


Fuzzy Private Set Intersection from VOLE

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

¹ Aarhus University, Cryptography and Cyber Security Group, Aarhus, Denmark

² CNRS, Research Institute on the Foundations of Computer Science (IRIF), Paris, France

Abstract. Private set intersection (PSI) is a well-researched cryptographic primitive that allows two parties to compute the intersection of their input sets without revealing any information about items outside of the intersection. Fuzzy private set intersection is a relatively new variant of PSI, where items are not matched exactly but “fuzzily”. Most commonly, items are points \mathbf{q}, \mathbf{w} in d -dimensional integer space \mathbb{Z}^d and a point is a fuzzy match to another if it lies within a ball of radius δ centered at this point, with respect to some distance metric.


Previous works either only support infinity (L_∞) distance metric and standard PSI functionality, or support general Minkowski (L_p , $p \in [1, \infty]$) distance metrics and realize richer functionalities but rely on expensive homomorphic encryptions. Our work aims to bridge this gap by giving the first construction of a fuzzy PSI protocol for general Minkowski distance metrics relying on significantly cheaper operations during the online phase.

Our main building block is a novel fuzzy matching protocol based on an oblivious pseudo-random function (OPRF), which can be realized very efficiently from vector oblivious linear evaluation (VOLE). Our protocol is able to *preserve* the asymptotic complexity as well as the simplicity of the fuzzy matching protocol from van Baarsen and Pu (Eurocrypt ’24), while being much more concretely efficient. Additionally, we achieve several asymptotic improvements by representing intervals succinctly. Finally, we present the first fuzzy PSI protocol for infinity distance that places *no assumptions* on the sets of points, while maintaining asymptotic complexities comparable to the state-of-the-art fuzzy PSI protocol.

^{*} This research was supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme under grant agreement number 101124977 (DECRYPSIS)

^{**} Sihang Pu gratefully acknowledges the support of the French Agence Nationale de la Recherche (ANR) through grant ANR-20-CE39-0001 (project SCENE). This work is also supported by the ERC grant OBELiSC (101115790).

Table of Contents

| | |
|--|----|
| Fuzzy Private Set Intersection from VOLE | 1 |
| <i>Aron van Baarsen and Sihang Pu</i>  aronvanbaarsen@gmail.com , sihang.pu@gmail.com | |
| 1 Introduction | 3 |
| 1.1 Related Work | 3 |
| 1.2 Our Contributions | 5 |
| 2 Technical Overview | 6 |
| 2.1 Fuzzy Matching from AHE | 6 |
| 2.2 VOLE-based Fuzzy Matching | 7 |
| 2.3 Prefix Trie Optimizations | 8 |
| 2.4 Towards Fuzzy PSI | 9 |
| 2.5 Arbitrarily Distributed Points | 10 |
| 2.6 Minkowski Distance | 10 |
| 3 Preliminaries | 11 |
| 3.1 Notation | 11 |
| 3.2 Oblivious Key-Value Store | 11 |
| 3.3 Definition of Fuzzy Matching | 12 |
| 3.4 Definition of Fuzzy Private Set Intersection | 13 |
| 3.5 Spatial Hashing Techniques | 13 |
| 3.6 Standard PSI Functionalities | 14 |
| 4 Weak Labeled PSI | 14 |
| 5 Prefix Trie | 15 |
| 6 Multi-Batch $O(P)$ PRF | 18 |
| 6.1 Multi-Batch OPPRF | 20 |
| 7 Fuzzy Matching from OPPRF | 23 |
| 7.1 Optimization from Prefix Trie Techniques | 25 |
| 8 Fuzzy PSI from OPPRF | 27 |
| 8.1 Minkowski Distance | 27 |
| 8.2 Sender Privacy | 29 |
| 8.3 Optimization from Prefix Trie Techniques | 32 |
| 9 Fuzzy PSI for High Dimensions | 33 |
| 10 Fuzzy PSI with Extended Functionalities | 35 |
| 11 Fuzzy PSI for Arbitrary Distribution | 36 |
| 11.1 Infinity Distance | 37 |
| 11.2 Minkowski Distance | 38 |

1 Introduction

Private set intersection (PSI) has been studied for several decades since [33]. It allows two parties to jointly compute the intersection $X \cap Y$ of their respective sets X and Y without revealing any additional information. PSI is often viewed as a natural application of oblivious pseudorandom functions (OPRF), where one party learns only the PRF evaluations on its inputs, and the other party learns the PRF key. Many modern PSI constructions are based on different implementations of OPRF, such as using oblivious transfer extension [29,31], or vector oblivious linear evaluation (VOLE) [38,36].

In this work, we focus on a variant called fuzzy PSI, which allows approximate matches between set elements (or points³). Specifically, given two sets X, Y of size N and M , held by a receiver and a sender, respectively, the receiver learns which elements in X are close to the sender’s elements. Here, closeness is measured by a metric like infinity or Euclidean distance, and the maximum allowable distance δ between two matched points also defines the radius of the ball surrounding each point, which is additionally parametrized by the dimension d of the space.

Fuzzy PSI can be seen as a generalization of standard PSI, as it allows a broader matching function, rather than a simple equality test as in standard PSI. This generalization is particularly useful in scenarios where elements are inherently “noisy”, such as biometric samples [3,30] where users want to compare its biometric readings (e.g., iris scans, fingerprints, or facial patterns) with samples recorded at a server, or for geographic locations [23,20], where passengers want to know whether there are available Ubers around. This latter application is also known as privacy-preserving ride sharing. Although fuzzy PSI is closely related to PSI, it typically requires more complicated constructions and follows different methodologies, which we outline below.

1.1 Related Work

Naive Enumeration. The most natural solution is enumerating every point contained in the δ -radius ball surrounding each element in the set, followed by a standard PSI on these expanded sets. This folklore approach utilizes standard PSI in a black-box way and therefore could benefit from any future improvement on PSI constructions. The downside is that it is only suitable for scenarios where the volume of the ball surrounding each element is small enough, because, for instance, in d -dimensional space, a δ -radius ball has a volume roughly $O(\delta^d)$.

Generic Approaches. A common approach involves using garbled circuits (GC) [43,4] or fully homomorphic encryption (FHE⁴) [25] to implement a “fuzzy matching” circuit [18] for each pair of elements, as shown in [27,41,40]. A fuzzy matching circuit takes two points as input and outputs a bit indicating whether the points are within a certain distance threshold δ . By leveraging GC (or FHE), these protocols can handle elements with polynomially large dimensions and therefore exponentially large balls.

However, a limitation of this approach is that it becomes inefficient when dealing with large sets. The pairwise comparison between elements from two sets results in a quadratic overhead in terms of set size, $O(N \cdot M)$. This is particularly expensive for communication (in the case of GC) or computation (in the case of FHE) when the sizes of the sets are, for instance, several millions.

Concurrent work [37] circumvents this quadratic overhead by encoding the messages from the GC-based fuzzy matching protocols in an oblivious key-value store (OKVS) which maps a

³ We use elements and points interchangeably in fuzzy PSI scenarios.

⁴ There is another line work use predicate encryption in fuzzy PSI for Hamming distance in a concurrent work [6], however, they still share the similar shortcoming with quadratic overhead.

hash identifier for each of the parties’ points to the corresponding next subprotocol message. Instantiating this hash function similarly to the spatial hashing used in [23,24,1,20], they achieve $O(M2^{d-s} + N2^s)d\log\delta$ communication complexity, where $s \in [0, d]$ is an integer that can be chosen freely. In case $s = 0$ or $s = d$, their “Disjoint hash” assumption is identical to the “Apart” assumption we use for our protocol in Section 8.1. For intermediate values of s , their hashing technique can actually also be applied to our construction to achieve a similar complexity, but we leave the details to future work.

Another very recent work [14] use secret-sharing based techniques and focus on Hamming distance. They handle other distance functions including L_2 distance by using Johnson-Lindenstrauss-style embeddings, but depend on a “gap” assumption which requires the parties’ points to either be close or far apart across two sets.

Function Secret Sharing (FSS). Recent works [23,24] have introduced “structure-aware” PSI protocols that can be adapted to achieve fuzzy PSI as well. These methods utilize specialized FSS [9,10] to implement secure membership tests within a δ -radius ball, enabling more efficient fuzzy PSI. Compared to naive enumeration, these protocols significantly reduce communication complexity from $O(\delta^d)$ to $O((2\log\delta)^d)$. In a very recent work [20], the authors optimized computational complexity to the same level, reducing it from $O(\delta^d)$ to $O((2\log\delta)^d)$.

These approaches primarily rely on symmetric cryptographic operations, making them concretely efficient for practical use in low-dimensional settings. However, achieving the $O((N + M) \cdot (2\log\delta)^d)$ complexity typically requires additional assumptions, such as ensuring that the points in the set are not too close to each other. If these assumptions are violated, the complexity would degrade to $O(N \cdot M \cdot u^d)$ for both communication and computation, as reported in [20], where 2^u represents the size of the universe for each element. Another drawback of FSS is we currently do not have any efficient instantiation suitable for generalized metrics (e.g., Euclidean distance), apart from infinity distance ⁵.

Additive Homomorphic Encryption (AHE). The first construction for generalized metrics supporting $L_{p \in [1, \infty]}$ distances was introduced in a recent work [1] using AHE, such as ElGamal-based encryption [17]. Leveraging the linear homomorphism of the encryption scheme, this approach can compute the p -powered L_p distance and securely compare it with a threshold to implement a fuzzy matching. This method effectively handles polynomially large dimensions and radii while avoiding the quadratic overhead appearing in generic solutions. It achieves complexities of $O(2^d M + \delta d N)$ for low dimensions or $O(N d^2 \delta^2 + M)$ for higher dimensions under various mild assumptions on the sets.

This construction is particularly suited for high-dimensional scenarios with large sets. While it offers better asymptotic complexity than FSS-based approaches, its computational cost is dominated by expensive public-key operations, making it less efficient in low-dimensional settings. A recent follow-up work [19] reduced the cost from $O(N d^2 \delta^2 + M)$ to $O((N + M)d\delta)$ in high-dimensional cases by introducing more interaction rounds, but it requires stronger assumptions (which does *not* hold in low-dimensional settings) and still relies on additive homomorphic encryptions. More precisely, they assume for both parties points that for each point there exists a dimension on which its projection is at least distance 2δ from all other points. When considering the private ride-sharing application [23], this assumption does not allow cars to be on parallel streets on both dimensions, which is not realistic in practice.

⁵ It’s important to note that in this work, we focus solely on protocols with negligible correctness errors and do not consider metric embedding or similar techniques. Please refer to [41,12] for constructions with non-negligible errors.

Table 1. Asymptotic complexities of fuzzy PSI protocols, where a receiver and a sender hold sets of size N and M , respectively. Each point is sampled from $\mathbb{Z}_{2^u}^d$. Denote κ as the security parameter, λ as the statistical parameter, and δ as the maximum allowable distance between two points, $s \in [0, d]$ is an integer. “*Arbitrary*” means points are sampled arbitrarily; “*Limited*” means points are mostly 2δ -apart, but could be closer for limited points; “*Apart*” means points are either $2\delta d^{\frac{1}{p}}$ - or $2\delta(d^{\frac{1}{p}} + 1)$ -apart (degrading to 2δ and 4δ for L_∞ distances); “*Locally Separated*” means that for each point there exists *at least one* dimension on which its projection is 2δ -apart from all other points; “*Globally Separated*” means that for each point its projection on *each* dimensions is 2δ -apart from all other points.

| Protocol | Communication | Computation |
|--|--|--|
| [1, Apart, L_∞] | $O(\kappa\delta dN + (\kappa + \lambda)2^d M)$ | $O(\delta dN + d2^d M)$ [#] |
| [1, Apart, L_p] | $O(\kappa\delta d2^d N + (\kappa + \lambda\delta^p)M)$ | $O(\delta d2^d N + dM + M\delta^p)$ [#] |
| [19, Locally Separated, L_∞] | $O(\kappa\delta dN + \kappa\delta dM)$ | $O(\delta dN + \delta dM)$ [#] |
| [19, Locally Separated, L_p] | $O(\kappa\delta dN + \kappa M(\delta d + p \log \delta))$ | $O(\delta dN + M(\delta d + p \log \delta))$ [#] |
| [20, Arbitrary, L_∞] | $O(\kappa^2 uNd + \lambda NMu^d)$ | $O(NM(u^d + \kappa du))$ |
| [20, Limited, L_∞] | $O(\kappa^2 Nd \log \delta + \lambda \cdot M(2 \log \delta)^d)$ | $O(N(\log \delta)^d + M2^d((\log \delta)^d + \kappa d \log \delta))$ |
| [37, Apart, L_∞] | $O(\kappa(M2^{d-s} + N2^s)d \log \delta)$ | $O((M2^{d-s} + N2^s)d \log \delta)$ |
| [37, Apart, L_1] | $O(\kappa(M2^{d-s} + N2^s)d \log(d\delta))$ | $O((M2^{d-s} + N2^s)d \log(d\delta))$ |
| [37, Apart, L_2] | $O(\kappa M2^{d-s}d \log(d\delta) + \kappa N2^s(d \log(d\delta) + \log(d\delta)^3))$ | $O((M2^{d-s} + N2^s) \cdot (d \log(d\delta) + \log(d\delta)^3))$ |
| Ours , Apart [†] , L_∞ | $O(\lambda \cdot \min(N, M) \cdot d2^d + \lambda d\delta \cdot \max(N, M))$ | $O(\min(N, M) \cdot d2^d + d\delta \cdot \max(N, M))$ |
| Ours , Apart [†] , L_∞ | $O(\lambda(M2^d + N)d \log \delta)$ | $O(N(\log \delta)^{d/2} + 2^d M d \log \delta)$ |
| Ours , Apart [†] , L_p | $O(\lambda N2^d(d + p \log \delta) + \lambda M(d\delta + p \log \delta))$ | $O(N2^d(d + p \log \delta) + M(d\delta + p \log \delta))$ |
| Ours , Arbitrary, L_∞ | $O(\lambda(M + N)(\log \delta)^d)$ | $O((M + N)(\log \delta)^d)$ |
| Ours , Globally Separated [†] , L_∞ | $O(\lambda dN + \lambda d\delta M)$ | $O(dN + \delta dM)$ |
| Ours , Globally Separated [†] , L_p | $O(\lambda N(d + p \log \delta) + \lambda M(d\delta + p \log \delta))$ | $O(N(d + p \log \delta) + M(d\delta + p \log \delta))$ |

[#] The computational complexity is primarily dominated by costly **public-key** operations.

[†] Requires the assumption to hold on *both* sets.

1.2 Our Contributions

We summarize our main contributions below and provide a table comparing the asymptotic complexities with previous works in Table 1. Additionally, a comparison of concrete communication costs is presented in Table 2.

New Fuzzy Matching from VOLE. Our key insight is that the linear homomorphism used in [1] is not necessary to achieve fuzzy matching. We propose a fuzzy matching protocol based on OPRF, as described in Section 7. This protocol retains the *same asymptotic* complexities as [1] but only relies on super cheap operations (dominated by hash evaluations and XORs during online phase), thanks to recent advancements in VOLE [38]. Prior works either have strictly worse asymptotic complexities [20], or rely on homomorphic encryptions [1, 19].

Moreover, FSS-based protocols [23, 24, 20] must repeat $O(\kappa)$ times to get negligible correctness error, where κ is the security parameter, resulting in an $O(\kappa^2)$ factor in communication overhead. In contrast, our communication costs scale linearly with κ , or are dependent only on the statistical parameter λ (when using subfield VOLE).

Improved Prefix Trie Optimizations. Inspired by the idea from [1, Remark 2], we explore the potential of applying prefix trie techniques from [12]. The algorithm in [12] requires $O(\delta)$ time to build a prefix trie for an interval of length $O(\delta)$. In contrast, we present a new algorithm in Section 5 that reduces the time complexity to $O(\log \delta)$ which is important in our applications. Moreover, by

applying the “meet-in-the-middle” trick at the receiver’s side, we further decrease the receiver’s computational complexity from $(\log \delta)^d$ to $(\log \delta)^{d/2}$.

First Protocol Supporting Arbitrary Distributions. In Section 11, we introduce a fuzzy PSI protocol for infinity distance in the most generalized setting, i.e., allowing *arbitrarily distributed* points. It was even conjectured “necessary” to place assumptions on datasets to get non-trivial complexities [19, Remark 1] since previous works only get trivial complexities: either incur a quadratic overhead of set size, $O(NM)$, or scale linearly with the volume of the balls, $O((N+M) \cdot \delta^d)$. Instead, our protocol achieves a complexity of $O((N+M) \cdot (\log \delta)^d)$ for both communication and computation⁶.

Sender Privacy for Free. In Section 8, we extend our fuzzy matching protocol to fuzzy PSI in low-dimensional settings, where we assume that the 2δ - or 4δ -apart assumptions hold for both parties’ sets. Particularly, our approach supports sender privacy (i.e., it does not leak the sender’s *exact* point as defined in Section 3.4) *without additional cost*, which is difficult to achieve in FSS-based methods as they typically require more advanced primitives like PSI cardinality or labeled PSI (only reveal labels), which lack symmetric key-based constructions.

High Dimensions and Extended Functionalities. In Section 9, we explore the potential for efficient fuzzy PSI protocols in high-dimensional spaces. However, achieving this requires a rather strong assumption that the points in the sets are *globally disjoint*, as our new fuzzy matching protocol is inherently non-reusable. Under this assumption, for the first time, we can eliminate the 2^d factor from *both* communication and computational complexities for generalized L_p distances. Furthermore, in Section 10, we extend fuzzy PSI to support more advanced functionalities, such as labeled PSI, PSI with cardinality, and circuit PSI.

2 Technical Overview

2.1 Fuzzy Matching from AHE

We briefly revisit the fuzzy matching protocol from [1] before moving on to our contributions. Recall that fuzzy matching targets the setting where a receiver holds a point $\mathbf{w} \in \mathbb{Z}^d$, a sender holds a point $\mathbf{q} \in \mathbb{Z}^d$, and the receiver learns whether $\text{dist}(\mathbf{w}, \mathbf{q}) \leq \delta$ with respect to some distance function dist and some maximum distance δ . The simplest case is the infinity distance L_∞ , defined as $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) := \max_{i \in [d]} |w_i - q_i|$. Their main building block can be seen as a set membership protocol based on additively homomorphic encryption (AHE). The fuzzy matching protocol works as follows:

- For each dimension $i \in [d]$, the receiver encodes an oblivious key-value store (OKVS) E_i with a list of key-value pairs $\{(w_i + j, c_{i,j})\}_{j \in [-\delta, +\delta]}$ where $c_{i,j} \leftarrow \text{Enc}(\text{pk}, 0)$ is fresh encryption of zero under an AHE scheme. The receiver sends over $\mathbf{E} := (E_1, \dots, E_d)$ together with pk .
- The sender decodes E_i at q_i for each $i \in [d]$ to obtain $c_i \leftarrow \text{Decode}(E_i, q_i)$, homomorphically combines these as $c \leftarrow c_1 + \dots + c_d$, and re-randomizes the result as $c' \leftarrow \text{Enc}(\text{pk}, 0) + c \cdot r$ before sending it back to the sender.
- The receiver outputs 1 if $\text{Dec}(\text{sk}, c') = 0$, and 0 otherwise.

⁶ To clarify, in [20, Section 5.2], the authors claim that their spatial hashing supports overlapping balls, but it works only in the *best-case* (i.e., the overlapping does not affect encoding an OKVS) and far from the average-case, whereas our protocol works in the *worst-case*.

Table 2. Concrete communication/computational costs of fuzzy PSI protocols for infinity distance L_∞ , where parameters and assumptions are defined similarly in Table 1, with $\lambda = 40, \kappa = 128$. We choose $N = M = 2^{16}, \delta = 32$, and note that δ is radius as in most of prior works, instead of diameter denoted in [20, Tab. 4]. We estimate the numbers for [20] by setting $u = 32, \ell_{OT} = 318$, mini-universe with *radius* of 2δ , and we report the numbers for [19, 37] from their papers. For runtime estimation, we simplify calculations by counting only hash calls and OKVS costs since XOR costs are typically negligible, and do not take the preprocessing phase of generating the VOLE correlation into account. We assume hashing applies AES-NI (2.2 ns/call on a 4.5 GHz CPU [36]), while OKVS encoding and decoding values (143.7 ns/item and 55.4 ns/item) are taken from [36, Tab. 1]. The standard PSI subprotocol are estimated according to [36, Tab. 2, PSI-fast].

| Protocols | Estimated Bandwidth (GB) | | | Estimated Runtime (Sec) | | |
|------------------------------|--------------------------|-------------|--------------|-------------------------|--------------|----------------|
| | $d = 2$ | $d = 6$ | $d = 10$ | $d = 2$ | $d = 6$ | $d = 10$ |
| [20, Limited] | 6.07 | 18446.91 | $> 10^9$ | 1.78 | 2460.94 | $> 10^8$ |
| [19, Separated] [†] | 5.35 | 15.97 | 26.59 | ~ 2218 | ~ 6366 | ~ 10779 |
| [37, Apart] | 0.44 | 5.31 | 35.39 | — | — | — [#] |
| Section 11, Arbitrary | 0.15 | 640.00 | $> 10^6$ | 1.75 | 7184.28 | $> 10^7$ |
| Section 8, Apart | 0.09 | 0.77 | 14.20 | 1.82 | 12.58 | 182.18 |

[†] We stress that the “separated” assumption is not quite realistic in low-dimensional settings (i.e., $d < \lambda$).

[#] Concrete computational costs not provided.

Given $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) \leq \delta$ if and only if $w_i \in [q_i - \delta, q_i + \delta]$ for all $i \in [d]$, this protocol’s correctness relies on the properties of OKVS (detailed in Section 3.2): if a correct key $q_i \in [w_i - \delta, w_i + \delta]$ is used for decoding, then c_i is a ciphertext encrypting zero; otherwise, c_i is a *random ciphertext* and the sum c' is unlikely to be a zero ciphertext. Security against a semi-honest sender is ensured by the OKVS obliviousness, which perfectly hides the inserted keys $w_i + j$. Security against a semi-honest receiver comes from the complete re-randomization of the AHE (i.e., both *plaintext* and random coins are re-randomized).

