be constructed using any $O(N)$ communication PSI protocol for sets of size N , and has resulting communication complexity $O(N \log \delta)$ [CFR23]. It is an interesting question whether their techniques can be extended to higher dimensions. Since the focus of our work is to construct fuzzy PSI protocols for general L_p and L_∞ distances, general dimension d , and with negligible error rate, it is not possible to make a meaningful comparison with these works.

Garimella et al. [GRS22, GRS23] initiated the study of structure-aware PSI, which covers fuzzy PSI as a special case. They introduce the definition of weak boolean function secret sharing (bFSS) for set membership testing and give a general protocol for structure-aware PSI from bFSS. They develop several new bFSS techniques, focusing on the case where the input set is the union of N balls of radius δ with respect to the L_∞ norm in d -dimensional space, which results in a fuzzy PSI protocol as the ones we concern ourselves with in this work. The techniques used in their protocols are fundamentally different from ours, except that we use similar spatial hashing techniques to obtain efficient fuzzy matching protocols in the low-dimensional setting. Moreover, their protocols are limited to the L_∞ and L_1 distance setting and only realize the standard PSI functionality where the receiver learns the exact sender’s points in the intersection. Finally, the receiver’s computational complexity in their protocols scales as $O((2\delta)^d N)$, which makes them unsuitable in the high-dimensional setting. See Table 1 for a more detailed comparison of communication and computational complexities.

1.3 Applications

Private proximity detection. There exist certain contexts where individuals need to know the proximity of others for varying purposes: In the realm of contact tracing, where individuals may seek to determine if they are in the vicinity of an infected person; Within the scope of ride-sharing platforms, users might wish to identify available vehicles in their surroundings. In both scenarios, the privacy of all involved parties should be preserved and fuzzy PSI protocols provide a direct solution to this problem.

Biometric and password identification. Fuzzy matching could also be useful in authentication or identification scenarios. Notable applications of this technique can be observed in the matching of similar passwords to enhance usability or security. A case in point is Facebook’s authentication protocol, which auto-corrects the capitalization of the initial character in passwords [Muf15]. Similarly, it can be useful to check if a user’s password is similar to a leaked password [PIB⁺22]. Furthermore, fuzzy matching can be employed to match biometric data, such as fingerprint and iris scans, thereby facilitating a blend of convenience and security [DHP⁺18]. In general, a fuzzy unbalanced PSI protocol is more useful since the server usually holds a large database of clients’ passwords (or biometric samples).

Illegal content detection. Recently, Bartusek et al. [BGJP23] introduced the study of illegal content detection within the framework of end-to-end secure messaging, focusing particularly on the detection of child sexual abuse material, encompassing photographs and videos. Central to

their protocol is a two-message PSI protocol, wherein the initial message is reusable and published once for the receiver’s database. After this, the computational overhead for both parties is rendered independent of the database size. The research notably leverages Apple’s PSI protocol [BBM⁺21], which, while only facilitating exact matches, serves its purpose effectively. Ideally, matching should be sufficiently fuzzy to ensure that illegal images remain detectable even following rotation or mild post-processing. Our fuzzy PSI constructions, encapsulated within two-round protocols and featuring a reusable initial message, may find potential applicability in such contexts.

2 Technical Overview

Before heading for the details of our fuzzy matching and PSI protocols, let us start by discussing a standard PSI protocol proposed by Apple [BBM⁺21].

2.1 Recap: Apple’s PSI Protocol

We simplify Apple’s PSI protocol to the basic setting where the receiver holds a set $W := \{w_1, \dots, w_n\}$, the sender holds an item q , and the receiver wants to learn q if $q \in W$ and nothing otherwise. Their main idea is a novel usage of random self-reduction of DDH tuples from Naor and Reingold [NR97] in PSI contexts. Given a cyclic group $\mathbb{G} := \langle g \rangle$ of prime order p , the tuple (g, h, h_1, h_2) can be re-randomized into (g, h, u, v) such that u, v are uniformly random over \mathbb{G} as long as (g, h, h_1, h_2) is *not* a well-formed DDH tuple (i.e., there is no $s \in \mathbb{Z}_p^*$ to satisfy $g^s = h \wedge h_1^s = h_2$). Otherwise, both (g, h, h_1, h_2) and the re-randomized tuple (g, h, u, v) are valid DDH tuples. This re-randomization basically utilizes two random coins $a, b \leftarrow_{\$} \mathbb{Z}_p^*$ to output

$$(u := g^a h_1^b, v := h^a h_2^b).$$

Now to obtain a PSI protocol, the receiver could sample $s \leftarrow_{\$} \mathbb{Z}_p^*$ and publish

$$(g, h := g^s, H(w_1)^s, \dots, H(w_n)^s),$$

where H is a hash-to-group function. Then the sender returns pairs

$$(u_i, \text{ct}_i := \text{Enc}_{v_i}(q))_{i \in [n]},$$

where (u_i, v_i) is the re-randomization output for each tuple $(g, h, H(q), H(w_i)^s)$, and Enc is some symmetric-key encryption scheme (e.g., a one-time pad). The receiver can try to decrypt each ct_i using the key u_i^s to learn q . For the sender’s privacy, the random self-reduction of DDH tuples guarantees that when $q \neq w_i$, the secret key v_i is uniformly random from the receiver’s view and thus q is hidden according to the security of this symmetric-key encryption. For the receiver’s privacy, $(H(w_1)^s, \dots, H(w_n)^s)$ is pseudorandom according to the generalized DDH assumption when H is modelled as a random oracle.

2.2 Fuzzy Matching for Infinity Distance

Our crucial observation is that the above approach can be naturally applied in fuzzy matching protocols where the receiver holds a point $\mathbf{w} \in \mathbb{Z}^d$ in a d -dimensional space, the sender holds a point $\mathbf{q} \in \mathbb{Z}^d$, and the receiver learns if $\text{dist}(\mathbf{w}, \mathbf{q}) \leq \delta$. Here, δ is the maximal allowed distance between

\mathbf{w} and \mathbf{q} . For the moment, let us focus on the simplest case where the distance is calculated over L_∞ , which means, the receiver gets 1 if

$$\forall i \in [d] : w_i - \delta \leq q_i \leq w_i + \delta,$$

and gets 0 otherwise. This problem is equivalent to the following: The receiver holds d sets $\{W_1, \dots, W_d\}$ where $W_i := \{w_i - \delta, \dots, w_i + \delta\}$, the sender holds d items $\{q_1, \dots, q_d\}$, and they want to run a membership test for each dimension simultaneously, without leaking the results for individual dimensions. Though the receiver can publish $H(w_i + j)^s$ for each $i \in [d], j \in [-\delta, +\delta]$ as above, the sender has to use random self-reduction for each possible match, which yields too much communication and computation effort for the sender. Namely, the entire volume of a d -dimensional δ -radius ball $O((2\delta + 1)^d)$.

Reducing the complexity. There is a standard trick to significantly reduce the complexity by using an oblivious key-value store (OKVS). Recall that an OKVS [GPR⁺21] will encode a key-value list $\{(\text{key}_j, \text{val}_j)\}_{j \in [n]}$ into a data structure E , such that decoding with a correct key_* returns the corresponding val_* , where the encoding time scales linearly to the list size and decoding a single key takes only a constant number of operations. So the above protocol can be improved as follows:

- 1) The receiver publishes $(g, h, E_i \leftarrow \text{Encode}(\{(w_i + j, H(w_i + j)^s)\}_{j \in [-\delta, +\delta]}))$ for each $i \in [d]$;
- 2) The sender retrieves $h_i \leftarrow \text{Decode}(E_i, q_i)$ for each $i \in [d]$ and sends the rerandomized tuple $(u := g^a \prod_{i=1}^d H(q_i)^b, v := h^a \prod_{i=1}^d h_i^b)$, where $a, b \leftarrow_{\$} \mathbb{Z}_p^*$, to the receiver;
- 3) The receiver checks if (g, h, u, v) is a valid DDH tuple.

The protocol is correct when $\text{dist}(\mathbf{q}, \mathbf{w}) \leq \delta$, according to the correctness of the underlying OKVS scheme, which says that decoding the structure E_i with a correct encoding key q_i will return the encoded value $h_i := H(q_i)^s$; When $\text{dist}(\mathbf{q}, \mathbf{w}) > \delta$, we typically need to rely on the independence property of OKVS, which says that decoding with a non-encoded key will yield a uniformly random result. Therefore, in this case, there exists at least one h_{i^*} that is uniformly random; hence $(g, h, H(q_{i^*}), h_{i^*})$ is not a DDH tuple except with negligible probability. The sender's privacy can be established as before from the random self-reduction of DDH tuples. To argue the receiver's privacy, we rely on the obliviousness property of the OKVS, namely, the encoded keys $\{w_i + j\}$ are completely hidden as long as the encoded values $\{H(w_i + j)^s\}$ are uniformly random. Since $(h, H(w_i + j), H(w_i + j)^s)$ is pseudorandom by the DDH assumption, then according to the obliviousness of OKVS, the receiver's message can be simulated by encoding random key-value pairs.

Note that our real construction shown in Section 5.1, is slightly different from what we described here. We encode the OKVS "over the exponent" to reduce heavy public-key operations over \mathbb{G} because our encoded values are pseudorandom over a *structured* group \mathbb{G} (i.e., the elliptic curves).

So far, we have obtained a two-message fuzzy matching protocol for L_∞ distance, with $O(d\delta)$ communication and computation complexity.

2.3 Generalized Distance Functions

When the distance function is calculated in L_p , the receiver would get 1 if

$$\text{dist}_p(\mathbf{w}, \mathbf{q}) := \left(\sum_{i=1}^d |w_i - q_i|^p \right)^{1/p} \leq \delta,$$

and 0 otherwise. To make the problem easier, we consider the \mathbf{p} -powered $L_{\mathbf{p}}$ distance, namely, we check if $\sum_{i=1}^d |w_i - q_i|^{\mathbf{p}} \leq \delta^{\mathbf{p}}$. Thanks to the homomorphism of DDH tuples, the sender can homomorphically evaluate the distance function. Moreover, since an $L_{\mathbf{p} \geq 1}$ ball must be confined in an L_{∞} ball, namely, $|w_i - q_i| \leq \delta$ for any $i \in [d]$ if $\text{dist}_{\mathbf{p}}(\mathbf{w}, \mathbf{q}) \leq \delta$, the protocol could work as follows:

- 1) The receiver publishes

$$\left(g, h, E_i \leftarrow \text{Encode} \left(\left\{ (w_i + j, \text{H}(w_i + j)^s \cdot g^{|j|^{\mathbf{p}}}) \right\}_{j \in [-\delta: \delta]} \right) \right),$$

for each $i \in [d]$;

- 2) The sender retrieves $h_i \leftarrow \text{Decode}(E_i, q_i)$ for each $i \in [d]$, and computes

$$\left(u := g^a \prod_{i=1}^d \text{H}(q_i)^b, v := h^a \prod_{i=1}^d h_i^b \right),$$

for random $a, b \leftarrow \mathbb{Z}_p^*$;

- 3) The sender generates a list $\text{list} := \{g^{b \cdot j}\}_{j \in [0: \delta^{\mathbf{p}}]}$ and outputs (u, v, list) ;
- 4) The receiver checks if there is any $x \in \text{list}$ such that $v = u^s \cdot x$.

Denote $t := \text{dist}_{\mathbf{p}}(\mathbf{w}, \mathbf{q})$. If $\forall i \in [d], |w_i - q_i| \leq \delta$, then the correctness holds naturally since each retrieved $h_i := \text{H}(q_i)^s \cdot g^{|w_i - q_i|^{\mathbf{p}}}$, implying that

$$\frac{v}{u^s} = g^{b \cdot t^{\mathbf{p}}},$$

which would be included in list if and only if $t \leq \delta$ ⁵. On the other hand, if there exists $i^* \in [d]$ such that $|w_{i^*} - q_{i^*}| > \delta$, then according to the independence property of OKVS the decoded h_{i^*} as well as v would be uniformly random over \mathbb{G} , such that $v/u^s \in \text{list}$ with only negligible probability.

Subtle issues and the fix. The receiver's privacy is almost the same as before, relying on the generalized DDH assumption and the obliviousness of OKVS. It is a little bit subtle to argue the sender's privacy: Currently, list would leak information on the sender's input. Precisely, given (u, v, list) , the receiver could check, for example, if $\frac{v}{u^s \cdot g^b} \in \text{list}$ to learn if $t^{\mathbf{p}} = \delta^{\mathbf{p}} + 1$ or not, since $g^b \in \text{list}$. Moreover, even in the case that $t \leq \delta$, the receiver could still deduce the *exact* t by checking which index is matched. The latter can be solved by shuffling the list, so we focus on the former issue. One approach is to hash each list item as $\text{list} := \{\text{H}'(g^{b \cdot j})\}_{j \in [0: \delta^{\mathbf{p}}]}$. By modeling $\text{H}' : \mathbb{G} \mapsto \{0, 1\}^*$ as a random oracle, the group structure is erased and the adversary cannot utilize $g^{b \cdot j}$ anymore. However, the issue still exists since the adversary could check if $\text{H}'((\frac{v}{u^s})^{1/\alpha}) \in \text{list}$ to learn if $t^{\mathbf{p}} \in \{0, \alpha, 2\alpha, \dots, \delta^{\mathbf{p}}\alpha\}$ for any α . Therefore, we have to apply a random linear function over $t^{\mathbf{p}}$ to make sure that $\frac{v}{u^s} = g^{b \cdot t^{\mathbf{p}} + c}$ where b, c are random scalars. The details can be found in Section 5.2.

Regarding the complexity, the communication and computation are increased by an additive term $O(\delta^{\mathbf{p}})$ from the infinity distance setting.

⁵ Assuming the group order p is large enough that $t^{\mathbf{p}} < p$.

2.4 Fuzzy PSI in Low Dimensions

For the moment, let us consider the fuzzy PSI *cardinality* problem, where the receiver holds a union of d -dimensional balls of radius δ represented by their centers $\{\mathbf{w}_1, \dots, \mathbf{w}_N\}$, the sender holds a set of points $\{\mathbf{q}_1, \dots, \mathbf{q}_M\}$ in the same space, and the receiver learns *the number of* sender’s points located inside the balls. When the dimension d of the space is low, e.g., $O(\log(\lambda))$, we can exploit the geometric structure of the space to efficiently match balls and points to avoid the quadratic blowup mentioned in the introduction. The high-level idea is to tile the entire space by d -dimensional hypercubes of side-length 2δ (also called *cells*, together a *grid*), then the receiver can encode a ball (represented by its center \mathbf{w}_i) in a way that the sender can efficiently match it with a point \mathbf{q}_j , without enumerating all balls. After that, both parties can run a fuzzy matching protocol between \mathbf{w}_i and \mathbf{q}_j as before.

The idea of Garimella et al. [GRS22] is to “shatter” each receiver’s ball into its intersected cells, however, to guarantee each cell is intersected with a single ball (otherwise collisions appear during encoding an OKVS⁶), the receiver’s balls typically need to be at least 4δ apart from each other. To tackle the case of disjoint balls, the authors improved their techniques [GRS23] by observing that each grid cell can only contain the center of a single receiver’s ball. Thus, the receiver could encode the identifier of each cell which contains a ball center, and the sender can try to decode the OKVS by iterating over all neighborhood cells⁷ surrounding its point. This approach yields a $O(3^d)$ factor for the sender’s computation and communication costs: Given a point \mathbf{q} , the center of any L_∞ ball intersected with \mathbf{q} is located in at most 3^d cells surrounding the cell containing \mathbf{q} .

New spatial hashing ideas. Here we provide a new hashing technique to reduce this blowup from 3^d to 2^d . Note that the 3^d factor comes from the fact that the entire neighborhood of the point \mathbf{q} is too large (i.e., a hypercube of side-length 6δ), but we only need to care about the neighbor cells that intersected with the receiver’s balls already. Specifically, if the grid is set properly, an L_∞ ball will intersect *exactly* 2^d cells, which constitute a hypercube of side-length 4δ , denoted as a *block*. Our crucial observation is that each block is *unique* for each disjoint ball, i.e., two disjoint balls must be associated with different blocks, as detailed in Lemma 6. Given this, the receiver could encode the identifier of each block, and the sender would decode by iterating all potential blocks. There are in total 2^d possible blocks for each sender’s point due to each block being comprised of 2^d cells and each cell contains at most a single ball’s center.

Compatible with L_p balls. Though we only considered L_∞ balls so far, we can generalize the “shattering” idea from [GRS22] to L_p balls as well. We still tile the space with hypercubes, but we show that as long as L_p balls are at least $2\delta(d^{1/p} + 1)$ apart from each other, then each grid cell intersects at most one L_p ball, as detailed in Lemma 5. In particular, when $p = \infty$, $2\delta(d^{1/p} + 1)$ degrades to the original 4δ . Combining it with our fuzzy matching protocols for the L_p distance, we immediately obtain fuzzy PSI for precise L_p distance (i.e., without approximation or metric embedding). An important step different from the L_∞ setting is to pad the key-value list to size $2^d N$ with random pairs since an L_p ball could intersect with a various number of cells. Otherwise, the receiver’s privacy would be compromised.

⁶ Note that this is not a problem in their setting as they use the function secret sharing (FSS) to handle each grid cell.

⁷ The neighborhood is a hypercube of side-length 6δ .

2.5 Extending to High Dimensions

To overcome the 2^d factor in the complexities, we first focus our attention on L_∞ distances: Ideally, if the receiver's balls are *globally disjoint* on every dimension, that is, the projection of the balls on each dimension never overlaps, then the ‘‘collision’’ issue mentioned above would disappear. In this way, for each dimension $i \in [d]$, the receiver could encode the OKVS as

$$E_i \leftarrow \text{Encode} \left(\left\{ (w_{k,i} + j, \quad \mathbf{H}(w_{k,i} + j)^s \cdot \zeta_{k,i}) \right\}_{k \in [N], j \in [-\delta, +\delta]} \right),$$

where $w_{k,i}$ is the projection of the ball center \mathbf{w}_k on dimension i and $\zeta_{k,i}$ is a random secret share of 1 such that $1 = \prod_{i=1}^d \zeta_{k,i}$, for each $k \in [N]$. The sender just behaves the same as in Section 2.2. This approach results in $O(\delta d N + M)$ communication and computation costs. However, as stated in [GRS22], this ideal setting is somewhat artificial and unrealistic.

Weaker assumptions by leveraging dummy OKVS instances. After taking a closer look at this approach, we realized that the global disjointness is not necessary to be satisfied on *every* dimension, as we actually could tolerate some collisions. Specifically, the value h_i decoded from E_i for some point \mathbf{q} (lying in one of the receiver's balls) would constitute a tuple $(g, h, \mathbf{H}(q_i), h_i)$. However, this tuple does not necessarily need to be a DDH tuple. We only need the final product

$$\left(g, h, \prod_{i=1}^d \mathbf{H}(q_i), \prod_{i=1}^d h_i \right)$$

to be a valid DDH tuple for correctness.

