Computation Efficient Structure-Aware PSI from Incremental Function Secret Sharing

Gayathri Garimella(✉), Benjamin Goff, and Peihan Miao

Brown University, Providence, USA
{gayathri_garimella,benjamin_goff,peihan_miao}@brown.edu

Abstract. Structure-Aware Private Set Intersection (sa-PSI), recently introduced by Garimella et al. (Crypto'22), is a PSI variant where Alice's input set S_A has a publicly known structure (for example, interval, ball or union of balls) and Bob's input S_B is an unstructured set of elements. Prior work achieves sa-PSI where the communication cost only scales with the description size of S_A instead of the set cardinality. However, the computation cost remains linear in the cardinality of S_A, which could be prohibitively large.

In this work, we present a new semi-honest sa-PSI framework where both computation and communication costs *only scale with the description size* of S_A. Our main building block is a new primitive that we introduce called Incremental Boolean Function Secret Sharing (ibFSS), which is a generalization of FSS that additionally allows for evaluation on input prefixes. We formalize definitions and construct a weak ibFSS for a d-dimensional ball with ℓ_∞ norm, which may be of independent interest. Independently, we improve spatial hashing techniques (from prior work) when S_A has structure union of d-dimensional balls in $(\{0,1\}^u)^d$, each of diameter δ, from $\mathcal{O}(u \cdot d \cdot (\log \delta)^d)$ to $\mathcal{O}(\log \delta \cdot d)$ in terms of both computation and communication. Finally, we resolve the following open questions from prior work with communication and computation scaling with the description size of the structured set.

- Our PSI framework can handle a union of overlapping structures, while prior work strictly requires a disjoint union.
- We have a new construction that enables Bob with unstructured input S_B to learn the intersection.
- We extend to a richer class of functionalities like structure-aware PSI Cardinality and PSI-Sum of associated values.

1 Introduction

Private Set Intersection (PSI) enables two distrusting parties, each holding a private set of elements, to jointly compute the intersection of their sets without revealing any additional information. Over the past decade, PSI has found numerous real-world applications, such as secure password breach checkup [1,2,5,9], mobile private contact discovery [6,48], online advertising measurement [4,45,52], privacy-preserving contact tracing in a global pandemic [3,13,65], and more. There has been enormous progress towards realizing

PSI efficiently to be deployed in practice, at scale. Most of these protocols have both the computational and communication complexity proportional to the size of the two sets [14,23,26,28,38,43,44,51,54,57,61,63,64,64].

Recent work by Garimella et al. [39] considers the scenario where one party's input set has a publicly known structure. They introduce *Structure-Aware PSI* (sa-PSI), where Alice has an input set $S_A \in \mathcal{S}$ from a known family of structured sets $\mathcal{S}$ (for example, a single-dimensional interval, high-dimensional ball, disjoint union of balls) and Bob's input set S_B consists of an unstructured collection of elements. The size of S_A could potentially be much larger than S_B, namely $|S_A| \gg |S_B|$, but the structure leads to a succinct *description size* of the set S_A. sa-PSI [39,40] allows Alice to learn the intersection $S_A \cap S_B$ with communication $\ll |S_A|$, that *scales with the description size* of S_A instead of its cardinality.

Structure-aware PSI finds many applications in practice. For instance, suppose Alice and Bob each have a set of points A and B respectively, and they want to identify which pairs of their points are *close by*, namely $(x, y) \in A \times B$ satisfying $D(x, y) \leq \delta$, where D is a public distance metric and δ is a public threshold. We can model this problem as an instance of sa-PSI, where Alice's structured input S_A is a union of δ-balls (for each point $x \in A$, consider a ball centered at x with radius δ under the distance metric D), and Bob's input is his unstructured input set B. This can be used for applications such as noisy/fuzzy biometric matching [34,66] and privacy-preserving ride sharing [35,62], among others.