2.2 VOLE-based Fuzzy Matching

Our key observation is that AHE can be replaced with *oblivious pseudorandom function* (OPRF) which has much more efficient instantiations. An OPRF allows a receiver to evaluate a PRF on a set of inputs while the sender remains oblivious to the inputs and the receiver remains oblivious to the function. Using this primitive, we can adapt the fuzzy matching protocol as follows by *switching* the roles of the sender and receiver:

- The receiver gets $\{F(w_1), \dots, F(w_d)\}$ and the sender gets $F(\cdot)$ by interacting with an OPRF functionality.
- For each dimension $i \in [d]$, the sender encodes an OKVS instance E_i with a list of key-value pairs $\{(q_i + j, r_i - F(q_i + j))\}_{j \in [-\delta, +\delta]}$ where r_i is a random mask. The sender sends over $\mathbf{E} := (E_1, \dots, E_d)$ together with the target value $r := \sum_{i \in [d]} r_i$.
- The receiver can now decode E_i at w_i and unmask using their OPRF output as $\text{Decode}(E_i, w_i) + F(w_i)$, then sums the results over all dimensions $i \in [d]$, and outputs 1 if and only if it equals the target value r .

It is clear that the correctness still holds as before, and the security against the sender is trivial since the receiver has no transcripts (except for interacting with the ideal functionality of OPRF). The security against the receiver follows from the fact that $F(q_i + j)$ is pseudorandom unless $q_i + j = w_i$.

One might recognize the technique of using an OPRF to mask values encoded in an OKVS as an oblivious *programmable* PRF (OPPRF) [32], which is typically used in circuit-PSI [35,38] and multi-party PSI [32,22] context. To the best of our knowledge, this is the first time a close connection between OPPRF and fuzzy protocols has been established. Details of our fuzzy matching protocol are provided in Section 7.

Instantiating OPRF. To instantiate the OPRF in our fuzzy matching protocol, we observe that the sender can use *different* random function F_i for each dimension $i \in [d]$, and that the receiver only needs to obtain a single evaluation $F_i(w_i)$ for each dimension $i \in [d]$. This allows us to use a more efficient OPRF protocol from subfield VOLE alone [11], instead of building OPRF from VOLE+OKVS [38]. When moving to the fuzzy PSI setting, where the sender and receiver hold multiple points, we actually need the receiver to obtain multiple evaluations of each random function. To this end, we formalize the notion of a (d, n) -OPRF in Section 6, where the receiver obtains n evaluations of each of the d random functions. We moreover give a protocol realizing this functionality from subfield VOLE and an OKVS, which can be seen as a combination of the protocols of [38] and [11]. Instantiating subfield VOLE by a pseudorandom correlation generator (PCG) [8,7], our OPRF protocol only needs public-key operations in the setup phase (i.e., expanding the seeds⁷), and only relies on cheap symmetric-key operations in the online phase (i.e., XOR and hash evaluations). We extend the notion of a (d, n) -OPRF to allow the sender to program points in Section 6.1, resulting in a (d, n) -OPPRF, which forms the main building block of our fuzzy matching and fuzzy PSI protocols.

2.3 Prefix Trie Optimizations

In Section 8.3, we explore how our fuzzy matching techniques can be combined with the prefix trie techniques from [12]. The idea behind the prefix trie technique is to identify a set of *common prefixes* for the binary representations of integers in an interval $[q - \delta, q + \delta]$. An integer w now lies in this interval if and only if it has a prefix lying in this set. Both the set of common prefixes representing the interval $[q - \delta, q + \delta]$ as well as the number of prefixes of w that need to be checked have size $O(\log \delta)$. The main difference between ours (Figure 8 in Section 5) and the Prefix Trie techniques used in [12] is that we improve the computational time from $O(\delta)$ to $O(\log \delta)$, for an interval of length $O(\delta)$. Instead of building a Prefix Trie naively as in [12], we directly identify these common prefixes in Figure 8 by exploiting the structure of the interval.

Hence our fuzzy matching protocol explained above can be adapted by the sender just encoding the $O(\log \delta)$ common prefixes $\tilde{q}_{i,j}$ representing the interval $[q_i - \delta, q_i + \delta]$ in E_i for each dimension $i \in [d]$. The receiver needs to evaluate the OPRF F_i and decode E_i at all of the $O(\log \delta)$ relevant prefixes $\tilde{w}_{i,j}$ of w_i , for each dimension $i \in [d]$. As a result, the receiver has to compare the target value r with all $O((\log \delta)^d)$ possible combinations

$$\sum_{i=1}^d F_i(\tilde{w}_{i,j_i}) + \text{Decode}(E_i, \tilde{w}_{i,j_i}).$$

⁷ The seed expansion for VOLE relies on linear codes that are quite efficient in practice. For instance, ordinary laptops can generate 2^{20} correlations in < 0.5 sec [39], while typical fuzzy PSIs take $50 \sim 100$ sec for moderate-sized sets [1,19]. In fact, the setup cost is marginal even in the standard PSI context as reported in [38, Table 2].

Meet-in-the-Middle. This $O((\log \delta)^d)$ factor is undesirable, and in fact, we can do better: this is a special case of the Knapsack problem to find a subset summing to the target value r . It is known to be NP-complete, however, the receiver can search for the solution in *quadratically* better time by using the “meet-in-the-middle” trick [26]. In detail, we can divide d lists of size ℓ' into roughly two halves, namely, $d_1 := \lceil \frac{d}{2} \rceil$ lists and $d_2 := \lfloor \frac{d}{2} \rfloor$ lists. Then we compute all possible sums of for each half, resulting in two sets of size $(\ell')^{d_1}$ and $(\ell')^{d_2}$. Checking if there is a match between two sets takes time $(\ell')^{d_2}$. In summary, we use $O((\log \delta)^{\lceil \frac{d}{2} \rceil})$ time and space to find a match.

Difference from FSS-based Approaches. Note that [21] also considers representing an interval succinctly but they follow a completely different approach. They build an FSS for a one-sided interval $[0, b]$ through GGM trees, which is done by identifying the critical path (representing point b) with u nodes (domain 2^u). Thus their overhead comes from the *depth* of the GGM tree for a one-sided interval. This overhead is actually *inherent* for FSS. For d -dimensional intervals, the overhead is $(2u)^d$ overhead, which can be reduced to $(2 \log \delta)^d$ with the help of spatial hashing by placing constraints on datasets.

On the contrary, our prefix trie does not rely on any cryptographic primitives and uses only algebraic operations without needing to expand PRG over GGM (i.e., much more concretely efficient). We represent a d -dimensional two-sided interval directly with $(\log \delta)^d$ costs, without putting any restrictions on datasets.

2.4 Towards Fuzzy PSI

When moving from the single point fuzzy matching setting to the multiple point fuzzy PSI setting in Section 8, we can use spatial hashing techniques similarly to previous works [23, 24, 1, 20] which requires the senders points are 2δ or 4δ apart to encode an OKVS successfully.

However, there is one important functional difference between our OPRF-based fuzzy matching and the AHE-based solution from [1]: The sender’s encoding \mathbf{E} can not be re-used with multiple arbitrary receiver points. That is, when the receiver obtains the OPRF evaluations $F_i(w_i), F_i(w'_i)$ of two different values w_i, w'_i lying in the same interval $[q_i - \delta, q_i + \delta]$, they can learn that

$$F_i(w_i) + \text{Decode}(E_i, w_i) = r_i = F_i(w'_i) + \text{Decode}(E_i, w'_i),$$

from which they can infer that on dimension $i \in [d]$, the projections w_i, w'_i lie in the projected interval $[q_i - \delta, q_i + \delta]$ with overwhelming probability.

As a result, we get around this issue by relying on a slightly stronger assumption that the receiver’s points are also 2δ - (or 4δ -) apart from each other⁸. This is still a realistic assumption, since: (1) Each party can independently verify their sets (unlike [14]), so they can decide whether to run the protocol depending on their own input distribution; (2) They can incrementally decrease δ until points are 2δ - (or 4δ -) apart, or merge too-close points into one (which is not possible for [19], since a point whose local projection on each dimension is close to another point is not necessarily close to these points with respect to the global distance function). In the private ride-sharing example, if two cars are too close, the system can use a merged one to represent them, without affecting other cars’ privacy. The details of our fuzzy PSI protocol can be found in Section 8.1.

Sender Privacy. The fuzzy PSI protocol as described in the previous paragraph reveals to the receiver the sender’s point \mathbf{q}_k that lies close to their point $\mathbf{w}_{k'}$. However, for some applications we

⁸ Placing constraints on both parties’ sets has already been considered in existing works [19, 14], where they require either separated or “clustered” points for both parties. Both of them are stronger than ours. Additionally, concurrent work [37] relies on a very similar assumption to ours.

only want the receiver to learn that *there exists* some point of the sender close to their point $\mathbf{w}_{k'}$. This functionality is known as fuzzy PSI with *sender privacy*, which we explore in Section 8.2.

Existing works [20] could support this but have to rely on PSI with cardinality or labeled PSI which are significantly more expensive. Instead, our protocol can be adapted by letting the sender send over the target values r_k for $k \in [M]$, the parties execute a *standard* PSI protocol where the sender learns whether the receiver's obtained values equal the target values. This effectively flips the role of the sender and receiver, and has the effect that the protocol achieves sender privacy, without increasing the asymptotic complexity, obtaining the protocol in Figure 18. Alternatively, we can also achieve sender privacy by letting the sender iterate over the 2^d cells intersecting the ball centered at each point \mathbf{q}_k when encoding the fuzzy matching instances, so that the receiver only has to evaluate the fuzzy matching instance at a single cell containing its point. For this protocol, detailed in Figure 19, the sender's points are required to be distance 4δ apart and the receiver's points 2δ apart.

2.5 Arbitrarily Distributed Points

While our new fuzzy PSI constructions impose extra conditions on the distribution of the parties' points compared to previous work, we on the other hand explore the setting where the parties' points are arbitrarily distributed, i.e., without imposing any restrictions on the distribution of the parties' points.

Our first observation is that by writing a d -dimensional L_∞ ball as the direct product of d one-dimensional interval, we generalize the prefix trie idea to compress a ball of $O(\delta^d)$ points into a set of $O((\log \delta)^d)$ common prefixes. That is, for each $k \in [M]$, the sender can compute a hash $H(\tilde{q}_{k,1}^{j_1}, \dots, \tilde{q}_{k,d}^{j_d})$ for all possible $O((\log \delta)^d)$ choices of (j_1, \dots, j_d) , where $\tilde{q}_{k,i}^j$ are the common prefixes representing the interval $[q_{k,i} - \delta, q_{k,i} + \delta]$. The receiver can similarly, for each $k \in [N]$, compute $H(\tilde{w}_{k,1}^{j_1}, \dots, \tilde{w}_{k,d}^{j_d})$ for all possible $O((\log \delta)^d)$ choices of (j_1, \dots, j_d) , where $\tilde{w}_{k,i}^j$ are the prefixes of $w_{k,i}$. Now the parties can run a standard PSI protocol between these sets of hash values, resulting in a fuzzy PSI protocol with complexity $O((N + M) \cdot (\log \delta)^d)$ based on very simple techniques.

We additionally show that for general Minkowski distance L_p , $p \in [1, \infty)$, we can use the prefix trie technique to obtain a modest improvement over the naive approach of expanding the entire input sets and obtain a protocol with complexity $O((N + M) \cdot \delta^{d-1} \log \delta)$ in this way. Please refer to Section 11 for more detail.

2.6 Minkowski Distance.

So far, we have focused on the infinity distance setting, but our protocols actually cover the general Minkowski distance L_p , defined as

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

and we will briefly sketch how our protocols can be adapted to this setting, but refer to the corresponding sections for more details. In our fuzzy matching protocol, the sender actually encodes the OKVS E_i to map $q_i + j$ to $r_i + s \cdot |j|^p - F(q_i + j)$, where s is an additional random value. This has the effect that if $w_i \in [q_i - \delta, q_i + \delta]$ for each dimension $i \in [d]$, the receiver obtains the value

$$u := \sum_{i=1}^d \text{Decode}(E_i, w_i) + F_i(w_i) = r + s \cdot \sum_{i=1}^d |w_i - q_i|^p, \quad (1)$$

and so the problem is reduced to checking whether $u \in \{r + s \cdot j' \mid j' \in [\delta^p]\}$.

The latter can be done by the sender simply sending over the random oracle evaluations $H(r + s \cdot j')$, adding a factor δ^p to the complexity of the fuzzy matching protocol. Alternatively, we can leverage the prefix trie technique to reduce this overhead from $O(\delta^p)$ to $O(p \log \delta)$ as follows. The sender can in fact put $s := 1$, which reduces the problem to checking whether $u \in [r, r + \delta^p]$. Then instead of sending over hash values, the sender can represent the interval $[r, r + \delta^p]$ by $O(p \log \delta)$ common prefixes using the prefix trie technique, and the receiver can compute the $O(p \log \delta)$ relevant prefixes \tilde{u}_j for the value u from equation (1). Now the parties can run a standard PSI protocol between these sets of prefixes. We refer to Section 8 for the details of how these techniques can be extended to the multiple-point setting.

3 Preliminaries

3.1 Notation

We denote λ for the statistical security parameter and κ for the computational security parameter. For $n \in \mathbb{N}$, we write $[n]$ for the set of integers $\{1, \dots, n\}$, and for $a, b \in \mathbb{Z}$, with $a < b$, write $[a, b]$ for the set $\{a, a + 1, \dots, b - 1, b\}$.

3.2 Oblivious Key-Value Store

An oblivious key-value store OKVS [22] is a datastructure consisting of an **Encode** algorithm, which takes as input a set of key-value pairs, and a **Decode** algorithm, which takes as input a key. Obliviousness roughly means that as long as the encoded values are chosen randomly, an adversary can not distinguish between different sets of encoded keys.

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 $\theta \in \{0, 1\}^\kappa$, 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 $\theta \in \{0, 1\}^\kappa$, 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 $\theta \in \{0, 1\}^\kappa$ 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:

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

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

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

| |
|--|
| $\mathcal{F}_{\text{FUZZYMATCH}}$ |
| 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 |
| $\text{dist}(\mathbf{w}, \mathbf{q})$ is defined as: <ul style="list-style-type: none"> – L_∞ Distance: $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) = \max_{i \in [d]} w_i - q_i$ – L_p Distance: $\text{dist}_p(\mathbf{w}, \mathbf{q}) = \left(\sum_{i=1}^d w_i - q_i ^p \right)^{1/p}$ |

Fig. 1. Ideal Functionality of Fuzzy Matching

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 expansion factor ϵ , characterizing the increase in size of the encoding $m := (1 + \epsilon)n$ relative to the number of key-value pairs n . Recent OKVS constructions [22,36,5] achieve: (1) encoding time $O(n\lambda)$; (2) decoding time $O(\lambda)$; (3) constant expansion factor.

For its application to construct an oblivious programmable PRF (OPPRF), we require OKVS to satisfy some additional properties, which are satisfied by all the state-of-the-art OKVS constructions [22,36,5].

- **Linearity:** There exists a function $\text{dec} : \mathcal{K} \times \{0, 1\}^\kappa \rightarrow \mathcal{V}^m$ such that for all $\mathbf{r} \in \mathcal{V}^m$, $k \in \mathcal{K}$ and $\theta \in \{0, 1\}^\kappa$ it holds that $\text{Decode}(\mathbf{r}, k; \theta) := \langle \text{dec}(k; \theta), \mathbf{r} \rangle$.
- **Double obliviousness:** For all sets of n distinct keys $\{k_1, \dots, k_n\} \subseteq \mathcal{K}$ and n values $\{v_1, \dots, v_n\} \leftarrow_{\$} \mathcal{V}$, it holds that $\text{Encode}(\{(k_i, v_i)_{i \in [n]}\}; \theta)$ is statistically indistinguishable from a uniformly random element from \mathcal{V}^m .

These properties imply another useful property, which is known as the *independence* or *random decoding* property. It is proven for the boolean case ($\mathcal{V} = \{0, 1\}^\ell$) in [23] and was extended to the more general linear case in [1].

Lemma 1 (Independence). *If OKVS satisfies linearity, double obliviousness and $\text{negl}(\lambda)$ failure probability, and θ is uniformly randomly chosen while $n < m$, 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.*

3.3 Definition of Fuzzy Matching

We recall the functionality $\mathcal{F}_{\text{FUZZYMATCH}}$ of fuzzy matching between two points in d -dimensional space \mathbb{Z}^d from [1] in Figure 1, with respect to either infinity (L_∞) distance or Minkowski (L_p) distance where $p \in [1, \infty)$.

| |
|--|
| $\mathcal{F}_{\text{FUZZYPSI}}$ |
| <p>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.</p> <p>Functionality :</p> <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 |
| <p>$\text{leakage}(\mathbf{W}, \mathbf{Q})$ is defined as:</p> <ul style="list-style-type: none"> – 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. – PSI-CA: $\text{leakage}(\mathbf{W}, \mathbf{Q}) = \sum_{i \in [N], j \in [M]} (\text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta)$. |

Fig. 2. Ideal Functionality of Fuzzy PSI

3.4 Definition of Fuzzy Private Set Intersection

Moreover, we recall the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ of fuzzy private set intersection (fuzzy PSI) from [1] in Figure 2. Similarly, this functionality is defined with respect to different distance functions, and additionally incorporates various possible leakage functions, as detailed in Figure 2.

3.5 Spatial Hashing Techniques

To move from single-point fuzzy matching to the multiple-point fuzzy PSI setting, previous works [23, 24, 1, 20] all make use of the idea to tile the space into smaller cells such that each point only needs to be matched to the points in a bounded number of cells. Since the number of relevant cells still scales exponentially in the dimension of the space, this technique is especially useful in low dimensions.

Specifically, the function $\text{cell}_{2\delta}(\mathbf{q})$ maps a point \mathbf{q} to the identifier of the cell (an L_∞ hypercube with side length 2δ) that contains \mathbf{q} . The index id_i of the corresponding cell $\mathcal{C} \leftarrow \text{cell}_{2\delta}(\mathbf{q})$ on dimension $i \in [d]$ is given by $\text{id}_i = \lfloor \frac{q_i}{2\delta} \rfloor$ and each cell is uniquely labeled by $\text{id}_1 \parallel \dots \parallel \text{id}_d$. The function $\text{ball}_\delta(\mathbf{q})$, on the other hand, maps \mathbf{q} to the set of points that are within a distance δ of \mathbf{q} .

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. A *cell* is defined as a L_∞ hypercube of side length 2δ . Given a point $\mathbf{w} \in \mathcal{U}^d$, the index id_i of the corresponding cell \mathcal{C} on dimension $i \in [d]$ is given by $\text{id}_i = \lfloor \frac{w_i}{2\delta} \rfloor$ and each cell is uniquely labeled by $\text{id}_1 \parallel \dots \parallel \text{id}_d$. We refer to [1] for the proofs of the following results.

Lemma 2 (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 $p = \infty$, $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) < 2\delta$.*

Lemma 3 (Unique Center). *Suppose there are multiple L_p balls ($p \in [1, \infty]$) with radius δ lying in a d -dimensional space which is tiled by cells with side length 2δ . If these balls' centers are at least $2\delta d^{\frac{1}{p}}$ apart from each other, then for each cell, there is at most one center lying in this cell. Specifically, if $p = \infty$, then this holds for disjoint balls, since $2\delta d^{\frac{1}{p}}$ degrades to 2δ in this case.*

Lemma 4 (Unique Ball). *Suppose there are multiple L_p balls ($p \in [1, \infty]$) with radius δ lying in a d -dimension space which is tiled by cells with 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 $p = \infty$, then this holds for L_∞ balls with 4δ -apart centers.*

3.6 Standard PSI Functionalities

We introduce some standard PSI functionalities, which we make black-box use of when realizing sender privacy in or other extended functionalities for our fuzzy PSI protocols in Sections 8.2 and 10, respectively. The standard PSI functionality \mathcal{F}_{PSI} is given in Figure 3, the PSI with cardinality (PSI-CA) functionality $\mathcal{F}_{\text{PSI-CA}}$ is given in Figure 4, and the Circuit-PSI functionality $\mathcal{F}_{\text{CPSI}}$ is given in Figure 5.

| |
|---|
| \mathcal{F}_{PSI} |
| Parameters : Input domain \mathcal{X} , cardinality of input sets N, M . |
| Functionality : |
| <ul style="list-style-type: none"> – RECEIVER inputs $X \in \mathcal{X}^N$. – SENDER inputs $Y \in \mathcal{X}^M$. – Output $X \cap Y$ to RECEIVER. |

Fig. 3. Ideal functionality of standard PSI.

| |
|---|
| $\mathcal{F}_{\text{PSI-CA}}$ |
| Parameters : Input domain \mathcal{X} , cardinality of input sets N and M . |
| Functionality : |
| <ul style="list-style-type: none"> – RECEIVER inputs $X \in \mathcal{X}^N$. – SENDER inputs $Y \in \mathcal{X}^M$. – Output $X \cap Y$ to RECEIVER. |

Fig. 4. Ideal functionality of PSI with cardinality.