Suppose there exists *at least one* dimension on which the projections of *all* balls are disjoint. This gives each ball a unique way to identify it from others. Our idea is to leverage OKVS instances recursively: For each dimension $i \in [d]$, the receiver encodes an *outer* OKVS as

$$E_i \leftarrow \text{Encode} \left(\left\{ (w_{k,i} + j, \quad \text{val}_{k,i,j}) \right\}_{k \in [N], j \in [-\delta, +\delta]} \right),$$

where $\text{val}_{k,i,j}$ differs in two cases:

- If the current dimension i is the globally separated dimension for all balls, then $\text{val}_{k,i,j}$ is an *inner* OKVS instance for fuzzy matching with \mathbf{w}_k , namely,

$$\text{val}_{k,i,j} \leftarrow \text{Encode} \left(\left\{ (i' \parallel w_{k,i'} + j', \quad h_{i',j'} \parallel h_{i',j'}^s) \right\}_{i' \in [d], j' \in [-\delta, +\delta]} \right),$$

where $h_{i',j'} \leftarrow_{\$} \mathbb{G}$;

- Otherwise, the $\text{val}_{k,i,j} := (\mathbf{r} \parallel \mathbf{r}^s)$ is a *dummy* instance where $\mathbf{r} \leftarrow_{\$} \mathbb{G}^m$ and m is the size of the inner OKVS instance.

For each point \mathbf{q} , the sender first decodes the outer OKVS to obtain a list $\{\text{val}_1, \dots, \text{val}_d\}$, then runs the decoding function on each $\text{val}_j \in [d]$ to get

$$(u_j \parallel v_j) := \prod_{i=1}^d \text{Decode}(\text{val}_j, q_i).$$

In the end, the sender re-randomizes the result from the tuple $(g, h, \prod_{j=1}^d u_j, \prod_{j=1}^d v_j)$.

For correctness, we expect that decoding a dummy instance on any key would output a valid DDH pair all the time. This can be guaranteed if the inner OKVS has a linear decoding function. Clearly, in this way, each \mathbf{q} would get either an inner OKVS instance or random garbage from *the globally separated* dimension. The latter results in valid DDH tuples with negligible probability, so we focus on the case that the sender gets an inner OKVS instance in the end. This reduces the fuzzy PSI problem to the fuzzy matching problem as other dummy instances won't affect the correctness. For security, the inner OKVS has to be doubly oblivious, namely, the encoded structure itself is uniformly random. Regarding the complexity, the receiver's communication and computation costs would be $O(d\delta)$ times larger.

Further weaken the assumption. The above assumption is weaker and milder than what was used in prior works, but it is still somewhat artificial. Here we show that we can even weaken this assumption to the following: For each ball, there *exists* at least one dimension on which *its* projection is separated from others. Note that the above approach doesn't work yet in this setting: There might exist a point whose projection on *each* dimension is inside the projection of a *non-separated* interval from some ball. In other words, the sender would get a list of dummy instances after decoding the outer OKVS. This results in a false positive since dummy instances always output a match. To rule out these false positives, we realize that we could encode additional information into each $\text{val}_{k,i,j}$.

For simplicity, let's assume the decoding function of the OKVS is determined by a binary vector with some *fixed hamming weight*, that is, given an instance $\mathbf{r} \in \mathbb{G}^m$ and some key, the decoding function outputs

$$\text{Decode}(\mathbf{r}, \text{key}) = \langle \mathbf{d}, \mathbf{r} \rangle = \prod_{i=1}^m r_i^{d_i},$$

where $\mathbf{d} \in \{0, 1\}^m$ is deterministically sampled by the key, and $\text{HammingWeight}(\mathbf{d}) = t$. The receiver samples two random shares $\zeta_{\perp}, \zeta_{\top}$ such that $\zeta_{\perp} \cdot \zeta_{\top} = 1$. We denote as I_k the *first* dimension on which \mathbf{w}_k projects a separated interval. Then the receiver could set $\text{val}_{k,i,j}$ for each \mathbf{w}_k in this way:

- If the current dimension $i = I_k$, then $\text{val}_{k,i,j}$ is an inner OKVS instance defined by

$$\text{val}_{k,i,j} \leftarrow \text{Encode} \left(\left\{ \left(i' \parallel w_{k,i'} + j', \quad h_{i',j'} \parallel h_{i',j'}^s \cdot \zeta_{\perp}^{t \cdot (d-1)} \right) \right\}_{i' \in [d], j' \in [-\delta, +\delta]} \right),$$

where $h_{i',j'} \leftarrow_{\$} \mathbb{G}$ and t is the hamming weight of \mathbf{d} ;

- Otherwise, the $\text{val}_{k,i,j} := (\mathbf{r} \parallel \mathbf{r}^s \cdot \zeta_{\top})$ for $\mathbf{r} \leftarrow_{\$} \mathbb{G}^m$.

The security follows as before, whereas the correctness is non-trivial. First, consider the sender's point \mathbf{q} intersecting some receiver's ball. After decoding the inner OKVS instance, the sender gets a pair $(u_* \parallel u_*^s \cdot \zeta_{\perp}^{td \cdot (d-1)})$ for some u_* ; After decoding a dummy instance, the sender gets $(r_* \parallel r_*^s \cdot \zeta_{\top}^{td})$ for some r_* instead. Now, by multiplying them together, the final tuple

$$(g, h, v \parallel v^s \cdot \zeta_{\perp}^{td \cdot (d-1)} \cdot \zeta_{\top}^{td \cdot (d-1)}) = (g, h, v \parallel v^s)$$

is a valid DDH tuple for some $v \in \mathbb{G}$.

Then consider the case that the sender's point \mathbf{q} is outside of all balls. The only way to report a match is to get a list of all dummy instances after decoding the outer OKVS instance, otherwise the inner OKVS instance will output a random garbage result. However, since dummy instances

only encode ζ_\top , the product of them equals 1 with negligible probability due to ζ_\top being randomly sampled and $td^2 \ll p$ if $t = O(\kappa)$.

Recall that we assume the decoding vector \mathbf{d} to have fixed hamming weight. This is not ideal since most modern OKVS instantiations (e.g., [GPR⁺21,RR22,BPSY23]) don't satisfy this requirement, whereas the only exception is the garbled bloom filters [DCW13] whose efficiency is not satisfactory. We managed to get rid of this assumption in our real protocol in the end, please refer to Section 7.1 for details.

Locality-sensitive hashing. The above approaches are heavily tailored to the L_∞ distance⁸ and require proper distribution of the receiver's input. To support L_p distance in high dimensions, we utilize locality-sensitive hashing (LSH) to identify matching balls. An LSH family with parameters $(\delta, c\delta, p_1, p_2)$ guarantees the following:

- If two points \mathbf{w} and \mathbf{q} are close enough, i.e., $\text{dist}_p(\mathbf{w}, \mathbf{q}) \leq \delta$, they would be hashed into the same bucket with at least p_1 probability;
- If they are far apart, i.e., $\text{dist}_p(\mathbf{w}, \mathbf{q}) > c\delta$, then the probability of hashing them into the same bucket is at most p_2 .

In other words, an LSH family bounds the false-positive and false-negative probability to p_2 and $1 - p_1$, respectively. Usually, false-positive and false-negative cannot be reduced to negligible simultaneously. However, given the existence of our fuzzy matching protocols, we can tolerate false positives by running fuzzy matching on each positive match. Therefore, the high-level strategy is that the receiver hashes each ball center via LSH to some LSH entry, and the sender would identify multiple positive LSH entries for each of its points. If we set the parameters properly, the total number of false positives for each sender's point can be upper-bounded by $O(N^\rho)$ for some $\rho < 1$ which gives us just a *sub-quadratic* blowup in total communication and computation complexities.

One caveat is that there is a constant gap between the calculation of false positives and false negatives mentioned above, namely, false positives are calculated when points are $c\delta$ -apart, whereas false negatives are calculated when points are δ -close. Fortunately, when the receiver's balls are disjoint (i.e., centers are 2δ -part), this gap can be filled by setting $c = 2$. Another caveat is that this approach does not support fuzzy PSI cardinality anymore due to the rationale behind the LSH: To guarantee a negligible false-negative rate, we typically have to prepare multiple LSH tables where a true positive might appear more than once.

3 Preliminaries

We represent the computational security parameter as $\lambda \in \mathbb{N}$, the statistical security parameter as $\kappa \in \mathbb{N}$, and the output of the algorithm \mathcal{A} on input in using $r \leftarrow \{0, 1\}$ as its randomness by $x \leftarrow \mathcal{A}(\text{in}; r)$. Randomness is often omitted and is only explicitly mentioned when necessary. Efficient algorithms are considered to be *probabilistic polynomial time* (PPT) machines. We use \approx_c to denote computational indistinguishability and \approx_s to denote statistical indistinguishability of probability distributions. The notation $[n]$ signifies a set $\{1, \dots, n\}$ and $[a : b]$ the set $\{a, a + 1, \dots, b - 1, b\}$. We use $\mathbf{c}[i : j]$ to represent a vector with a defined length of $[c_i, \dots, c_j]$ and \mathbf{c} to indicate a vector of c .

All protocols presented in this work are two-party protocols. Security is proven against semi-honest adversaries via the standard simulation-based paradigm (see, e.g., [Lin16]).

⁸ It can also be generalized to the L_p setting as briefly explained in Remark 4.

3.1 Oblivious Key-Value Store (OKVS)

The concept of an oblivious key-value store (OKVS) was introduced by Garimella et al. [GPR+21] to capture the properties of data structures commonly used in PSI protocols. Subsequent work proposed OKVS constructions offering favorable trade-offs between encoding/decoding time and encoding size [RR22,BPSY23].

Definition 1 (Oblivious Key-Value Store). An oblivious key-value store OKVS is parameterized by a key space \mathcal{K} , a value space \mathcal{V} , computational and statistical security parameters λ, κ , respectively, and consists of two algorithms:

- **Encode** : takes as input a set of key-value pairs $L \in (\mathcal{K} \times \mathcal{V})^n$ and randomness $\theta \in \{0, 1\}^\lambda$, and outputs a vector $\mathbf{r} \in \mathcal{V}^m$ or a failure indicator \perp .
- **Decode** : takes as input a vector $\mathbf{r} \in \mathcal{V}^m$, a key $k \in \mathcal{K}$ and randomness $\theta \in \{0, 1\}^\lambda$, and outputs a value $v \in \mathcal{V}$.

That satisfies:

- **Correctness**: For all $L \in (\mathcal{K} \times \mathcal{V})^n$ with distinct keys and $\theta \in \{0, 1\}^\lambda$ for which $\text{Encode}(L; \theta) = \mathbf{r} \neq \perp$, it holds that $\forall (k, v) \in L: \text{Decode}(\mathbf{r}, k; \theta) = v$.
- **Low failure probability**: For all $L \in (\mathcal{K} \times \mathcal{V})^n$ with distinct keys:

$$\Pr_{\theta \leftarrow \mathcal{S}\{0,1\}^\lambda} [\text{Encode}(L; \theta) = \perp] \leq 2^{-\kappa}.$$

- **Obliviousness**: For any $\{k_1, \dots, k_n\}, \{k'_1, \dots, k'_n\} \subseteq \mathcal{K}$ of n distinct keys and any $\theta \in \{0, 1\}^\lambda$, if Encode does not output \perp , then for $v_1, \dots, v_n \leftarrow \mathcal{S}\mathcal{V}$:

$$\{\mathbf{r} \leftarrow \text{Encode}(\{(k_i, v_i)_{i \in [n]}\}; \theta)\} \approx_c \{\mathbf{r}' \leftarrow \text{Encode}(\{(k'_i, v_i)_{i \in [n]}\}; \theta)\}.$$

- **Double obliviousness**: For all sets of n distinct keys $\{k_1, \dots, k_n\} \subseteq \mathcal{K}$ and n values $\{v_1, \dots, v_n\} \leftarrow \mathcal{S}\mathcal{V}$, there is $\text{Encode}(\{(k_i, v_i)_{i \in [n]}\}; \theta)$ statistically indistinguishable from the uniformly random element from \mathcal{V}^m .

The efficiency of OKVS is characterized by: (1) the time it takes to encode n key-value pairs; (2) the time it takes to decode a single key; (3) the ratio n/m between the number of key-value pairs n and the encoding size m , also called *rate*. Recent OKVS constructions [GPR+21,RR22,BPSY23] achieve: (1) encoding time $O(n\kappa)$; (2) decoding time $O(\kappa)$; (3) constant rate.

For this work, we will need OKVS to support the value space \mathcal{V} being equal to a cyclic group \mathbb{G} of prime order p . A sufficient condition for this, which is satisfied by the efficient constructions of [GPR+21,RR22,BPSY23] is:

- **\mathbb{F}_p -Linear**: There exists a function $\text{dec} : \mathcal{K} \times \{0, 1\}^\lambda \rightarrow \mathbb{F}_p^m$ such that for all $\mathbf{r} \in \mathbb{G}^m$, $k \in \mathcal{K}$ and $\theta \in \{0, 1\}^\lambda$ it holds that $\text{Decode}(\mathbf{r}, k; \theta) := \langle \text{dec}(k; \theta), \mathbf{r} \rangle$, where $\text{dec} : \mathcal{K} \times \{0, 1\}^\lambda \mapsto \mathbb{F}_p^m$. For $\mathbf{d} \in \mathbb{F}_p^m$ and $\mathbf{g} \in \mathbb{G}^m$ we define $\langle \mathbf{d}, \mathbf{g} \rangle := g_1^{d_1} \cdots g_m^{d_m}$.

When OKVS is \mathbb{F}_p -linear, the encoding procedure for a set of key-value pairs $\{(k_1, v_1), \dots, (k_n, v_n)\} \in (\mathcal{K} \times \mathbb{G})^n$ basically consists of finding a solution $\mathbf{r} \in \mathbb{G}^m$ to a system of equations $\mathbf{M} \cdot \mathbf{r} = \mathbf{v}$, where $\mathbf{M} \in \mathbb{F}_p^{n \times m}$ with rows $\mathbf{m}_i := \text{dec}(k_i; \theta)$. Hence, in this case an encoding can be computed through linear algebra over \mathbb{F}_p followed by evaluating the action of \mathbb{F}_p^n on \mathbb{G}^m as defined above. In our instantiation, we choose the function $\text{dec} : \mathcal{K} \times \{0, 1\}^\lambda \mapsto \{0, 1\}^m \in \mathbb{F}_p^m$ which will not affect correctness or security but will make the decoding process more efficient.

We require an additional property for OKVS which says decoding a non-encoded key will yield a uniformly random result. It is only about correctness rather than security and is proved in [GRS22] for binary OKVS (and naturally extends to \mathbb{F}_p -linear case) as the *independence* property⁹. We sketch the proof for completeness. One caveat is that the proof below (as well as in [GRS22]) requires the rate of OKVS to be smaller than 1, and thus the original polynomial-based OKVS does not satisfy this property, unless we assume the encoded values to be uniformly random.

Lemma 1 (Independence). *If OKVS satisfies \mathbb{F}_p -linearity, double obliviousness and $\text{negl}(\kappa)$ failure probability, and θ is uniformly randomly chosen while $n < m$ holds, then for any $L := \{(k_i, v_i)_{i \in [n]}\}$ with distinct keys, and any key $k \notin \{k_i\}_{i \in [n]}$, it holds that $\text{Decode}(\text{Encode}(L; \theta), k)$ is indistinguishable from random, where $\text{Encode} : (\mathcal{K} \times \mathcal{V})^n \rightarrow \mathcal{V}^m$.*

Proof. Following the proof of Lemma 30 in [GRS22], \mathbb{F}_p -linearity means that the output \mathbf{r} of $\text{Encode}(L; \theta)$ is a vector satisfying the linear system $\mathbf{M} \cdot \mathbf{r} = \mathbf{v}$ where $\mathbf{m}_i := \text{dec}(k_i; \theta)$ and $(k_i, v_i) \in L$. If we require the $\text{negl}(\kappa)$ failure probability of the encoding algorithm, then the matrix \mathbf{M} has the full rank with overwhelming probability. Moreover, when $n < m$, and θ is randomly chosen, \mathbf{r} is a random solution in the subspace by the property of double obliviousness. With overwhelming probability, any $k_* \notin L$ results in a linear independent vector $\mathbf{m}_* := \text{dec}(k_*; \theta)$, which makes $\langle \mathbf{m}_*, \mathbf{r} \rangle$ uniformly random distributed and independent of \mathbf{v} . \square

3.2 Random Self-Reductions of DDH Tuples

The well-known decisional Diffie-Hellman (DDH) assumption for a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p states that the distribution of Diffie-Hellman (DH) tuples $(g, h := g^s, h_1, h_2 := h_1^s)$, where $s \leftarrow \mathbb{Z}_p$, $h_1 \leftarrow \mathbb{G}$, is computationally indistinguishable from the distribution of random tuples $(g, h := g^s, h_1, h_2)$, where $s \leftarrow \mathbb{Z}_p$, $h_1, h_2 \leftarrow \mathbb{G}$. Naor and Reingold [NR97] show that deciding whether an arbitrary tuple (g, h, h_1, h_2) with $h, h_1, h_2 \in \mathbb{G}$ is a DH tuple can be reduced to breaking the DDH assumption. For this work, we consider a special case of this reduction where $h := g^s$ is fixed.

Lemma 2 (Random Self-Reduction [NR97]). *Let $\mathbb{G} := \langle g \rangle$ be a cyclic group of order p , let $h := g^s$ for $s \in \mathbb{Z}_p$ and $h_1, h_2 \in \mathbb{G}$. If $h'_1 := g^a \cdot h_1^b$ and $h'_2 := h^a \cdot h_2^b$, where $a, b \leftarrow \mathbb{Z}_p$, then:*

- h'_1 is uniformly random in \mathbb{G} and $h'_2 = (h'_1)^s$ if $h_2 = h_1^s$.
- (h'_1, h'_2) is a uniformly random pair of group elements otherwise.

3.3 Locality-Sensitive Hashing

We recall the definition of locality-sensitive hashing (LSH) [IM98] and adapt it according to our constructions.

Definition 2 (LSH). *A hash family $\mathcal{F}_{\mathcal{D}}: \{\mathbb{F}_{\mathcal{D}} | \mathbb{Z}^d \mapsto \{0, 1\}^*\}$ is a $(\delta, c\delta, p_1, p_2)$ -LSH family for some distance metric \mathcal{D} if it maps points in \mathbb{Z}^d into string labels, such that the following two conditions hold for any two points $\mathbf{w}, \mathbf{q} \in \mathbb{Z}^d$:*

- $\text{dist}_{\mathcal{D}}(\mathbf{w}, \mathbf{q}) \leq \delta \implies \Pr[\mathbb{F}_{\mathcal{D}}(\mathbf{w}) = \mathbb{F}_{\mathcal{D}}(\mathbf{q})] \geq p_1$, and
- $\text{dist}_{\mathcal{D}}(\mathbf{w}, \mathbf{q}) > c\delta \implies \Pr[\mathbb{F}_{\mathcal{D}}(\mathbf{w}) = \mathbb{F}_{\mathcal{D}}(\mathbf{q})] \leq p_2$,

where $p_1 > p_2$ and the probabilities are over random choices of $\mathbb{F}_{\mathcal{D}} \in \mathcal{F}_{\mathcal{D}}$.

⁹ It is also mentioned and proved in [BPSY23] as *random decoding*.