Prior works [39,40] present a general framework for semi-honest and malicious secure sa-PSI, respectively, with specific construction for set family of union of d-dimensional balls in the ℓ_∞ norm, where the communication cost is *independent of the total volume of the balls* in the structured input. However, an inherent limitation is that Alice's local *computation cost* remains linear in the size of her structured input $|S_A|$ (that is, total volume of the balls), which can be exponentially large. This leaves us with the following open problem that is explicitly mentioned in [39]:

*Can we achieve sa-PSI where **both** computation and communication costs only scale with the description size of the structured set?*

1.1 Our Contributions

Structure-Aware PSI with Efficient Computation. In this work, we answer the above question in the affirmative. We present a new semi-honest sa-PSI protocol for union of ℓ_∞-balls in a d-dimensional space, where *both* computation and communication costs only scale with the description size of the structured set. We achieve both communication and communication complexity of $\mathcal{O}(N_A \cdot u^d)$, in a d dimensional space $(\{0, 1\}^u)^d$ where N_A is the number of balls in Alice's set, instead of scaling with $|S_A|$, which could potentially be as large as $2^{u \cdot d}$.

Incremental Boolean Function Secret Sharing. The main building block underneath our construction is a new notion we introduce, called *Incremental*

Boolean Function Secret Sharing (ibFSS). Intuitively speaking, FSS [17] splits a function f from a function family $\mathcal{F}$ into two functions, f_0, f_1, such that each function f_b hides the function f and that $f_0(x)+f_1(x) = f(x)$ for all input x. Our new notion ibFSS is an instance of Boolean FSS [39], for a function that captures set membership of a structured set S_A, meaning that $f(x) = 0$ if $x \in S_A$ and $f(x) = 1$ otherwise. Additionally, ibFSS also enables evaluation on *prefixes* of all the points while preserving the FSS properties. We formalize the notion of ibFSS and give a construction for d-dimensional ℓ_∞-balls. We then present a generic framework of constructing sa-PSI from ibFSS, and instantiate the protocol with d-dimensional ℓ_∞-balls. As a result, Alice's computation cost is reduced from $\mathcal{O}(|S_A|)$ to $\mathcal{O}(u^d)$ for a single ball (see Table 1 in Sect. 7.1). It is worth noting that our ibFSS construction only uses lightweight operations of pseudorandom generators (PRGs) and may be of independent technical interest.

New, Efficient Construction for Union of (Overlapping) Structured Sets. We develop a new approach for PSI with union of structures as input, that improves *both computation and communication costs* from exponential to linear in the dimension d, namely from $\mathcal{O}(u^d)$ to $\mathcal{O}(u \cdot d)$ (see Table 2 in Sect. 7.2). The construction in prior works [39,40] crucially requires that Alice's structure is a union of *disjoint* balls, because it is unclear how to construct an efficient FSS scheme for a union of *overlapping* structures. However, such a requirement may not be feasible in certain applications such as privacy-preserving ride sharing. We answer the following question in the affirmative:

Can we achieve sa-PSI for a union of ***overlapping*** *balls?*

New Spatial Hashing Techniques. Garimella et al. [39] introduced a *spatial hashing* technique to improve the performance of structure-aware PSI for a union of N_A disjoint d-dimensional ℓ_∞-balls of the same diameter δ. This technique was further improved in [40]. In this work, we introduce a new perspective on spatial hashing, which improves both computation and communication costs from $\mathcal{O}(u \cdot d \cdot (\log \delta)^d)$ to $\mathcal{O}(\log \delta \cdot d)$ (see Table 3 in Sect. 7.3). Furthermore, we relax the restrictions on the balls, supporting overlapping balls with different diameters. The new spatial hashing techniques is compatible with the previous framework [39,40] and improves the communication cost of all prior constructions.

Extended Functionalities. The previous construction [39,40] is critically restricted to *one-sided*, *plain* PSI, where Alice, the party holding the structured set, learns the entire set intersection. It is unclear how to extend it to more advanced functionalities. For instance, certain real-world applications may require *only Bob* to learn the output. A naive approach that does not work is to have Alice send the FSS evaluations of all her elements in S_A to Bob, which is highly inefficient. Even worse, these evaluations can reveal critical information about Alice's structured set; in fact, they reveal Alice's entire set to Bob! As another extension, we may want to reveal only aggregate information about the intersection, while hiding the actual contents of the intersection. One notable functionality, *PSI-Cardinality*, allows two parties to only learn the cardinality of their intersection $|S_A \cap S_B|$.