4 Weak Labeled PSI

We introduce a functionality called weak labeled PSI in Figure 6, which is sufficient to realize labeled fuzzy PSI as described in Section 10, and is moreover used for our fuzzy PSI protocols making

| |
|---|
| $\mathcal{F}_{\text{CPSI}}$ |
| <p>Parameters : Input domain \mathcal{X}, cardinality of input sets N, M, associated data length σ. Reorder : $\mathcal{X}^N \rightarrow (\pi : [N] \rightarrow [m])$ which on input X outputs an injective function π.</p> <p>Functionality :</p> <ul style="list-style-type: none"> – RECEIVER inputs $X \in \mathcal{X}^N$ and associated data $\tilde{X} \in \{0, 1\}^{\sigma \times N}$. – RECEIVER inputs $Y \in \mathcal{X}^M$ and associated data $\tilde{Y} \in \{0, 1\}^{\sigma \times M}$. – For each $k \in [m]$, sample $a_k, b_k \leftarrow \{0, 1\}^{1+2\sigma}$ such that: $a_k \oplus b_k = 1 \parallel \tilde{x}_i \parallel \tilde{y}_j$ if $\exists i \in [N]$ s.t. $k = \pi(i) \wedge x_i = y_j$, $a_k \oplus b_k = 0^{1+2\sigma}$ otherwise. – Return $\pi, (a_k)_{k \in [m]}$ to RECEIVER and $(b_k)_{k \in [m]}$ to SENDER. |

Fig. 5. Ideal functionality of Circuit-PSI

use of the prefix trie techniques in Sections 8.3 and 11. Weak labeled PSI differs from the usual notion of labeled PSI in that the functionality outputs tuples (y_i, label_i) instead of just the label_i of the items in the intersection. We additionally present a simple protocol realizing weak labeled PSI from a $(1, N)$ -OPRF in Figure 7, which can be instantiated using for example the VOLE-based protocol from Figure 12, whereas labeled PSI protocols typically require more expensive public-key techniques [13, 15].

| |
|--|
| $\mathcal{F}_{\text{WLPSI}}$ |
| <p>Parameters : Input domain \mathcal{X}, cardinality of input sets N, M, label length σ.</p> <p>Functionality :</p> <ul style="list-style-type: none"> – RECEIVER inputs $X \in \mathcal{X}^N$. – SENDER inputs $Y = (y_i, \text{label}_i)_{i \in [M]} \in (\mathcal{X} \times \{0, 1\}^\sigma)^M$. – Output $\{(y_i, \text{label}_i) \mid i \in [M], y_i \in X\}$ to RECEIVER. |

Fig. 6. Ideal functionality of weak labeled PSI.

Remark 1. Note that the protocol Π_{WLPSI} reduces to the standard $(1, N)$ -OPRF based PSI protocol, for example, as in [38], realizing \mathcal{F}_{PSI} , when $\sigma = 0$.

Theorem 1. *The protocol Π_{WLPSI} realizes the functionality $\mathcal{F}_{\text{WLPSI}}$ against semi-honest adversaries in the $\mathcal{F}_{(1, N)\text{-OPRF}}$ hybrid model if $\ell \geq \lambda + \log_2(NM)$.*

Proof. (sketch) The proof proceeds analogously as in [38], with the only difference being that the OPRF outputs $F(y_i)$ now completely hide the label_i by definition of $\mathcal{F}_{(1, N)\text{-OPRF}}$ with output length $\ell + \sigma$. \square

5 Prefix Trie

Inspired by [12], we provide the following theorem about succinctly and efficiently representing an integer interval. The main difference between ours and the Prefix Trie techniques used in [12] is that

Π_{WLPSI}

Parameters : Input domain \mathcal{X} , cardinality of input sets N, M , label length σ , OPRF output length $\ell + \sigma$. RECEIVER with input $X = (x_i)_{i \in [N]} \in \mathcal{X}^N$ and SENDER with input $Y = (y_i, \text{label}_i)_{i \in [M]} \in (\mathcal{X} \times \{0, 1\}^\sigma)^M$. The output I is initiated empty, i.e. $I := \emptyset$.

Protocol :

- RECEIVER sends (RECEIVER, EVALUATE, X) and SENDER sends (SENDER, EVALUATE) to $\mathcal{F}_{(1,N)\text{-OPRF}}$.
- RECEIVER gets $(F(x_i))_{i \in [N]}$ and SENDER gets \mathcal{O}^F from $\mathcal{F}_{(1,N)\text{-OPRF}}$.
- SENDER sends a shuffled $L = \{F(y_i) \oplus (0^\ell \parallel \text{label}_i) \mid i \in [M]\}$ to RECEIVER.
- RECEIVER checks for $i \in [N]$ whether there exists $\tilde{y} \in L$ such that $F(x_i) \oplus \tilde{y} = (0^\ell \parallel z)$ for some $z \in \{0, 1\}^\sigma$ and updates $I \leftarrow I \cup \{(x_i, z)\}$ if this is the case.

Fig. 7. Weak labeled PSI protocol from $(1, N)$ -OPRF.

we improve the computational time from $O(\delta)$ in [12] to $O(\log \delta)$, for an interval of length $O(\delta)$. For completeness, we provide the detailed algorithms **PrefixTrie** and **PrefixPath** in Figure 8 and Figure 9. We also denote the 2^u -sized universe as \mathbb{Z}_{2^u} , and drop the subscript when the context is clear. To describe the **PrefixTrie** algorithm, we introduce a helper function **Bin** where $\mathbf{b} \in \mathbb{Z}_2^u \leftarrow \text{Bin}(c \in \mathbb{Z}_{2^u})$ is a binary decomposition function such that $c = \sum_{i=0}^{u-1} b_i \cdot 2^i$.

Theorem 2. *Given an integer interval $[w - \delta, w + \delta] \in \mathbb{Z}_{2^u}^{2\delta+1}$, there is an efficient algorithm to succinctly encode the interval into a list of prefix nodes*

$$\{\tilde{w}_1, \dots, \tilde{w}_\ell\} \leftarrow \text{PrefixTrie}(w - \delta, w + \delta),$$

where $\ell \leq \ell_{\max}$ and $\tilde{w}_i \in \{0, 1\}^u$. Particularly, $\ell_{\max} = 1 + \lfloor \log(2\delta + 1) \rfloor$ when δ is a power of 2, and $\ell_{\max} = 2 \cdot \lfloor \log(2\delta + 1) \rfloor$ otherwise. Moreover, there is another efficient algorithm to expand each query point $q \in \mathbb{Z}$ into a path of prefix nodes

$$\{\tilde{q}_1, \dots, \tilde{q}_{\ell'}\} \leftarrow \text{PrefixPath}(q, \delta),$$

where $\ell' = 2 + \lfloor \log(2\delta + 1) \rfloor$ and $\tilde{q}_i \in \{0, 1\}^u$. Both algorithms have $O(\log \delta)$ computation complexity. Importantly, $\tilde{q}_i \in \{\tilde{w}_1, \dots, \tilde{w}_\ell\}$ for some unique $i \in [\ell']$ if and only if $q \in [w - \delta, w + \delta]$.

Proof. The algorithm **PrefixPath**(c, δ) presented in Figure 9 is precisely the same as in [12], such that it takes ℓ' iterations and outputs a set of size $|J| = \ell'$.

Let us focus on the algorithm **PrefixTrie**(x_l, x_r) presented in Figure 8. First, every integer $x_i \in [x_l, x_r]$ has a common prefix of $u - (d + 1)$ bits where $d = \lfloor \log(x_r - x_l + 1) \rfloor$: if there are $x_i, x_j \in [x_l, x_r]$ differing more than lower $d + 1$ bits, then the distance between them is greater than $2^{d+1} - 1 > x_r - x_l$ which is beyond the maximal distance within $[x_l, x_r]$.

Next, consider an interval $[e_l, e_r]$ of length 2^{d+1} which has a common prefix of $u - (d + 1)$ bits. We consider the following cases.

1. If $[x_l, x_r]$ has length 2^d , then it might align perfectly with the left half or the right half of $[e_l, e_r]$. In this case, the interval $[x_l, x_r]$ can be represented as a single prefix.
2. Otherwise, $[x_l, x_r]$ must span across two halves of $[e_l, e_r]$. We denote two halves as $[e_l, e'_l]$ and $[e'_r, e_r]$ respectively. Given the distance between $s = e'_l - x_l + 1$, we can decompose it into a d -bit binary vector and its Hamming weight indicates the number of disjoint prefixes required to represent the half $[e_l, e'_l]$. The other half $[e'_r, e_r]$ is similar.

In conclusion, in the worst case, we can represent the entire interval $[x_l, x_r]$ with at most $\ell_{\max} := 2 \cdot d$ prefixes, and the algorithm PrefixTrie takes $O(2d)$ steps at most.

Particularly, if δ is a power of 2, then the interval can be represented by $\ell_{\max} = d + 1$ prefixes in the worst case. Imaging $[x'_l, x'_r]$ has length 2^d and spans across two halves of $[e_l, e_r]$, then it takes at most $d + 1$ prefixes to represent $[x'_l, x'_r]$. To see this, consider a divide-and-conquer algorithm:

1. Check if the middle point of $[x'_l, x'_r]$ is greater than the ‘break’ point e'_r or not;
2. Then, use one prefix to represent the half not containing e'_r ;
3. Next, run the algorithm recursively on the other half. It ends with only two integers which can be represented at most 2 prefixes.

Summing up together, we have $d + 1$ prefixes at most. When δ is a power of 2, $[x_l, x_r]$ has length $2^d + 1$ and it is either $[x'_l - 1, x'_r]$ or $[x'_l, x'_r + 1]$. We argue the number of required prefixes is unchanged (i.e., $d + 1$) for both $[x_l, x_r]$ and $[x'_l, x'_r]$ by the following cases.

1. If there is a prefix p in J' of length u where J' is the prefix set for $[x'_l, x'_r]$, then x'_l and x'_r must be a singleton prefix. In this case, $[x_l, x_r]$ can still be represented as $|J'|$ prefixes, just preserve $u - 1$ bits of either x'_l or x'_r as the new prefix.
2. On the other hand, if there is no prefix of length u which means $|J'| < d + 1$, then $|J| = |J'| + 1 \leq d + 1$.

□

| PrefixTrie |
|---|
| <p>Inputs : Two integers $x_l, x_r \in \mathbb{Z}_{2^u}$, representing an interval $[x_l, x_r]$.</p> <p>Algorithm :</p> <ol style="list-style-type: none"> 1. Set $d = \lfloor \log(x_r - x_l + 1) \rfloor$, and prepare a set $J = \emptyset$, 2. Set $e_l = x_l - (x_l \bmod 2^{d+1})$ and $e_r = e_l + 2^{d+1} - 1$. 3. return $\text{Bin}\left(\frac{e_l}{2^d}\right)$ if $e_l = x_l \wedge e_l + 2^d - 1 = x_r$, or if $e_r = x_r \wedge e_r - (2^d - 1) = x_l$. 4. Set $e'_l = e_l + 2^d - 1$ and $e'_r = e_r - (2^d - 1)$. 5. Decompose $\mathbf{b}_l = \text{Bin}(e'_l - x_l + 1)$ and $\mathbf{b}_r = \text{Bin}(x_r - e'_r + 1)$. 6. Calculate the distance to get $\mathbf{s}_l, \mathbf{s}_r$ where $s_{l,i} = \sum_{j=i}^{d-1} 2^j \cdot b_{l,j}, \quad s_{r,i} = \sum_{j=i+1}^{d-1} 2^j \cdot b_{r,j}$ for $i \in [d - 1, 0]$. 7. for each $i \in [d - 1, 0]$, do: <ol style="list-style-type: none"> (a) If $b_{l,i} = 1$, set $J = J \cup \text{Bin}\left(\frac{1 + e'_l - s_{l,i}}{2^i}\right)$. (b) If $b_{r,i} = 1$, set $J = J \cup \text{Bin}\left(\frac{e'_r + s_{r,i}}{2^i}\right)$. 8. return J. |

Fig. 8. Algorithm descriptions of PrefixTrie

| |
|--|
| PrefixPath |
| Inputs : An integer $c \in \mathbb{Z}_{2^u}$ and a radius δ . Algorithm : <ol style="list-style-type: none"> 1. Prepare a set $J = \emptyset$. 2. for each $i \in [0, \lfloor \log(2\delta + 1) \rfloor + 1]$ do: <ul style="list-style-type: none"> – Update $J = J \cup \text{Bin}(\lfloor \frac{c}{2^i} \rfloor)$. 3. return J. |

Fig. 9. Algorithm descriptions of PrefixPath

6 Multi-Batch O(P)PRF

We introduce the notion of a multi-batch oblivious pseudorandom function (OPRF), where batches of points are evaluated under multiple independent random functions. More precisely, there are d independent random functions, the receiver inputs d batches of n points and receives the evaluation of each batch under the corresponding random function. We will refer to this functionality as a (d, n) -OPRF and formally define it in Figure 10. Note that by setting $d = 1$, one recovers the usual notion of an OPRF as in [38]. By setting $n = 1$, one recovers the notion of a vector OPRF from [2]. Our notion of a (d, n) -OPRF is moreover related to the notion of a membership batch, related-key OPRF (mBaRK-OPRF) from [11], except that we work in the random oracle model, which allows for a cleaner presentation.

| |
|--|
| $\mathcal{F}_{(d,n)\text{-OPRF}}$ |
| Parameters : Number of batches d , batch size n , input space \mathcal{X} , output space \mathcal{Y} . Functionality : <ul style="list-style-type: none"> – RECEIVER inputs (RECEIVER, EVALUATE, \mathbf{X}), where $\mathbf{X} := (\mathbf{X}_1, \dots, \mathbf{X}_d) \in \mathcal{X}^{d \cdot n}$. – SENDER inputs (SENDER, EVALUATE). – Sample $F_1, \dots, F_d \leftarrow \mathcal{S} \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$ – Output $(F_i(x_{i,j}))_{i \in [d], j \in [n]}$ to RECEIVER and $(\mathcal{O}^{F_i})_{i \in [d]}$ to SENDER. |

Fig. 10. Ideal functionality of (d, n) -multi-batch oblivious PRF $((d, n)$ -OPRF).

We moreover give a construction $\Pi_{(d,n)\text{-OPRF}}$ of a multi-batch OPRF based on subfield vector linear oblivious evaluation (sVOLE) in Figure 12. This can be seen as a combination of the semi-honest VOLE-based OPRF construction from [38] and the sVOLE-based mBaRK-OPRF construction from [11]. The protocol makes use of the subfield VOLE functionality defined in Figure 11. While keeping the total number $N := dn$ constant, the protocol becomes more efficient as d increases, since this means that the number of points $n = N/d$ the sender needs to encode in each OKVS decreases. For example, the state-of-the-art OKVS construction [5] has encoding complexity $O(n(\lambda/\epsilon + \log n))$ to encode n points and decoding complexity $O(\lambda/\epsilon + \log n)$ to decode a single point, where ϵ is the expansion factor of the OKVS. Hence, to encode dn values in a single OKVS takes time $O(dn(\lambda/\epsilon + \log(dn)))$, whereas encoding d separate OKVS's with n values each

takes time $O(dn(\lambda/\epsilon + \log n))$, saving a factor $O(\frac{\log d}{\lambda/\epsilon + \log n})$. The same factor is saved when decoding dn values.

| |
|---|
| $\mathcal{F}_{\text{sVLE}}$ |
| <p>Parameters : Base field \mathbb{B}, finite extension field \mathbb{F}.</p> <p>Functionality :</p> <ul style="list-style-type: none"> – RECEIVER inputs (RECEIVER, EVALUATE, m). – SENDER inputs (SENDER, EVALUATE, m). – Sample $\Delta \leftarrow \mathbb{F}$, $\mathbf{A} \leftarrow \mathbb{B}^m$, $\mathbf{B} \leftarrow \mathbb{F}^m$ and put $\mathbf{C} := \Delta \cdot \mathbf{A} + \mathbf{B}$. – Output \mathbf{A}, \mathbf{C} to RECEIVER and Δ, \mathbf{B} to SENDER. |

Fig. 11. Ideal functionality of subfield VOLE.

| |
|--|
| $\Pi_{(d,n)\text{-OPRF}}$ |
| <p>Parameters : RECEIVER with input $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_d) \in \mathcal{X}^{d \cdot n}$, number of batches d, batch size n, input field \mathbb{B}, \mathbb{F} a finite extension field of \mathbb{B}, output space \mathcal{Y}. $\mathbf{H} : \{0, 1\}^* \rightarrow \mathcal{Y}$ is a random oracle, OKVS = (Encode, Decode) is an oblivious key-value store with value space \mathbb{B} and expansion factor ϵ.</p> <p>Protocol :</p> <ol style="list-style-type: none"> 1. RECEIVER computes $\mathbf{P}_i \leftarrow \text{Encode}(\{(x_{i,j}, 0) \mid j \in [n]\})$ for each $i \in [d]$. Let $\mathbf{P} := (\mathbf{P}_1, \dots, \mathbf{P}_d) \in \mathbb{B}^m$, $m := d \cdot (1 + \epsilon)n$. If $n = 1$, RECEIVER puts $\mathbf{P}_i := \mathbf{X}_i$. 2. RECEIVER sends (RECEIVER, EVALUATE, m) and SENDER sends (SENDER, EVALUATE, m) to $\mathcal{F}_{\text{sVLE}}$. 3. RECEIVER gets $\mathbf{A} \in \mathbb{B}^m$, $\mathbf{C} \in \mathbb{F}^m$ and SENDER gets $\Delta \in \mathbb{F}$, $\mathbf{B} \in \mathbb{F}^m$ from $\mathcal{F}_{\text{sVLE}}$, satisfying the relation $\mathbf{C} = \Delta \cdot \mathbf{A} + \mathbf{B}$. 4. RECEIVER sends $\mathbf{Z} := \mathbf{P} + \mathbf{A}$ to SENDER. 5. SENDER defines $\mathbf{K} := \Delta \cdot \mathbf{Z} + \mathbf{B}$ and $F_i(x) := \mathbf{H}(i, \text{Decode}(\mathbf{K}_i, x))$, for each $i \in [d]$, where $\mathbf{K}_i := (k_{(i-1) \cdot (1+\epsilon)n+1}, \dots, k_{i \cdot (1+\epsilon)n})$. 6. RECEIVER computes $F_i(x_{i,j}) = \mathbf{H}(i, \text{Decode}(\mathbf{C}_i, x_{i,j}))$, for each $i \in [d]$ and $j \in [n]$, where $\mathbf{C}_i := (c_{(i-1) \cdot (1+\epsilon)n+1}, \dots, c_{i \cdot (1+\epsilon)n})$. |

Fig. 12. Subfield VOLE-based (d, n) -OPRF protocol

Lemma 5. *The protocol $\Pi_{(d,n)\text{-OPRF}}$ (Fig. 12) has computation complexity $O(dn)$ and communication complexity $O(dn\lambda)$, ignoring logarithmic factors in d and n .*

Proof. This can be achieved using a PCG to generate the subfield VOLE correlations, which achieves [8, 7]: computation and communication scaling logarithmically in the vector length in the offline phase to distribute the seeds; computation scaling linearly in the vector length in the online phase to expand the seeds, while requiring no communication. Additionally we can instantiate OKVS by a construction with constant expansion factor, linear encoding time and constant decoding time, such as [36, 5]. \square

Theorem 3. *The protocol $\Pi_{(d,n)\text{-OPRF}}$ (Fig. 12) securely realizes the ideal functionality $\mathcal{F}_{(d,n)\text{-OPRF}}$ against semi-honest adversaries in the random oracle, $\mathcal{F}_{\text{sVOLE}}$ -hybrid model if the subfield size $\log_2 |\mathbb{B}| \geq \lambda + \log_2(n)$ and extension field size $\log_2 |\mathbb{F}| \geq \kappa$ and OKVS satisfies the independence property from Lemma 1.*

Proof. If both parties behave honestly, it is straightforward to check that the receiver obtains the correct evaluations. Namely, on inputs $x_{i,j} \in \mathbf{X}_i$, by definition of $\mathcal{F}_{\text{sVOLE}}$ and the linearity of OKVS, by expanding \mathbf{K}_i , it follows that

$$\begin{aligned} F_i(x_{i,j}) &:= \text{H}(i, \text{Decode}(\mathbf{K}_i, x_{i,j})) \\ &= \text{H}(i, \text{Decode}(\Delta \cdot \mathbf{Z}_i + \mathbf{B}_i, x_{i,j})) \\ &= \text{H}(i, \text{Decode}(\Delta \cdot \mathbf{P}_i + \Delta \cdot \mathbf{A}_i + \mathbf{B}_i, x_{i,j})) \\ &= \text{H}(i, \text{Decode}(\Delta \cdot \mathbf{P}_i + \mathbf{C}_i, x_{i,j})) \\ &= \text{H}(i, \Delta \cdot \text{Decode}(\mathbf{P}_i, x_{i,j}) + \text{Decode}(\mathbf{C}_i, x_{i,j})) \\ &= \text{H}(i, \text{Decode}(\mathbf{C}_i, x_{i,j})). \end{aligned}$$

Note that the functions $F_i(x)$ are independent for different $i \in [d]$ by domain separation in the input to the random oracle H . Moreover, each function $F_i(x)$ behaves as a random function because the vector $\mathbf{K}_i \in \mathbb{F}^{(1+\epsilon)n}$ is uniformly random and the decoding vectors $\text{dec}(x)$ for different x are linearly independent with overwhelming probability since OKVS has negligible failure probability.

Consider a semi-honestly corrupted sender. The simulator samples the VOLE outputs $\Delta \leftarrow \$ \mathbb{F}$ and $\mathbf{B} \leftarrow \$ \mathbb{F}^m$ as the ideal functionality $\mathcal{F}_{\text{sVOLE}}$ does, and samples $\mathbf{Z} \leftarrow \$ \mathbb{B}^m$, which is indistinguishable from the real protocol execution since \mathbf{P} is completely masked by \mathbf{A} . Then the simulator computes $\mathbf{K} := \Delta \cdot \mathbf{Z} + \mathbf{B}$ and programs the random oracle at queries of the form $(i, \text{Decode}(\mathbf{K}_i, y))$, $i \in [d]$, $y \in \mathcal{X}$ to return the $\mathcal{F}_{(d,n)\text{-OPRF}}$ output $F_i(y)$. These outputs are indistinguishable from the real protocol outputs by the same reasoning as for the correctness above.