Theorem 1 (Locality-Sensitive Hash Table [IM98]). *Given a set of $c\delta$ -apart points $\mathbf{W} \in \mathbb{Z}^{d \times N}$ in d dimensional space, and a $(\delta, c\delta, \frac{1}{N^\rho}, \frac{1}{N})$ -LSH family $\mathcal{F}_{\mathcal{D}}$ as in Definition 2 where $\rho < 1$, there exists an efficient algorithm that hashes the set into $L = \frac{\kappa}{\log e} N^\rho$ hash tables with statistical parameter κ , such that:*

- for any point \mathbf{q} for which $\text{dist}_{\mathcal{D}}(\mathbf{w}, \mathbf{q}) \leq \delta$ holds for some $\mathbf{w} \in \mathbf{W}$, there exists $i \in [L]$ such that \mathbf{q} and \mathbf{w} are hashed to the same bucket in this table with probability at least $1 - \text{negl}(\kappa)$;
- space complexity for L tables is $O(N\lambda \cdot L)$ and encoding complexity $O(NdL)$;
- query complexity is $O(Ld)$ per lookup point.

Proof. It is folklore that if we choose $L = \kappa' N^\rho$ where $\kappa' = \kappa / \log e$, then for two points \mathbf{q}, \mathbf{w} that $\text{dist}(\mathbf{q}, \mathbf{w}) \leq \delta$, there is

$$\Pr[\exists i \in [L], F_{\mathcal{D},i}(\mathbf{q}) = F_{\mathcal{D},i}(\mathbf{w})] \geq 1 - (1 - \frac{1}{N^\rho})^L \quad (1)$$

$$\geq 1 - e^{-LN^{-\rho}} \quad (2)$$

$$= 1 - e^{-\kappa'} = 1 - \text{negl}(\kappa), \quad (3)$$

where κ is the statistical parameter and (2) is followed by the Bernoulli's inequality. \square

Note that we only guarantee that the false-negative rate is negligible but not the false-positive rate, which will be taken care of in our concrete constructions. Moreover, we provide an efficient instantiation of LSH family $\mathcal{F}_{\mathcal{D}}$ for $\mathcal{D} = L_2$ and $\mathcal{D} = L_1$ distance from existing work as follows.

Theorem 2 (L_2 [AI06]). *There exists a $(\delta, c\delta, \frac{1}{N^\rho}, \frac{1}{N})$ -LSH family \mathcal{F}_{L_2} for L_2 distance with $\rho \leq 0.365$ when $c = 2$ and $\rho \leq 0.168$ when $c = 4$.*

Theorem 3 (L_1 [DIIM04]). *There exists a $(\delta, c\delta, \frac{1}{N^\rho}, \frac{1}{N})$ -LSH family \mathcal{F}_{L_1} for L_1 distance with $\rho \leq 0.5$ when $c = 2$ and $\rho \leq 0.25$ when $c = 4$.*

For theoretical interests, it is also possible to get near-linear space complexity (instead of super-linear $N^{1+\rho}$ as above).

Theorem 4 (Near-linear Space [AI06]). *There exists an efficient algorithm that can generically transform the above hash tables into one with space complexity $\tilde{O}(N)$ and query time $O(N^{O(\rho)})$.*

3.4 Oblivious Programmable PRF

An oblivious programmable PRF (OPPRF) [KMP⁺17] is a two-party primitive that allows the sender to program a PRF such that it maps certain inputs to certain outputs and to let the receiver obviously evaluate this programmed function on a number of points. It can be realized by using an OKVS to encode “corrections” for an oblivious PRF (OPRF) [KMP⁺17,RS21]. A basic ideal functionality is given in Figure 1.

3.5 Partially Homomorphic Encryption

As mentioned in Section 1.1, a main ingredient in our fuzzy matching protocols is a partially homomorphic public key encryption scheme. In this section we introduce the necessary definitions and conditions this scheme needs to satisfy, but in the main body we explicitly instantiate it by ElGamal's cryptosystem (see Example 1).

$\mathcal{F}_{\text{OPPRF}}$
Parameters : cardinality of sets n, m , input space \mathcal{X} , output space \mathcal{Y} . Functionality : <ul style="list-style-type: none"> - RECEIVER inputs $X \in \mathcal{X}^n$. - SENDER inputs $L := \{(y_1, z_1), \dots, (y_m, z_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$. - Sample $F \leftarrow \{f : \mathcal{X} \rightarrow \mathcal{Y} \mid \forall j \in [m] : f(y_j) = z_j\}$ - Output $\{(x, F(x)) \mid x \in X\}$ to RECEIVER and \mathcal{O}^F to SENDER.

Fig. 1. Ideal functionality of oblivious programmable PRF (OPPRF).

A partially homomorphic public key encryption scheme is an encryption scheme for which it is possible to compute an encryption of the sum of two messages by just performing operations on encryptions of these messages. It is parametrized by a message space family $\mathcal{M} := (\mathcal{M}_\lambda)_{\lambda \in \mathbb{N}}$, where each \mathcal{M}_λ is a finite abelian group (written additively), a ciphertext space family $\mathcal{C} := (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ and is given by a tuple of PPT algorithms $\text{PKE} := (\text{Gen}, \text{Enc}, \text{Dec}, \boxplus)$, where:

- $\text{Gen}(1^\lambda) \rightarrow (\text{pk}, \text{sk})$: takes as input the security parameter λ and outputs a key pair (pk, sk) .
- $\text{Enc}_{\text{pk}}(m) \rightarrow c$: takes as input the public key pk and a message $m \in \mathcal{M}_\lambda$, and outputs a ciphertext $c \in \mathcal{C}_\lambda$.
- $\text{Dec}_{\text{sk}}(c) \rightarrow m$: takes as input the private key sk and a ciphertext $c \in \mathcal{C}_\lambda$, and outputs a message $m \in \mathcal{M}_\lambda$. For all $m \in \mathcal{M}_\lambda$ and $c \leftarrow \text{Enc}_{\text{pk}}(m)$, we require that $\text{Dec}_{\text{sk}}(c) = m$.
- $c' \boxplus_{\text{pk}} c'' \rightarrow c$: takes as input the public key pk and two ciphertexts $c', c'' \in \mathcal{C}_\lambda$ and outputs a ciphertext $c \in \mathcal{C}_\lambda$. For all $m', m'' \in \mathcal{M}_\lambda$, $c' \leftarrow \text{Enc}_{\text{pk}}(m')$, $c'' \leftarrow \text{Enc}_{\text{pk}}(m'')$, $c \leftarrow c' \boxplus_{\text{pk}} c''$, we require that $\text{Dec}_{\text{sk}}(c) = m' + m''$.

We require PKE to satisfy IND- \mathcal{S} CPA security [CLOS02]. That is, for any $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$ and $m \in \mathcal{M}_\lambda$, the distributions $\{\text{pk}, c \leftarrow \text{Enc}_{\text{pk}}(m)\}$ and $\{\text{pk}, c \leftarrow \mathcal{C}_\lambda\}$ are computationally indistinguishable. For example, ElGamal's [ELG85] cryptosystem is partially homomorphic and satisfies the above security notion under the decisional Diffie-Hellman (DDH) assumption.

Example 1 (ElGamal). ElGamal's cryptosystem [ELG85] is defined over a prime-order cyclic group family $\mathcal{G} := (\mathcal{G}_\lambda)_{\lambda \in \mathbb{N}}$ and consists of the following algorithms:

- $\text{Gen}(1^\lambda) \rightarrow (\text{pk}, \text{sk})$: samples $\mathbb{G} \leftarrow \mathcal{G}_\lambda$ of prime order p with generator g , samples $s \leftarrow \mathbb{Z}_p$ and puts $h := g^s$. Let $\text{pk} := (\mathbb{G}, p, g, h)$ and $\text{sk} := s$.
- $\text{Enc}_{\text{pk}}(m) \rightarrow (c_1, c_2)$: samples $r \leftarrow \mathbb{Z}_p$ and puts $(c_1, c_2) := (g^r, m \cdot h^r)$.
- $\text{Dec}_{\text{sk}}(c_1, c_2) \rightarrow m$: puts $m := c_1^{-s} \cdot c_2$.
- $(c'_1, c'_2) \boxplus_{\text{pk}} (c''_1, c''_2) \rightarrow (c_1, c_2)$: puts $(c_1, c_2) := (c'_1 \cdot c''_1, c'_2 \cdot c''_2)$.

It is clear that the above gives a partially homomorphic public key encryption scheme with respect to the group operation of \mathbb{G} . It is moreover known to be IND- \mathcal{S} CPA secure under the decisional Diffie-Hellman (DDH) assumption. For some applications it might be desirable to obtain an additive homomorphism with respect to \mathbb{Z}_p , in which case one can encrypt a message $m := g^x$ using the above algorithm. In this case there does however not exist an efficient decryption algorithm, since decrypting comes down to solving the discrete logarithm problem over \mathbb{G} , which we assume to be hard. For applications where one only needs to check whether a message belongs to some relatively small set of valid messages this does not pose a problem, which is the case for our fuzzy matching protocols in Section 5.1 and Section 5.2

$\mathcal{F}_{\text{FUZZYMATCH}}$
<hr/> Parameters : dimension d , radius δ , and a distance function $\text{dist}(\cdot, \cdot)$.
Functionality :
<ul style="list-style-type: none"> - RECEIVER inputs $\mathbf{w} \in \mathbb{Z}^d$. - SENDER inputs $\mathbf{q} \in \mathbb{Z}^d$. - Output 1 to RECEIVER if $\text{dist}(\mathbf{w}, \mathbf{q}) \leq \delta$, and 0 otherwise.

Possible Distance Functions
<hr/> $\text{dist}(\mathbf{w}, \mathbf{q})$ is defined as:
<ul style="list-style-type: none"> - L_∞ Distance: $\text{dist}(\mathbf{w}, \mathbf{q}) = \max_{i \in [d]} w_i - q_i$ - Hamming Distance: $\text{dist}(\mathbf{w}, \mathbf{q}) = \sum_{i=1}^d (w_i \neq q_i)$ - Conjunction of Hamming Distance and L_∞ on δ_∞: $\text{dist}(\mathbf{w}, \mathbf{q}) = \sum_{i=1}^d (w_i - q_i > \delta_\infty)$ - L_p Distance: $\text{dist}(\mathbf{w}, \mathbf{q}) = \left(\sum_{i=1}^d w_i - q_i ^p \right)^{1/p}$

Fig. 2. Ideal Functionality of Fuzzy Matching

Furthermore note that the random self-reduction of DH tuples Lemma 2 essentially says that if one rerandomizes an ElGamal ciphertext $c := (h_1, h_2)$ as $c' := c^b \cdot c_1$, where $c_1 := (g^a, h^a \cdot 1) \leftarrow \text{Enc}_{\text{pk}}(1)$, then the resulting ciphertext c' is a fresh random encryption of 1 if c is an encryption of 1, and an encryption of a uniformly random element otherwise. For the generalized version of our protocols using a partially homomorphic encryption scheme we will need the analogous condition.

4 Definitions and Functionalities

We define the two-message protocol as below, consisting of three algorithms:

- $\text{Receiver}_1(\text{INPUT}_R)$: The algorithm takes the RECEIVER's INPUT_R , outputs the first message msg_1 and its secret state st ;
- $\text{Sender}_1(\text{INPUT}_S, \text{msg}_1)$: The algorithm takes the SENDER's INPUT_S and msg_1 , outputs the second message msg_2 ;
- $\text{Receiver}_2(\text{st}, \text{msg}_2)$: The algorithm takes the state st and the second message msg_2 , outputs the final OUTPUT .

4.1 Definition of Fuzzy Matching

We define the functionality of fuzzy matching between two points in in Figure 2, with different distance functions including both infinity (L_∞) and Minkowski (L_p) distance where $p \in [1, \infty)$.

4.2 Definition of Fuzzy (Circuit) Private Set Intersection

We define the functionality of fuzzy PSI and fuzzy circuit PSI in Figure 3 and Figure 4, respectively. Note that for standard fuzzy PSI, we also consider a slightly stronger functionality (compared to

$\mathcal{F}_{\text{FUZZYPSI}}$
Parameters : dimension d , radius δ , cardinality of sets N, M , a distance function $\text{dist}(\cdot, \cdot)$, a leakage function $\text{leakage}(\cdot, \cdot)$, label length σ , and a concise description for receiver's and sender's points $\mathcal{D}_R, \mathcal{D}_S$, respectively.
Functionality :
<ul style="list-style-type: none"> - RECEIVER inputs $\mathbf{W} \in \mathbb{Z}^{d \times N}$ according to \mathcal{D}_R. - SENDER inputs $\mathbf{Q} \in \mathbb{Z}^{d \times M}$ according to \mathcal{D}_S. For Labeled PSI, SENDER inputs $\text{Label}_{\mathbf{Q}} \in \{0, 1\}^{\sigma \times M}$. - Return $\text{leakage}(\mathbf{W}, \mathbf{Q})$ to RECEIVER.

Possible Leakage Functions
$\text{leakage}(\mathbf{W}, \mathbf{Q})$ is defined as:
<ul style="list-style-type: none"> - PSI-CA: $\text{leakage}(\mathbf{W}, \mathbf{Q}) = \sum_{i \in [N], j \in [M]} (\text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta)$. - PSI: $\text{leakage}(\mathbf{W}, \mathbf{Q}) = \{\mathbf{q}_j \mid \exists i \in [N], \text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta\}$. - PSI-SP: $\text{leakage}(\mathbf{W}, \mathbf{Q}) = \{\mathbf{w}_i \mid \exists j \in [M], \text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta\}$. - Labeled PSI: $\text{leakage}(\mathbf{W}, \mathbf{Q}) = \{\text{label}_j \mid \exists i \in [N], \text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta\}$, where label_j is the label associated with \mathbf{q}_j.

Fig. 3. Ideal Functionality of Fuzzy PSI

prior works) where the receiver only learns whether their points are in the intersection, but not the sender's exact points, which we call PSI with sender privacy (PSI-SP). We extend the functionality of fuzzy PSI to many closely related variants including PSI cardinality (PSI-CA), labeled PSI, and circuit PSI.

5 Fuzzy Matching

We start by presenting a fuzzy matching protocol for two points in hyperspace with infinity distance (L_∞) and hamming distance, then we extend it into a more general setting with Minkowski distance ($L_{p \in [1, \infty)}$).

$\mathcal{F}_{\text{FUZZYCPSI}}$
Parameters : dimension d , radius δ , cardinality of sets N, M , a distance function $\text{dist}(\cdot, \cdot)$, associated data length σ , and a concise description for receiver's and sender's points $\mathcal{D}_R, \mathcal{D}_S$, respectively.
Functionality :
<ul style="list-style-type: none"> - RECEIVER inputs $\mathbf{W} \in \mathbb{Z}^{d \times N}$ according to \mathcal{D}_R and associated data $\tilde{\mathbf{W}} \in \{0, 1\}^{\sigma \times N}$. - SENDER inputs $\mathbf{Q} \in \mathbb{Z}^{d \times M}$ according to \mathcal{D}_S and associated data $\tilde{\mathbf{Q}} \in \{0, 1\}^{\sigma \times M}$. - For each $j \in [M]$, sample $r_j, s_j \leftarrow \{0, 1\}^{1+2\sigma}$ such that: <ul style="list-style-type: none"> $r_j \oplus s_j = 1 \ \tilde{\mathbf{w}}_i\ \tilde{\mathbf{q}}_j$ if $\exists i \in [N]$ s.t. $\text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta$, and $r_j \oplus s_j = 0^{1+2\sigma}$ otherwise. - Return $(r_j)_{j \in [M]}$ to RECEIVER and $(s_j)_{j \in [M]}$ to SENDER.

Fig. 4. Ideal Functionality of Fuzzy Circuit PSI

$\text{GetList}_\infty(h, s, \mathbf{w}, \Delta_w)$	$\text{GetTuple}_\infty(g, h, \mathbf{q}, \Delta_q, \mathbf{E})$
For each $i = 1 \dots d$:	For each $i = 1 \dots d$:
For each $j = -\delta \dots \delta$:	Set $\text{key}_i \leftarrow \text{H}_\gamma(\Delta_q \ q_i)$
Set $\text{key}_j \leftarrow \text{H}_\gamma(\Delta_w \ w_i + j)$	$(u_i \ v_i) \leftarrow \text{Decode}(E_i, \text{key}_i)$
Set $h_j \leftarrow \mathbb{G}$	Sample $a, b \leftarrow \mathbb{Z}_p$
Set $\text{val}_j = (h_j \ h_j^s)$	Set $u_* = g^a \cdot \left(\prod_{i=1}^d u_i\right)^b$
Set $\text{list}_i = \{(\text{key}_j, \text{val}_j)_{j \in [-\delta:\delta]}\}$	Set $v' = h^a \cdot \left(\prod_{i=1}^d v_i\right)^b$
Return $\text{list}_1, \dots, \text{list}_d$	Set $v_* \leftarrow \text{H}_{\kappa'}(v')$
	Return (u_*, v_*)

$\text{Receiver}_1(\mathbf{w} \in \mathbb{Z}^d)$	$\text{Sender}_1(\mathbf{q} \in \mathbb{Z}^d, \text{msg}_1)$
Sample $g \leftarrow \mathbb{G}, s \leftarrow \mathbb{Z}_p$	Parse $\text{msg}_1 := (g, h, \mathbf{E})$
Compute $h = g^s$	$(u_*, v_*) \leftarrow \text{GetTuple}_\infty(g, h, \mathbf{q}, 0^\kappa, \mathbf{E})$
Get $\{\text{list}_i\}_{i \in [d]} \leftarrow \text{GetList}_\infty(h, s, \mathbf{w}, 0^\kappa)$	Output $\text{msg}_2 := (u_*, v_*)$
Set $E_i \leftarrow \text{Encode}(\text{list}_i)$ for each $i \in [d]$	$\text{Receiver}_2(\text{st}, \text{msg}_2)$
Set $\mathbf{E} = \{E_1, \dots, E_d\}$	Parse $\text{msg}_2 := (u_*, v_*)$ and $\text{st} := s$
Output $\text{msg}_1 := (g, h, \mathbf{E}), \text{st} := s$	Output 1 if $\text{H}_{\kappa'}(u_*^s) = v_*$ and 0 otherwise

Fig. 5. Fuzzy Matching for L_∞ Distance

5.1 Fuzzy Matching for Infinity Distance

We provide the protocol for infinity distance in Figure 5. Intuitively, the receiver encodes its point as a list of DDH values and then encodes this list into an oblivious key-value store. This guarantees that the sender will decode a correct DDH value if its i -th dimension lies in the range, and will decode a random group element otherwise. Finally, according to random self-reduction of DDH tuples, the sender will send a uniformly random pair if its point is δ apart from the receiver's point and will send a randomized DDH tuple otherwise.