Can we achieve sa-PSI *that enables* ***only Bob*** *to learn the output?*
Can we achieve sa-PSI *that enables Alice (or Bob) to learn* ***PSI-Cardinality****?*

In this work, we present new sa-PSI protocols for these extended functionalities, allowing them to learn *PSI-Cardinality* as well as *PSI-Sum* [45], in which Bob has an integer value associated with every element in his set and they jointly learn the summation of all the associated values for elements in the intersection. We further allow *only Bob* to learn the output in all these functionalities, including PSI, PSI-Cardinality, and PSI-Sum. All our protocols have communication and computation costs that only scale with the description size of the structured set. Our constructions only use lightweight symmetric-key cryptographic operations (such as pseudorandom generators and hash functions) and oblivious transfer (OT) [60], which can be instantiated efficiently with OT extension [46].

1.2 Related Work

Conventional PSI and Related Notions. In recent years, many paradigms for PSI protocols have evolved, including circuit-based [43,56–58], Diffie-Hellman [30,44,45,47], oblivious polynomial evaluation [29,50], RSA [11, 30], fully homomorphic encryption [24,25,27], bloom filters [31,63], oblivious transfer [23,51,54,59], and vector oblivious linear evaluation [28,38,61,64] approaches, achieving both semi-honest and malicious security [14,24,26,53,55, 63,64].

As discussed earlier, certain applications require PSI with more refined functionalities, such as allowing only a designated party to learn the output or enabling restricted computation on the elements in the intersection [22,37,42,42,49,57,64]. For instance, PSI-Cardinality and PSI-Sum model many applications like aggregated ads measurement [45,52] and privacy-preserving contact tracing [13,65].

Structure-Aware PSI. Recent work of Garimella et al. [39] introduced the notion of structure-aware PSI, where Alice holds a set with a publicly known structure and Bob holds an unstructured set of points. In such a setting where one party's set could be substantially larger than the other, namely *unbalanced PSI*, it is possible to construct protocols with communication sublinear in the larger set using RSA accumulators [11,30], leveled fully homomorphic encryption [24,25,27], or laconic PSI [8,10,32,36]. However, they require expensive public-key cryptographic operations, and the computation cost remains linear the larger set.

In contrast, the work of Garimella et al. [39] constructs a semi-honest structure-aware PSI using lightweight cryptographic tools including PRG, hash functions, and OT, taking advantage of the efficient OT extension [46]. A follow-up work [40] extends the protocol to achieve malicious security. Both works achieve communication cost that only scales with the description size instead of cardinality of Alice's structured set. However, Alice's computation cost still scales with the cardinality of her set. Another recent work [67] achieves both communication and communication costs that scale linearly with the diameter δ and exponentially in the dimension d, in two interaction messages. They rely on the Decisional Diffie-Hellman (DDH) assumption and public-key operations, especially leveraging the homomorphism of ElGamal encryption [33].

The following works [12,41,68–70] study fuzzy matching or threshold PSI, that reveals the intersection, only when it is above a certain threshold. Chakraborti *et al.* [21] and [66] construct fuzzy PSI protocols (which they call *distance-aware PSI*) for Hamming distance metric, based on homomorphic encryption.

Function Secret Sharing. Function secret sharing (FSS) was first introduced by Boyle et al. [17] who proposed efficient FSS constructions for point functions, comparison functions, and a few other interesting classes. These original FSS constructions were further optimized and extended in a sequence of works for point functions, multi-point functions, comparison functions and d-dimensional intervals [16,18,20]. Boneh et al. [15] extended the construction of Distributed Point Function (DPF), namely FSS for point functions, to Incremental DCF, which allows for evaluation on any prefix of a string. In this work, we further extend the notion to Incremental Distributed Comparison Function (DCF) and give an efficient construction of it from PRG, which may be of independent interest. *Boolean FSS (bFSS)* was introduced by Garimella et al. [39] to capture structured sets, and they present bFSS constructions for union of disjoint structured sets. We formalize an extended notion of Incremental Boolean FSS (ibFSS) and present an ibFSS construction for d-dimensional ℓ_∞ balls, from which we construct our sa-PSI protocol.