Now consider a semi-honestly corrupted receiver. The simulator samples the VOLE outputs $\mathbf{A} \leftarrow \$ \mathbb{B}^m$ and $\mathbf{C} \leftarrow \$ \mathbb{F}^m$, which is indistinguishable from the ideal functionality $\mathcal{F}_{\text{sVOLE}}$ since \mathbf{B} completely randomizes \mathbf{C} . Upon receiving random oracle queries of the form $(i, \text{Decode}(\mathbf{C}_i, x_{i,j}))$, $i \in [d]$, $j \in [n]$, the simulator returns the $\mathcal{F}_{(d,n)\text{-OPRF}}$ output $F_i(x_{i,j})$. These are indistinguishable from the real functionality outputs since each $\mathbf{C}_i \in \mathbb{F}^{(1+\epsilon)n}$ is uniformly random and the decoding vectors $\text{dec}(x_{i,j})$, $j \in [n]$, are linearly independent with overwhelming probability. Consider a hybrid which behaves as a real protocol execution but aborts when the receiver poses a query of the form $(i, \Delta \cdot \text{Decode}(\mathbf{P}_i, x) + \text{Decode}(\mathbf{C}_i, x))$ for any $i \in [d]$ and $x \in \mathcal{X} \setminus \mathbf{X}_i$. This hybrid aborts with negligible probability in κ since $\Delta \leftarrow \$ \mathbb{F}$ from the receiver's point of view and $|\mathbb{F}| \geq 2^\kappa$. Moreover, it is infeasible for the receiver to find $x \in \mathcal{X} \setminus \mathbf{X}_i$ such that $\text{Decode}(\mathbf{P}_i, x) = 0$ by the independence property from Lemma 1. Finally, the simulated execution is computationally indistinguishable from this hybrid, which completes our proof. \square

6.1 Multi-Batch OPRF

Extending our multi-batch OPRF with the ability for the sender to program certain input points to map to certain output points, we obtain the notion of a multi-batch oblivious programmable pseudorandom function (OPPRF). In addition to the receiver inputting d batches of n points, the sender inputs d lists of (input, output) pairs. If one of the receiver's inputs in batch $i \in [d]$ matches one of the sender's inputs in the same list $i \in [d]$, the receiver obtains the corresponding output value. Otherwise, the receiver obtains an independently random output value, as in (d, n) -OPRF.

We formalize this functionality as a (d, n) -OPPRF and present it in Figure 13. By setting $d = 1$, one recovers the usual notion of an OPPRF as in [38] and, by setting $n = 1$, one recovers the notion of a vector OPPRF from [2]. Our notion of a (d, n) -OPPRF is also related to the notion of a batch OPPRF from [35], except that we allow the receiver to obtain multiple evaluations of each programmed function (if $n > 1$), and that working in the random oracle model allows us to give a cleaner presentation.

| |
|---|
| $\mathcal{F}_{(d,n)\text{-OPPRF}}$ |
| Parameters : Number of batches d , batch size n , input space \mathcal{X} , output space \mathcal{Y} . Functionality : <ul style="list-style-type: none"> – RECEIVER inputs (RECEIVER, EVALUATE, \mathbf{X}), $\mathbf{X} := (\mathbf{X}_1, \dots, \mathbf{X}_d)$ with $\mathbf{X}_i \in \mathcal{X}^n$. – SENDER inputs (SENDER, EVALUATE, \mathbf{L}), where $\mathbf{L} := (L_1, \dots, L_d)$ with $L_i \subset \mathcal{X} \times \mathcal{Y}$. – Sample $F_i \leftarrow \{f : \mathcal{X} \rightarrow \mathcal{Y} \mid \forall (y, z) \in L_i : f(y) = z\}$ for each $i \in [d]$ – Output $(L_i)_{i \in [d]}, (F_i(x_{i,j}))_{i \in [d], j \in [n]}$ to RECEIVER and $(\mathcal{O}^{F_i})_{i \in [d]}$ to SENDER. |

Fig. 13. Ideal functionality of (d, n) -multi-batch oblivious programmable PRF ((d, n) -OPPRF).

Additionally, we present a protocol $\Pi_{(d,n)\text{-OPPRF}}$ of a multi-batch OPPRF from a multi-batch OPRF in Figure 14. This protocol follows the usual blueprint of the OPPRF protocols presented in [32, 35, 38] of using a data structure to encode some offset to an OPRF, such that decoding the data structure and adding the OPRF evaluation gives the programmed point. The conditions that this data structure needs to satisfy were recently formalized under the notion of an oblivious key-value store (OKVS) [22], and a formal proof of an OPPRF from an OPRF and a general OKVS was given in [2]. Our protocol can be seen as a combination of the protocols from [38, 2] to allow the receiver to obtain multiple evaluations under multiple independent programmed functions.

| |
|--|
| $\Pi_{(d,n)\text{-OPPRF}}$ |
| Parameters : Number of batches d , batch size n , input space \mathcal{X} , output space \mathcal{Y} . RECEIVER with input $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_d)$, where $\mathbf{X}_i \in \mathcal{X}^n$, and SENDER with input $\mathbf{L} = (L_1, \dots, L_d)$, where $L_i \subset \mathcal{X} \times \mathcal{Y}$. Protocol <ol style="list-style-type: none"> 1. RECEIVER sends (RECEIVER, EVALUATE, \mathbf{X}) and SENDER sends (SENDER, EVALUATE) to $\mathcal{F}_{(d,n)\text{-OPPRF}}$. 2. RECEIVER gets $(F'_i(x_{i,j}))_{i \in [d], j \in [n]}$ and SENDER gets $(\mathcal{O}^{F'_i})_{i \in [d]}$ from $\mathcal{F}_{(d,n)\text{-OPPRF}}$. 3. SENDER computes $E_i \leftarrow \text{Encode}(\{(y, z - F'_i(y)) \mid (y, z) \in L_i\})$ for each $i \in [d]$. 4. SENDER sends $\mathbf{E} := (E_1, \dots, E_d)$ to RECEIVER. 5. SENDER defines the function $F_i(x) := F'_i(x) + \text{Decode}(E_i, x)$ for each $i \in [d]$. 6. RECEIVER computes $F_i(x_{i,j}) = F'_i(x_{i,j}) + \text{Decode}(E_i, x_{i,j})$ for each $i \in [d], j \in [n]$. |

Fig. 14. (d, n) -OPRF-based (d, n) -OPPRF protocol

Lemma 6. *The protocol $\Pi_{(d,n)\text{-OPPRF}}$ has computation complexity $O(dn + |\mathbf{L}|)$ for the sender, computation complexity $O(dn)$ for the receiver, and communication complexity $O(dn\lambda + |\mathbf{L}| \cdot \log_2 |\mathcal{Y}|)$, where $|\mathbf{L}| := \sum_{i=1}^d |L_i|$, ignoring logarithmic factors in d , n and $|\mathbf{L}|$.*

Proof. This can be achieved realizing $\mathcal{F}_{(d,n)\text{-OPRF}}$ by the protocol $\Pi_{(d,n)\text{-OPPRF}}$ from Figure 14 and using the corresponding complexities from Lemma 5, and again instantiating OKVS by construction with constant expansion factor, linear encoding time and constant decoding time. \square

Theorem 4. *The protocol $\Pi_{(d,n)\text{-OPPRF}}$ realizes the ideal functionality $\mathcal{F}_{(d,n)\text{-OPPRF}}$ against semi-honest adversaries in the $\mathcal{F}_{(d,n)\text{-OPPRF}}$ hybrid model if OKVS is linear and doubly oblivious.*

Proof. If both parties behave honestly, correctness on programmed points $(y, z) \in L_i$ follows from the correctness of OKVS, since

$$\begin{aligned} F_i(y) &:= F'_i(y) + \text{Decode}(E_i, y) \\ &= F'_i(y) + z - F'_i(y) \\ &= z. \end{aligned}$$

On unprogrammed points $x \in \mathcal{X}$, the vector E_i is independent from $F'_i(x)$, and thus the value $F_i(x) := F'_i(x) + \text{Decode}(E_i, x)$ is uniformly random since F'_i is a random function by definition of $\mathcal{F}_{(d,n)\text{-OPPRF}}$.

First, consider a semi-honestly corrupted sender. Since the sender does not receive any messages from the receiver, we only need to simulate the outputs $(\mathcal{O}^{F'_i})_{i \in [d]}$ of $\mathcal{F}_{(d,n)\text{-OPPRF}}$ such that they are consistent with the outputs $(\mathcal{O}^{F_i})_{i \in [d]}$ of $\mathcal{F}_{(d,n)\text{-OPPRF}}$. On inputs $y \in \mathcal{X}$ for which there exists $(y, z) \in L_i$, the simulator samples the value $F'_i(y)$ uniformly random as the ideal functionality $\mathcal{F}_{(d,n)\text{-OPPRF}}$ does. Using these values, the simulator can compute $E_i \leftarrow \text{Encode}(\{(y, z - F'_i(y)) \mid (y, z) \in L_i\})$ as the sender does. The simulator can now simulate the OPRF oracle $\mathcal{O}^{F'_i}$ on unprogrammed points as $F'_i(x) := F_i(x) - \text{Decode}(E_i, x)$, which is indistinguishable from a real protocol execution since the values $F_i(x)$ are uniformly random on unprogrammed points.

Now consider a semi-honestly corrupted receiver. The simulator obtains the receiver's input $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_d)$ and the ideal $\mathcal{F}_{(d,n)\text{-OPPRF}}$ functionality's outputs $(|L_i|)_{i \in [d]}$ and $(F_i(x_{i,j}))_{i \in [d], j \in [n]}$. The simulator samples \mathbf{E} by inserting $|L_i|$ random key-value pairs into each E_i , and simulates the $\mathcal{F}_{(d,n)\text{-OPPRF}}$ outputs as $F'_i(x_{i,j}) := F_i(x_{i,j}) - \text{Decode}(E_i, x_{i,j})$, which guarantees correctness on programmed points. The OKVS encodings E_i are statistically indistinguishable from a real protocol execution since OKVS is doubly oblivious. Indistinguishability of the values $F'_i(x_{i,j})$ can be seen from the following two cases. First, if $x_{i,j} = y$ for some programmed point $(y, z) \in L_i$, then $\text{Decode}(E_i, y) := \langle \text{dec}(y), E_i \rangle$ is distributed uniformly random independent from z since E_i is uniformly random independent from L_i and the vectors $\text{dec}(y)$, for $(y, z) \in L_i$, are linearly independent with overwhelming probability by the correctness of OKVS; hence

$$\begin{aligned} F'_i(x_{i,j}) &= F'_i(y) \\ &= F_i(y) - \text{Decode}(E_i, y) \\ &= z - \text{Decode}(E_i, y) \end{aligned}$$

is distributed statistically close to uniformly random. Second, if $x_{i,j}$ is an unprogrammed point, the $\mathcal{F}_{(d,n)\text{-OPPRF}}$ output $F_i(x_{i,j})$ is distributed uniformly random independent from E_i ; hence $F'_i(x_{i,j}) := F_i(x_{i,j}) - \text{Decode}(E_i, x_{i,j})$ is uniformly random. \square

| GetList _p ($r, s, \mathbf{q}, \Delta_q$) |
|---|
| If $p = \infty$, replace $s \leftarrow 0$ |
| For each $i = 1, \dots, d$ |
| If $i < d$, sample $r_i \leftarrow_{\$} \{0, 1\}^{\lambda'}$ |
| If $i = d$, set $r_d := r - \sum_{i=1}^{d-1} r_i$ |
| For each $j = -\delta, \dots, \delta$ |
| Set $\text{key}_{i,j} \leftarrow H_\gamma(q_i + j \parallel \Delta_q)$ |
| Set $\text{val}_{i,j} := r_i + s \cdot j ^p$ |
| Set $\text{list}_i := \{(\text{key}_{i,j}, \text{val}_{i,j})_{j \in [-\delta, \delta]}\}$ |
| Return $(\text{list}_1, \dots, \text{list}_d)$ |

Fig. 15. GetList subprotocol for Minkowski distance $p \in [1, \infty]$.

7 Fuzzy Matching from OPPRF

Fuzzy matching targets the setting where there are two parties holding points $\mathbf{w} = (w_1, \dots, w_d)$ and $\mathbf{q} = (q_1, \dots, q_d)$ in d -dimensional space \mathbb{Z}^d , who wish to learn whether their points are close to each other, that is, whether $\text{dist}(\mathbf{w}, \mathbf{q}) \leq \delta$, with respect to some distance metric. In particular, we focus our attention on the generalized Minkowski distance

$$\text{dist}_p(\mathbf{w}, \mathbf{q}) := \begin{cases} \max_{i \in [d]} |w_i - q_i| & \text{if } p = \infty, \\ (\sum_{i=1}^d |w_i - q_i|^p)^{1/p} & \text{if } p \neq \infty. \end{cases}$$

The ideal functionality $\mathcal{F}_{\text{FUZZYMATCH}}$ is detailed in Figure 1 and we present a protocol $\Pi_{\text{FUZZYMATCH}}^p$ realizing this functionality for generalized Minkowski distance $p \in [1, \infty]$ from a $(d, 1)$ -OPPRF in Figure 16. Our protocol is inspired by the fuzzy matching protocol from [1]. Under the hood, the OPPRF protocol from Figure 14 can be seen as using the OPRF outputs to one-time-pad encrypt the programmed values, which the receiver can only decrypt if their input matches a programmed point. Our protocol can therefore be seen as replacing the additively homomorphic encryption in [1] by an OPRF, which can be super efficiently instantiated from OKVS and VOLEs, as shown in Figure 12.

The main idea behind our protocol for infinity distance ($p = \infty$) is that the sender programs the OPPRF to map the intervals $([q_i - \delta, q_i + \delta])_{i \in [d]}$, to a random d -out-of- d secret sharing $(r_i)_{i \in [d]}$ of zero. The receiver obtains the OPPRF evaluations $(F_i(w_i))_{i \in [d]}$, which will sum up to zero if and only if $w_i \in [q_i - \delta, q_i + \delta]$ for all $i \in [d]$. That is, if and only if $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) \leq \delta$.

For Minkowski distance ($p \neq \infty$), the sender instead programs the OPPRF to map the values $q_i + j$ to $r_i + s \cdot |j|^p$ with random r_i, s , for $i \in [d], j \in [-\delta, \delta]$. This means that if $w_i \in [q_i - \delta, q_i + \delta]$, the receiver will obtain the OPPRF evaluation $F_i(w_i) := r_i + s \cdot |w_i - q_i|^p$. If this holds for all dimensions, the receivers OPPRF evaluations will sum to $r + s \cdot \text{dist}_p(\mathbf{w}, \mathbf{q})^p$, where $r := \sum_{i=1}^d r_i$. As a final step, the sender can therefore send over the set $\{H(r + s \cdot j) \mid j \in [\delta^p]\}$ of hash values, and the receiver can check if $H(\sum_{i=1}^d F_i(w_i))$ is present in this set.

This leads to our protocol having almost identical asymptotic complexities to the protocol from [1], the main difference being the δ^p term in the sender's instead of the receiver's computation complexity, but using significantly cheaper individual operations (i.e., only containing XOR and

Parameters : RECEIVER with input $\mathbf{w} \in \mathbb{Z}^d$ and SENDER with input $\mathbf{q} \in \mathbb{Z}^d$. Dimension d and radius δ .

Protocol :

1. SENDER samples $r, s \leftarrow \{0, 1\}^{\lambda'}$, and sets $r = 0$ if $\mathbf{p} = \infty$.
2. SENDER gets $(\text{list}_1, \dots, \text{list}_d) \leftarrow \text{GetList}_p(r, s, \mathbf{q}, 0^\lambda)$.
3. RECEIVER sends (RECEIVER, EVALUATE, \mathbf{w}) and SENDER sends (SENDER, EVALUATE, $(\text{list}_1, \dots, \text{list}_d)$) to $\mathcal{F}_{(d,1)\text{-OPPRF}}$.
4. RECEIVER gets $(F_i(w_i))_{i \in [d]}$ and SENDER gets $(\mathcal{O}^{F_i})_{i \in [d]}$ from $\mathcal{F}_{(d,1)\text{-OPPRF}}$.
5. If $\mathbf{p} \neq \infty$, SENDER puts $S := \{H_{\lambda'}(r + s \cdot j) \mid j \in [\delta^{\mathbf{p}}]\}$, shuffles S and sends it to RECEIVER.
6. If $\mathbf{p} = \infty$, RECEIVER puts $S := \{H_{\lambda'}(0)\}$.
7. RECEIVER outputs 1 if $H_{\lambda'}(\sum_{i=1}^d F_i(w_i)) \in S$, and 0 otherwise.

Fig. 16. Fuzzy matching for Minkowski distance from $(d, 1)$ -OPPRF.

hash evaluations during the online phase). Additionally, the communication complexity of our protocol relies only on the statistical security parameter λ , not on the computational security parameter κ , due to the use of subfield VOLE in the underlying OPRF. However, this gain in efficiency does come at the cost of our fuzzy matching protocol not being *reusable*. That is, since the sender programs all points in the interval $[q_i - \delta, q_i + \delta]$ to map to the same r_i , the receiver is not allowed to receive OPRF evaluations of two different $w_i, w'_i \in [q_i - \delta, q_i + \delta]$, as this reveals more information about partial matches on individual dimensions. This does make it more complicated to move from our fuzzy matching protocol to a fuzzy PSI protocol, which we will discuss in more detail in Section 8.

Lemma 7. *The protocol $\Pi_{\text{FUZZYMATCH}}^p$ has*

- *sender's computational complexity: $O(\delta d)$ if $\mathbf{p} = \infty$, and $O(\delta d + \delta^{\mathbf{p}})$ if $\mathbf{p} \neq \infty$;*
- *receiver's computational complexity $O(d)$;*
- *communication complexity $O(\delta d \lambda)$ if $\mathbf{p} = \infty$, and $O(\delta d \lambda + \delta^{\mathbf{p}} \lambda)$ if $\mathbf{p} \neq \infty$;*

ignoring logarithmic factors in all complexities.

Proof. The above complexities can be realized by instantiating $\mathcal{F}_{(d,1)\text{-OPPRF}}$ by a protocol with sender computational complexity $O(\sum_{i=1}^d |L_i| + d)$, receiver computational complexity $O(d)$, and communication complexity $O((d + \sum_{i=1}^d |L_i|) \cdot \lambda)$, as in Lemma 6. For example, by instantiating $\mathcal{F}_{(d,1)\text{-OPPRF}}$ by the subfield VOLE-based protocol from Figure 12 with subfield size $\log_2 |\mathbb{B}| := \gamma$ and output length $\log_2 |\mathcal{Y}| := \lambda'$ as in Theorem 5, using the complexities from Lemma 5. Moreover, we can instantiate S by a data structure with constant lookup time and linear build time, such as a Cuckoo hash table. \square

Theorem 5. *The protocol $\Pi_{\text{FUZZYMATCH}}^p$ securely realizes the ideal functionality $\mathcal{F}_{\text{FUZZYMATCH}}$ for L_p distance, $\mathbf{p} \in [1, \infty]$, against semi-honest adversaries in the $\mathcal{F}_{(d,1)\text{-OPPRF}}$ -hybrid model if $H_\gamma : \{0, 1\}^* \rightarrow \{0, 1\}^\gamma$ is a universal hash function with $\gamma \geq \lambda + \log \delta$ and:*

- *If $\mathbf{p} = \infty$, $H_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a universal hash function with $\lambda' \geq \lambda$.*
- *If $\mathbf{p} \neq \infty$, $H_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a random oracle with $\lambda' \geq \lambda + \mathbf{p} \cdot \log \delta$.*

Proof. For correctness, we will show that the protocol outputs are correct with overwhelming probability when both parties behave honestly. We will handle the cases $\mathbf{p} = \infty$ and $\mathbf{p} \neq \infty$ separately.

Correctness when $\mathbf{p} = \infty$. If $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) \leq \delta$, this implies that $w_i \in [q_i - \delta, q_i + \delta]$ for all $i \in [d]$. Hence $F_i(w_i) = r_i$ for all $i \in [d]$ by definition of $\mathcal{F}_{(d,1)\text{-OPPRF}}$, and thus $H_\lambda(\sum_{i=1}^d F_i(w_i)) = H_\lambda(0)$. If $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) > \delta$, this means that there exists at least one dimension $j \in [d]$ such that $w_j \notin [q_j - \delta, q_j + \delta]$, and thus $F_j(w_j)$ is distributed uniformly random in $\{0, 1\}^{\lambda'}$ by the definition of $\mathcal{F}_{(d,1)\text{-OPPRF}}$. Hence $\sum_{i=1}^d F_i(w_i) = 0$ with negligible probability by choosing $\lambda' \geq \lambda$.

Correctness when $\mathbf{p} \neq \infty$. If $\text{dist}_\mathbf{p}(\mathbf{w}, \mathbf{q}) \leq \delta$, then in particular $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) \leq \delta$ and $F_i(w_i) = r_i + s \cdot |w_i - q_i|^\mathbf{p}$ by the definition of $\mathcal{F}_{(d,1)\text{-OPPRF}}$. Thus $H_{\lambda'}(\sum_{i=1}^d F_i(w_i)) = H_{\lambda'}(r + s \cdot \text{dist}_\mathbf{p}(\mathbf{w}, \mathbf{q})^\mathbf{p}) \in S$ if $\text{dist}_\mathbf{p}(\mathbf{w}, \mathbf{q}) \leq \delta$. If $\text{dist}_\mathbf{p}(\mathbf{w}, \mathbf{q}) > \delta$, then two cases can occur. Either $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) > \delta$, in which case it follows similarly to the case $\mathbf{p} = \infty$ that the receiver outputs 1 with negligible probability. Or $\text{dist}_\infty(\mathbf{w}, \mathbf{q}) \leq \delta$, in which case $\sum_{i=1}^d F_i(w_i) = r + s \cdot \text{dist}_\mathbf{p}(\mathbf{w}, \mathbf{q})^\mathbf{p}$ but $\text{dist}_\mathbf{p}(\mathbf{w}, \mathbf{q})^\mathbf{p} > \delta$; thus $H_{\lambda'}(\sum_{i=1}^d F_i(w_i)) \in S$ with negligible probability.