Theorem 5 (Correctness). *The protocol provided in Figure 5 is correct with $1 - \text{negl}(\kappa)$ probability if OKVS satisfies perfect correctness defined in Section 3.1 and independence property from Lemma 1, and $\text{H}_\gamma : \{0, 1\}^* \mapsto \{0, 1\}^\gamma, \text{H}_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ are universal hash functions where $\gamma = \kappa + \log \delta$ and $\kappa' = \kappa$.*

Proof. It is clear that the correctness holds in the case that $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) \leq \delta$: For each $i \in [d]$, there is $q_i \in [w_i - \delta, w_i + \delta]$, then according to the perfect correctness of OKVS, we have $u_i^s = v_i$ at the sender's side which implies $u_*^s = v'$, $\text{H}_{\kappa'}(u_*^s) = v_*$, and $1 \leftarrow \text{Receiver}_2$. Now we turn to the case that $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) > \delta$: Since the OKVS satisfies independence property, there exists at least one $j \in [d]$ such that $(u_j \| v_j)$ is uniformly random over $\mathbb{G} \times \mathbb{G}$, which implies $u_*^s = v'$ with only $\text{negl}(\kappa)$ probability. Moreover, since $\text{H}_{\kappa'}$ and H_γ are universal hash functions, the probability that

any different inputs resulting the same hash value is negligible. Taking a union bound over all $2\delta + 1$ possible values, we have $0 \leftarrow \text{Receiver}_2$ with $1 - \text{negl}(\kappa)$ probability. \square

Theorem 6 (Security). *The protocol provided in Figure 5 realizes the functionality defined in Figure 2 for L_∞ distance function against semi-honest adversaries if OKVS is oblivious and the DDH assumption holds.*

Proof. First, consider the sender is corrupted. Since the sender doesn't receive any output from the functionality, its view contains only the msg_1 . The simulator can simulate msg_1 by inserting $2\delta + 1$ dummy key-value pairs (i.e., $\text{key}'_j \leftarrow_{\$} \{0, 1\}^\gamma$, $\text{val}'_j \leftarrow_{\$} \mathbb{G} \times \mathbb{G}$) into each of d OKVSs, denoted as E'_i for $i \in [d]$. We show the simulated $\text{msg}'_1 := (g, R \leftarrow_{\$} \mathbb{G}, \mathbf{E}')$ is computationally indistinguishable from msg_1 in the real game as below. According to the DDH assumption, (g, h, h_j, h_j^s) in the real game is a DDH tuple and thus $\text{val}_j := (h_j \parallel h_j^s)$ is indistinguishable from a uniformly random pair of group elements. Then according to the obliviousness of OKVS, as long as inserted $\{\text{val}_j\}$ are uniformly random, then \mathbf{E}' and \mathbf{E} are perfectly indistinguishable. Therefore we have $\text{msg}'_1 \approx_c \text{msg}_1$ which is sufficient to simulate the sender's view.

Now consider the case that the receiver is corrupted. Since the output can be easily simulated by invoking $\mathcal{F}_{\text{FUZZYMATCH}}$, we focus on simulating the msg_2 . If $1 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$, the simulator generates $\text{msg}'_2 := (g^r, \mathbf{H}_\kappa(h^r))$ where $r \leftarrow_{\$} \mathbb{Z}_p$; otherwise, generates $\text{msg}'_2 := (R, \mathbf{H}_\kappa(R'))$ where $R, R' \leftarrow_{\$} \mathbb{G}$. We show the simulated msg'_2 is indistinguishable from msg_2 in the real game as follows. If $1 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$, then there is $u_i^s = v_i$ for each $i \in [d]$ thus $(\prod_{i=1}^d u_i)^s = (\prod_{i=1}^d v_i)$; otherwise, there is at least one $j \in [d]$ such that $u_j^s \neq v_j$ and thus $(\prod_{i=1}^d u_i)^s \neq (\prod_{i=1}^d v_i)$ with overwhelming probability. According to the random self-reduction of DDH tuples shown in Lemma 2, (g, h, u_*, v') is a randomized DDH tuple when $1 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$, and (u_*, v') are truly uniformly random over $\mathbb{G} \times \mathbb{G}$ otherwise. Therefore, we have $\text{msg}'_2 \approx_s \text{msg}_2$, and the receiver's view can be simulated. \square

Theorem 7 (Complexity). *The communication complexity is $O(2\delta d\lambda + \lambda + \kappa)$ where λ, κ are the security and statistical parameters; The computational complexity is $O(2\delta d)$ for the receiver and $O(d)$ for the sender.*

Proof. It is clear to see that the above complexity holds if the instantiation of OKVS has a constant rate, linear encoding time, and constant decoding time. Also, note that the sender can already compute their keys before receiving msg_1 , reducing the computation during online time. \square

Remark 1 (Potential Trade-off). Note that we can save half of the receiver's communication cost by sampling $h_j \leftarrow \mathbf{H}_\mathbb{G}(\text{key}_j)$ instead of being uniformly random in Figure 5 where $\mathbf{H}_\mathbb{G} : \{0, 1\}^* \mapsto \mathbb{G}$ is a hash-to-group mapping and can be modeled as a random oracle in the security analysis. However, the sender's concrete computational cost would be dramatically increased, as during OKVS encoding it needs to perform computations over \mathbb{G} which is much more expensive than working over \mathbb{F}_p .

Remark 2 (Prefix Encoding). We can use the techniques of Chakraborti et al. [CFR23] to obtain a communication complexity of $O(2d\lambda \log \delta + (\lambda + \kappa) \cdot (\log \delta)^d)$ as follows. Instead of encoding the entire interval $[w_i - \delta, w_i + \delta]$ in the OKVS for each dimension $i \in [d]$, the receiver encodes the $O(\log \delta)$ common prefixes representing the interval, as in [CFR23, Algorithm 1]. The sender now similarly has to decode $O(\log \delta)$ values for each dimension, leading to $O((\log \delta)^d)$ potential tuples (u_*, v_*) . We expect this to lead to a lower communication complexity for low dimensions, but will leave it to future work to explore this further.

Conjunction of Hamming and Infinity Distance. If the dimension is low enough, then the construction in Section 5.1 can be easily adapted to support the conjunction of hamming and infinity distance, which is defined in Figure 2. Suppose that the infinity distance threshold is δ_∞ and the hamming distance threshold is δ (noting that $\delta_\infty = 0$ implies the pure hamming distance). Intuitively, we make a random self-reduction for each (u_i, v_i) pair and then utilize threshold secret sharing to encode a “zero message” in Sender_1 . In more detail, we set $f_i = g^{a_i} u_i^{b_i}$ and $h_i = h^{a_i} v_i^{b_i}$ where $a_i, b_i \leftarrow_{\$} \mathbb{Z}_p$ for $i \in [d]$. Since the hamming distance threshold is δ , we generate a $(d - \delta, d)$ -secret sharing of 0 as $(t_1, \dots, t_d) \leftarrow \text{share}_{d-\delta}^d(0)$. In the end, the output of Sender_1 is

$$\text{msg}_2 := \{f_i, t'_i := H_\kappa(h_i) \oplus t_i\}_{i \in [d]}.$$

The algorithm Receiver_2 is modified accordingly: It computes $\tilde{t}_i = H_\kappa(f_i^s) \oplus t'_i$ for $i \in [d]$. Then, it outputs 1 if $0 \leftarrow \text{rec}_{d-\delta}^d(\{\tilde{t}_1, \dots, \tilde{t}_d\})$ and outputs 0 otherwise. In the end, the receiver can succeed by invoking rec_δ^d at most $\binom{d}{d-\delta} = \binom{d}{\delta}$ times. Also, if we would like to tolerate some non-negligible correctness error, we can use *robust secret sharing* [CDD⁺15, DHP⁺18] which lets the receiver reconstruct the secret in an efficient way even some shares are randomly distributed.

5.2 Fuzzy Matching for Minkowski Distance

We provide the protocol for L_p distance where $1 \leq p < \infty$ in Figure 6. For simplicity, we assume p is an integer for the moment. Similarly, the receiver encodes its point as a list of OKVS, each one encoding all possible values for one dimension. When $p \geq 1$, if $\left(\sum_{i=1}^d |w_i - q_i|^p\right)^{1/p} \leq \delta$, this implies that $|w_i - q_i| \leq \delta$ for each $i \in [d]$. Thus the receiver can encode from $w_i - \delta$ to $w_i + \delta$ as before. The sender still uses its point to decode each OKVS and then homomorphically sums up $|w_i - q_i|^p$ for every $i \in [d]$ before applying a random linear function. In the end, the sender gets an encryption of $t = (\text{dist}_p(\mathbf{w}, \mathbf{q}))^p$, then it can rerandomize both the ciphertext and plaintext, and send back the result.

Theorem 8 (Correctness). *The protocol provided in Figure 6 is correct with $1 - \text{negl}(\kappa)$ probability if OKVS satisfies perfect correctness defined in Section 3.1 and independence property from Lemma 1, and $H_\gamma : \{0, 1\}^* \mapsto \{0, 1\}^\gamma$, $H_\kappa : \mathbb{G} \mapsto \{0, 1\}^\kappa$ are universal hash functions where $\gamma = \kappa + \log \delta$ and $\kappa' = \kappa + p \log \delta$.*

Proof. Similar to Theorem 5, with the perfect correctness of the OKVS, if $\text{dist}_p(\mathbf{w}, \mathbf{q}) \leq \delta$, it is clear that the correctness always holds since $v_i = u_i^s \cdot g^{|w_i - q_i|^p}$ for $i \in [d]$ and thus

$$h_* = \left(g^c \cdot \prod_{i=1}^d u_i^b\right)^s \cdot g^{a+b \cdot (\sum_{i=1}^d |w_i - q_i|^p)} = f_*^s \cdot g^{a+b \cdot t},$$

where $t^{1/p} = \text{dist}_p(\mathbf{w}, \mathbf{q})$. Moreover, $t^{1/p} \leq \delta$ and $p \geq 1$ implies $t \leq \delta^p$ and thus there must be some $j_* \in [0 : \delta^p]$ such that $H_{\kappa'}(h_*/f_*^s) = x_{j_*} \in \text{list}_*$.

Now consider the case that $\text{dist}_p(\mathbf{w}, \mathbf{q}) > \delta$. If $q_i \in [w_i - \delta, w_i + \delta]$ for each $i \in [d]$, then we still get $h_* = f_*^s \cdot g^{a+b \cdot t}$, but now $t > \delta^p \pmod p$, then x_t will not be included in the $\{g^{a+b \cdot 0}, \dots, g^{a+b \cdot \delta^p}\}$, thus $\Pr[H_{\kappa'}(h_*/f_*^s) \in \text{list}_*] \leq \frac{\delta^p + 1}{2^{\kappa'}}$ if $H_{\kappa'}$ is a universal hash function; On the other side, if there is some $q_{i'} \notin [w_{i'} - \delta, w_{i'} + \delta]$, then according to the independence property in Lemma 1, $v_{i'}$ and thus h_*/f_*^s would be uniform random over \mathbb{G} , and thus equal to $x_{j \in [0, \delta^p]}$ with negligible probability. \square

$\text{GetList}_p(h, s, \mathbf{w}, \Delta_w)$	$\text{GetTuple}_p(g, h, \mathbf{q}, \Delta_q, \mathbf{E})$
For each $i = 1 \dots d$: For each $j = -\delta \dots \delta$: Set $\text{key}_j \leftarrow \mathbf{H}_\gamma(\Delta_w \ w_i + j)$ Set $h_j \leftarrow \mathbb{G}$ Set $\text{val}_j = (h_j \ h_j^s \cdot g^{ j ^p})$ Set $\text{list}_i = \{(\text{key}_j, \text{val}_j)_{j \in [-\delta:\delta]}\}$ Return $\text{list}_1, \dots, \text{list}_d$	For each $i = 1 \dots d$: Set $\text{key}_i \leftarrow \mathbf{H}_\gamma(\Delta_q \ q_i)$ $(u_i \ v_i) \leftarrow \text{Decode}(E_i, \text{key}_i)$ Sample $a, b, c \leftarrow \mathbb{Z}_p$ Set $f_* = g^c \cdot \left(\prod_{i=1}^d u_i\right)^b$ Set $h_* = h^c \cdot \left(\prod_{i=1}^d v_i\right)^b \cdot g^a$ Set $\text{list}_* = \emptyset$ For each $j = 0 \dots \delta^p$: Set $x_j = \mathbf{H}_{\kappa'}(g^{a+b \cdot j})$ Set $\text{list}_* = \text{list}_* \cup x_j$ Shuffle list_* Return $(f_*, h_*, \text{list}_*)$

$\text{Receiver}_1(\mathbf{w} \in \mathbb{Z}^d)$	$\text{Sender}_1(\mathbf{q} \in \mathbb{Z}^d, \text{msg}_1)$
Sample $g \leftarrow \mathbb{G}, s \leftarrow \mathbb{Z}_p$ Compute $h = g^s$ Get $\{\text{list}_{i \in [d]}\} \leftarrow \text{GetList}_p(h, s, \mathbf{w}, 0^\kappa)$ Get $E_i \leftarrow \text{Encode}(\text{list}_i)$ for each $i \in [d]$ Set $\mathbf{E} = \{E_1, \dots, E_d\}$ Output $\text{msg}_1 := (g, h, \mathbf{E}), \text{st} := s$	Parse $\text{msg}_1 := (g, h, \mathbf{E})$ $(f_*, h_*, \text{list}_*) \leftarrow \text{GetTuple}_p(g, h, \mathbf{q}, 0^\kappa, \mathbf{E})$ Output $\text{msg}_2 := (f_*, h_*, \text{list}_*)$ <hr/> Receiver ₂ (st, msg ₂) Parse $\text{msg}_2 := (f_*, h_*, \text{list}_*)$ and $\text{st} := s$ Set $x = \mathbf{H}_{\kappa'}(f_*^{-s} \cdot h_*)$ Output 1 if $x \in \text{list}_*$ and 0 otherwise

Fig. 6. Fuzzy Matching for L_p Distance

Theorem 9 (Security). *The protocol provided in Figure 6 realizes the functionality defined in Figure 2 for L_p distance function, against semi-honest adversaries if OKVS is oblivious, the hash function $\mathbf{H}_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ is modeled as a random oracle, and the DDH assumption holds.*

Proof. Consider the case that the sender is corrupted. The simulator only needs to simulate msg_1 . The simulator can simulate it the same as in the proof of Theorem 6, according to DDH assumption, $(h_j \| h_j^s)$ is indistinguishable from uniformly random pairs over $\mathbb{G} \times \mathbb{G}$, thus val_j is also random. Then the simulator can simulate msg_1 by inserting dummy random key-value pairs into each E_i since OKVS is oblivious.

Then we consider the case that the receiver is corrupted. The output can be simulated via invoking $\mathcal{F}_{\text{FUZZYMATCH}}$. We focus on simulating the msg_2 . The simulator samples

$$\text{list}'_* := \{r_0, r_1, \dots, r_{\delta^p}\} \leftarrow \mathbb{S} \{0, 1\}^{\kappa' \times (\delta^p + 1)},$$

and sets $\text{msg}'_2 := (g^r, h^r \cdot R', \text{list}'_*)$ where $r \leftarrow_{\$} \mathbb{Z}_p, R' \leftarrow_{\$} \mathbb{G}$. If $1 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$, the simulator additionally programs $H_{\kappa'}$ such that $H_{\kappa'}(R') = r_j$ with a randomly chosen $j \in [0 : \delta^p]$. Now we argue that msg'_2 is statistically indistinguishable from msg_2 in the real game. According to the correctness analysis above, h_* is either $f_*^s \cdot g^{a+bt}$ for $t \leq d\delta^p$ or uniformly random over \mathbb{G} in the real game.

If $1 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$, then (f_*, h_*) is distributed exactly the same as $(g^r, h^r \cdot R')$ due to $a, b, c \leftarrow_{\$} \mathbb{Z}_p$. Moreover, since $H_{\kappa'}$ is modelled as a random oracle, list_* and list'_* are perfectly indistinguishable, unless the adversary queried $H_{\kappa'}$ for some $g^{a+b \cdot j}$ for $j \neq t \wedge j \in [0 : \delta^p]$. However, since any query $g^a \in \mathbb{G}$ can be represented as $g^{a+b \cdot j'}$ for some $j' \in \mathbb{Z}_p$. With uniformly random a, b , given a single point g^{a+bt} , the linear system is under-determined, thus j' is still uniformly random over \mathbb{Z}_p . This means the adversary can only query $H_{\kappa'}$ for some unknown random j after seeing g^{a+bt} , and thus the probability that they query for some $j \in [0 : \delta^p]$ is less than $\frac{\delta^p+1}{|\mathbb{G}|}$. Thus, $\text{msg}'_2 \approx_s \text{msg}_2$ in this case.

For the case that $0 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$, it is similar to the above case, except that $h_* = f_*^s g^{a+bt}$ for $t > \delta^p$ when $\text{dist}_{\infty}(\mathbf{w}, \mathbf{q}) \leq \delta < \text{dist}_p(\mathbf{w}, \mathbf{q})$, or $h_* \leftarrow_{\$} \mathbb{G}$ when $\delta < \text{dist}_{\infty}(\mathbf{w}, \mathbf{q})$. And (f_*, h_*) is perfectly indistinguishable from $(g^r, h^r \cdot R')$ in either case. Also, $\text{msg}'_2 \approx_s \text{msg}_2$ if the adversary didn't query $H_{\kappa'}$ with $g^{a+b \cdot j'}$ for any $j' \in [0 : \delta^p]$. This probability is $\text{negl}(\lambda)$ as we argued above. Therefore, the receiver's view can be statistically simulated. \square

When list_* supports constant lookup time and linear insertion time (e.g., instantiated by a Cuckoo Hash table), we can achieve the following complexity.

Theorem 10 (Complexity). *The communication complexity is $O(2\delta d\lambda + 2\lambda + \delta^p \kappa)$ where λ, κ are the security and statistical parameters; The computational complexity is $O(2\delta d)$ for the receiver and $O(d + \delta^p)$ for the sender.*

Proof. The above complexity can be achieved if list_* supports constant lookup time and linear insertion time (e.g., instantiated by a Cuckoo Hash table). \square

Remark 3 ($p \notin \mathbb{Z}$). When p is not an integer, we can scale the space to round p into an integer. This, however, loses some precision during rounding and increases the computation and communication overhead by a p power of the scaling factor.

6 Fuzzy PSI in Low-Dimension Space

Clearly, with a fuzzy matching protocol in hand, we could straightforwardly execute a protocol instance for every pair of points from both the sender and receiver. Yet, this approach would lead to a quadratic increase in computational and communicative overheads. In the following sections, we depict some methods to circumvent this quadratic overhead, addressing both low-dimensional (in Section 6) and high-dimensional (in Section 7) spaces separately. We will deal with PSI-CA first (i.e., only let the receiver learn the cardinality of the intersection), then show how to extend PSI-CA to broader functionalities in Section 8, including standard PSI, labeled PSI, and circuit PSI.

6.1 Spatial Hashing Techniques

Consider the case that points are located in a low-dimension space \mathcal{U}^d (e.g., $d = o(\log(\lambda))$) where \mathcal{U} is the universe for each dimension. We use a similar idea from [GRS22] to tile the entire space into hypercubes with side length 2δ , but we consider a more general L_p distance setting. That

is, we consider L_p distance over a space tiled by L_∞ hypercubes. We denote each hypercube as a *cell*. Specifically, given a point $\mathbf{w} \in \mathcal{U}^d$, the index id_i of each cell \mathcal{C} on each dimension $i \in [d]$ is determined by $\text{id}_i = \lfloor \frac{w_i}{2\delta} \rfloor$ and each cell is labeled by $\text{id}_0 \parallel \dots \parallel \text{id}_d$.