2 Technical Overview

Prior Framework for sa-PSI. We first briefly revisit the framework in prior works [39,40]. The main building block is a weak Boolean FSS scheme t-bFSS (with evaluation output bit length t) for Alice's set family (e.g., a single-dimensional interval, d-dimensional ball, or union of balls). The scheme succinctly secret shares a set S_A and produces a pair of keys $(k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, S_A)$ with the property that each key share k_b hides the structure S_A and that $\mathsf{Eval}(k_0, \vec{y}) \oplus \mathsf{Eval}(k_1, \vec{y}) = 0^t$ when $\vec{y} \in S_A$ and $\mathsf{Eval}(k_0, \vec{y}) \oplus \mathsf{Eval}(k_1, \vec{y}) \neq 0^t$ otherwise. The authors leverage this tool for set-membership testing and realize a secure PSI scheme as follows.

Alice generates κ *independent* sharings of her set and obtains (short) keys $(k_0^{(i)}, k_1^{(i)}) \leftarrow \mathsf{Share}(1^\kappa, S_A)$ for each instance $i \in [\kappa]$, where κ is the security parameter. Bob randomly samples a string $s \leftarrow \{0,1\}^\kappa$ to indicate his choice bits for each sharing. Alice and Bob perform an oblivious transfer (OT) protocol [60], where Bob, as the receiver, learns one of the two shares $k_{s[i]}^{(i)}$ based on his choice bit $s[i]$, while Alice remains oblivious to Bob's choice bits. Bob defines a special function that can be computed as follow:

$$\mathsf{F}(\vec{y}) = \mathsf{H}\left(\mathsf{Eval}(k_{s[i]}^{(1)}, \vec{y}) \| \dots \| \mathsf{Eval}(k_{s[i]}^{(\kappa)}, \vec{y})\right).$$

Bob sends $\{\mathsf{F}(\vec{y}) \mid \vec{y} \in S_B\}$ to Alice. Alice can *only* compute this special function if $\vec{y} \in S_A$, otherwise the output looks pseudorandom. Why? When $\vec{y} \in S_A$, both key shares evaluate to the same value, namely $\mathsf{Eval}(k_0^{(i)}, \vec{y}) = \mathsf{Eval}(k_1^{(i)}, \vec{y})$ by the t-bFSS property, hence the function evaluation is independent of Bob's choice bits. She simply computes the function as follows:

$$\mathsf{F}(\vec{y}) = \mathsf{H}\left(\mathsf{Eval}(k_0^{(1)}, \vec{y}) \| \dots \| \mathsf{Eval}(k_0^{(\kappa)}, \vec{y})\right).$$

When $\vec{y} \notin S_A$, it holds that $\mathsf{Eval}(k_0^{(i)}, \vec{y}) \neq \mathsf{Eval}(k_1^{(i)}, \vec{y})$, so Alice must simultaneously guess all of Bob's choice bits (which is negligibly likely), to compute the function. The output $\mathsf{F}(\vec{y})$ is pseudorandom to Alice assuming H is a correlation robust hash function. Therefore, Alice only recognizes evaluations on values in the intersection.

Cost Analysis. If there exists a t-bFSS scheme with succinct keys (of size $\sigma \ll |S_A|$) for Alice's set family, then the $\mathsf{sa\text{-}PSI}$ protocol is communication-efficient with cost $\mathcal{O}((\sigma + |S_B|) \cdot \kappa)$. However, an inherent drawback of this approach is that Alice must compute the following set of size $|S_A|$

$$\{\mathsf{H}(\mathsf{Eval}(k_0^{(1)}, \vec{x}) \| \dots \| \mathsf{Eval}(k_0^{(\kappa)}, \vec{x})) \mid \vec{x} \in S_A\}$$

in order to locally compare with Bob's message of hash evaluations, and identify matches to learn the intersection. Alice's computation cost scales with the cardinality of her structured input $|S_A|$, which can be prohibitively large for many applications.

Our New Framework for sa-PSI. Our first contribution is a new framework for $\mathsf{sa\text{-}PSI}$ where *both computation and communication costs scale with the (short) description size* of Alice's set, from a new tool called weak Incremental Boolean Function Secret Sharing ibFSS. At a high-level, the idea is that Bob will craft and send additional "hints" along with hash evaluations on his inputs. The hints will help Alice *efficiently identify and search for values* that are in the intersection, without having to expand and compute on every item in her full set.