To argue security, let us first consider a semi-honestly corrupted sender. Since the sender does not receive any output from the ideal functionality, nor any messages from the receiver, the simulation is trivial by simulating the ideal $\mathcal{F}_{(d,1)\text{-OPPRF}}$ functionality.

Now consider a semi-honestly corrupted receiver. The simulator again sends \mathbf{w} to $\mathcal{F}_{\text{FUZZYMATCH}}$ to obtain the output $b \in \{0, 1\}$. The simulator now simulates the adversary's view in different cases.

Corrupted receiver when $\mathbf{p} = \infty$. The simulator only needs to simulate the output from $\mathcal{F}_{(d,1)\text{-OPPRF}}$ by putting $|L_i| = 2\delta + 1$ for each $i \in [d]$, and sampling $\{F_i(w_i) \leftarrow_{\$} \{0, 1\}^{\lambda'}\}_{i \in [d]}$ such that $\sum_{i=1}^d F_i(w_i) = 0$ if $b = 1$, or sampling them randomly if $b = 0$. The simulated outputs are indistinguishable from the real protocol outputs. If $b = 0$, the receiver obtains $d - 1$ additive shares of 0 and some uniformly random evaluation (guaranteed by the functionality $\mathcal{F}_{(d,1)\text{-OPPRF}}$). They are all uniformly random. If $b = 1$, the receiver obtains d shares of 0 which is precisely the same as the simulated values.

Corrupted receiver when $\mathbf{p} \neq \infty$. The simulator needs to simulate the output of $\mathcal{F}_{(d,1)\text{-OPPRF}}$ and the set S . The simulator simulates $\mathcal{F}_{(d,1)\text{-OPPRF}}$ outputs $\{F_i(w_i)\}_{i \in [d]}$ by sampling them uniformly random from $\{0, 1\}^{\lambda'}$ and by putting $|L_i| = 2\delta + 1$ for each $i \in [d]$. Since the sender programs a uniformly random target value $r_i + s \cdot |j|^\mathbf{p}$ in the function F_i for all points $q_i - \delta, \dots, q_i + \delta$, and the receiver only obtains a single evaluation $F_i(w_i)$, which may or may not lie in $[q_i - \delta, q_i + \delta]$, for each dimension $i \in [d]$, the simulated outputs are indistinguishable from the real protocol outputs. If the ideal functionality outputs $b = 1$, the simulator computes $h := H_{\lambda'}(\sum_{i=1}^d F_i(w_i))$, inserts it in a random location $j_* \leftarrow_{\$} [\delta^\mathbf{p}]$ of S , and samples the remaining locations of S uniformly random from $\{0, 1\}^{\lambda'}$. If $b = 0$, the simulator samples all entries of S uniformly random from $\{0, 1\}^{\lambda'}$. The simulator sends S to the receiver. \square

7.1 Optimization from Prefix Trie Techniques

We can combine our fuzzy matching protocol from Figure 16 with the prefix trie techniques from Section 5 in multiple ways.

Infinity Distance. If $\mathbf{p} = \infty$, we can use the prefix trie technique to reduce the number of points programmed in the OPPRF for each dimension from $O(\delta)$ to $O(\log \delta)$, inspired by [1, Remark 2]. Furthermore, we manage to optimize the computational cost at the receiver side from $(\log \delta)^d$ in [1] to $(\log \delta)^{\frac{d}{2}}$. For points $\mathbf{q}, \mathbf{w} \in \mathbb{Z}_{2^u}^d$, we adapt the protocol as follows:

1. SENDER computes $(\tilde{q}_{i,1}, \dots, \tilde{q}_{i,\ell}) \leftarrow \text{PrefixTrie}(q_i - \delta, q_i + \delta)$ for each $i \in [d]$ and puts $\widetilde{\text{list}}_i := \{(\tilde{q}_{i,j}, r_i) \mid j \in [\ell]\}$, where $r_i \leftarrow_{\$} \{0,1\}^{\lambda'}$ such that $\sum_{i=1}^d r_i = 0^{\lambda'}$. SENDER pads each $\widetilde{\text{list}}_i$ to size ℓ_{\max} as defined in Theorem 2 with random pairs.
2. RECEIVER puts $\tilde{\mathbf{w}}_i := (\tilde{w}_{i,1}, \dots, \tilde{w}_{i,\ell'}) \leftarrow \text{PrefixPath}(w_i, \delta)$ for each $i \in [d]$.
3. RECEIVER inputs (RECEIVER, EVALUATE, $(\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_d)$) and SENDER inputs (SENDER, EVALUATE, $(\text{list}_1, \dots, \text{list}_d)$) to $\mathcal{F}_{(d,\ell')\text{-OPPRF}}$.
4. RECEIVER gets $(F_i(\tilde{w}_{i,j}))_{i \in [d], j \in [\ell']}$, SENDER gets $(\mathcal{O}^{F_i})_{i \in [d]}$ from $\mathcal{F}_{(d,\ell')\text{-OPPRF}}$.
5. RECEIVER checks if $\exists \mathbf{j} := (j_1, \dots, j_d) \in [\ell']^d$ such that $\sum_{i=1}^d F_i(\tilde{w}_{i,j_i}) = 0^{\lambda'}$, then outputs 1 if this is the case, and 0 otherwise.

This reduces the communication cost of the protocol to $O(d\lambda \log \delta)$ and the sender's computation cost to $O(d \log \delta)$, while increasing the receiver's computation cost to $O(d \log \delta + (\log \delta)^{\frac{d}{2}})$, compared to Figure 16. We argue that RECEIVER can run Step 5 in $O((\log \delta)^{\frac{d}{2}})$ time: this is a special case of the Knapsack problem which is known to be NP complete, however, we can search for the solution in slightly better time by using the standard “meet-in-the-middle” trick [26]. We can divide d lists of size ℓ' into roughly two halves, namely, $d_1 := \lceil \frac{d}{2} \rceil$ lists and $d_2 := \lfloor \frac{d}{2} \rfloor$ lists. Then we compute all possible sums of the first λ' bits for each half (e.g., $\sum_{i=1}^{d_1} (F_i(\tilde{w}_{i,j_i}))_{[1:\lambda']}$ for all \mathbf{j}), resulting in two sets of size $(\ell')^{d_1}$ and $(\ell')^{d_2}$. Checking if there is a match between two sets takes time $(\ell')^{d_2}$. In summary, RECEIVER takes $O((\log \delta)^{\lceil \frac{d}{2} \rceil})$ time and space to find a match in Step 5.

The correctness follows from Theorem 2, that is, RECEIVER obtains all of the shares r_i if and only if $w_i \in [q_i - \delta, q_i + \delta]$ for each $i \in [d]$. However, regarding the security, the vector $\mathbf{j}_* \in [\ell']^d$ that leads to a match, reveals more information about the distance between \mathbf{w} and \mathbf{q} . Security can be proven for an adapted fuzzy matching functionality with additional leakage where the receiver learns \mathbf{q} if $\text{dist}_{\mathbf{p}}(\mathbf{w}, \mathbf{q}) \leq \delta$, and nothing otherwise. This is fine for standard PSI applications where the receiver learns the sender's points close to theirs. Note that according to Theorem 2, the match is unique, which is crucial for the security to prevent RECEIVER learns multiple $F_i(\tilde{q}_{i,j})$ for some $i \in [d]$.

For an application where one requires sender privacy, the parties can run a standard PSI protocol with SENDER acting as the receiver with input S and RECEIVER acting as the sender with input $\{\mathbf{H}_{\lambda'}(\sum_{i=1}^d F_i(\tilde{w}_{i,j_i})) \mid (j_1, \dots, j_d) \in [\ell']^d\}$. Using a PSI protocol with linear complexity, this will add a factor $O((\log \delta)^d)$ to the communication and computation complexity, and will ultimately flip the role of the sender and receiver in the fuzzy matching protocol.

Minkowski Distance. If $\mathbf{p} \neq \infty$, the above method will unfortunately not work, but we can instead leverage the prefix trie technique to get rid of the $\delta^{\mathbf{p}}$ factor in the communication and computation complexity, which was left as an open question in [1]. For points $\mathbf{q}, \mathbf{w} \in \mathbb{Z}_{2^u}^d$, this can be done as follows:

1. SENDER sets $\text{val}_{i,j} := r_i + |j|^{\mathbf{p}}$ instead of $r_i + s \cdot |j|^{\mathbf{p}}$, which has the effect that SENDER can represent the interval $[r, r + \delta^{\mathbf{p}}]$ using the prefix trie technique as $S := (\tilde{r}_1, \dots, \tilde{r}_{\ell}) \leftarrow \text{PrefixTrie}(r, r + \delta^{\mathbf{p}})$. Padding S to size ℓ_{\max} , where $\ell_{\max} = O(\mathbf{p} \log \delta)$ as defined in Theorem 2.
2. RECEIVER now gets $U := (\tilde{u}_1, \dots, \tilde{u}_{\ell'}) \leftarrow \text{PrefixPath}(u, \delta^{\mathbf{p}}/2)$, where $u := \sum_{i=1}^d F_i(w_i)$ and $\ell' = O(\mathbf{p} \log \delta)$.
3. Now the parties can run a PSI protocol between S and U to let RECEIVER learn \tilde{u}_{j_*} for the $j_* \in [\ell']$ that leads to a match when $u \in [r, r + \delta^{\mathbf{p}}]$.

Using a PSI protocol with linear complexity, this will reduce the communication cost of the protocol to $O(\delta d \lambda + p \lambda \log \delta)$ sender's computational cost to $O(\delta d + p \log \delta)$, while increasing the receiver's computational cost to $O(d + p \log \delta)$.

However, the item \tilde{u}_{j_*} that leads to a match reveals more information about the distance between \mathbf{w} and \mathbf{q} when $\text{dist}(\mathbf{w}, \mathbf{q}) \leq \delta$. The protocol again realizes an adapted fuzzy matching functionality with additional leakage where the receiver learns the sender's point \mathbf{q} if $\text{dist}(\mathbf{q}, \mathbf{w}) \leq \delta$ and nothing otherwise, which is fine for standard PSI applications where the receiver learns the close sender's points. For an application where one requires sender privacy, one can instead use a PSI cardinality protocol to compare S and U .

8 Fuzzy PSI from OPPRF

In this section, we expand our OPPRF-based fuzzy matching protocols to support fuzzy PSI. We start by introducing an efficient protocol for standard fuzzy PSI, then work on adapting it to ensure sender privacy (allowing the receiver to only learn which of its own points are close to the sender's points). After that, we apply prefix trie techniques to our fuzzy PSI protocols to achieve a better trade-off in terms of complexity. Finally, we explore additional functionalities, such as labeled PSI, PSI with cardinality, and circuit PSI.

8.1 Minkowski Distance

To move from single-point fuzzy matching to the multiple-point fuzzy PSI setting, we leverage spatial hashing techniques (see Section 3.5), similar to [1], by tiling the space with cells of side-length 2δ . If the sender's points are distance $2\delta d^{\frac{1}{p}}$ apart, there is at most one of their points lying in each cell by Lemma 3. The sender can therefore program a fuzzy matching instance for \mathbf{q}_k in the OPPRF at $\mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{q}_k)$ for each $k \in [M]$. The receiver now needs to evaluate the fuzzy matching instances at the at most 2^d cells $\mathcal{C}_{k,j}$ intersecting $\text{ball}_\delta(\mathbf{w}_k)$, for each $k \in [N]$, to check for potential matches. To make sure that the receiver does not notice any collisions in the OPPRF outputs, we additionally require the receiver's points to be distance $2\delta(d^{\frac{1}{p}} + 1)$ apart, which guarantees that there is at most one $\text{ball}_\delta(\mathbf{w}_k)$ intersecting with the each cell \mathcal{C} by Lemma 4. This results in the protocol Π_{FUZZYPSI}^p in Figure 17, whose complexity is given in Lemma 8. and security is proven in Theorem 6.

Lemma 8. *The protocol Π_{FUZZYPSI}^p has*

- *sender's computational complexity $O(\delta d M + d 2^d N)$ if $p = \infty$, and $O(\delta d M + \delta^p M + d 2^d N)$ if $p \neq \infty$;*
- *receiver's computational complexity $O(d 2^d N)$;*
- *communication complexity $O(\delta d M \lambda + d 2^d N \lambda)$ if $p = \infty$, and $O(\delta d M \lambda + \delta^p M \lambda + d 2^d N \lambda)$ if $p \neq \infty$;*

ignoring logarithmic factors in all complexities.

Proof. These complexities can be achieved similarly to Lemma 7 with subfield size $\log_2 |\mathbb{B}| := \gamma$ and OPPRF output length $\log_2 |\mathcal{Y}| := \lambda'$ where λ' is as in Theorem 6. \square

Theorem 6. *The protocol Π_{FUZZYPSI}^p realizes the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ for L_p distance, $p \in [1, \infty]$, with standard PSI leakage against semi-honest adversaries in the $\mathcal{F}_{(d, 2^d \cdot N)\text{-OPPRF}}$ -hybrid model if*

Parameters : RECEIVER with input $\mathbf{W} = (\mathbf{w}_1, \dots, \mathbf{w}_N) \in \mathbb{Z}^{d \times N}$ and SENDER with input $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_M) \in \mathbb{Z}^{d \times M}$. Dimension d and radius δ .

Protocol :

1. For $k \in [M]$, SENDER puts $\mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{q}_k)$, samples $r_k, s_k \leftarrow_{\$} \{0, 1\}^{\lambda'}$.
2. For $i \in [d]$, SENDER updates $\text{list}_i \leftarrow \text{list}_i \cup \text{list}_{k,i}$, where $(\text{list}_{k,1}, \dots, \text{list}_{k,d}) \leftarrow \text{GetList}_p(r_k, s_k, \mathbf{q}_k, \mathcal{C}_k)$ and $k \in [M]$.
3. RECEIVER sends $(\text{RECEIVER, EVALUATE, } (\tilde{\mathbf{W}}_1, \dots, \tilde{\mathbf{W}}_d))$ and SENDER sends $(\text{SENDER, EVALUATE, } (\text{list}_1, \dots, \text{list}_d))$ to $\mathcal{F}_{(d, 2^d \cdot N)\text{-OPPRF}}$, where

$$\forall i \in [d], \tilde{\mathbf{W}}_i := (\mathbf{H}_\gamma(w_{k,i} \| \mathcal{C}_{k,j}))_{k \in [N], j \in [2^d]},$$

- $\mathcal{C}_{k,j}$ ranges over all cells intersecting $\text{ball}_\delta(\mathbf{w}_k)$, and each $\tilde{\mathbf{W}}_i$ is padded with random items to size $2^d \cdot N$.
4. RECEIVER gets $(F_i(\mathbf{H}_\gamma(w_{k,i} \| \mathcal{C}_{k,j})))_{i \in [d], k \in [N], j \in [2^d]}$ and SENDER gets $(\mathcal{O}^{F_i})_{i \in [d]}$ from $\mathcal{F}_{(d, 2^d \cdot N)\text{-OPPRF}}$.
 5. RECEIVER computes $u_{k,j} := \sum_{i=1}^d F_i(\mathbf{H}_\gamma(w_{k,i} \| \mathcal{C}_{k,j}))$ for all $k \in [N]$, $j \in [2^d]$.
 6. SENDER puts
 - $S := \{\mathbf{H}_{\lambda'+du}(r_k) \oplus (0^{\lambda'} \| \mathbf{q}_k) \mid k \in [M]\}$ if $p = \infty$;
 - $S := \{\mathbf{H}_{\lambda'+du}(r_k + s_k \cdot j) \oplus (0^{\lambda'} \| \mathbf{q}_k) \mid k \in [M], j \in [\delta^p]\}$ if $p \neq \infty$;
 then shuffles S and sends it to RECEIVER.
 7. RECEIVER sets $I := \emptyset$, and, for each $k \in [N]$, if there exists $j_* \in [2^d]$ such that $\mathbf{H}_{\lambda'+du}(u_{k,j_*}) \oplus s = 0^{\lambda'} \| \mathbf{q}$ for some $s \in S$ and $\mathbf{q} \in \mathbb{Z}_{2^u}^d$, updates $I \leftarrow I \cup \{\mathbf{q}\}$.

Fig. 17. Fuzzy standard PSI for Minkowski distance $p \in [1, \infty]$, sender's points $2\delta d^{\frac{1}{p}}$ apart (2δ apart when $p = \infty$), receiver's points $2\delta(d^{\frac{1}{p}} + 1)$ apart (4δ apart when $p = \infty$).

the sender's points are $2\delta d^{\frac{1}{p}}$ apart (2δ when $p = \infty$) and the receiver's points are $2\delta(d^{\frac{1}{p}} + 1)$ apart (4δ apart when $p = \infty$). Moreover, $\mathbf{H}_\gamma : \{0, 1\}^* \rightarrow \{0, 1\}^\gamma$ is a universal hash function with $\gamma \geq \lambda + d + \log(\delta NM)$, $\mathbf{H}_{\lambda'+du} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'+du}$ is a random oracle, and:

- If $p = \infty$, $\lambda' \geq \lambda + d + \log N$;
- If $p \neq \infty$, $\lambda' \geq \lambda + d + \log N + \log(\delta^p M)$.

Proof. First consider correctness. Since the sender's points are $2\delta d^{\frac{1}{p}}$ apart, there is at most one point \mathbf{q} lying in each cell by Lemma 3. Hence there are no collisions in the keys of each list_i , $i \in [d]$. Now since the receiver's points are $2\delta(d^{\frac{1}{p}} + 1)$ apart, the cells $\{\mathcal{C}_{k,j}\}_{j \in [2^d]}$ and $\{\mathcal{C}_{k',j}\}_{j \in [2^d]}$ are disjoint for different points $\mathbf{w}_k, \mathbf{w}_{k'}$ by Lemma 4. Moreover, each $\text{ball}_\delta(\mathbf{w})$ intersects with at most 2^d cells, and any point \mathbf{q} with $\text{dist}_p(\mathbf{w}, \mathbf{q}) \leq \delta$ must lie in one of these cells. The correctness now follows from the correctness of the fuzzy matching protocol Theorem 5.

Assume the sender is semi-honestly corrupted. Since the sender does not receive any output from the ideal functionality, nor any messages from the sender, the simulation is trivial by simulating the $\mathcal{F}_{(d, 2^d \cdot N)\text{-OPPRF}}$ outputs.

Now assume the receiver is semi-honestly corrupted. The simulator receives the intersection $I = \{\mathbf{q}_j \mid \exists k \in [N], \text{dist}_p(\mathbf{w}_k, \mathbf{q}_j) \leq \delta\}$ from the ideal functionality, and uses this to get $\mathcal{C}_j \leftarrow \text{cell}_{2\delta}(\mathbf{q}_j)$ and the corresponding \mathbf{w}_{k_j} with $\text{dist}_p(\mathbf{w}_{k_j}, \mathbf{q}_j)$ for each $\mathbf{q}_j \in I$. The simulator simulates the $\mathcal{F}_{(d, 2^d \cdot N)\text{-OPPRF}}$ outputs $(F_i(\mathbf{H}_\gamma(w_{k,i} \| \mathcal{C}_{k,j})))_{i \in [d], k \in [N], j \in [2^d]}$ by sampling them uniformly random from $\{0, 1\}^{\lambda'}$ and sets the corresponding programmed set size $M \cdot (2\delta + 1)$. Note that the simulated

OPPRF outputs are indistinguishable from the real protocol execution similar to Theorem 5: the receiver obtains evaluations for different cells $\{\mathcal{C}_{k,j}\}_{j \in [2^d]}$ for each $k \in [N]$, since the receiver's points are $2\delta(d^{\frac{1}{p}} + 1)$ apart; and the sender programs $\{r_{k,i}\}_{i \in [d]}$ as independent random target values for each cell $\mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{q}_k)$, $k \in [M]$. Finally, the simulator computes $H_{\lambda'+du}(\sum_{i=1}^d F_i(H_\gamma(w_{k,j,i} \parallel \mathcal{C}_j))) \oplus (0^{\lambda'} \parallel \mathbf{q}_j)$ for each $\mathbf{q}_j \in I$, inserts these in random locations of S , and samples the remaining entries uniformly random from $\{0, 1\}^{\lambda'+du}$. \square

Remark 2 (Limited Overlap). In the infinity distance setting $p = \infty$, the sender can instead of encoding fuzzy matching instances at the $\mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{q}_k)$ assign a block $\mathcal{B}_k \leftarrow \text{block}_{4\delta}(\mathbf{q}_k)$ of side-length 4δ and encode the fuzzy matching instance at this block. The receiver can now similarly iterate over all blocks $\mathcal{B}_{k,j}$ containing $\text{cell}_{2\delta}(\mathbf{w}_k)$. Now, similarly to [20, Section 5.2], as long as the sender can assign a unique block to each point $k \in [M]$, they can program the OPPRF correctly. As a result, we can allow the sender's balls to have limited overlap under this condition (i.e., the points \mathbf{q}_k can be distance $< 2\delta$ from each other). The receiver will again not learn any collisions from the OPPRF evaluations since they will iterate over each block \mathcal{B} at most once if their points are distance 4δ apart.

8.2 Sender Privacy

The protocol Π_{FUZZYPSI}^p from Figure 17 realizes the standard fuzzy PSI functionality where the receiver learns the sender's points $\mathbf{q} \in \mathbf{Q}$ that lie close to their points, that is, for which there exists $\mathbf{w} \in \mathbf{W}$ such that $\text{dist}_p(\mathbf{w}, \mathbf{q}) \leq \delta$. However, for some applications it might be desirable to not reveal the exact point \mathbf{q} , but only reveal that the sender has a point lying close to the receiver's point \mathbf{w} . We refer to this variant as fuzzy PSI with *sender privacy* (PSI-SP), see Figure 2 for the ideal functionality. There are two ways we can adapt the protocol in Figure 17 to realize this functionality, which we will detail below.