Lemma 3 (Maximal Distance in a Cell). *Given two points $\mathbf{w}, \mathbf{q} \in \mathcal{U}^d$ located in the same cell with side length 2δ , then the distance between them is $\text{dist}_p(\mathbf{w}, \mathbf{q}) < 2\delta d^{\frac{1}{p}}$ where $p \in [1, \infty]$. Specifically, if $p = \infty$, $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) < 2\delta$.*

Proof. Suppose the distance is equal to or greater than $2\delta d^{\frac{1}{p}}$, then there must exist some $i_* \in [d]$ such that $|w_{i_*} - q_{i_*}| \geq 2\delta$: Since $p > 1$, if $|w_i - q_i| < 2\delta$ holds for each $i \in [d]$, then

$$\left(\sum_i^d |w_i - q_i|^p \right)^{1/p} < (2^p \delta^p d)^{1/p} = 2\delta \cdot d^{1/p}.$$

Without loss of generality, we assume $w_{i_*} \geq q_{i_*} + 2\delta$ which implies $\lfloor \frac{w_{i_*}}{2\delta} \rfloor \geq \lfloor \frac{q_{i_*}}{2\delta} \rfloor + 1$ and they are not in the same cell. \square

Lemma 4 (Unique Center). *Suppose there are multiple L_p balls ($p \in [1, \infty]$) with radius δ lying in a d -dimension space which is tiled by hypercubes (i.e., cells) with side length 2δ . If these balls' centers are at least $2\delta d^{\frac{1}{p}}$ apart, then for each cell, there is at most one center of the balls lying in this cell. Specifically, if $p = \infty$, then the unique center holds for disjoint balls since $2\delta d^{\frac{1}{p}}$ degrades to 2δ in this case.*

Proof. Given any pair of centers $\mathbf{c}_1, \mathbf{c}_2$ satisfying that $\text{dist}_p(\mathbf{c}_1, \mathbf{c}_2) > 2\delta d^{\frac{1}{p}}$. Then it is clear to see that for any two centers $\mathbf{c}_1, \mathbf{c}_2$ of different balls, if they are located in the same cell, then $\text{dist}_p(\mathbf{c}_1, \mathbf{c}_2) < 2\delta d^{\frac{1}{p}}$ according to Lemma 3. This completes the proof. \square

Lemma 5 (Unique Ball). *Suppose there are multiple δ -radius L_p balls ($p \in [1, \infty]$) distributed in a d -dimension space which is tiled by hypercubes (cells) of side length 2δ . If these balls' centers are at least $2\delta(d^{\frac{1}{p}} + 1)$ apart from each other, then there exists at most one ball intersecting with the same cell. Specifically, if $p = \infty$, this holds for L_∞ balls with 4δ -apart centers.*

Proof. It is clear to check that the maximal distance in a cell is less than $2\delta d^{\frac{1}{p}}$ according to Lemma 3. In other words, any two points \mathbf{w}, \mathbf{q} in the cell have $\text{dist}_p(\mathbf{w}, \mathbf{q}) < 2\delta d^{\frac{1}{p}}$. Therefore, if the centers of two balls are $2\delta(d^{\frac{1}{p}} + 1)$ apart, any points \mathbf{w}, \mathbf{q} inside (or on) the ball will be at least $2\delta d^{\frac{1}{p}}$ apart because:

$$\begin{aligned} \text{dist}_p(\mathbf{w}, \mathbf{q}) &\geq \text{dist}_p(\mathbf{c}_1, \mathbf{q}) - \text{dist}_p(\mathbf{c}_1, \mathbf{w}) \\ &\geq \text{dist}_p(\mathbf{c}_1, \mathbf{c}_2) - \text{dist}_p(\mathbf{c}_2, \mathbf{q}) - \text{dist}_p(\mathbf{c}_1, \mathbf{w}) \\ &\geq 2\delta(d^{\frac{1}{p}} + 1) - \delta - \delta \\ &= 2\delta d^{\frac{1}{p}} \end{aligned}$$

holds by the triangle inequality, where $\mathbf{c}_1, \mathbf{c}_2$ are centers of the balls containing \mathbf{w}, \mathbf{q} , respectively. \square

Lemma 6 (Unique Block). *Any L_∞ ball with radius δ will intersect with **exactly** 2^d cells with side length 2δ 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.*

Proof. Consider vertices of a L_∞ ball as $\{\mathbf{v}_1, \dots, \mathbf{v}_{2^d}\}$. Since the ball has radius δ , the distance between each pair of adjacent vertices is 2δ . This implies each vertex lies in different but adjacent cells according to Lemma 3: For each adjacent $\mathbf{v}_i, \mathbf{v}_j$ that differ in only one dimension k , there is

$$\left| \left\lfloor \frac{v_{i,k}}{2\delta} \right\rfloor - \left\lfloor \frac{v_{j,k}}{2\delta} \right\rfloor \right| = 1.$$

Therefore, each vertex of a L_∞ ball is located in each cell of the block. Then we show the block is unique for each disjoint ball.

Suppose two disjoint balls are lying in the same block. For each cell of the block, there is a vertex lying in this cell for both balls. From Lemma 3, the distance between each pair of vertices in the same cell is smaller than 2δ . On the other hand, disjoint balls imply the distance between two centers $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) > 2\delta$ where \mathbf{w}, \mathbf{q} are two center points. This means that there is some dimension $i \in [d]$, such that $|w_i - q_i| > 2\delta$. Moreover, on this dimension, assuming $w_i > q_i$, there is

$$(w_i + \delta) - (q_i \pm \delta) > 2\delta.$$

$w_i + \delta$ is the projection on this dimension from some vertex, and $q_i \pm \delta$ is the projection from any vertex of the other ball. In other words, a vertex \mathbf{v}_* of the ball centered at \mathbf{w} , which maintains a distance greater than 2δ from *any vertex* of the ball centered at \mathbf{q} . However, as per the preceding discussion, the vertex that shares the same cell with \mathbf{v}_* must have a distance less than 2δ . This completes the proof. \square

6.2 Fuzzy PSI-CA for Infinity Distance

With the spatial hashing techniques, we are prepared to get around the quadratic overhead mentioned above. It is worth noting that, a fuzzy PSI functionality for two parties holding a point set, can be framed as the so-called *structure-aware* PSI [GRS22]: The receiver holds a set of balls with radius δ (i.e., structured), and the sender holds a set of points (i.e., non-structured). In this and the next section, we mainly focus on the fuzzy PSI-CA functionality as defined in Figure 3, or equivalently, counting how many sender's points lie inside the receiver's balls.

We provide the detailed protocol in Figure 7 realizing fuzzy PSI-CA for infinity distance where the receiver's points are 2δ apart from each other (i.e., the receiver's δ -radius balls are disjoint). In the figure, $\text{block}_{4\delta}$ returns the label of the block of side-length 4δ , $\text{cell}_{2\delta}$ returns the label of the cell of side-length 2δ , and $\text{GetList}, \text{GetTuple}$ are provided in Figure 5. The high-level intuition is that the receiver encodes each of its balls as a unique block, and then the sender checks all possible 2^d blocks containing the cell where its point is located.

Theorem 11 (Correctness). *The protocol presented in Figure 7 is correct with probability $1 - \text{negl}(\kappa)$ if OKVS satisfies perfect correctness defined in Section 3.1 and the independence property from Lemma 1, $H_\gamma : \{0, 1\}^* \mapsto \{0, 1\}^\gamma$, $H_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ used in $\text{GetList}, \text{GetTuple}$ are universal hash functions where $\gamma = \kappa + d + \log(MN\delta)$, $\kappa' = \kappa + d + \log M$, and the receiver's points are 2δ apart.*

Proof. Note that if $\text{dist}_\infty(\mathbf{w}_i, \mathbf{q}_j) \leq \delta$ for $i \in [N], j \in [M]$, its correctness reduces to Theorem 5 when $\mathcal{B}_k = \mathcal{B}_j$. Thus, in this case, we only need to show that, with overwhelming probability, the sender will iterate to the same block the receiver encodes its point. This is true because, according to Lemma 6, the receiver's ball will intersect with 2^d cells which means the sender's point lies in one of these cells and thus lies in the same block. When $\text{dist}_\infty(\mathbf{w}_i, \mathbf{q}_j) > \delta$, the sender will not get the same key to decode OKVS for at least one dimension if H_γ is a universal hash, except with

Receiver ₁ ($\mathbf{W} \in \mathbb{Z}^{d \times N}$)	Sender ₁ ($\mathbf{Q} \in \mathbb{Z}^{d \times M}, \text{msg}_1$)
Sample $g \leftarrow \mathbb{G}, s \leftarrow \mathbb{Z}_p$	Parse $\text{msg}_1 := (g, h, \mathbf{E})$
Compute $h = g^s$	For each $k = 1 \dots M$:
Set $\text{list}_i = \emptyset$ for each $i \in [d]$	For each $\mathcal{B}_{j \in [2^d]}$ containing $\text{cell}_{2\delta}(\mathbf{q}_k)$:
For each $k = 1 \dots N$:	$(u_{k,j}, v_{k,j}) \leftarrow \text{GetTuple}_\infty(g, h, \mathbf{q}_k, \mathcal{B}_j, \mathbf{E})$
$\mathcal{B}_k \leftarrow \text{block}_{4\delta}(\mathbf{w}_k)$	Set $\text{msg}_2 = \{(u_{k,j}, v_{k,j})_{k \in [M], j \in [2^d]}\}$
$\{\text{list}'_{i \in [d]}\} \leftarrow \text{GetList}_\infty(h, s, \mathbf{w}_k, \mathcal{B}_k)$	Shuffle msg_2 by j and then by k
$\text{list}_i = \text{list}_i \cup \text{list}'_i$ for each $i \in [d]$	Output msg_2
Get $\forall i \in [d], E_i \leftarrow \text{Encode}(\text{list}_i)$	Receiver ₂ (st, msg_2)
Set $\mathbf{E} = \{E_1, \dots, E_d\}$	Parse $\text{msg}_2 := \{(u_{k,j}, v_{k,j})_{k \in [M], j \in [2^d]}\}, \text{st} := s$
Output $\text{msg}_1 := (g, h, \mathbf{E}), \text{st} := s$	Set $c = 0$
	For $k = 1 \dots M$:
	If $\exists j_* \in [2^d], H_{\kappa'}(u_{k,j_*}^s) = v_{k,j_*}$:
	Set $c = c + 1$
	Output c

Fig. 7. Fuzzy PSI-CA, infinity distance, receiver's points are 2δ apart (i.e., disjoint balls)

$\frac{2\delta+1}{2^\gamma}$ probability. After taking a union bound over each pair of points, it is $\text{negl}(\kappa)$. This results in inconsistent $(u_{k,j}, v_{k,j})$ pairs with $1 - \text{negl}(\kappa)$ probability as in Theorem 5, and the probability of existing some $j_* \in [2^d]$ such that $H_{\kappa'}(u_{k,j_*}^s) = v_{k,j_*}$ is still negligible if $\kappa' = \kappa + d \log M$ where κ is the statistical parameter. \square

Theorem 12 (Security). *The protocol presented in Figure 7 realizes the fuzzy PSI-CA functionality defined in Figure 3 for infinity distance against semi-honest adversaries if OKVS is oblivious, and the DDH assumption holds.*

Proof. Intuitively, the usage of spatial hashing doesn't affect the security but only the correctness. To see this, we still consider two cases.

If the receiver is corrupted, the simulator needs to simulate msg_2 to complete the receiver's view. The simulator invokes the ideal functionality to get the cardinality of the intersection $c \leftarrow \mathcal{F}_{\text{FUZZYPSI}}$. It then samples c random indices denoted as $\mathcal{I} := \{I_i \in [M]\}_{i \in [c]}$ and sets $\tilde{L} := \{L_1, \dots, L_M\}$, where

$$L_k := \{(u_{k,j}, v_{k,j})_{j \in [2^d]}\} \leftarrow \mathbb{G} \times \mathbb{G}^{2^d}.$$

Additionally, for each $k \in \mathcal{I}$, the simulator puts $(u_{k,j_*}, v_{k,j_*}) = (g^r, h^r)$ in L_k where $j_* \leftarrow [d], r \leftarrow \mathbb{Z}_p^*$. In the end, the simulated message would be $\text{msg}'_2 := \tilde{L}$. The simulated message msg'_2 is indistinguishable from msg_2 in the real game. According to Theorem 6, each $\{(u_{k,j}, v_{k,j})_{j \in [2^d]}\}$ in msg_2 is either uniformly random over $(\mathbb{G} \times \mathbb{G})^{2^d}$, or has a DDH tuple at some location $j_* \in [d]$. Since msg_2 is shuffled, it is clear to see that $\text{msg}'_2 \approx_s \text{msg}_2$.

If the sender is corrupted, the simulator invokes $\mathcal{F}_{\text{FUZZYPSI}}$ to get the output and simulate msg_1 to complete the sender's view. The simulator samples

$$L_i := \{(\text{key}_j, \text{val}_j)\} \leftarrow \mathbb{G} \times \mathbb{G}^{N(2\delta+1)}$$

for each dimension $i \in [d]$, sets $E'_i \leftarrow \text{OKVS.Encode}(L_i)$, and sets $\text{msg}'_1 := \{E'_1, \dots, E'_d\}$. Since each \mathcal{B}_k is unique to \mathbf{w}_k , the key j generated during GetList_∞ in the real game should be distinct from each other with $1 - \text{negl}(\kappa)$ probability. Thus each list'_i in Figure 7 will not have the intersection with the existing list_i which means $\|\text{list}'_i\| = N(2\delta + 1)$. Then according to Theorem 6, each $E'_i \approx_c E_i$ and thus $\text{msg}'_1 \approx_c \text{msg}_1$. This completes the proof. \square

Theorem 13 (Complexity). *The protocol provided in Figure 7 has communication complexity $O(2\delta dN\lambda + 2^d M(\lambda + \kappa'))$ where $\lambda, \kappa = \kappa' - d \log M$ are the security and statistical parameters; The computational complexity is $O(2\delta dN + 2^d M)$ for the receiver and $O(2^d dM)$ for the sender.*

Structured Sets for Both Parties. Consider the case that both the sender and the receiver hold a structured set [GRS22]. Namely, the receiver's set consists of δ -radius balls whose centers are 4δ apart, and the sender's set consists of multiple δ -radius balls. Our construction in Figure 7 can also work in this case: We tile the space with cells of side length 4δ and each block has side length 8δ . Then the protocol would be exactly the same as in Figure 7 except that we replace $\{\delta, \text{block}_{4\delta}, \text{cell}_{2\delta}\}$ with $\{2\delta, \text{block}_{8\delta}, \text{cell}_{4\delta}\}$ instead. The intuition is that computing the intersection between two balls is equivalent to comparing the distance between two centers of the balls (i.e., checking if they are $\leq 2\delta$ or not).

6.3 Fuzzy PSI-CA for Minkowski Distance

Assuming that the receiver's points are spaced $2\delta(d^{\frac{1}{p}} + 1)$ apart, we can allow the receiver to iterate through each possible location, as depicted in Figure 8. The correctness and security naturally follow from the above discussion, given that each $\text{ball}_\delta(\mathbf{w}_k)$ will intersect with at most 2^d cells and is unique for each intersecting cell according to Lemma 5. Note that if $L_p = L_\infty$, each ball will intersect with exact 2^d cells according to Lemma 6; For other $1 \leq p < \infty$, L_p ball is smaller than L_∞ thus intersects with at most 2^d cells. Specifically, when $p = \infty$, the receiver's points are spaced 4δ apart.

Theorem 14 (Correctness). *The protocol presented in Figure 8 is correct with probability $1 - \text{negl}(\kappa)$ if OKVS satisfies the perfect correctness defined in Section 3.1 and the independence property from Lemma 1, $H_\gamma : \{0, 1\}^* \mapsto \{0, 1\}^\gamma$, $H_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ used in $\text{GetList}, \text{GetTuple}$ are universal hash functions where $\gamma = \kappa + d + \log(\delta NM)$, $\kappa' = \kappa + \log(M\delta^p)$ if $p < \infty$ and $\kappa' = \kappa + \log M$ if $p = \infty$, and the receiver's points are $2\delta(d^{\frac{1}{p}} + 1)$ apart for $p \in [1, \infty]$.*

Proof. The correctness holds as long as the receiver can encode OKVS successfully and the cell of the sender's point is iterated during the encoding process. Since for each cell, there is only one ball intersecting with it according to Lemma 5, thus each cell $\mathcal{C}_{k'}$ is unique and results in distinct key with $1 - \text{negl}(\kappa)$ probability. Also, each ball intersects at most 2^d cells where any point inside (or on) the ball must lie in one of these cells. \square

Theorem 15 (Security). *The protocol presented in Figure 8 realizes the fuzzy PSI-CA functionality defined in Figure 3 for $L_{p \in [1, \infty]}$ distance against semi-honest adversaries if OKVS is oblivious and the DDH assumption holds. Additionally, if $p < \infty$, the hash function $H_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ is modeled as a random oracle.*

Proof. When $p = \infty$, both views can be simulated similarly as in Theorem 12. When $p < \infty$, the general strategy is the same as in the infinity setting, but the output of GetTuple_p will be simulated by the approach in Theorem 9. \square

Receiver ₁ ($\mathbf{W} \in \mathbb{Z}^{d \times N}$)	Sender ₁ ($\mathbf{Q} \in \mathbb{Z}^{d \times M}, \text{msg}_1$)
Sample $g \leftarrow \mathbb{G}, s \leftarrow \mathbb{Z}_p$	Parse $\text{msg}_1 := (g, h, \mathbf{E})$
Compute $h = g^s$	For each $k = 1 \dots M$:
Set $\text{list}_i = \emptyset$ for each $i \in [d]$	$\mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{q}_k)$
For each $k = 1 \dots N$:	$t_k \leftarrow \text{GetTuple}_p(g, h, \mathbf{q}_k, \mathcal{C}_k, \mathbf{E})$
For each cell $\mathcal{C}_{k'}$ intersecting $\text{ball}_\delta(\mathbf{w}_k)$:	Set $\text{msg}_2 = \{t_{k \in [M]}\}$ and shuffle
$\{\text{list}'_{i \in [d]}\} \leftarrow \text{GetList}_p(h, s, \mathbf{w}_k, \mathcal{C}_{k'})$	Output msg_2
$\text{list}_i = \text{list}_i \cup \text{list}'_i$ for each $i \in [d]$	Receiver ₂ (st, msg_2)
Pad list_i to size $2^d N$ with random pairs	Parse $\text{msg}_2 := \{t_{k \in [M]}\}, \text{st} := s$
Get $\forall i \in [d], E_i \leftarrow \text{Encode}(\text{list}_i)$	Set $c = 0$
Set $\mathbf{E} = \{E_1, \dots, E_d\}$	For $k = 1 \dots M$:
Output $\text{msg}_1 := (g, h, \mathbf{E}), \text{st} := s$	If $p = \infty$: Parse $t_k := (u_k, v_k)$
	Set $c = c + 1$ if $\mathbf{H}_{\kappa'}(u_k^s) = v_k$
	Else: Parse $t_k := (f_k, h_k, \mathcal{X}_k)$
	Set $c = c + 1$ if $\mathbf{H}_{\kappa'}(f_k^{-s} h_k) \in \mathcal{X}_k$
	Output c

Fig. 8. Fuzzy PSI-CA, L_p distance with $p \in [1, \infty]$, receiver's points are $2\delta(d^{\frac{1}{p}} + 1)$ apart

Theorem 16 (Complexity). *The protocol provided in Figure 8 has communication complexity $O(2\delta d 2^d N \lambda + M(\lambda + \kappa'))$ when $L_p = L_\infty$ and $O(2\delta d 2^d N \lambda + M(2\lambda + \delta^p \kappa'))$ when $p \in [1, \infty)$ where λ, κ are the security and statistical parameters. Specifically, $\kappa = \kappa' - \log M$ if $p = \infty$ and $\kappa = \kappa' - p \log(M\delta)$ otherwise. The receiver's computational complexity is $O(2\delta d 2^d N + M)$; The sender's computational complexity is $O(dM)$ if $p = \infty$ and $O(dM + \delta^p M)$ otherwise.*

Structured Sets for Both. Consider the case that both the sender and the receiver hold a structured set [GRS22], namely, the receiver's set consists of δ -radius balls whose centers are 4δ apart, and the sender's set consists of multiple δ -radius balls. Our construction in Figure 7 can also work in this case: We tile the space with cells of side length 4δ and each block has side length 8δ . Then the protocol would be exactly the same as in Figure 7 except that we replace $\{\delta, \text{block}_{4\delta}, \text{cell}_{2\delta}\}$ with $\{2\delta, \text{block}_{8\delta}, \text{cell}_{4\delta}\}$ instead. The intuition is that computing the intersection between two balls is equivalent to comparing the distance between two centers of the balls (i.e., checking if they are smaller than 2δ or not).