Firstly, we observe that a structured set from a set family $S_A \in \mathcal{S}$ can be succinctly represented by a distinct and bounded set of w *critical prefixes*, where w is a direct function of $\mathcal{S}$. Each of these critical prefixes is a placeholder for all inputs in S_A that share the same prefix. Stated differently, every input in S_A has one of the w critical prefixes as its prefix. For instance, if Alice's structured set is a one-sided interval $(0, \alpha]$, where the partition point $\alpha \in \{0,1\}^u$, then at most u critical prefixes will span her interval (since α is a string of length u). Alternatively, if the structured set is a d-dimensional ℓ_∞-ball, which is a cross product of $2 \cdot d$ one-dimensional intervals, then at most $u^{2 \cdot d}$ critical prefixes will span such an input. In our new $\mathsf{sa\text{-}PSI}$ framework, we exploit the fact that the set of critical prefixes is much smaller than the total set size to reduce computation. For this, we will require an additional property on the weak Boolean FSS (from prior work [39]) that allows evaluation not only on the inputs strings but also on all its prefixes. The guarantee will be $\mathsf{Eval}(k_0, \vec{y}) \oplus \mathsf{Eval}(k_1, \vec{y}) = 0^t$ for all $\vec{y}$ that is a critical prefix or contains a critical prefix (this implicitly includes every input in Alice's set S_A). For all other $\vec{y}$, it holds that $\mathsf{Eval}(k_0, \vec{y}) \oplus \mathsf{Eval}(k_1, \vec{y}) \neq 0^t$. We formalize these requirements and define **weak Incremental Boolean Function Secret Sharing** ibFSS in Sect. 4, and present a construction for d-dimensional ℓ_∞-balls in Sect. 6.

So, how does our sa-PSI protocol work and what are these "hints"? Let's use a simple example. Alice's input S_A is a single d-dimensional ball (or any other structure in d-dimensional space), while Bob's input S_B contains a single point $\vec{y} = (y_1, \ldots, y_d) \in [2^u]^d$. Following prior work, Alice secret shares her input κ times using ibFSS. Bob then samples a uniform string $s \leftarrow \{0,1\}^\kappa$ and via OT learns one of the two keys for each sharing. Now, Bob defines a special function using his shares and evaluates on his input $\vec{y}$:

$$\mathsf{F}(\vec{y}) = \mathsf{H}\left(\mathsf{Eval}(k_{s[i]}^{(1)}, \vec{y}) \| \ldots \| \mathsf{Eval}(k_{s[i]}^{(\kappa)}, \vec{y})\right).$$

Additionally, he evaluates on all prefixes $\vec{y'}$ of his input as "hints" and sends to Alice:

$$\left\{ \mathsf{F}(\vec{y}) \;\middle|\; \begin{array}{l} \ell_1 \in [u], \ldots, \ell_d \in [u], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\}.$$

Alice identifies a set of critical prefixes for her input ball. The claim is that if Bob's input $\vec{y} \in S_A$, then there is exactly one critical prefix $\vec{p} = (p_1, \ldots, p_d)$ that is a prefix match of $\vec{y}$. As guaranteed by the ibFSS scheme, Alice can compute Bob's function on this prefix $\vec{p}$ as follows:

$$\mathsf{F}(\vec{p}) = \mathsf{H}\left(\mathsf{Eval}(k_0^{(1)}, \vec{p}) \| \ldots \| \mathsf{Eval}(k_0^{(\kappa)}, \vec{p})\right).$$

Alice can recognize $\mathsf{F}(\vec{p})$ from Bob's set of hints. Furthermore, it's not hard to see that exactly one of the two prefixes $(p_1 \| 0, p_2, \ldots, p_d), (p_1 \| 1, p_2, \ldots, p_d)$ is a prefix of $\vec{y}$. Alice can correctly compute the function on the matching prefix and is guaranteed to find this value in Bob's set of hints (since he sends evaluations on every prefix of his input). Similarly, Alice can iteratively figure out the rest of the bits of $\vec{y}$ using binary search, to learn the intersection. Finally, if Bob's input $\vec{y} \notin S_A$, then Alice cannot compute Bob's function correctly on any of her prefixes, and as a result, learns nothing about his private input. This is the crux of our new approach and how we achieve savings in computation at the cost of slightly increased communication from Bob.