Infinity Distance. In the infinity distance case, $p = \infty$, we can let the parties run a normal PSI protocol in the end (replacing the Step 6 and 7 in Figure 17), where the sender acts as receiver and inputs the set $S := \{\sum_{i=1}^d r_{k,i}\}_{k \in [M]}$, whereas the receiver acts as sender and inputs the set $U := \{u_{k,j}\}_{j \in [N], j \in [2^d]}$. This effectively flips the role of the sender and the receiver in the resulting protocol. For completeness, we present the protocol in Figure 18 and prove security in Theorem 7. The complexity of the resulting protocol is given in Lemma 9. Note that this approach does not work in the $p \neq \infty$ case, since the intersection $S \cap U$ reveals the exact distance between the matching points to the receiver.

Lemma 9. *The protocol $\Pi_{\text{FUZZYPSI-SP}}^\infty$ has computational complexity $O(d2^d M + N)$ for the sender, computational complexity $O(\delta d N + d2^d M)$ for the receiver, and communication complexity $O(\delta d N \lambda + d2^d M \lambda)$, ignoring logarithmic values.*

Proof. By instantiating \mathcal{F}_{PSI} by a PSI protocol with linear communication and sender computation in the size of both input sets, and receiver computation linear in the receiver's set size. For example, by an OPRF-based PSI protocol such as [36]. Moreover, we instantiate $\mathcal{F}_{(d,2^d,M)\text{-OPPRF}}$ by $\Pi_{(d,2^d,M)\text{-OPPRF}}$ from Figure 14 as in Lemma 7, with the parameters λ' and γ chosen as in Theorem 7. \square

Theorem 7. *The protocol $\Pi_{\text{FUZZYPSI-SP}}^\infty$ realizes the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ for L_∞ distance with PSI-SP leakage against semi-honest adversaries in the $\mathcal{F}_{\text{PSI}}, \mathcal{F}_{(d,N)\text{-OPPRF}}$ -hybrid model if the*

$\Pi_{\text{FUZZYPSI-SP}}^\infty$

Parameters : RECEIVER with input $\mathbf{W} = (\mathbf{w}_1, \dots, \mathbf{w}_N) \in \mathbb{Z}^{d \times N}$ and SENDER with input $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_M) \in \mathbb{Z}^{d \times M}$. Dimension d and radius δ .

Protocol :

1. For each $k \in [N]$, RECEIVER puts $\mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{w}_k)$, samples $r_k \leftarrow \$ \{0, 1\}^{\lambda'}$, and updates $\text{list}_i \leftarrow \text{list}_i \cup \text{list}_{k,i}$ for each $i \in [d]$, where $(\text{list}_{k,1}, \dots, \text{list}_{k,d}) \leftarrow \text{GetList}_p(r_k, 0, \mathbf{w}_k, \mathcal{C}_k)$.
2. SENDER sends $(\text{RECEIVER}, \text{EVALUATE}, (\tilde{\mathbf{Q}}_1, \dots, \tilde{\mathbf{Q}}_d))$ and RECEIVER sends $(\text{SENDER}, \text{EVALUATE}, (\text{list}_1, \dots, \text{list}_d))$ to $\mathcal{F}_{(d, 2^d \cdot M)\text{-OPPRF}}$, where

$$\forall i \in [d], \tilde{\mathbf{Q}}_i := (\mathbf{H}_\gamma(q_{k,i} \| \mathcal{C}_{k,j}))_{k \in [M], j \in [2^d]},$$

$\mathcal{C}_{k,j}$ ranges over all cells intersecting $\text{ball}_\delta(\mathbf{q}_k)$, and each $\tilde{\mathbf{Q}}_i$ is padded with random items to size $2^d \cdot M$.

3. SENDER gets $(F_i(\mathbf{H}_\gamma(q_{k,i} \| \mathcal{C}_{k,j})))_{i \in [d], k \in [M], j \in [2^d]}$ and RECEIVER gets $(\mathcal{O}^{F_i})_{i \in [d]}$ from $\mathcal{F}_{(d, 2^d \cdot M)\text{-OPPRF}}$.
4. RECEIVER puts $S := \{\mathbf{H}_{\lambda'}(r_k) \mid k \in [N]\}$.
5. SENDER puts $U := \{\mathbf{H}_{\lambda'}(\sum_{i=1}^d F_i(\mathbf{H}_\gamma(q_{k,i} \| \mathcal{C}_{k,j}))) \mid k \in [M], j \in [2^d]\}$.
6. RECEIVER sends $(\text{RECEIVER}, \text{EVALUATE}, S)$ and SENDER sends $(\text{SENDER}, \text{EVALUATE}, U)$ to \mathcal{F}_{PSI} .
7. RECEIVER gets $S \cap U$ from \mathcal{F}_{PSI} and outputs $I := \{\mathbf{w}_k \mid k \in [N], \mathbf{H}_{\lambda'}(r_k) \in S \cap U\}$.

Fig. 18. Fuzzy PSI-SP for infinity distance, making black-box use of PSI, sender's points 4δ apart, receiver's points 2δ apart.

sender's points are 4δ apart and the receiver's points are 2δ apart. Moreover, $\mathbf{H}_\gamma : \{0, 1\}^* \rightarrow \{0, 1\}^\gamma$ and $\mathbf{H}_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ are universal hash functions with $\gamma \geq \lambda + d + \log(\delta NM)$ and $\lambda' \geq \lambda + d + \log M + \log N$.

Proof. Correctness of the protocol follows from the correctness of the protocol Π_{FUZZYPSI}^p from Theorem 6, since by definition the ideal functionality \mathcal{F}_{PSI} outputs the values $\mathbf{H}_{\lambda'}(r_k)$, $k \in [N]$, for which there exists $k' \in [M]$ and $j \in [2^d]$ such that $\mathbf{H}_{\lambda'}(\sum_{i=1}^d F_i(\mathbf{H}_\gamma(q_{k',i} \| \mathcal{C}_{k',j}))) = \mathbf{H}_{\lambda'}(r_k)$.

Simulation proceeds similarly to the proof of Theorem 6, except that the simulator for a corrupt receiver now obtains S from the receiver's input to \mathcal{F}_{PSI} and inserts the k -th element of S into the output from \mathcal{F}_{PSI} if \mathbf{w}_k is in the output from $\mathcal{F}_{\text{FUZZYPSI}}$. \square

Minkowski Distance. For general Minkowski distance, $p \in [1, \infty]$, we can basically take a “dual” approach to the protocol in Figure 17, where now the sender iterates over all cells intersecting $\text{ball}_\delta(\mathbf{q}_k)$ when programming the OPPRF and the receiver just receives the evaluation at a single cell for each point \mathbf{w}_k . This means that now the sender's points have to be $2\delta(d^{\frac{1}{p}} + 1)$ apart (4δ when $p = \infty$) and the receiver's points $2\delta d^{\frac{1}{p}}$ apart (2δ when $p = \infty$). The protocol is given in 19 and security is proven in Theorem 8, following the discussion about the complexity in Lemma 10.

In the case that $p = \infty$, compared to the protocol in Figure 18, the protocol from Figure 19 will be desirable in a setting where $N > 2^d \cdot M$ and $\delta > 1$. To see this, the overhead in Figure 18 is $C_1 = \delta Nd + d2^d M$, whereas the overhead in Figure 19 is $C_2 = \delta d2^d M + dN$. If $k = \frac{N}{2^d M} > 1$, we have $C_1 - C_2 = dN(\delta - 1)(1 - \frac{1}{k}) > 0$.

Lemma 10. *The protocol $\Pi_{\text{FUZZYPSI-SP}}^p$ has computation complexity $O(\delta d2^d M + dN)$ for the sender if $p = \infty$ and $O(\delta d2^d M + \delta^p M + dN)$ if $p \neq \infty$, computation complexity $O(dN)$ for the receiver, and communication complexity $O(\delta d2^d M \lambda + dN \lambda)$ if $p = \infty$, $O(\delta d2^d M \lambda + \delta^p M \lambda + dN \lambda)$ if $p \neq \infty$, ignoring logarithmic factors in all complexities.*

$\Pi_{\text{FUZZYPSI-SP}}^{\mathbf{p}}$

Parameters : RECEIVER with input $\mathbf{W} = (\mathbf{w}_1, \dots, \mathbf{w}_N) \in \mathbb{Z}^{d \times N}$ and SENDER with input $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_M) \in \mathbb{Z}^{d \times M}$. Dimension d and radius δ .

Protocol :

1. For each $k \in [M]$, SENDER samples $r_k, s_k \leftarrow \{0, 1\}^{\lambda'}$ and sets $r_k = 0$ when $\mathbf{p} = \infty$.
2. For each $k \in [M]$ and each cell \mathcal{C}_j intersecting $\text{ball}_\delta(\mathbf{q}_k)$, SENDER gets

$$(\text{list}_{k,j,1}, \dots, \text{list}_{k,j,d}) \leftarrow \text{GetList}_{\mathbf{p}}(r_k, s_k, \mathbf{q}_k, \mathcal{C}_j)$$

and updates $\text{list}_i \leftarrow \text{list}_i \cup \text{list}_{k,j,i}$ for each $i \in [d]$. Pads each list_i to size $2^d \cdot M(2\delta + 1)$ with random pairs.

3. RECEIVER sends $(\text{RECEIVER}, \text{EVALUATE}, (\tilde{\mathbf{W}}_1, \dots, \tilde{\mathbf{W}}_d))$ and SENDER sends $(\text{SENDER}, \text{EVALUATE}, (\text{list}_1, \dots, \text{list}_d))$ to $\mathcal{F}_{(d,N)\text{-OPPRF}}$, where

$$\forall i \in [d], \tilde{\mathbf{W}}_i := (\mathbf{H}_\gamma(w_{k,i} \parallel \mathcal{C}_k))_{k \in [N]}, \mathcal{C}_k \leftarrow \text{cell}_{2\delta}(\mathbf{w}_k).$$

4. RECEIVER gets $(F_i(\mathbf{H}_\gamma(w_{k,i} \parallel \mathcal{C}_k)))_{i \in [d], k \in [N]}$ and SENDER gets $(\mathcal{O}^{F_i})_{i \in [d]}$ from $\mathcal{F}_{(d,N)\text{-OPPRF}}$.
5. If $\mathbf{p} \neq \infty$, SENDER puts $S := \{\mathbf{H}_{\lambda'}(r_k + s_k \cdot j) \mid k \in [M], j \in [\delta^{\mathbf{p}}]\}$, shuffles S and sends it to RECEIVER.
6. If $\mathbf{p} = \infty$, RECEIVER puts $S := \{\mathbf{H}_{\lambda'}(0)\}$.
7. RECEIVER computes $u_k := \mathbf{H}_{\lambda'}\left(\sum_{i=1}^d F_i(\mathbf{H}_\gamma(w_{k,i} \parallel \mathcal{C}_k))\right)$ for all $k \in [N]$.
8. RECEIVER sets $I := \emptyset$, and, for each $k \in [N]$, if $u_k \in S$, updates $I \leftarrow I \cup \{(\mathbf{w}_k)\}$.

Fig. 19. Fuzzy PSI-SP for Minkowski distance $\mathbf{p} \in [1, \infty]$, sender's points $2\delta(d^{\frac{1}{\mathbf{p}}} + 1)$ apart (4δ apart when $\mathbf{p} = \infty$), receiver's points $2\delta d^{\frac{1}{\mathbf{p}}}$ apart (2δ apart when $\mathbf{p} = \infty$).

Proof. These complexities can be achieved similarly to Lemma 7 with subfield size $\log_2 |\mathbb{B}| := \gamma$ and OPRF output length $\log_2 |\mathcal{Y}| := \lambda'$ as in Theorem 8. \square

Theorem 8. The protocol $\Pi_{\text{FUZZYPSI-SP}}^{\mathbf{p}}$ realizes the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ for $L_{\mathbf{p}}$ distance, $\mathbf{p} \in [1, \infty]$, with PSI-SP leakage against semi-honest adversaries in the $\mathcal{F}_{(d,N)\text{-OPPRF}}$ -hybrid model if the sender's points are $2\delta(d^{\frac{1}{\mathbf{p}}} + 1)$ apart (4δ apart when $\mathbf{p} = \infty$) and the receiver's points are $2\delta d^{\frac{1}{\mathbf{p}}}$ apart (2δ apart when $\mathbf{p} = \infty$). Moreover, $\mathbf{H}_\gamma : \{0, 1\}^* \rightarrow \{0, 1\}^\gamma$ is a universal hash functions with $\gamma \geq \lambda + d + \log(\delta NM)$ and:

- If $\mathbf{p} = \infty$, $\mathbf{H}_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a universal hash function, $\lambda' \geq \lambda + \log N$;
- If $\mathbf{p} \neq \infty$, $\mathbf{H}_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a random oracle, $\lambda' \geq \lambda + \log N + \log(\delta^{\mathbf{p}} M)$.

Proof. First, we argue correctness. For any receiver's point \mathbf{w} , since the sender's points are $2\delta(d^{\frac{1}{\mathbf{p}}} + 1)$ apart, there exists at most a single sender's point \mathbf{q} such that $\text{ball}_\delta(\mathbf{q})$ intersects $\text{cell}_{2\delta}(\mathbf{w})$ by Lemma 4. Moreover, the ball $\text{ball}_\delta(\mathbf{q})$ intersects with at most 2^d cells and any receiver's point \mathbf{w} with $\text{dist}_{\mathbf{p}}(\mathbf{w}, \mathbf{q}) \leq \delta$ must lie in one of these cells. Moreover, since the receiver's points are $2\delta d^{\frac{1}{\mathbf{p}}}$ apart, there is at most one of the receiver's points \mathbf{w} lying in each cell by Lemma 3. Finally, since the sender's points are $2\delta(d^{\frac{1}{\mathbf{p}}} + 1)$ apart, the cells $\{\mathcal{C}_{k,j}\}_{j \in [2^d]}$ and $\{\mathcal{C}_{k',j}\}_{j \in [2^d]}$ are disjoint for different points $\mathbf{q}_k, \mathbf{q}_{k'}$ by Lemma 4, thus there are no collisions in the keys of each list_i , $i \in [d]$. Hence the correctness of the fuzzy PSI protocol follows from the correctness of the fuzzy matching protocol Theorem 5.

Consider a semi-honestly corrupted sender. Since the sender does not receive any output from the ideal functionality, nor any messages from the sender, the simulation is trivial by simulating the $\mathcal{F}_{(d,N)\text{-OPPRF}}$ outputs.

Now consider a semi-honestly corrupted receiver. The simulator receives the intersection $I := \{\mathbf{w}_k \mid \exists j \in [M], \text{dist}_{\mathbf{p}}(\mathbf{w}_k, \mathbf{q}_j) \leq \delta\}$ from the ideal functionality. The simulator can simulate the $\mathcal{F}_{(d,N)\text{-OPPRF}}$ outputs $F_i(\mathbf{H}_{\gamma}(w_{k,i} \parallel \mathcal{C}_k))$ by sampling them uniformly random from $\{0, 1\}^{\lambda'}$ (or additive shares 0 when $\mathbf{p} = \infty$), and send the corresponding programmed set size $2^d \cdot M(2\delta + 1)$. The simulated outputs are indistinguishable from the real protocol OPPRF outputs since the sender programs independent d -out-of- d sharings $\{r_{k,j,i}\}_{i \in [d]}$ of a random value r_k (or 0) for each cell \mathcal{C}_j intersecting $\text{ball}_{\delta}(\mathbf{q}_k)$ and, since the receiver's points are $2\delta d^{\frac{1}{d}}$ apart, they decode at most one point $\mathbf{w}_{k'}$ for each cell $\mathcal{C}_{j'}$ by Lemma 3, and can not have two points $\mathbf{w}_{k'}, \mathbf{w}_{k''}$ in different cells $\mathcal{C}_{j'}, \mathcal{C}_{j''}$ both lying in $\text{ball}_{\delta}(\mathbf{q}_k)$. Finally, for each $\mathbf{w}_k \in I$, the simulator adds $\mathbf{H}_{\lambda'}(\sum_{i=1}^d F_i(\mathbf{H}_{\gamma}(w_{k,i} \parallel \mathcal{C}_k)))$ to a random location in S , and samples the remaining entries uniformly random from $\{0, 1\}^{\lambda'}$. \square

8.3 Optimization from Prefix Trie Techniques

In this section, we will explore how the results of Section 7.1 on combining the prefix trie techniques from Section 5 with our fuzzy matching protocol transfer to the fuzzy PSI protocols presented in the previous sections.

Infinity Distance. Combining the protocols in Figures 17, for $\mathbf{p} = \infty$, with the prefix trie technique discussed in Section 7.1 requires using a weak labeled PSI (as in Section 4)⁹. The sender generates the lists of points to be programmed in the OPPRF in the same way, except replacing the interval $(q_{k,i} + j)_{j \in [-\delta, \delta]}$ by the list of prefixes $(\tilde{q}_{k,i,j})_{j \in [\ell]} \leftarrow \text{PrefixTrie}(q_{k,i} - \delta, q_{k,i} + \delta)$ for each $k \in [M]$, $i \in [d]$, and pads each list to size ℓ_{\max} with random strings. Additionally, the sender replaces $r_k \leftarrow 0^{\lambda} \parallel v_k$ for each $k \in [M]$ in Figure 17, where $v_k \leftarrow_{\$} \{0, 1\}^{\sigma}$ and $\sigma = O(\lambda)$. Similarly, for each receiver's point to be evaluated under the OPPRF, the role of $w_{k,i}$ is replaced by $(\tilde{w}_{k,i,j})_{j \in [\ell]} \leftarrow \text{PrefixPath}(w_{k,i}, \delta)$, increasing the number of points by a factor $\ell' = O(\log \delta)$.

Furthermore, Step 5 of the protocol again changes as the receiver now additionally needs to iterate over $\mathbf{j} \in [\ell']^d$ to find whether these OPPRF evaluations sum to $0^{\lambda} \parallel v$ for some $v \in \{0, 1\}^{\sigma}$, where we can again use the “meet-in-the-middle” trick sketched in Section 7.1 to find such v .

In the end, instead of following Step 6 and 7, both parties run a weak labeled PSI protocol between $S' := \{(v_k, \mathbf{q}_k) \mid k \in [M]\}$ and $U := \{v_k \mid \mathbf{q}_k \text{ is close to some } \mathbf{w}\}$ such that the receiver learns (v_k, \mathbf{q}_k) for each \mathbf{q}_k which is close to receiver's points. Notably, the adapted versions of all these protocols realize the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ with *standard PSI* leakage. The Figure 19 can be adapted similarly with standard PSI leakage. The complexities of the resulting protocols can be found in Table 3.

The protocol in Figure 18 is adapted similarly but with the roles of sender and receiver reversed and the final comparison is made by the ideal PSI functionality \mathcal{F}_{PSI} . Because of this, the adapted protocol realizes $\mathcal{F}_{\text{FUZZYPSI}}$ with PSI-SP leakage. Sender privacy can similarly be achieved for the other adapted protocols by reversing the parties' roles and letting the final comparison step be done by a standard PSI functionality. Note that the protocol in Figure 17 adapted in this way becomes identical to the adapted version of the protocol in Figure 18. Moreover, to preserve sender privacy, we require the sender to program uniformly random values instead of additive shares of 0,

⁹ Note that Figure 20 is also compatible with the prefix trie techniques but the computation cost scales to $(\log \delta)^{\frac{d}{2}}$, which does not make too much sense in high-dimensional spaces.

Table 3. Complexities for fuzzy standard PSI protocols for infinity distance using prefix trie encoding.

| Protocol | Communication | Sender comp. | Receiver comp. |
|----------|--------------------------------------|-------------------------------|---|
| Fig. 17 | $O((M + 2^d N)d\lambda \log \delta)$ | $O((M + 2^d N)d \log \delta)$ | $O(2^d N(d \log \delta + (\log \delta)^{\frac{d}{2}}))$ |
| Fig. 19 | $O((2^d M + N)d\lambda \log \delta)$ | $O((2^d M + N)d \log \delta)$ | $O(N(d \log \delta + (\log \delta)^{\frac{d}{2}}))$ |

Table 4. Complexities for fuzzy PSI with sender privacy for infinity distance using prefix trie encoding.

| Protocol | Communication | Sender comp. | Receiver comp. |
|----------|--|---|-------------------------------|
| Fig. 18 | $O((2^d M + N)d\lambda \log \delta + 2^d M\lambda(\log \delta)^d)$ | $O(2^d M(d \log \delta + (\log \delta)^d) + N)$ | $O((2^d M + N)d \log \delta)$ |
| Fig. 19 | $O((M + 2^d N)d\lambda \log \delta + M\lambda(\log \delta)^d)$ | $O(M(d \log \delta + (\log \delta)^d) + N)$ | $O((M + 2^d N)d \log \delta)$ |

Table 5. Complexities for fuzzy standard PSI protocols for Minkowski distance using prefix trie encoding.

| Protocol | Communication | Sender comp. | Receiver comp. |
|----------|---|--|--|
| Fig. 17 | $O((\delta M + 2^d N)d\lambda + (M + 2^d N)\lambda \mathbf{p} \log \delta)$ | $O((\delta M + 2^d N)d + (M + 2^d N)\mathbf{p} \log \delta)$ | $O(2^d N(d + \mathbf{p} \log \delta))$ |
| Fig. 19 | $O((\delta 2^d M + N)d\lambda + (M + N)\lambda \mathbf{p} \log \delta)$ | $O((\delta 2^d M + N)d + (M + N)\mathbf{p} \log \delta)$ | $O(N(d + \mathbf{p} \log \delta))$ |
| Fig. 20 | $O((\delta M + N)d\lambda + (M + N)\lambda \mathbf{p} \log \delta)$ | $O((\delta M + N)d + (M + N)\mathbf{p} \log \delta)$ | $O(N(d + \mathbf{p} \log \delta))$ |

which renders our “meet-in-the-middle” trick mentioned in Section 7.1 no longer available. The complexities of the resulting protocols can be found in Table 4.