7 Fuzzy PSI in High-Dimension Space

In this section, we construct an efficient fuzzy PSI protocol in a high-dimensional space, i.e., of a polynomially large dimension. For infinity distance, we provide a fuzzy PSI-CA protocol in Section 7.1 and extend it to richer functionalities in Section 8; For Minkowski distance, we provide a standard fuzzy PSI protocol in Section 7.2 as it doesn't support PSI-CA or other stronger functionalities.

7.1 Infinity Distance

Suppose we assume the receiver's set has good distribution in a high-dimensional space, particularly if each ball has disjoint projections (i.e., separated) from others on *at least one* dimension. In this case, we can get communication and computation complexity both scaling *polynomially* in the dimension. For instance, if balls are uniformly distributed, then it satisfies this predicate with overwhelming probability.

Definition 3 (Separated Balls). *The set of δ -radius balls are separated in a d -dimension space if and only if the projections are separated on at least one dimension for each ball. Specifically, for the center \mathbf{w}_k of each ball in the set, there exists some dimension $i_* \in [d]$ such that*

$$\forall j \in [-\delta : \delta], w_{k,i_*} + j \notin \{w_{k',i_*} + j'\}_{k' \neq k, j' \in [-\delta : \delta]},$$

where $\{w_{k',i_*} + j'\}$ is the set of projections from other balls.

Lemma 7 (Uniform Distribution). *If centers of the balls are uniformly distributed ($\mathbf{W} \leftarrow \mathcal{U}^{d \times N}$) where $\mathcal{U} := \mathbb{Z}_{2^u}$, then it has the property defined in Definition 3 with probability $1 - \text{negl}(d)$.*

Proof. For each \mathbf{w}_k with $k \in [N]$, its projection on each dimension i is an interval of length $2\delta + 1$ and thus the probability that it collides with the projection of other $\mathbf{w}_{k'}$ with $k' \neq k$ on some specific dimension is

$$\Pr[\text{Not separated on some dimension}] \leq \frac{(N-1)(2\delta+1) \cdot 2}{2^u}.$$

The probability of colliding with others for each dimension is smaller than $\left(\frac{2(N-1)(2\delta+1)}{2^u}\right)^d$. Therefore, according to the union bound, the probability that there exists some ball collides with other balls onto *every* dimension is

$$\Pr[\text{Not a good distribution}] \leq \left(\frac{2N^{\frac{1}{d}+1}(2\delta+1)}{2^u}\right)^d,$$

which is clearly $\text{negl}(d)$ when $\frac{\log N + \log \delta + 2}{u}$ is a constant smaller than 1. \square

Given that the receiver's balls are separated as defined in Definition 3, we provide an efficient protocol in Figure 9 that removes the term 2^d for communication and computation. The high-level intuition is the following: For each \mathbf{w}_k , the receiver encodes an inner OKVS of size $O(2\delta d)$ following the same approach in Figure 5, then encode an outer OKVS for each dimension while using this inner OKVS as the encoded value for the unique separated dimension and dummy instances for other dimensions.

Theorem 17 (Correctness). *The protocol presented in Figure 9 is correct with probability $1 - \text{negl}(\kappa)$ if OKVS satisfies the perfect correctness, \mathbb{F}_p -linearity defined in Section 3.1 and the independence property from Lemma 1, $\mathbf{H}_\gamma : \{0, 1\}^* \mapsto \{0, 1\}^\gamma$, $\mathbf{H}_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ are universal hash functions where $\gamma = \kappa + \log NM\delta$, $\kappa' = \kappa + \log M$, and the receiver's set are separated as defined in Definition 3. Particularly, we require that the decoding vector satisfies $\text{dec}(\cdot) \in \{0, 1\}^m$ and $\text{HammingWeight}(\text{dec}(\cdot)) = O(\kappa)$ where m is the size of the OKVS.*

Receiver ₁ ($\mathbf{W} \in \mathbb{Z}^{d \times N}$)	
Sample $g \leftarrow \mathbb{G}, s \leftarrow \mathbb{Z}_p$, compute $h = g^s$, and set $\text{list}_i = \emptyset$ for each $i \in [d]$	
Set $m = (2\delta + 1)d \cdot (1 + \epsilon)$ where ϵ is the expansion factor of the OKVS scheme	
Denote dec for the \mathbb{F}_p -linear decoding function of the OKVS scheme	
Denote as I_k the <i>first</i> separated dimension for the ball $\mathbf{w}_k \in \mathbf{W}$	
Sample $\zeta \leftarrow \mathbb{G}$	
For each $k = 1 \dots, N$:	
For each $i = 1 \dots d$:	
For each $j \in [-\delta, +\delta]$:	
If $i = I_k$:	
Set $\text{val}_j \leftarrow \text{Encode}\left(\left\{(\mathbf{H}_\gamma(i', w_{k,i'} + j'), h_{i',j'} \parallel h_{i',j'}^s \cdot \zeta^{-x_{i',j'}})\right\}_{i' \in [d], j' \in [-\delta, +\delta]}\right)$	
where $h_{i',j'} \leftarrow \mathbb{G}$ and $x_{i',j'} := (d - 1) \cdot \sum_{\ell=1}^m \text{dec}(H_\gamma(i', w_{k,i'} + j'))_\ell$	
Else:	
Set $\text{val}_j := (\mathbf{r}_j \parallel \zeta \cdot \mathbf{r}_j^s)$ where $\mathbf{r}_j \leftarrow \mathbb{G}^m$	
Set $\text{list}_i = \text{list}_i \cup (\mathbf{H}_\gamma(i, w_{k,i} + j), \text{val}_j)$	
Pad each list_i to size $(2\delta + 1)N$ with random key-val pairs	
Set $E_i \leftarrow \text{Encode}(\text{list}_i)$ for each $i \in [d]$, and $\mathbf{E} = \{E_1, \dots, E_d\}$	
Output $\text{msg}_1 := (g, h, \mathbf{E}), \text{st} := s$	
Sender ₁ ($\mathbf{Q} \in \mathbb{Z}^{d \times M}, \text{msg}_1$)	Receiver ₂ (st, msg_2)
Parse $\text{msg}_1 := (g, h, \mathbf{E})$	Parse $\text{msg}_2 := \{(u_k, v_k)_{k \in [M]}\}, \text{st} := s$
For each $k = 1 \dots M$:	Set $c = 0$
Sample $a, b \leftarrow \mathbb{Z}_p$	For $k = 1 \dots M$:
For each $i = 1 \dots d$:	If $\mathbf{H}_{\kappa'}(u_k^s) = v_k$, set $c = c + 1$
Set $\mathbf{e}_i \leftarrow \text{Decode}(E_i, \mathbf{H}_\gamma(i, q_{k,i}))$	Output c
For each $j = 1 \dots d$:	
Set $(f'_j \parallel h'_j) \leftarrow \text{Decode}(\mathbf{e}_i, \mathbf{H}_\gamma(j, q_{k,j}))$	
Set $f_i := \prod_{j=1}^d f'_j, h_i := \prod_{j=1}^d h'_j$	
Set $u_k = g^a \cdot \prod_{i=1}^d f_i^b, v'_k = h^a \cdot \prod_{i=1}^d h_i^b$, and $v_k = \mathbf{H}_{\kappa'}(v'_k)$	
Set $\text{msg}_2 = \{(u_k, v_k)_{k \in [M]}\}$	
Shuffle and output msg_2	

Fig. 9. Fuzzy PSI-CA, infinity distance, each ball is separated on at least one dimension

Proof. First, consider the receiver's encoding process. If the receiver's set satisfies the Definition 3, then there will be at least one dimension that is "free" for each \mathbf{w}_k to encode the inner OKVS instance which guarantees the receiver's set can be successfully encoded. Now consider the sender's point \mathbf{q}_k for some $k \in [M]$. If $\text{dist}_\infty(\mathbf{w}, \mathbf{q}_k) \leq \delta$, then after decoding the outer OKVS, the sender will get a correct inner OKVS instance \mathbf{e}_{i^*} for some dimension $i^* \in [d]$ on which there exists a

separated interval, or a dummy instance $(\mathbf{r} \parallel \zeta \cdot \mathbf{r}^s)$ for other dimensions. Because the OKVS has \mathbb{F}_p -linear decoding function dec , decoding a dummy instance $(\mathbf{r} \parallel \zeta \cdot \mathbf{r}^s)$ at $H_\gamma(j, q_j)$ will output:

$$\langle \text{dec}(H_\gamma(j, q_j)), \mathbf{r} \parallel \mathbf{r}^s \cdot \zeta \rangle = \prod_{\ell=1}^m (r_\ell \parallel r_\ell^s \cdot \zeta)^{d_{j,\ell}} = \prod_{\ell=1}^m r_\ell^{d_{j,\ell}} \parallel \zeta^{x_j} \cdot \left(\prod_{\ell=1}^m r_\ell^{d_{j,\ell}} \right)^s,$$

where $d_{j,\ell} := \text{dec}(H_\gamma(j, q_j))_\ell$ and $x_j := \sum_{\ell=1}^m d_{j,\ell}$. Thus, decoding a dummy instance d times and then multiplying them together would get an ElGamal encryption of $\zeta^{\sum_{j=1}^d x_j}$, whereas decoding a correct inner OKVS instance \mathbf{e}_{i^*} for all dimensions and multiplying them together yields an ElGamal encryption of $\zeta^{-(d-1) \cdot \sum_{j=1}^d x_j}$. Given the fact that the sender's point \mathbf{q}_k is located inside a ball, and there exists exactly one dimension to get an inner OKVS instance, the sender would get $\left(\prod_{i=1}^d f_i, \prod_{i=1}^d h_i \right)$ which is an ElGamal encryption of

$$\zeta^{-(d-1) \cdot \sum_{j=1}^d x_j} \cdot \prod_{i=1, i \neq i^*}^d \zeta^{\sum_{j=1}^d x_j}$$

which is equal to 1. Therefore, $(g, h, \prod_{i=1}^d f_i, \prod_{i=1}^d h_i)$ is a valid DDH tuple. Using a similar argument as before we know the receiver can correctly verify the sender's output (u_k, v_k) .

On the other hand, if $\text{dist}_\infty(\mathbf{w}_j, \mathbf{q}_k) > \delta$ for every $j \in [N]$, any incorrect decoding (i.e., using a non-encoded key) on either outer or inner OKVS results in a random output over $\mathbb{G} \times \mathbb{G}$, making $(g, h, \prod_{i=1}^d f_i, \prod_{i=1}^d h_i)$ a DDH tuple with probability at most $\text{negl}(\kappa)$. Consider the case that the outer OKVS is decoded correctly for each dimension, and then the sender gets either dummy instances or inner OKVS instances. Since we assume each inner OKVS instance has to be correctly decoded, a non-match point \mathbf{q}_k can only report a false positive when those instances are *all* dummy. However, in this case, the final product is an ElGamal encryption of $\prod_{i=1}^d \zeta^{\sum_{j=1}^d x_j}$, which is equal to 1 with only $\text{negl}(\kappa)$ probability since $\zeta \leftarrow_{\$} \mathbb{G}$ and $d \cdot \sum_{i=1}^d x_j = O(\kappa \cdot d^2) \ll p$. \square

Theorem 18 (Security). *The protocol presented in Figure 9 satisfies the fuzzy PSI-CA functionality defined in Figure 3 for infinity distance against semi-honest adversaries if OKVS is doubly oblivious, and the DDH assumption holds.*

Proof. We can still simulate $\text{msg}_1, \text{msg}_2$ as before and only need to take care of the size of the OKVS. Because the receiver's balls might overlap on some dimension $i \in [d]$ which in turn affects the size of list_i , the receiver pads list_i with some dummy key-value pairs to simulate it. Accordingly, in the simulation against a corrupted sender, the simulator generates a list of size $(2\delta + 1)N$, containing uniformly random key-value pairs, for each dimension and encodes it over OKVS. This is indistinguishable due to the double obliviousness property, where the double obliviousness property also guarantees that the inner OKVS is uniformly random. \square

Theorem 19 (Complexity). *The protocol presented in Figure 9 has communication complexity $O((2\delta d)^2 N \lambda + M(\lambda + \kappa))$ where $\lambda, \kappa = \kappa' - \log M$ are computational and statistical parameters; The computational complexity is $O((2\delta d)^2 N + M)$ for the receiver and $O(2d^2 M)$ for the sender.*

Remark 4. Note that the above approach in Figure 9 can also be extended to the L_p setting by replacing the underlying fuzzy matching protocol from L_∞ to L_p . Moreover, since L_p balls are confined in L_∞ balls, Lemma 7 still holds in this setting.

7.2 Minkowski Distance

Our crucial observation is that locality-sensitive hash (LSH) defined in Section 3.3 can be combined with our fuzzy matching protocol to achieve $\text{negl}(\kappa)$ correctness error and sub-quadratic complexity in the fuzzy PSI setting. The high-level intuition is that we can use LSH to do a “coarse mapping”, and then use a fuzzy matching protocol for “refined filtering”. Usually, LSH will map similar points ($\text{dist} \leq \delta$) into the same bucket with at least $1 - \text{negl}(\kappa)$ probability, and different points ($\text{dist} > c\delta$) into different buckets with a high probability. In fuzzy PSI setting, $c\delta$ is equivalent to the minimal distance among the receiver’s points, e.g., $c = 2$ is the same as the disjoint balls and $c = 4$ reflects the case that points are 4δ apart. There are only false positives, but no false negatives, thus we can then use fuzzy matching to do a refined filtering.

The detailed construction is shown in Figure 10, where we denote as L the number of LSH tables, and T the bucket capacity of each LSH entry. The receiver tries to insert all ball centers into each LSH table. Particularly, when there is a collision during inserting to some entry, the receiver tries to insert the ball into the bucket of the same entry. When parameters are set properly, the insertion will be successful without exceeding the capacity T of each bucket with overwhelming probability. Note that, no matter how many collisions occurred, the total number inserted in each LSH table, as well as in each OKVS, is still the same as the receiver’s set size. Given a sender’s point, an LSH function F_{L_p} would map it to L entries across L tables, and for each entry, there are T possible locations in the bucket that could be the true positive. The sender needs to iterate each location to run the underlying fuzzy matching protocol, thus the computational overhead would be $L \cdot T$ times larger.

Theorem 20 (Correctness). *The protocol presented in Figure 10 is correct with probability $1 - \text{negl}(\lambda)$ if OKVS satisfies the perfect correctness and independence property defined in Section 3.1 and Lemma 1, $H_\lambda : \{0, 1\}^* \mapsto \{0, 1\}^\gamma$, $H_{\kappa'} : \mathbb{G} \mapsto \{0, 1\}^{\kappa'}$ used in GetList_p , GetTuple_p and $H_\Delta : \{0, 1\}^* \mapsto \{0, 1\}^\Delta$ are universal hash functions where $\gamma = \kappa + \log(\delta M N L T)$, $\kappa' = \kappa'' + du$ with $\kappa'' = \kappa + \log(\delta^p M L T)$ and u the bit-length of the coordinates, $\Delta = \kappa + \log(M N L T)$ and \mathcal{F}_{L_p} is a $(\delta, c\delta, \frac{1}{N^p}, \frac{1}{N})$ -LSH family defined in Section 3.3 with $T = O(\frac{\log N}{\log \log N})$ and $L = \frac{\kappa}{\log e} N^\rho$.*

Proof. According to Definition 2, there is a collision for $F_{L_p}^\ell(\mathbf{w}_k)$ and $F_{L_p}^\ell(\mathbf{w}_{k'})$ with probability at most $\frac{1}{N}$. With a similar analysis as maximal-load of balls-into-bins problem, for each $\ell \in [L]$ and some specific LSH bucket $R_{k_*}^\ell$ in ℓ -th table, we have

$$\Pr[\text{Hash to } R_{k_*}^\ell \text{ exactly } t_* \text{ times}] = \binom{N-1}{t_*} \prod_{k=1}^{t_*} p_k \prod_{k=1}^{N-t_*-1} (1-p_k) \leq \frac{1}{t_*!}$$

where $p_k \leq \frac{1}{N}$ for $k \in [N]$ is the probability that each $\mathbf{w}_{k \neq k_*}$ hashes to the same bucket $R_{k_*}^\ell$. It is less than $\frac{1}{N^{x-o(1)}}$ if $t_* = x \log N / \log \log N$. Taking a union bound on each LSH bucket and each $t > t_*$, the probability that there exists an LSH bucket which has $\geq t_*$ collisions is less than $\frac{1}{N^{x-3-o(1)}}$. If we choose $x = 3 + \frac{\kappa}{\log e}$, then the probability is $\text{negl}(\kappa)$ that there exists a bucket in L tables which exceeds the capacity $T = \frac{(3 + \frac{\kappa}{\log e}) \log N}{\log \log N}$.

Then, according to Theorem 1, if sender’s point \mathbf{q} satisfies $\text{dist}(\mathbf{q}, \mathbf{w}) \leq \delta$, there exists at least one list^{ℓ_*} for $\ell_* \in [L]$ which contains the correct key-value pair and the sender can decode them correctly for $\mathbf{E}_1^{\ell_*}, \dots, \mathbf{E}_d^{\ell_*}$. If the sender’s point \mathbf{q} is far away from every \mathbf{w} , then the correctness follows from Theorem 8 directly. \square

Receiver ₁ ($\mathbf{W} \in \mathbb{Z}^{d \times N}$)	Sender ₁ ($\mathbf{Q} \in \mathbb{Z}^{d \times M}, \text{msg}_1$)
Sample $g \leftarrow \mathbb{G}, s \leftarrow \mathbb{Z}_p$ Compute $h = g^s$ Set $\text{list}_i^\ell = \emptyset, \mathcal{L}_\ell = \emptyset$ for $i \in [d], \ell \in [L]$ For each $\ell = 1 \dots L$: For each $k = 1 \dots N$: Set $R_k^\ell \leftarrow F_{L_p}^\ell(\mathbf{w}_k)$ For $r = 1 \dots T$: Set $\Delta_{k,\ell} \leftarrow H_\Delta(R_k^\ell \ r)$ If $(\cdot, \Delta_{k,\ell}) \notin \mathcal{L}_\ell$ Exit For Set $\mathcal{L}_\ell = \mathcal{L}_\ell \cup (k, \Delta_{k,\ell})$ For each $\ell = 1 \dots L$: For each $k = 1 \dots N$: Retrieve $(k, \Delta_{k,\ell})$ from \mathcal{L}_ℓ $\{\text{list}'_i\}_{i \in [d]} \leftarrow \text{GetList}_p(h, s, \mathbf{w}_k, \Delta_{k,\ell})$ Set $\text{list}_i^\ell = \text{list}_i^\ell \cup \text{list}'_i$ Get $E_i^\ell \leftarrow \text{Encode}(\text{list}_i^\ell)$ for $i \in [d]$ Set $\mathbf{E}^\ell = \{E_1^\ell, \dots, E_d^\ell\}$ Output $\text{msg}_1 := (g, h, \{\mathbf{E}^\ell\}_{\ell \in [L]}), \text{st} := s$	Parse $\text{msg}_1 := (g, h, \{\mathbf{E}^\ell\}_{\ell \in [L]})$ Set $\mathcal{M}_k = \emptyset$ for $k \in [M]$ For each $k = 1 \dots M$: For each $\ell = 1 \dots L$: Get $R_k^\ell \leftarrow F_{L_p}^\ell(\mathbf{q}_k)$ For each $r = 1 \dots T$: $\Delta = H_\Delta(R_k^\ell \ r)$ $(f, h, \mathcal{X}) \leftarrow \text{GetTuple}_p(g, h, \mathbf{q}_k, \Delta, \mathbf{E}^\ell)$ For each $x \in \mathcal{X}$, set $x = x \oplus (0^{\kappa''} \ \mathbf{q}_k)$ $\mathcal{M}_k = \mathcal{M}_k \cup (f, h, \mathcal{X})$ Shuffle \mathcal{M}_k Set $\text{msg}_2 := \{\mathcal{M}_1, \dots, \mathcal{M}_M\}$ and shuffle Output msg_2 <hr/> Receiver ₂ (st, msg_2) <hr/> Parse $\text{msg}_2 := \{\mathcal{M}_1, \dots, \mathcal{M}_M\}, \text{st} := s$ Parse each $\mathcal{M}_k := \{(f_j, h_j, \mathcal{X}_j)\}_{j \in [L \cdot T]}$ Set $I = \emptyset$ For each $k = 1 \dots M$: If $\exists (f_*, h_*, \mathcal{X}_*) \in \mathcal{M}_k$ and $x \in \mathcal{X}_*$ s.t. $H_{\kappa'}(f_*^{-s} \cdot h_*) \oplus x_* = 0^{\kappa''} \ \mathbf{q}_k$: Set $I = I \cup \mathbf{q}_k$ Output I

Fig. 10. Fuzzy PSI, L_p distance in high dimension space, using LSH, disjoint balls

Theorem 21 (Security). *The protocol presented in Figure 10 satisfies the fuzzy PSI functionality defined in Figure 3 for L_p distance against semi-honest adversaries if OKVS is oblivious and the DDH assumption holds.*

Proof. The security follows naturally from the underlying fuzzy matching protocol proved in Theorem 9. Specifically, the msg_1 can be simulated by encoding L lists of $N(2\delta + 1)$ random key-value pairs, for each dimension. The indistinguishability comes from the obliviousness of OKVS and the DDH assumption. Now consider the receiver is corrupted and the simulator needs to simulate msg_2 . The simulator invokes $\mathcal{F}_{\text{FUZZYPSI}}$ to obtain the intersection

$$\mathcal{I} := \{\mathbf{q}_i \mid \exists \mathbf{w} \in \mathbf{W}, \text{dist}_p(\mathbf{w}, \mathbf{q}_i) \leq \delta\}.$$

Then, it simulates $(f_j, h_j, \mathcal{X}_j)$ as in the case that $0 \leftarrow \mathcal{F}_{\text{FUZZYMATCH}}$ in Theorem 9, for each $j \in [M \cdot L \cdot T]$. In the end, it samples $|\mathcal{I}|$ random indices $\{I_i \in [M]\}_{i \in [|\mathcal{I}|]}$, and for each I_i , it replaces $\{(f_k, h_k, \mathcal{X}_k)\}_{k \in [I_i, I_i + L \cdot T]}$ to be the tuples corresponding to \mathbf{q}_i in the real game. The indistinguishability can be argued the same as in Theorem 9. \square

Theorem 22 (Complexity). *The protocol presented in Figure 10 has communication complexity $O(2\delta dN^{1+\rho}\lambda + MN^\rho(2\lambda + \delta^p\kappa)\log N)$; The computational complexity is $O(2\delta dN^{1+\rho} + MN^\rho\log N)$ for the receiver and $O((d + \delta^p)MN^\rho\log N)$ for the sender. Also, $\rho \leq \frac{1}{c}$ when receiver's points are c -apart.*

Proof. Just choose $L = O(N^\rho)$ and $T = O(\log N)$ according to Section 3.3 and Theorem 20. \square
As a concrete example, for L_2 distance and disjoint balls (2δ -apart points), we have communication complexity $O(2\kappa\delta dN^{1+\rho}\lambda + \kappa^2MN^\rho\log N(2\lambda + \delta^2\kappa))$. After dropping constant terms, we have a sub-quadratic complexity $O(N^{1+\rho} + MN^\rho\log N)$ where $\rho = 0.365$. Combined with Theorem 4, this complexity can be asymptotically reduced to $O((N + M)\log N)$.

8 Extending to Broader Functionalities

We show above protocols can be extended to a broader class of functionalities, including standard PSI, PSI with sender privacy, labeled PSI, and circuit PSI, with small tweaks and therefore preserving the efficiency. We describe extensions for all protocols in this work except for the L_p distance protocol in high dimensional space since currently, the simulator for a corrupt receiver needs to know the points of the sender that lie in the intersection, i.e., only works for the standard PSI functionality.

8.1 Labeled PSI.

For labeled PSI, the sender has some labels $\text{label}_k \in \{0, 1\}^\sigma$ attached to their input points \mathbf{q}_k , $k \in [M]$, and the receiver wishes to learn the labels of the points for which there exists an $i \in [N]$ such that $\text{dist}(\mathbf{w}_i, \mathbf{q}_k) \leq \delta$ (see Figure 3 for the ideal functionality). It can be realized for the protocol in Figure 7 (and similar for the protocols in Figure 8 and Figure 9 by ignoring the index j in these cases) by letting the sender use $v_{k,j}$ as a one-time pad to encrypt label_k together with a special prefix, e.g., 0^κ , indicating that the label belongs to a valid match. For the protocol in Figure 8 with $p \neq \infty$, the sender instead uses the $x_{k,j} \in \mathcal{X}_k$ as a one-time pad to encrypt $0^\kappa \parallel \text{label}_k$.

More formally, let $\text{label}_k \in \{0, 1\}^\sigma$ denote the sender's labels associated to their inputs \mathbf{q}_k , $k \in [M]$. The protocol in Figure 7 can be adapted to realize labeled PSI (see Figure 3) as follows, and the protocols in Figure 8 with $p = \infty$ and Figure 9 can be adapted analogously by ignoring the index j .

- The sender puts $v_{k,j} := H_{\kappa'+\sigma}(v'_{k,j})$ for each $k \in [M]$ and $j \in [2^d]$, and puts $\text{msg}_2 := \left\{ \left(u_{k,j}, v_{k,j} \oplus (0^{\kappa'} \parallel \text{label}_k) \right)_{k \in [M], j \in [2^d]} \right\}$.
- The receiver parses $\text{msg}_2 := \{(u_{k,j}, C_{k,j})_{k \in [M], j \in [2^d]}\}$ and, for each $k \in [M]$, if there exists $j_* \in [2^d]$ for which $H_{\kappa'+\sigma}(u_{k,j_*}^s) \oplus C_{k,j_*} = 0^{\kappa'} \parallel z_{k,j}$ for some $z_{k,j} \in \{0, 1\}^\sigma$, adds $z_{k,j}$ to their output.

The simulation for a corrupt sender remains identical. In case of a corrupt receiver, the only thing that changes is that the simulator puts $u_{k,j_*} := g^r$, $C_{k,j_*} := H_{\kappa'+\sigma}(h^r) \oplus (0^{\kappa'} \parallel \text{label}_k)$, where $r \leftarrow \mathbb{Z}_p$, $j_* \leftarrow \mathbb{[}2^d]$, if label_k is in the output, and $u_{k,j} := R$, $C_{k,j} := H_{\kappa'+\sigma}(R')$, where $R, R' \leftarrow \mathbb{G}$, otherwise.

The protocol from Figure 8 with $p \neq \infty$ can be adapted similarly by letting the sender use the $x_{k,j}$ as a one-time pad to encrypt their labels. Formally:

- The sender constructs the items in \mathcal{X}_k as $x_{k,j} := H_{\kappa'+\sigma}(g^{a_k+b_k \cdot j}) \oplus (0^{\kappa'} \parallel \text{label}_k)$ for $j \in [\delta^p]$ and each $k \in [M]$, shuffles \mathcal{X}_k and puts $\text{msg}_1 := \{(f_k, h_k, \mathcal{X}_k)_{k \in [M]}\}$.
- The receiver checks for each $k \in [M]$ if there exists $j \in [\delta^p]$ for which $H_{\kappa'+\sigma}(f_k^{-s} \cdot h_k) \oplus x_{k,j} = 0^{\kappa'} \parallel z_{k,j}$ for some $z_{k,j} \in \{0, 1\}^\sigma$, adds $z_{k,j}$ to their output.

The simulation proceeds similar to before, where the simulator for a corrupt receiver now puts $f_k := g^r$, $h_k := h^r \cdot R$, $x_{k,j} := H_{\kappa'+\sigma}(R) \oplus (0^{\kappa'} \parallel \text{label}_k)$ with $r \leftarrow \mathbb{Z}_p$, $R \leftarrow \mathbb{G}$ for a random $j \leftarrow [\delta^p]$ if label_k is in the output, and $x_{k,j} \leftarrow \{0, 1\}^{\kappa'+\sigma}$ otherwise.

8.2 Standard PSI.

By letting the labels be a description of the sender's points, we can realize standard PSI, where the receiver learns the sender's points \mathbf{q}_k for which there exists an $i \in [N]$ such that $\text{dist}(\mathbf{w}_i, \mathbf{q}_k) \leq \delta$ (see Figure 3 for the ideal functionality).

8.3 Standard PSI with Sender Privacy (PSI-SP).

We saw above that standard PSI where the receiver learns the sender's points within distance δ of theirs can be seen as a special case of labeled PSI. Adapting the protocol such that the receiver only learns which of their balls have a non-empty intersection with any of the sender's points, which we refer to as *standard PSI with sender privacy* (PSI-SP) requires a bit more work. The basic idea is to again let the sender use $v_{k,j}$ as a one-time pad, but now to encrypt a value that can be used to identify the receiver's point $\mathbf{w}_{k'}$ in case $\text{dist}(\mathbf{w}_{k'}, \mathbf{q}_k) \leq \delta$. For the protocol in Figure 7 the sender can encrypt $H_{\kappa}(\mathcal{B}_{k,j})$ as an identifier for the receiver's point. For the protocol in Figure 8 the receiver first encodes an OKVS P that maps all cells intersecting $\text{ball}_{\delta}(\mathbf{w}_{k'})$ to fresh encryptions of the same identifier $R_{k'}$, which the sender decodes at $H_{\lambda}(\mathcal{C}_k)$, masks with v_k and sends the result to the receiver together with u_k . The receiver can then decrypt, unmask with u_k^s and compare to $R_{k'}$. For the Minkowski distance protocol, we can reduce the number of decryptions and comparisons the receiver needs to perform at the end of the protocol by letting the sender program an oblivious programmable PRF (OPPRF, see Section 3.4) that sends the points in \mathcal{X}_k to $\text{Decode}(P, H_{\lambda}(\mathcal{C}_k))$.

More formally, we can achieve sender privacy (see Figure 3) for the protocol in Figure 7 as follows:

- The sender puts $\text{msg}_2 := \{(u_{k,j}, v_{k,j} \oplus H_{\kappa'}(\mathcal{B}_{k,j}))_{k \in [M], j \in [2^d]}\}$, where $\mathcal{B}_{k,j}$, $j \in [2^d]$, ranges over all the blocks containing $\text{cell}_{2\delta}(\mathbf{q}_k)$.
- The receiver parses $\text{msg}_2 := \{(u_{k,j}, C_{k,j})_{k \in [M], j \in [2^d]}\}$ and, for each $k \in [M]$, if there exists $j_* \in [2^d]$ and $k' \in [N]$ for which $H_{\kappa'}(u_{k,j_*}^s) \oplus C_{k,j_*} = H_{\kappa'}(\mathcal{B}_{k'})$, adds $\mathbf{w}_{k'}$ to their output.

The simulator for a corrupt sender again remains identical. In case of a corrupt receiver, if $\mathbf{w}_{k'}$ is in the output, the simulator puts $u_{k,j_k} := g^r$, $C_{k,j_k} := H_{\kappa'}(h^r) \oplus H_{\kappa'}(\mathcal{B}_{k'})$, where $k \leftarrow [M]$, $j_k \leftarrow [2^d]$, $r \leftarrow \mathbb{Z}_p$, and otherwise puts $u_{k,j} := R$, $C_{k,j} := H_{\kappa'}(R')$, where $R, R' \leftarrow \mathbb{G}$.

To achieve sender privacy for the other protocols we need to do a bit more work. The receiver encodes encryptions of the identifiers for their points in an OKVS such that the sender does not learn any information if multiple points are close to the same point of the receiver. The sender subsequently homomorphically masks the encrypted identifiers such that the receiver can only unmask them for points in the intersection. Let $\text{PKE} := (\text{Gen}, \text{Enc}, \text{Dec}, \boxplus)$ be an additively homomorphic IND- \mathbb{S} CPA secure public-key encryption scheme as defined in Section 3.5 with plaintext space $\{0, 1\}^{\kappa'}$. Just as in our main protocols, one can also choose to instantiate PKE by ElGamal's cryptosystem, and let the receiver use the same key-pair s, h , but we leave the details of this to future work. Using the subprotocols from Figure 11 we can describe the adaptations as follows:

GetOKVS(pk, \mathbf{R} , $(\Delta^k)_{k \in [N]}$)	HomMask(pk, P , S , Δ)
For each $k = 1, \dots, N$:	$X \leftarrow \text{Enc}_{\text{pk}}(S)$
For each $j = 1, \dots, \Delta^k $:	$Y \leftarrow \text{Decode}(P, \Delta)$
$C_{k,j} \leftarrow \text{Enc}_{\text{pk}}(R_k)$	$Z \leftarrow X \boxplus Y$
list $\leftarrow \text{list} \cup \{(\Delta_j^k, C_{k,j})\}$	Output Z
$P \leftarrow \text{Encode}(\text{list})$	
Output P	

Fig. 11. Subprotocols GetOKVS and HomMask

- **Figure 8** ($p = \infty$): Receiver samples $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$, $\mathbf{R} \leftarrow_{\$} \{0, 1\}^{\kappa \times N}$ and sends pk , $P \leftarrow \text{GetOKVS}(\text{pk}, \mathbf{R}, ((H_\gamma(\mathcal{C}_{k,j}))_{j \in [2^d]}))_{k \in [N]}$ in addition to msg_1 . Sender computes $Z_k \leftarrow \text{HomMask}(\text{pk}, P, v_k, H_\gamma(\mathcal{C}_k))$ for each $k \in [M]$ and sends a shuffled $\text{msg}_2 := \{(u_k, Z_k)_{k \in [M]}\}$. For each $k \in [M]$, if there exists $k' \in [N]$ such that $H_{\kappa'}(u_k^s) \oplus \text{Dec}_{\text{sk}}(Z_k) = R_{k'}$, the receiver adds $\mathbf{w}_{k'}$ to their output.
- **Figure 8** ($p \neq \infty$): Receiver generates pk, P same as above. Sender computes $Z_{k,j} \leftarrow \text{HomMask}(\text{pk}, P, x_{k,j}, H_\gamma(\mathcal{C}_k))$, where $x_{k,j} \in \mathcal{X}_k$, for each $k \in [M]$ and $j \in [\delta^p]$, and sends a shuffled $\text{msg}_2 := \{(f_k, h_k, (Z_{k,j})_{j \in [\delta^p]})_{k \in [M]}\}$. For each $k \in [M]$, $j \in [\delta^p]$, if there exists $k' \in [N]$ such that $H_{\kappa'}(f_k^{-s} \cdot h_k) \oplus \text{Dec}_{\text{sk}}(Z_{k,j}) = R_{k'}$, the receiver adds $\mathbf{w}_{k'}$ to their output.

To simulate a corrupt sender can encode an OKVS P with random key-value pairs, and indistinguishability follows from the IND- $\$$ CPA security of the encryption scheme and the obliviousness of the OKVS. In case of a corrupt receiver, if $\mathbf{w}_{k'}$ is in the output, the simulator can insert the relation being checked by the receiver into a random entry in \mathbf{Z}_k for a random $k \in [M]$, and sample the remaining entries uniformly random. That is, if $p = \infty$ it puts $u_k := g^r$, $Z_k := \text{Enc}_{\text{pk}}(H_{\kappa'}(h^r) \oplus R_{k'})$, where $r \leftarrow_{\$} \mathbb{Z}_p$, for a random $k \leftarrow_{\$} [M]$ if $\mathbf{w}_{k'}$ is in the output, and otherwise puts $u_k := R$, $Z_k := \text{Enc}_{\text{pk}}(R')$ where $R \leftarrow_{\$} \mathbb{G}$ and $R' \leftarrow_{\$} \{0, 1\}^{\kappa'}$. The simulator for $p \neq \infty$ can be adapted similarly.

Finally, the protocol in Figure 9 can be adapted in a similar way as follows:

- For every $k \in [N]$, the receiver samples a random $R_k \leftarrow_{\$} \{0, 1\}^{\kappa'}$ and picks a dimension $i_* \in [d]$ where the projected interval $[w_{k,i_*} - \delta : w_{k,i_*} + \delta]$ is disjoint from $[w_{k',i_*} - \delta : w_{k',i_*} + \delta]$ for all $k' \neq k$. For each $k \in [M]$, the receiver puts $C_{k,i_*,j} \leftarrow \text{Enc}_{\text{pk}}(R_k)$ and $C_{k,i,j} \leftarrow \text{Enc}_{\text{pk}}(0)$ for $i \neq i_*$, using fresh randomness for each $j \in [-\delta : \delta]$, and puts $\text{list}_i := \{(H_\gamma(i, w_{k,i} + j), C_{k,i,j}) : k \in [M], j \in [-\delta : \delta]\}$ padded to size $2\delta \cdot M$ with random key-value pairs. The receiver encodes $P_i \leftarrow \text{Encode}(\text{list}_i)$ and sends $\mathbf{P} := (P_1, \dots, P_n)$ to the sender in addition to msg_1 .
- The sender computes $Y_{k,i} \leftarrow \text{Decode}(P_i, H_\gamma(i, q_{k,i}))$ for each $k \in [M]$ and $i \in [d]$, puts $X_k \leftarrow \text{Enc}_{\text{pk}}(v_k)$ and $Z_k \leftarrow Y_{k,1} \boxplus \dots \boxplus Y_{k,d} \boxplus X_k$ for each $k \in [M]$ and sends a shuffled $\text{msg}_2 := \{(u_k, Z_k)_{k \in [M]}\}$.
- For each $k \in [M]$, if there exists $k' \in [N]$ for which $H_{\kappa'}(u_k^s) \oplus \text{Dec}_{\text{sk}}(Z_k) = R_{k'}$, the receiver adds $\mathbf{w}_{k'}$ to their output.

The simulator can be adapted similarly to before.