Minkowski Distance. In the $\mathbf{p} \neq \infty$ setting, as discussed in Section 7.1, we can use the prefix trie technique to reduce the $\delta^{\mathbf{p}}$ factor in the communication to $\mathbf{p} \log \delta$. We describe the adaptations for the protocol in Figure 17, but the protocols in Figures 19 and 20 can be adapted similarly. In all `GetList` calls, the sender replaces $s_k \leftarrow 1$, and forms the set S as $S \leftarrow S \cup \{\tilde{r}_{k,1}, \dots, \tilde{r}_{k,\ell}\}$ where $(\tilde{r}_{k,1}, \dots, \tilde{r}_{k,\ell}) \leftarrow \text{PrefixTrie}(r_k, r_k + \delta^{\mathbf{p}})$ for each $k \in [M]$, and pads S with random values to have size $\ell_{\max} \cdot M$. The receiver proceeds as usual, but instead puts $u_{k,j} := \sum_{i=1}^d F_i(\mathbf{H}_\gamma(w_{k,i} \| \mathcal{C}_{k,j}))$ and computes $(\tilde{u}_{k,j,1}, \dots, \tilde{u}_{k,j,\ell'}) \leftarrow \text{PrefixPath}(u_{k,j}, \delta^{\mathbf{p}}/2)$ for each $k \in [N]$, $j \in [2^d]$. Then the receiver forms the set $U := \{\tilde{u}_{k,j,i} \mid k \in [N], j \in [2^d], i \in [\ell']\}$ and pads it with random values to have size $\ell' \cdot 2^d \cdot N$. The parties run a weak labeled PSI protocol (as in Appendix 4) between $S' := \{(\tilde{r}_{k,i}, \mathbf{q}_k) \mid k \in [M], i \in [\ell_{\max}]\}$ and U such that the receiver learns $(\tilde{u}_{k,j,i_*}, \mathbf{q}_{k'})$ for each $k \in [N]$ for which there exists $j \in [2^d]$ such that $u_{k,j} \in [r_{k'}, r_{k'} + \delta^{\mathbf{p}}]$ for some $k' \in [M]$, and for which there thus exists a matching prefix \tilde{u}_{k,j,i_*} , $i_* \in [\ell']$. Since the receiver learns the matching sender’s points $\mathbf{q}_{k'}$, all adapted protocols realize $\mathcal{F}_{\text{FUZZYPSI}}$ with *standard PSI* leakage. The complexities of these adapted protocols can be found in Table 5. Depending on the size of M and N , the parties might also choose to reverse the roles of the parties when executing the final PSI protocol to achieve a different trade-off in computation complexity.

9 Fuzzy PSI for High Dimensions

If the dimension d is large, the factor 2^d in the protocols from the previous sections becomes prohibitive. Therefore, we explore under which assumptions our novel fuzzy matching approach

$\Pi_{\text{FUZZYPSI}}^{\mathbf{p}, \text{GD}}$

Parameters : RECEIVER with input $\mathbf{W} = (\mathbf{w}_1, \dots, \mathbf{w}_N) \in \mathbb{Z}^{d \times N}$ and SENDER with input $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_M) \in \mathbb{Z}^{d \times M}$. Dimension d and radius δ .

Protocol :

1. For each $k \in [M]$, SENDER samples $r_k, s_k \leftarrow \{0, 1\}^{\lambda'}$, and sets $r_k = 0$ when $\mathbf{p} = \infty$.
2. For each $i \in [d]$, SENDER updates $\text{list}_i \leftarrow \text{list}_i \cup \text{list}_{k,i}$ where $(\text{list}_{k,1}, \dots, \text{list}_{k,d}) \leftarrow \text{GetList}_{\mathbf{p}}(r_k, s_k, \mathbf{q}_k, 0)$ for each $k \in [M]$.
3. RECEIVER sends $(\text{RECEIVER}, \text{EVALUATE}, (\tilde{\mathbf{W}}_1, \dots, \tilde{\mathbf{W}}_d))$ and SENDER sends $(\text{SENDER}, \text{EVALUATE}, (\text{list}_1, \dots, \text{list}_d))$ to $\mathcal{F}_{(d,N)\text{-OPPRF}}$, where $\tilde{\mathbf{W}}_i := (\mathbf{H}_{\gamma}(w_{k,i}))_{k \in [N]}$ for each $i \in [d]$.
4. RECEIVER gets $(F_i(\mathbf{H}_{\gamma}(w_{k,i})))_{i \in [d], k \in [N]}$ and SENDER gets $(\mathcal{O}^{F_i})_{i \in [d]}$ from $\mathcal{F}_{(d,N)\text{-OPPRF}}$.
5. If $\mathbf{p} \neq \infty$, SENDER puts $S := \{\mathbf{H}_{\lambda'}(r_k + s_k \cdot j) \mid k \in [M], j \in [\delta^{\mathbf{p}}]\}$, shuffles S and sends it to RECEIVER.
6. If $\mathbf{p} = \infty$, RECEIVER puts $S := \{\mathbf{H}_{\lambda'}(0)\}$.
7. RECEIVER computes $u_k := \mathbf{H}_{\lambda'}\left(\sum_{i=1}^d F_i(\mathbf{H}_{\gamma}(w_{k,i}))\right)$ for all $k \in [N]$.
8. RECEIVER sets $I := \emptyset$, and, for each $k \in [N]$, if $u_k \in S$, updates $I \leftarrow I \cup \{(\mathbf{w}_k)\}$.

Fig. 20. Fuzzy PSI-SP protocol for high-dimension Minkowski distance when both the sender's and receiver's points are globally disjoint (GD)

can be leveraged to provide a more efficient fuzzy PSI protocol in the high-dimensional setting. Unfortunately, since the sender's message in our fuzzy matching protocol, that is, the OKVS encoding in the underlying OPPrf protocol, can not be re-used to match against multiple receiver's points, we have to make rather strong assumptions on the distribution of both the sender's as well as the receiver's points. We recall the definition of globally (axis) disjoint balls from [23] in Definition 2. Assuming that both the sender's and receiver's balls are globally disjoint, we are able to construct a fuzzy PSI-SP protocol whose complexity depends only linearly on the dimension d . We have to admit that this assumption is rather strong and not quite realistic, but it serves to illustrate the limits of our new fuzzy matching technique. The protocol is given in Figure 20, achieves the complexity as described in Lemma 11 and its security is proven in Theorem 9

Definition 2 (Globally Disjoint). A set of d -dimensional balls of radius δ with centers $\mathbf{Q} \in \mathbb{Z}^{d \times N}$ is globally (axis) disjoint if, for each dimension $i \in [d]$, the projections $[q_{k,i} - \delta, q_{k,i} + \delta]$ are disjoint for all $k \in [N]$.

Lemma 11. The protocol $\Pi_{\text{FUZZYPSI}}^{\mathbf{p}, \text{GD}}$ has computation complexity $O(\delta d M + d N)$ for the sender if $\mathbf{p} = \infty$ and $O(\delta d M + \delta^{\mathbf{p}} M + d N)$ if $\mathbf{p} \neq \infty$, computation complexity $O(d N)$ for the receiver, and communication complexity $O(\delta d M \lambda + d N \lambda)$ if $\mathbf{p} = \infty$, $O(\delta d M \lambda + \delta^{\mathbf{p}} M \lambda + d N \lambda)$ if $\mathbf{p} \neq \infty$.

Theorem 9. The protocol $\Pi_{\text{FUZZYPSI}}^{\mathbf{p}, \text{GD}}$ realizes the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ for $L_{\mathbf{p}}$, $\mathbf{p} \in [1, \infty]$, with PSI-SP leakage against semi-honest adversaries in the $\mathcal{F}_{(d,N)\text{-OPPRF}}$ -hybrid model if the sender's and receiver's balls are globally disjoint as in Definition 2. Moreover, $\mathbf{H}_{\gamma} : \{0, 1\}^* \rightarrow \{0, 1\}^{\gamma}$ is a universal hash function with $\gamma \geq \lambda + \log(\delta N M)$ and:

- If $\mathbf{p} = \infty$, $\mathbf{H}_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a universal hash function, $\lambda' \geq \lambda + \log N + \log M$;
- If $\mathbf{p} \neq \infty$, $\mathbf{H}_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a random oracle, $\lambda' \geq \lambda + \log N + \log(\delta^{\mathbf{p}} M)$.

Proof. For correctness, first note that since the sender’s balls are globally disjoint, there are no collisions in the keys of each list_i , $i \in [d]$. Moreover, since the receiver’s points are also globally disjoint, there can not be two points $\mathbf{w}_k, \mathbf{w}_{k'}$ whose projections $w_{k,i}, w_{k',i}$ for some dimension $i \in [d]$ lie in the same interval $[q_i - \delta, q_i + \delta]$. Hence the OPPRF is programmed on a disjoint union of fuzzy matching instances and the receiver obtains OPPRF evaluations at disjoint instances. So the correctness of the protocol follows from the correctness of the fuzzy matching protocol Theorem 5.

The simulation proceeds similarly to the proof of Theorem 8, except that the independent randomness of the receiver’s OPPRF evaluations on different $w_{k,i}, w_{k',i}$ now follows from the assumption that the receiver’s points are globally disjoint. \square

10 Fuzzy PSI with Extended Functionalities

So far we have focussed on fuzzy PSI protocols achieving standard PSI or standard PSI with sender privacy. However, one might want to achieve a different variant of fuzzy PSI depending on one’s use case. For example, labeled PSI or PSI with cardinality (PSI-CA) as defined in Figure 2, or Circuit-PSI as defined in Figure 21.

Labeled PSI. Labeled PSI is relatively easy to achieve for all our protocols that realize PSI with sender privacy (PSI-SP). Note that labeled PSI implies standard PSI, by letting the label be a description of the sender’s point, but not necessarily the other way around, since the parties might want to exchange the label without revealing the sender’s point. For the protocols in Figures 19 and 20 labeled PSI, for labels of length σ , can be achieved by replacing the items in the sender’s set S by $H_{\lambda'+\sigma}(r_k) \oplus (0^{\lambda'} \parallel \text{label}_k)$ if $\mathbf{p} = \infty$, or $H_{\lambda'+\sigma}(r_k + s_k \cdot j) \oplus (0^{\lambda'} \parallel \text{label}_k)$ for all $j \in [\delta]^{\mathbf{p}}$ if $\mathbf{p} \neq \infty$. The receiver then checks for each $k \in [N]$ if there exists $s \in S$ such that $u_k \oplus s = 0^{\lambda'} \parallel z$ for some $z \in \{0, 1\}^\sigma$, and adds (\mathbf{w}_k, z) to their output if this is the case. To not increase the receiver’s complexity in doing this final step, the set S can be instantiated as follows. The sender uses a data structure with constant lookup time and linear insertion time, such as a Cuckoo hash table, where the first λ' bits of $s \in S$ are used to compute the bin index, and all $\lambda' + \sigma$ bits of s are subsequently stored in this bin. The receiver then similarly uses the first λ' bits of u_k to compute the potential bin indice, and subsequently checks for all items s in these bins if $u_k \oplus s = 0^{\lambda'} \parallel z$ for some $z \in \{0, 1\}^\sigma$. In this way, the labeled version of the protocols in Figures 19 and 20 have the same asymptotic complexity as the original versions.

For the protocol in Figure 18, labeled PSI can be achieved by replacing the ideal functionality \mathcal{F}_{PSI} by an ideal “weak” labeled PSI functionality $\mathcal{F}_{\text{WLPSI}}$. With “weak”, we mean a labeled PSI functionality that outputs the point in the intersection in addition to the label. This weaker functionality is sufficient to obtain a labeled fuzzy PSI protocol since the value $(H_{\lambda'}(r_k), \text{label}_{k'})$ in the output only reveals that for the receiver’s point \mathbf{w}_k , there exists some close sender point $\mathbf{q}_{k'}$ with corresponding $\text{label}_{k'}$. We formalize the notion of weak labeled PSI and give a simple protocol with linear communication and computation complexity from a $(1, n)$ -OPRF in Appendix 4.

PSI with Cardinality. We can realize fuzzy PSI with cardinality (defined in Figure 2) for all the protocols in Figures 17, 19 and 20 by letting the parties send the sets S and $U := \{u_{k,j} \mid k \in [N], j \in [2^d]\}$ (resp. $U := \{u_k \mid k \in [N]\}$) to an ideal PSI cardinality functionality $\mathcal{F}_{\text{PSI-CA}}$ (as in Figure 3). Similarly, the protocol in Figure 18 can be adapted by replacing the ideal functionality \mathcal{F}_{PSI} by $\mathcal{F}_{\text{PSI-CA}}$. Compared to the fuzzy PSI protocols from [1], which naturally realize fuzzy PSI-CA, it seems like our protocols will incur a relatively large overhead to realize this functionality. However,

$\mathcal{F}_{\text{FUZZYCPSI}}$

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. **Reorder** : $\mathbb{Z}^{d \times N} \rightarrow (\pi : [N] \rightarrow [m])$ which on input \mathbf{W} outputs an injective function π .

Functionality :

- 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 $k \in [m]$, sample $a_k, b_k \leftarrow_{\$} \{0, 1\}^{1+2\sigma}$ such that:

$$a_k \oplus b_k = 1 \|\tilde{\mathbf{w}}_i\| \tilde{\mathbf{q}}_j \text{ if } \exists i \in [N] \text{ s.t. } i' = \pi(i) \wedge \text{dist}(\mathbf{w}_i, \mathbf{q}_j) \leq \delta,$$

$$a_k \oplus b_k = 0^{1+2\sigma} \text{ otherwise.}$$
- Return $\pi, (a_k)_{k \in [m]}$ to RECEIVER and $(b_k)_{k \in [m]}$ to SENDER.

Fig. 21. Ideal Functionality of Fuzzy Circuit-PSI

since to the best of our knowledge, all PSI-CA protocols rely on public-key techniques such as additively homomorphic encryption (except from resorting to generic solutions) [18, 16, 28, 34, 42], it seems inherent that we will require more effort to achieve PSI-CA for our protocols. Additionally, by composing with a PSI-CA protocol at the end, we only need public-key operations on sets of relatively small size. Hence by using a PSI-CA protocol with linear communication and computation complexity, we expect our protocols to still outperform those of [1].

Circuit-PSI. We present a slightly different functionality $\mathcal{F}_{\text{FUZZYCPSI}}$ for fuzzy Circuit-PSI, compared to [1], in Figure 21, where we allow the indices of the output shares to be reordered according to an injective function $\pi : [N] \rightarrow [m]$ which is known to the receiver. This is in line with the standard Circuit-PSI functionality from [38].

To achieve fuzzy Circuit-PSI, we can similarly leverage an ideal Circuit-PSI functionality $\mathcal{F}_{\text{CPSI}}$ (as in Figure 5). We will describe how the protocols in Figures 19 and 20 can be adapted.

- RECEIVER inputs $U := \{(u_k, \tilde{\mathbf{w}}_k) \mid k \in [N]\}$ to $\mathcal{F}_{\text{CPSI}}$.
- If $\mathbf{p} = \infty$, SENDER inputs $S := \{(s_k, \tilde{\mathbf{q}}_k) \mid k \in [M]\}$ to $\mathcal{F}_{\text{CPSI}}$.
- If $\mathbf{p} \neq \infty$, SENDER inputs $S := \{(s_{k,j}, \tilde{\mathbf{q}}_k) \mid k \in [M], j \in [\delta^{\mathbf{p}}]\}$ to $\mathcal{F}_{\text{CPSI}}$.
- RECEIVER gets $\pi : [N] \rightarrow [m]$ and $(a_i)_{i \in [m]}$ from $\mathcal{F}_{\text{CPSI}}$.
- SENDER gets $(b_i)_{i \in [m]}$ from $\mathcal{F}_{\text{CPSI}}$.

To realize circuit-PSI for the protocol in Figure 17 can be done similarly, but here the shares $(a_i)_{i \in [m]}, (b_i)_{i \in [m]}$ are indexed by $\pi : [2^d] \times [N] \rightarrow [m]$. So, for each $k \in [N]$, the parties need to sum over the shares $a_{\pi(j,k)}$ and $b_{\pi(j,k)}$ for $j \in [2^d]$, without the sender learning the indices $\pi(j, k)$.

11 Fuzzy PSI for Arbitrary Distribution

In this section, we consider the most generalized setting (or, the weakest assumption): we do not assume that points are 2δ - or 4δ -apart from each other, instead, they can be *distributed arbitrarily* in the space. In other words, if we consider each point as the center of a ball, then these balls can be overlapped arbitrarily. Note that this is more general than the setting captured in [20], which only allows limited overlapping to ensure a successful encoding.

11.1 Infinity Distance

The key observation is that we can utilize the prefix trie idea from [12] to represent an integer interval succinctly. Then a d -dimensional hypercube can be represented as the direct product of d intervals. Though the previous work [20] uses a similar idea, they represent each interval through a GGM-tree to build function secret sharing (FSS), which incurs an $O(u^d)$ overhead, where 2^u is the universe. Our protocol is much simpler, not relying on FSS, and incurs only an $O((\log \delta)^d)$ overhead. We present the protocol in Figure 22 where the invoked algorithms $\{\text{PrefixTrie}, \text{PrefixPath}\}$ are introduced in Theorem 2. Notably, we improve the computational time of PrefixTrie from $O(\delta)$ in [12] to $O(\log \delta)$. Our protocol makes use of an ideal weak labeled PSI functionality $\mathcal{F}_{\text{WLPSI}}$, as defined in Appendix 4.

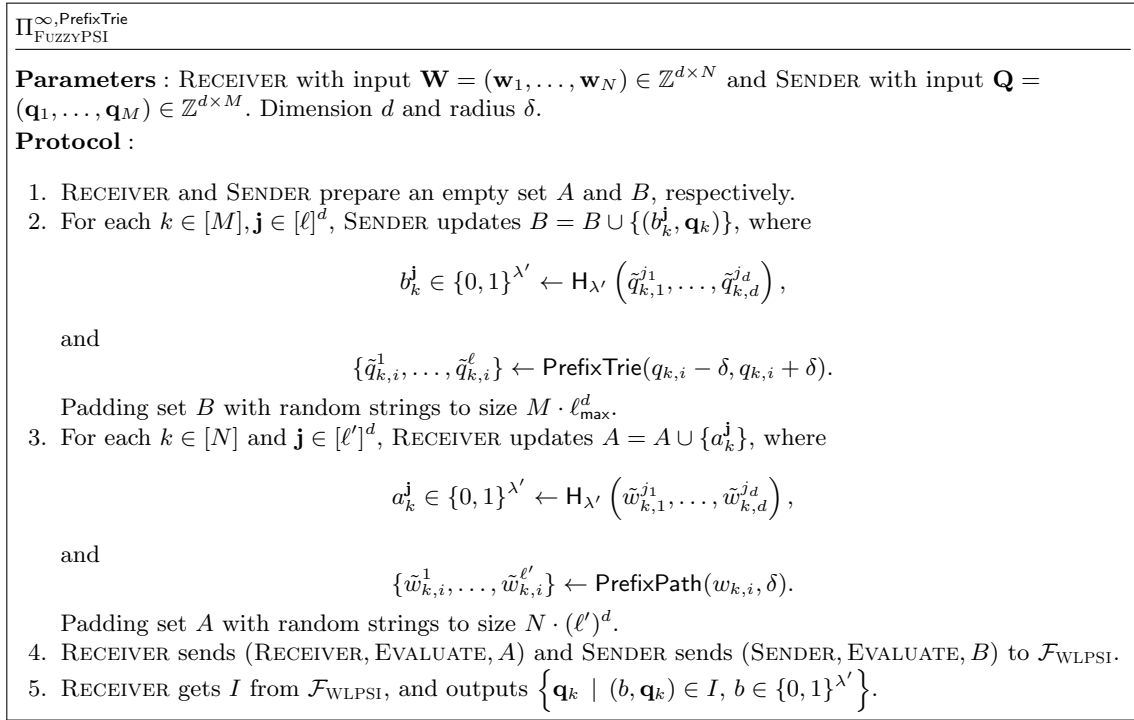


Fig. 22. Fuzzy PSI for Infinity distance from prefix trie, points are arbitrarily distributed

Lemma 12. *The protocol $\Pi_{\text{FUZZYPSI}}^{\infty, \text{PrefixTrie}}$ presented in Figure 22 has communication and computational complexity $O(\ell_{\max}^d \cdot M + N \cdot (\ell')^d)$, where $\ell' = \lfloor \log(2\delta + 1) \rfloor$, and $\ell_{\max} = \lfloor \log(2\delta + 1) \rfloor$ when δ is a power of 2, and $\ell_{\max} = 2\lfloor \log(2\delta + 1) \rfloor$ otherwise. We assume there is a protocol realizing $\mathcal{F}_{\text{WLPSI}}$ with linear communication and computational complexities, such as, Π_{WLPSI} from Figure 7.*

Theorem 10. *The protocol $\Pi_{\text{FUZZYPSI}}^{\infty, \text{PrefixTrie}}$ presented in Figure 22 realizes the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ for L_{∞} distance, with standard PSI leakage defined in Figure 2, against semi-honest adversaries in the $\mathcal{F}_{\text{WLPSI}}$ -hybrid model if $\{\text{PrefixTrie}, \text{PrefixPath}\}$ satisfy Theorem 2 and $H_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a universal hash functions with $\lambda' \geq \lambda + \log M + \log N + d \log \log \delta$.*

Proof. Consider that a semi-honest adversary \mathcal{A} corrupts RECEIVER. Since there are no transcripts between parties that only interact with the ideal functionality \mathcal{F}_{PSI} , the view of \mathcal{A} includes the intersection I from $\mathcal{F}_{\text{WLPSI}}$ and the final output. The simulator \mathcal{S} performs the step 3 in Figure 22 to get the set A , and sends the input of RECEIVER \mathbf{W} into $\mathcal{F}_{\text{FUZZYPSI}}$ to learn the intersection $J := \{\mathbf{q}_k \mid \exists j \in [N] : \text{dist}(\mathbf{w}_j, \mathbf{q}_k) \leq \delta\}$. Then, \mathcal{S} performs step 2 on J to get the set B' , and simulates $I' = \{(b_k^{\mathbf{j}}, \mathbf{q}_k) \mid (b_k^{\mathbf{j}}, \mathbf{q}_k) \in B', b_k^{\mathbf{j}} \in A\}$ as the output from $\mathcal{F}_{\text{WLPSI}}$. The simulation is statistically indistinguishable from the adversary: Suppose there is a pair $(c, \mathbf{q}_y) \in I$ but $(c, \mathbf{q}_y) \notin I'$, then there must be some \mathbf{w}_x , for $y \in [M], x \in [N]$, such that

$$\mathbf{H}_{\lambda'}(\tilde{w}_{x,1}^{j'_1}, \dots, \tilde{w}_{x,d}^{j'_d}) = \mathbf{H}_{\lambda'}(\tilde{q}_{y,1}^{j_1}, \dots, \tilde{q}_{y,d}^{j_d}) \wedge \text{dist}(\mathbf{w}_x, \mathbf{q}_y) > \delta,$$

exists for some $\mathbf{j} \in [\ell]^d, \mathbf{j}' \in [\ell']^d$. However, according to Theorem 2, there is at least one $i \in [d]$ such that $\tilde{w}_{x,i}^{j'_i} \neq \tilde{q}_{y,i}^{j_i}$ for any $(j'_1, j_i) \in [\ell'] \times [\ell]$. This implies above equation holds with negligible probability when $\mathbf{H}_{\lambda'}$ is a universal hash function. Suppose the other way around, that is, there is a pair $(c, \mathbf{q}_y) \notin I$ but $(c, \mathbf{q}_y) \in I'$, implying there must be some \mathbf{w}_x , for $y \in [M], x \in [N]$, such that

$$\mathbf{H}_{\lambda'}(\tilde{w}_{x,1}^{j'_1}, \dots, \tilde{w}_{x,d}^{j'_d}) \neq \mathbf{H}_{\lambda'}(\tilde{q}_{y,1}^{j_1}, \dots, \tilde{q}_{y,d}^{j_d}) \wedge \text{dist}(\mathbf{w}_x, \mathbf{q}_y) \leq \delta,$$

holds for every $\mathbf{j} \in [\ell]^d, \mathbf{j}' \in [\ell']^d$. This cannot happen due to Theorem 2 and the correctness of $\mathbf{H}_{\lambda'}$.