8.4 Circuit PSI.

In the circuit PSI setting, we want none of the parties to learn which or even how many points are close to each other. Instead, they only learn secret shares encoding the intersection, which they can use as the input to a secure follow-up computation (see Figure 4 for the ideal functionality). The main idea behind the tweaks needed to achieve this functionality is to perform the comparisons that the receiver does at the end of the protocol inside of a secure computation to let the parties obtain secret shares of bits encoding the intersection. Additionally, the receiver encodes encryptions of their associated values into an OKVS similar to the PSI-SP extension and lets the sender homomorphically mask these with random values such that the parties obtain secret shares of the receiver's associated values for points in the intersection. We can reduce the number of secure comparisons the parties need to perform at the end of the protocol from $\delta^p \cdot M$ to M by letting the sender program an OPRF that maps \mathcal{X}_k to a random a_k and securely comparing this value to the OPRF evaluation at $H_\kappa(f_k^{-s} \cdot h_k)$.

More formally, let $\tilde{w}_k, \tilde{q}_k \in \{0, 1\}^\sigma$ be the associated data for the inputs $\mathbf{w}_k, \mathbf{q}_k$, respectively. Furthermore, let $\text{PKE} := (\text{Gen}, \text{Enc}, \text{Dec}, \boxplus)$ be an additively homomorphic IND- $\$$ CPA secure public key encryption scheme as defined in Section 3.5 with plaintext space $\{0, 1\}^\sigma$. Then the fuzzy PSI protocols can be adapted to realize circuit PSI (see Figure 4) as follows:

- **Figure 7:** Receiver samples $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$ and sends the sender pk together with $P \leftarrow \text{GetOKVS}(\text{pk}, (\tilde{w}_k)_{k \in [N]}, (H_\gamma(\mathcal{B}_k))_{k \in [N]})$ in addition to msg_1 . Sender samples $s_{k,j} \leftarrow_{\$} \{0, 1\}^\sigma$, computes $Z_{k,j} \leftarrow \text{HomMask}(\text{pk}, P, s_{k,j}, H_\gamma(\mathcal{B}_{k,j}))$ for each $k \in [M], j \in [2^d]$, and sends $\text{msg}_2 := \{(u_{k,j}, Z_{k,j})_{k \in [M], j \in [2^d]}\}$. Receiver inputs $y_{k,j} := H_\kappa(u_{k,j}^s)$ and $r_{k,j} := \text{Dec}_{\text{sk}}(Z_{k,j})$ and sender inputs $v_{k,j}, s_{k,j}$ and \tilde{q}_k into a generic MPC functionality that outputs secret shares of $\sum_{j \in [2^d]} (y_{k,j} = v_{k,j}) \cdot (1 \parallel (r_{k,j} \oplus s_{k,j}) \parallel \tilde{q}_k)$ for each $k \in [M]$.
- **Figure 8 ($p = \infty$):** Receiver samples $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$ and sends the sender $\text{pk}, P \leftarrow \text{GetOKVS}(\text{pk}, (\tilde{w}_k)_{k \in [N]}, ((H_\gamma(\mathcal{C}_{k,j}))_{j \in [2^d]})_{k \in [N]})$ in addition to msg_1 . Sender samples $s_k \leftarrow_{\$} \{0, 1\}^\sigma$, computes $Z_k \leftarrow \text{HomMask}(\text{pk}, P, s_k, H_\gamma(\mathcal{C}_k))$ for each $k \in [M]$ and sends $\text{msg}_2 := \{(u_k, Z_k)_{k \in [M]}\}$. Receiver inputs $y_k := H_\kappa(u_k^s)$ and $r_k := \text{Dec}_{\text{sk}}(Z_k)$, sender inputs v_k, s_k and \tilde{q}_k into an MPC functionality that computes secret shares of $(y_k = v_k) \cdot (1 \parallel (r_k \oplus s_k) \parallel \tilde{q}_k)$ for each $k \in [M]$.
- **Figure 8 ($p \neq \infty$):** Receiver generates pk, P identical to above. Sender computes Z_k identical to above and sends $\text{msg}_2 := \{(f_k, h_k, Z_k)_{k \in [M]}\}$ to the receiver. Receiver inputs $y_k := H_{\kappa'}(f_k^{-s} \cdot h_k)$ and $r_k := \text{Dec}_{\text{sk}}(Z_k)$, sender inputs \mathcal{X}_k, s_k and \tilde{q}_k into an MPC functionality that computes secret shares of $\sum_{x_{k,j} \in \mathcal{X}_k} (y_k = x_{k,j}) \cdot (1 \parallel (r_k \oplus s_k) \parallel \tilde{q}_k)$ for each $k \in [M]$.

The simulator for a corrupt sender can encode an OKVS P with random key-value pairs similar to before, and return the sender's output shares as the output of the ideal MPC functionality. In case of a corrupt receiver, the simulator generates msg_2 by sampling the first entries uniformly random, and setting the entries of Z_k to be ciphertexts of uniformly random values. Finally, it returns the receiver's output shares as the output of the ideal MPC functionality.

Finally, the protocol in Figure 9 can be adapted as follows:

- The receiver samples $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda)$. For each $k \in [M]$ and $j = -\delta, \dots, \delta$, generates fresh encryptions $C_{k,i_*,j} \leftarrow \text{Enc}_{\text{pk}}(\tilde{w}_k)$, where $i_* \in [d]$ is randomly sampled from the dimensions where the projected interval $[w_{k,i_*} - \delta : w_{k,i_*} + \delta]$ is disjoint from $[w_{k',i_*} - \delta : w_{k',i_*} + \delta]$ for all $k' \neq k$. For all other $k \in [M], i \neq i_*$ and $j \in [-\delta : \delta]$, generates a fresh $C_{k,i,j} \leftarrow \text{Enc}_{\text{pk}}(0)$. For each $i \in [d]$, the receiver encodes $P_i \leftarrow \text{Encode}(\text{list}_i)$, where list_i is $\{(H_\gamma(i, w_{k,i} + j), C_{k,i,j}) : k \in [M], j \in [-\delta : \delta]\}$ padded to size $2\delta \cdot M$ with random key-value pairs. The receiver sends $\text{pk}, \mathbf{P} := (P_1, \dots, P_d)$ to the sender in addition to msg_1 .

- The sender computes $Y_{k,i} := \text{Decode}(P_i, H_\gamma(i, q_{k,i}))$ for each $k = 1, \dots, M$ and $i = 1, \dots, d$, samples $s_k \leftarrow_{\$} \{0, 1\}^\sigma$, puts $X_k \leftarrow \text{Enc}_{\text{pk}}(s_k)$ and puts $Z_k := Y_{k,1} \boxplus \dots \boxplus Y_{k,d} \boxplus X_k$. The sender puts $\text{msg}_2 := \{(u_k, Z_k)_{k \in [M]}\}$.
- The receiver inputs $y_k := H_{\kappa'}(u_k^s)$ and $r_k := \text{Dec}_{\text{sk}}(Z_k)$, and the sender inputs v_k, s_k and \tilde{q}_k into a generic MPC functionality that computes secret shares of $(y_k = v_k) \cdot (1 \parallel (r_k \oplus s_k) \parallel \tilde{q}_k)$ for each $k \in [M]$.

The simulator can be adapted similarly to before.

8.5 Reducing Comparisons for PSI-SP and Circuit PSI.

Using an oblivious programmable PRF (OPPRF) as defined in Section 3.4, we can reduce the number of comparisons the receiver needs to perform for the PSI-SP extensions of the Minkowski distance protocols. The main idea (in terms of the protocol in Figure 8 with $p \neq \infty$) is to let the sender, in addition to sending (f_k, h_k) , program an OPPRF F_k that maps the points in \mathcal{X}_k to a randomized encryption of $\text{Decode}(P, H_\gamma(C_k))$ and to let the receiver evaluate the OPPRF at $H_{\kappa'}(f_k^{-s} \cdot h_k)$. If the decryption of this evaluation equals $R_{k'}$ for some $k' \in [N]$, the receiver adds $\mathbf{w}_{k'}$ to their output. This reduces the number of comparisons from $\delta^p \cdot M$ to M , assuming that $\{(R_{k'})_{k' \in [N]}\}$ has constant lookup time.

Similarly, we can reduce the number of secure comparisons the sender and receiver need to perform for the circuit PSI extensions of the Minkowski distance protocols. For the protocol in Figure 8, in addition to sending the (f_k, h_k, Z_k) , the sender can program an OPPRF F_k that maps the points in \mathcal{X}_k to a random a_k and to let the receiver evaluate the OPPRF at $H_{\kappa'}(f_k^{-s} \cdot h_k)$ to obtain b_k . Now the MPC functionality just needs to compute secret shares of $(a_k = b_k) \cdot (1 \parallel (r_k \oplus s_k) \parallel \tilde{q}_k)$ for each $k \in [M]$. This reduces the number of secure comparisons needed from $\delta^p \cdot M$ to M .

9 Performance Evaluation

In this section, we provide a micro-benchmark for our fuzzy PSI protocols for $L_{p \in \{1, 2, \infty\}}$ in low-dimension settings.

Implementation. We implement the standard fuzzy PSI variant (i.e., the receiver learns the sender’s points in the intersection) in three different metrics (L_∞, L_1, L_2) in a d -dimension space where $d = \{2, 3, 5, 10\}$, following the Figure 7, and Figure 8. The proof-of-concept implementation¹⁰ is written in Rust, with less than 1000 lines of code. We use Ristretto and curve25519-dalek to instantiate the underlying group \mathbb{G} , use FxHash and Blake3 to instantiate the hash function $H_\gamma, H_{\kappa'}$. We choose the security parameter $\lambda = 128$ and statistical parameter $\kappa = 40$ as usual. To instantiate the OKVS, we follow the construction from [BPSY23] but working in \mathbb{F}_p and the expansion rate $\epsilon = 0.5$ to make sure we have $2^{-\kappa}$ correctness error rate. Though it can be optimized to $\epsilon = 0.1 \sim 0.25$ to have a more compact size, the encoding and decoding time would also increase accordingly.

Environment. We run the experiments on an ordinary laptop over a single thread: Macbook Air (M1 2020) with 8GB RAM and a 2.1 GHz CPU, without using SIMD (e.g., AVX, NEON) optimizations. We measure the entire protocol time in a local network setting (i.e., LAN-like) without considering latency.

¹⁰ The open-sourced repository: https://github.com/sihangpu/fuzzy_PSI

Table 2. Fuzzy PSI when points are $> 2\delta$ (i.e., disjoint balls)

Metric	Radius δ	Dimension d	Receiver’s N	Sender’s M	Bandwidth	Total Time
L_∞ [GRS22]	30	2	2^{11}	2^{20}	≈ 9865 ² MB	$\gg 1500$ ¹ s
L_∞	30	2	2^{11}	2^{20}	173 MB	257.25 s
L_∞	30	5	2^{13}	2^{11}	231 MB	177.18 s
L_∞	1000	2	2^{11}	2^{11}	753 MB	303.59 s

¹ Estimated by assuming each PRG evaluation is about 4.8 nanoseconds and hash evaluation is about 4.8 nanoseconds/byte [II23]. Only consider the computational costs at the receiver’s side.

² Estimated by the concrete bFSS size provided in [GRS22].

9.1 Concrete Performance

Fuzzy PSI. We mainly consider three cases for fuzzy PSI protocols: The receiver’s points are 2δ -apart (shown in Table 2), and $2\delta(d^{\frac{1}{p}} + 1)$ -apart (shown in Table 3). It is worth noting that any distribution of the receiver’s points can be reduced to the disjoint setting by varying the radius. Specifically, for the L_∞ metric, the second case degrades to 4δ -apart points; For the $L_{\{1,2\}}$ metric, our protocol only supports the second case. Our protocols can support large-volume balls since our computation and communication cost scaled only sublinearly to the total volume.

For comparison, we estimate the concrete communication cost for [GRS22] based on their concrete bFSS size table reported in the paper. For the setting of disjoint balls, we use the reported share sizes for their `spatial hash ◦ sum ◦ tensor ◦ ggm` (0.5, 1)-bFSS, assume bFSS evaluation to cost $(2 \log \delta)^d$ PRG calls, estimate PRG calls to take 10 machine cycles using AES-NI, and put $\ell = 440$. For the distance $> 4\delta$ setting we use the reported share sizes for their `spatial hash ◦ concat ◦ tt` $(1 - 1/2^d, d)$ -bFSS, assume bFSS evaluation to cost 1 machine cycle and put $\ell = 162$ for dimension $d = 5$, $\ell = 139$ for dimension $d = 10$. In all settings we estimate the correlation-robust hash calls at the end of the protocol to take around 10 machine cycles/byte, based on the fastest performance reported in [II23] on 64-byte inputs. We assumed a universe size of 32-bit integers for each dimension. Note that here we report the most conservative estimates for their running time, which can only be considered as a loose lower bound.

PSI with Structured Sets. We also explore the setting that both receiver and sender hold a structured set, and their bandwidth and computation time should only depend on the succinct description of the set. Specifically, we consider each one to hold a union of balls. The centers of the balls can efficiently represent the set. For instance, if we work in a two-dimension space with L_∞ metric, and each party holds 4δ -apart 2^{11} balls with radius $\delta = 64$, then it reflects the classical PSI setting with 32 million points from each side. As comparison, prior works [GRS22] or trivial PSI [KKRT16] requires from 156 to 1184 MB bandwidth when receiver’s set contains 10 million points and sender’s set contains only 1.2 million points.

10 Conclusion

In this work, we explored the fuzzy PSI in a more general setting, including higher dimensional space, comprehensive L_p distance metric, and extended functionality variants. We also demonstrate the

Table 3. Fuzzy PSI when points are $> 2\delta(d^{\frac{1}{p}} + 1)$

Metric	Radius δ	Dimension d	Receiver's N	Sender's M	Bandwidth	Total Time
L_∞ [GRS22]	10	5	2^{11}	2^{20}	-	$\gg 4300$ ¹ s
L_∞ [GRS22]	30	10	2^5	2^{20}	$\approx 10^{11}$ ² MB	$\gg 10^{13}$ ¹ s
L_∞	30	2	2^{11}	2^{20}	134 MB	99.28 s
L_∞	10	5	2^{11}	2^{20}	1240 MB	432.42 s
L_∞	30	10	2^5	2^{20}	1844 MB	1135.63 s
L_1	10	2	2^{11}	2^{20}	107 MB	94.10 s
L_1	30	2	2^{11}	2^{20}	369 MB	111.07 s
L_2	10	2	2^{11}	2^{20}	467 MB	97.37 s
L_2	30	2	2^{11}	2^{20}	3727 MB	121.21 s

¹ Estimated by assuming each PRG evaluation is about 4.8 nanoseconds and hash evaluation is about 4.8 nanoseconds/byte [II23]. Only consider the computational costs at the receiver's side.

² Estimated by the concrete bFSS size provided in [GRS22].

Table 4. PSI where both parties hold a structured set

Metric	Radius δ	Dimension d	Receiver's N	Sender's M	Bandwidth	Total Time
[KKRT16]	1	2	2^{25}	2^{25}	≈ 4325 MB	-
L_∞ (2^{25} points)	64	2	2^{11}	2^{11}	97 MB	39.36 s
L_∞ (2^{46} points)	64	5	2^{11}	2^{11}	241 MB	97.48 s

practicality of our protocols by experimental results. However, there are still many open problems to be solved, such as, our L_p protocols have an additional $O(\delta^p)$ communication overhead for each sender's point which might be expensive when δ or p is too large. Another interesting problem to think is how to get a more efficient protocol in polynomially large dimension space for L_2 distance, or if we can weaken the separated assumption further for L_∞ distance? We leave them as well as the concrete efficiency optimization to future works. Also, current fuzzy PSI protocols with negligible correctness error require disjoint balls at least. What if the receiver's balls are intersected? Any non-trivial approaches without quadratic overhead would be interesting to explore.

References

- ABD⁺21. Navid Alapati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 94–125, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.
- AI06. Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual Symposium on Foundations of Computer Science*, pages 459–468, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press.

- BBM⁺21. Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The apple psi system, 2021.
- BDP21. Pedro Branco, Nico Döttling, and Sihang Pu. Multiparty cardinality testing for threshold private intersection. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 32–60, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany.
- BGJP23. James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 35–66, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany.
- BMRR21. Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multiparty threshold private set intersection with sublinear communication. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 349–379, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany.
- 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 Symposium*, pages 301–318. USENIX Association, 2023.
- CDD⁺15. Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 313–336, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
- CFR23. Anrin Chakraborti, Giulia Fanti, and Michael K. Reiter. Distance-aware private set intersection. In *USENIX Security Symposium*. USENIX Association, 2023.
- CGN98. Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. *Cryptology ePrint Archive*, Report 1998/003, 1998. <https://eprint.iacr.org/1998/003>.
- CH08. Lukasz Chmielewski and Jaap-Henk Hoepman. Fuzzy private matching (extended abstract). In *ARES*, pages 327–334. IEEE Computer Society, 2008.
- CHLR18. Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1223–1237, Toronto, ON, Canada, October 15–19, 2018. ACM Press.
- CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.
- CLR17. Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1243–1255, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.
- CM20. Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 34–63, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.
- CMdG⁺21. Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 1135–1150, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.
- DCW13. Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung,

- editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 789–800, Berlin, Germany, November 4–8, 2013. ACM Press.
- DHP⁺18. Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. Fuzzy password-authenticated key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 393–424, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.
- DIIM04. Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery.
- DPT20. Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated PSI cardinality with applications to contact tracing. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 870–899, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.
- ElG85. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
- FNP04. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.
- GPR⁺21. Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 395–425, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.
- GRS22. Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 323–352, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.
- GRS23. Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Malicious secure, structure-aware private set intersection. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 577–610, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.
- GS19. Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 3–29, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.
- GS23. Satrajit Ghosh and Mark Simkin. Threshold private set intersection with better communication complexity. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 251–272, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.
- HEK12. Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *ISOC Network and Distributed System Security Symposium – NDSS 2012*, San Diego, CA, USA, February 5–8, 2012. The Internet Society.
- II23. ECRYPT II. ebacs encrypt benchmarking of cryptographic systems, 2023. <https://bench.cr.yp.to/results-sha3>.
- 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 *EuroS&P*, pages 370–389. IEEE, 2020.

- IM98. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing*, pages 604–613, Dallas, TX, USA, May 23–26, 1998. ACM Press.
- IW06. Piotr Indyk and David P. Woodruff. Polylogarithmic private approximations and efficient matching. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 245–264, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.
- KKRT16. Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 818–829, Vienna, Austria, October 24–28, 2016. ACM Press.
- KMP⁺17. Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1257–1272, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.
- Lin16. Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. <https://eprint.iacr.org/2016/046>.
- Mea86. Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *S&P*, pages 134–137. IEEE Computer Society, 1986.
- Muf15. Alec Muffett. Facebook: Password hashing & authentication, 2015.
- NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.
- PIB⁺22. Bijeeta Pal, Mazharul Islam, Marina Sanusi Bohuk, Nick Sullivan, Luke Valenta, Tara Whalen, Christopher A. Wood, Thomas Ristenpart, and Rahul Chatterjee. Might I get pwned: A second generation compromised credential checking service. In *USENIX Security Symposium*, pages 1831–1848. USENIX Association, 2022.
- PSTY19. Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 122–153, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.
- RR22. Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 2505–2517, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.
- RS21. Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 901–930, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany.
- UCK⁺21. Erkam Uzun, Simon P. Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy labeled private set intersection with applications to private real-time biometric search. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 911–928. USENIX Association, August 11–13, 2021.
- YSPW10. Qingsong Ye, Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. Efficient fuzzy matching and intersection on private datasets. In Donghoon Lee and Seokhie Hong, editors, *ICISC 09: 12th International Conference on Information Security and Cryptology*, volume 5984 of *Lecture Notes in Computer Science*, pages 211–228, Seoul, Korea, December 2–4, 2010. Springer, Heidelberg, Germany.