Regarding the case that \mathcal{A} corrupts SENDER, \mathcal{S} sends the input of SENDER \mathbf{Q} into $\mathcal{F}_{\text{FUZZYPSI}}$, which is sufficient since SENDER has no output or transcripts.

Regarding the correctness, that is, when both parties are honest, the final output is consistent with the output from $\mathcal{F}_{\text{FUZZYPSI}}$: as discussed in the case where RECEIVER is semi-honest, the output I from $\mathcal{F}_{\text{WLPSI}}$ contains the pair (a_x, \mathbf{q}_y) if and only if \mathbf{w}_x is close to some point $\mathbf{q}_y \in \mathbf{Q}$. \square

11.2 Minkowski Distance

The above approach unfortunately does not work for Minkowski distance $L_p, p < \infty$: the reason is simple, a distance-preserving mapping (isometry) does not exist from L_p to L_∞ . One exception is the normalized Euclidean distance (equivalently, the Cosine distance) in a two-dimensional space. Not surprisingly, normalized 2-dim vectors are on a unit circle, and vectors within some distance are exactly an arc. Switching the coordinate system from Cartesian to Polar, the arc on the unit circle can be represented as an interval of radius δ . Then we can use the prefix trie to obtain a fuzzy PSI protocol for normalized Euclidean distance on two-dimensional space with overhead $O(\log \delta)$. However, normalized Euclidean distance does not make much sense in a low-dimensional space.

However, we show that the prefix trie technique can still be leveraged to save approximately a factor $O(\frac{\delta}{\log \delta})$ compared to the naive approach of expanding the input balls, resulting in a protocol with communication and computation complexity $O(N \log \delta + \delta^{d-1} \cdot M \log \delta)$. We present the protocol in Figure 23. The intuition is we can map each d -dimensional ball to multiple $(d-1)$ -dimensional spheres and there are at most $2\delta + 1$ such hyperspheres, thus the remaining dimension can be handled by the prefix trie.

Lemma 13. *The protocol $\Pi_{\text{FUZZYPSI}}^{\text{p,PrefixTrie}}$ presented in Figure 23 has communication and computational complexity $O(N\ell' + \delta^{d-1} \cdot M\ell_{\max})$, where $\ell' = \lfloor \log(2\delta + 1) \rfloor$ and $\ell_{\max} = O(\log(2\delta + 1))$. We assume there is a protocol realizing $\mathcal{F}_{\text{WLPSI}}$ with linear communication and computational complexities, such as Π_{WLPSI} from Figure 7.*

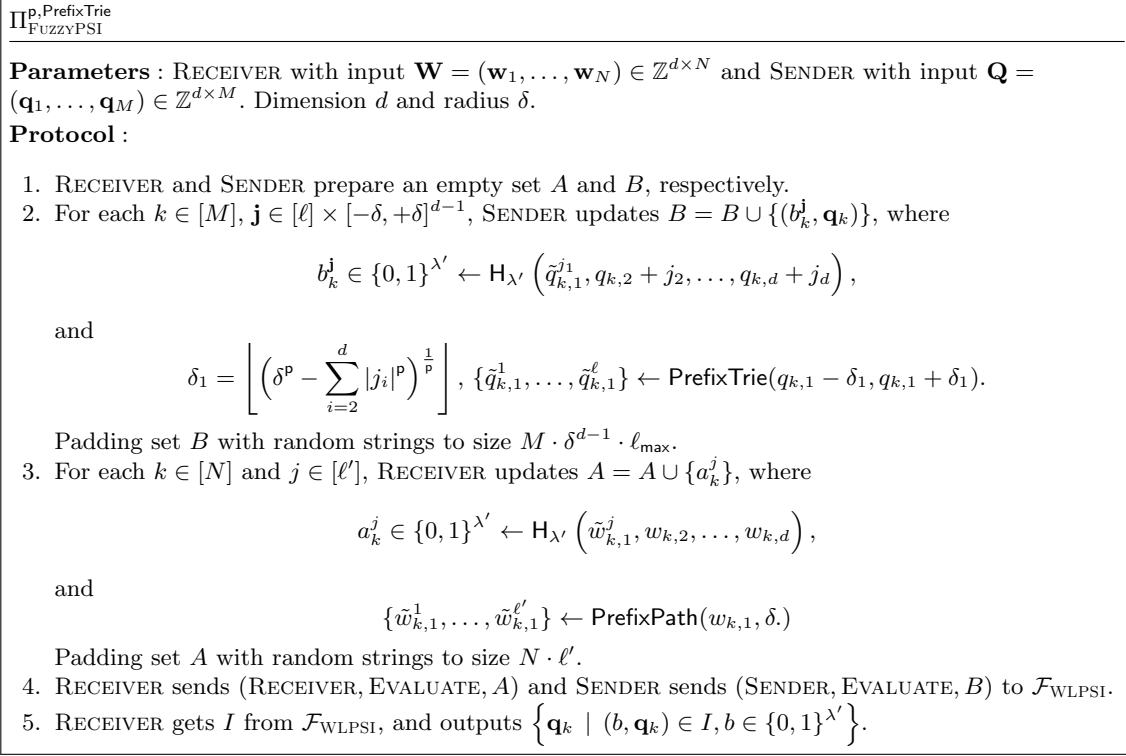


Fig. 23. Fuzzy PSI for Minkowski distance from prefix trie, points are arbitrarily distributed

Theorem 11. The protocol $\Pi_{\text{FUZZYPSI}}^{\text{p, PrefixTrie}}$ presented in Figure 22 realizes the functionality $\mathcal{F}_{\text{FUZZYPSI}}$ for L_p distance, with standard PSI leakage defined in Figure 2, against semi-honest adversaries in the $\mathcal{F}_{\text{WLPSI}}$ -hybrid model if $\{\text{PrefixTrie}, \text{PrefixPath}\}$ satisfy Theorem 2 and $H_{\lambda'} : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda'}$ is a universal hash functions with $\lambda' \geq \lambda + \log M + \log N + \log \log \delta + (d-1) \log \delta$.

Proof. The proof follows the same framework as the proof of Theorem 10. Consider that a semi-honest adversary \mathcal{A} corrupts RECEIVER. Since there are no transcripts between parties that only interact with the ideal functionality \mathcal{F}_{PSI} , the view of \mathcal{A} includes the output I from $\mathcal{F}_{\text{WLPSI}}$ and the final output. The simulator \mathcal{S} performs the step 3 in Figure 22 to get the set A , and sends the input of RECEIVER \mathbf{W} into $\mathcal{F}_{\text{FUZZYPSI}}$ to learn the intersection $J := \{\mathbf{q}_k \mid \exists j \in [N], \text{dist}(\mathbf{w}_j, \mathbf{q}_k) \leq \delta\}$. Then, \mathcal{S} performs step 2 on J to get the set B' , and simulates $I' := \{(b_k^{\mathbf{j}}, \mathbf{q}_k) \mid (b_k^{\mathbf{j}}, \mathbf{q}_k) \in B', b_k^{\mathbf{j}} \in A\}$ as the output from $\mathcal{F}_{\text{WLPSI}}$. The simulation is statistically indistinguishable from the adversary: Suppose there is a pair $(c, \mathbf{q}_y) \in I$ but $(c, \mathbf{q}_y) \notin I'$, then there must be some \mathbf{w}_x , for $y \in [M]$, $x \in [N]$, such that

$$H_{\lambda'}(\tilde{w}_{x,1}^{j'}, w_{x,2}, \dots, w_{x,d}) = H_{\lambda'}(\tilde{q}_{y,1}^{j_1}, q_{y,2} + j_2, \dots, q_{y,d} + j_d) \wedge \text{dist}(\mathbf{w}_x, \mathbf{q}_y) > \delta,$$

exists for some $\mathbf{j} \in [\ell] \times [-\delta, +\delta]^{d-1}$, $j' \in [\ell']$. However, $\text{dist}(\mathbf{w}_x, \mathbf{q}_y) > \delta$ implies that \mathbf{w}_x is outside the p -ball centered at \mathbf{q}_y and there must be some $i \in [d]$ such that either $w_{x,i} \notin [q_{y,i} - \delta, q_{y,i} + \delta]$, or $i = 1$ and $w_{x,1} \notin [q_{y,1} - \delta_1, q_{y,1} + \delta_1]$, where $2\delta_1 + 1$ is the length of the interval of the projection of $p-1$ -ball on this dimension. According to Theorem 2, $w_{x,1} \notin [q_{y,1} - \delta_1, q_{y,1} + \delta_1]$ means $\tilde{w}_{x,1}^{j'} \neq \tilde{q}_{y,1}^{j_1}$

for any $(j', j_1) \in [\ell'] \times [\ell]$. Thus, the equation holds with negligible probability when $H_{\lambda'}$ is a universal hash function.

Suppose the other way around, that is, there is a pair $(c, \mathbf{q}_y) \notin I$ but $(c, \mathbf{q}_y) \in I'$, implying there must be some \mathbf{w}_x , for $y \in [M], x \in [N]$, such that

$$H_{\lambda'}(\tilde{w}_{x,1}^{j'}, w_{x,2}, \dots, w_{x,d}) \neq H_{\lambda'}(\tilde{q}_{y,1}^{j_1}, q_{y,2} + j_2 \dots, q_{y,d} + j_d) \wedge \text{dist}(\mathbf{w}_x, \mathbf{q}_y) \leq \delta,$$

holds for every $\mathbf{j} \in [\ell] \times [-\delta, +\delta]^{d-1}$, $j' \in [\ell']$. Note that $\text{dist}(\mathbf{w}_x, \mathbf{q}_y) \leq \delta$ implies $w_{x,i} \in [q_{y,i} - \delta, q_{y,i} + \delta]$ for every $i \in [d]$, and $w_{x,1} \in [q_{y,1} - \delta_1, q_{y,1} + \delta_1]$. According to Theorem 2 and the correctness of $H_{\lambda'}$, the hash evaluations must match for some \mathbf{j}, j' except with $\text{negl}(\lambda)$ probability.

Regarding the case that \mathcal{A} corrupts SENDER and the case that both parties are honest (correctness), it is the same as proof of Theorem 10. \square

References

1. van Baarsen, A., Pu, S.: Fuzzy private set intersection with large hyperballs. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024, Part V*. Lecture Notes in Computer Science, vol. 14655, pp. 340–369. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58740-5_12
2. van Baarsen, A., Stevens, M.: Amortizing circuit-psi in the multiple sender/receiver setting. *IACR Commun. Cryptol.* **1**(3), 2 (2024). <https://doi.org/10.62056/AOFHSGVTW>
3. Bartusek, J., Garg, S., Jain, A., Policharla, G.V.: End-to-end secure messaging with traceability only for illegal content. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023, Part V*. Lecture Notes in Computer Science, vol. 14008, pp. 35–66. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30589-4_2
4. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) *ACM CCS 2012: 19th Conference on Computer and Communications Security*. pp. 784–796. ACM Press, Raleigh, NC, USA (Oct 16–18, 2012). <https://doi.org/10.1145/2382196.2382279>
5. Bienstock, A., Patel, S., Seo, J.Y., Yeo, K.: Near-optimal oblivious key-value stores for efficient PSI, PSU and volume-hiding multi-maps. In: Calandrino, J.A., Troncoso, C. (eds.) *USENIX Security 2023: 32nd USENIX Security Symposium*. pp. 301–318. USENIX Association, Anaheim, CA, USA (Aug 9–11, 2023)
6. Blass, E.O., Noubir, G.: Assumption-free fuzzy PSI via predicate encryption. *Cryptology ePrint Archive*, Paper 2025/217 (2025), <https://eprint.iacr.org/2025/217>
7. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) *ACM CCS 2019: 26th Conference on Computer and Communications Security*. pp. 291–308. ACM Press, London, UK (Nov 11–15, 2019). <https://doi.org/10.1145/3319535.3354255>
8. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) *Advances in Cryptology – CRYPTO 2019, Part III*. Lecture Notes in Computer Science, vol. 11694, pp. 489–518. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_16
9. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) *Advances in Cryptology – EUROCRYPT 2015, Part II*. Lecture Notes in Computer Science, vol. 9057, pp. 337–367. Springer, Berlin, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_12
10. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) *ACM CCS 2016: 23rd Conference on Computer and Communications Security*. pp. 1292–1303. ACM Press, Vienna, Austria (Oct 24–28, 2016). <https://doi.org/10.1145/2976749.2978429>

11. Bui, D., Couteau, G.: Improved private set intersection for sets with small entries. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 13941, pp. 190–220. Springer, Cham, Switzerland, Atlanta, GA, USA (May 7–10, 2023). https://doi.org/10.1007/978-3-031-31371-4_7
12. Chakraborti, A., Fanti, G., Reiter, M.K.: Distance-aware private set intersection. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023: 32nd USENIX Security Symposium. pp. 319–336. USENIX Association, Anaheim, CA, USA (Aug 9–11, 2023)
13. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018: 25th Conference on Computer and Communications Security. pp. 1223–1237. ACM Press, Toronto, ON, Canada (Oct 15–19, 2018). <https://doi.org/10.1145/3243734.3243836>
14. Chongchitmate, W., Lu, S., Ostrovsky, R.: Approximate PSI with near-linear communication. Cryptology ePrint Archive, Paper 2024/682 (2024), <https://eprint.iacr.org/2024/682>
15. Cong, K., Moreno, R.C., da Gama, M.B., Dai, W., Iliashenko, I., Laine, K., Rosenberg, M.: Labeled PSI from homomorphic encryption with reduced computation and communication. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021: 28th Conference on Computer and Communications Security. pp. 1135–1150. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484760>
16. De Cristofaro, E., Gasti, P., Tsudik, G.: Fast and private computation of cardinality of set intersection and union. In: Pieprzyk, J., Sadeghi, A.R., Manulis, M. (eds.) CANS 12: 11th International Conference on Cryptology and Network Security. Lecture Notes in Computer Science, vol. 7712, pp. 218–231. Springer, Berlin, Heidelberg, Germany, Darmstadt, Germany (Dec 12–14, 2012). https://doi.org/10.1007/978-3-642-35404-5_17
17. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology – CRYPTO’84. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 1984). https://doi.org/10.1007/3-540-39568-7_2
18. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 1–19. Springer, Berlin, Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004). https://doi.org/10.1007/978-3-540-24676-3_1
19. Gao, Y., Qi, L., Liu, X., Luo, Y., Wang, L.: Efficient fuzzy private set intersection from fuzzy mapping. In: Advances in Cryptology – ASIACRYPT 2024, Part VI. pp. 36–68. Lecture Notes in Computer Science, Springer, Singapore, Singapore (Dec 7–11, 2024). https://doi.org/10.1007/978-981-96-0938-3_2
20. Garimella, G., Goff, B., Miao, P.: Computation efficient structure-aware PSI from incremental function secret sharing. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024, Part VIII. Lecture Notes in Computer Science, vol. 14927, pp. 309–345. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68397-8_10
21. Garimella, G., Goff, B., Miao, P.: Computation efficient structure aware PSI from incremental function secret sharing. CRYPTO (2024), <https://eprint.iacr.org/2024/842>
22. Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious key-value stores and amplification for private set intersection. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part II. Lecture Notes in Computer Science, vol. 12826, pp. 395–425. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84245-1_14
23. Garimella, G., Rosulek, M., Singh, J.: Structure-aware private set intersection, with applications to fuzzy matching. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022, Part I. Lecture Notes in Computer Science, vol. 13507, pp. 323–352. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15802-5_12
24. Garimella, G., Rosulek, M., Singh, J.: Malicious secure, structure-aware private set intersection. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part I. Lecture Notes in Computer Science, vol. 14081, pp. 577–610. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38557-5_19

25. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing. pp. 169–178. ACM Press, Bethesda, MD, USA (May 31 – Jun 2, 2009). <https://doi.org/10.1145/1536414.1536440>
26. Horowitz, E., Sahni, S.: Computing partitions with applications to the knapsack problem. *J. ACM* **21**(2), 277–292 (Apr 1974). <https://doi.org/10.1145/321812.321823>, <https://doi.org/10.1145/321812.321823>
27. Indyk, P., Woodruff, D.P.: Polylogarithmic private approximations and efficient matching. In: Halevi, S., Rabin, T. (eds.) TCC 2006: 3rd Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 3876, pp. 245–264. Springer, Berlin, Heidelberg, Germany, New York, NY, USA (Mar 4–7, 2006). https://doi.org/10.1007/11681878_13
28. Ion, M., Kreuter, B., Nergiz, A.E., Patel, S., Saxena, S., Seth, K., Raykova, M., Shanahan, D., Yung, M.: On deploying secure computing: Private intersection-sum-with-cardinality. In: EuroS&P. pp. 370–389. IEEE (2020). <https://doi.org/10.1109/EUROSP48549.2020.00031>
29. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 145–161. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003). https://doi.org/10.1007/978-3-540-45146-4_9
30. Katsumata, S., Matsuda, T., Nakamura, W., Ohara, K., Takahashi, K.: Revisiting fuzzy signatures: Towards a more risk-free cryptographic authentication system based on biometrics. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021: 28th Conference on Computer and Communications Security. pp. 2046–2065. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484586>
31. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016: 23rd Conference on Computer and Communications Security. pp. 818–829. ACM Press, Vienna, Austria (Oct 24–28, 2016). <https://doi.org/10.1145/2976749.2978381>
32. Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., Trieu, N.: Practical multi-party private set intersection from symmetric-key techniques. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security. pp. 1257–1272. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017). <https://doi.org/10.1145/3133956.3134065>
33. Meadows, C.: A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: S&P. pp. 134–137. IEEE Computer Society (1986)
34. Miao, P., Patel, S., Raykova, M., Seth, K., Yung, M.: Two-sided malicious security for private intersection-sum with cardinality. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 3–33. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_1
35. Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based PSI with linear communication. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 122–153. Springer, Cham, Switzerland, Darmstadt, Germany (May 19–23, 2019). https://doi.org/10.1007/978-3-030-17659-4_5
36. Raghuraman, S., Rindal, P.: Blazing fast PSI from improved OKVS and subfield VOLE. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022: 29th Conference on Computer and Communications Security. pp. 2505–2517. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). <https://doi.org/10.1145/3548606.3560658>
37. Richardson, D., Rosulek, M., Xu, J.: Fuzzy PSI via oblivious protocol routing. Cryptology ePrint Archive, Paper 2024/1642 (2024), <https://eprint.iacr.org/2024/1642>
38. Rindal, P., Schoppmann, P.: VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021, Part II. Lecture Notes in Computer Science, vol. 12697, pp. 901–930. Springer, Cham, Switzerland, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77886-6_31
39. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019: 26th

- Conference on Computer and Communications Security. pp. 1055–1072. ACM Press, London, UK (Nov 11–15, 2019). <https://doi.org/10.1145/3319535.3363228>
40. Son, H., Paik, S., Kim, Y., Kim, S., Chung, H., Seo, J.H.: Doubly efficient fuzzy private set intersection for high-dimensional data with cosine similarity. Cryptology ePrint Archive, Paper 2025/054 (2025), <https://eprint.iacr.org/2025/054>
 41. Uzun, E., Chung, S.P., Kolesnikov, V., Boldyreva, A., Lee, W.: Fuzzy labeled private set intersection with applications to private real-time biometric search. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021: 30th USENIX Security Symposium. pp. 911–928. USENIX Association (Aug 11–13, 2021)
 42. Wu, M., Yuen, T.H.: Efficient unbalanced private set intersection cardinality and user-friendly privacy-preserving contact tracing. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023: 32nd USENIX Security Symposium. pp. 283–300. USENIX Association, Anaheim, CA, USA (Aug 9–11, 2023)
 43. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science. pp. 162–167. IEEE Computer Society Press, Toronto, Ontario, Canada (Oct 27–29, 1986). <https://doi.org/10.1109/SFCS.1986.25>