Multiple (Overlapping) Balls with Multiple Points. Next, we consider the case when Alice's input is a union of multiple balls (or any structure), namely $S_A = \vec{S}_A^{(1)} \cup \ldots \cup \vec{S}_A^{(N_A)}$, and Bob's input contains multiple points $|S_B| > 1$. Prior work [39,40] introduced an elegant sum transformation to obtain a 1-bFSS scheme for a *disjoint* union of structures. However, this technique is only compatible with a restricted class of *strong* 1-bFSS schemes that output a single bit, whose constructions are significantly more expensive. In particular, both the key share sizes and evaluation costs scale exponentially in the dimension d.

In this work, we circumvent this limitation to realize $\mathsf{sa\text{-}PSI}$ for union of structures (even possibly *overlapping* structures, under certain conditions) from the more efficient *weak* t-bFSS schemes with both key share sizes and evaluation costs linear in d. Our improvement comes from the following observation. If Alice learns from the PSI ideal functionality that an element $\vec{y}$ is in the intersection, then it implicitly reveals the underlying structured set(s) that contain $\vec{y}$, that is, every $i \in [N_A]$ for which $y \in \vec{S}_A^{(i)}$. Hence, we propose a new protocol where Alice compares each of her structures $\vec{S}_A^{(i)}, i \in [N_A]$, individually, with Bob's input $\vec{y}$ (effectively, N_A instances of PSI with a single structure against Bob's input $\vec{y}$). Since we evaluate each structure separately, the underlying t-bFSS scheme can be instantiated more efficiently (no longer restricted by the sum transformation) and it's compatible even when the structures are overlapping.

To handle multiple points in Bob's set, he will send prefix evaluations for each of his inputs to Alice. It could happen that some prefixes overlap across different inputs, which may leak extra information to Alice. A simple fix is to have Bob send the union of his evaluations across all inputs and pad with dummy values, which we will show is sufficient. We present the full construction in Sect. 5.1.

Spatial Hashing. One drawback of our $\mathsf{sa\text{-}PSI}$ protocol discussed so far is that Bob's communication cost scales with the number of structures in Alice's set N_A. Garimella et al. [39] introduced a technique called *spatial hashing*, that removes this dependence for a *disjoint union* of d-dimensional ℓ_∞-balls of the same, fixed diameter δ. This technique was further improved in [40]. Their high-level idea is to divide the input space into contiguous grid cells, construct FSS keys for all the active grid cells that contain or intersect with Alice's input balls, and then pack these FSS keys into an oblivious key value store (OKVS) data structure [38]. In our work, we present a new, improved spatial hashing technique, that is composable with our new (and prior) framework instantiated with *weak* FSS, and supports overlapping structures of varying size. We discuss more in Sect. 5.2.

Extended Functionalities. We solve another open question from prior work – can we construct $\mathsf{sa\text{-}PSI}$ where Bob (with unstructured set) learns the intersection? We leverage the fact that, Alice can identify a (bounded) set of $w \ll |S_A|$ critical prefixes to represent her set. For a given input $\vec{y} \in S_B$, Bob only needs to compare all its prefixes with Alice's critical prefixes. If Bob learns that there is exactly one match, then he includes $\vec{y}$ in the intersection; otherwise not. A straightforward method is to run a standard PSI-Caridinality between both sets of prefixes, for each of Bob's input. We present the construction in Sect. 8.1.

Finally, we present the first construction for Structure-Aware PSI-Cardinality. A naive approach is that both parties run the best known semi-honest PSI-Cardinality protocol on their inputs. However, since the communication and computation will scale with Alice's set size $|S_A|$, the protocol is not efficient. Instead, since Alice can identify a set of $w \ll |S_A|$ critical prefixes, if

Bob's input intersects/matches with any of the critical prefixes, then it should be counted for intersection cardinality, otherwise not. This idea can be further generalized to compute PSI-Sum. We discuss the full details in Sect. 8.2.

3 Preliminaries

Notation. Let κ and λ denote the computational and statistical security parameter, respectively. For an integer $m \in \mathbb{N}$, let $[m] = \{1, 2, \ldots, m\}$. PPT stands for probabilistic polynomial time. By $\cong_\kappa$ and $\cong_\lambda$ we mean two distributions are computationally indistinguishable and statistically close, respectively. Given a string $\alpha \in \{0,1\}^*$, we use $|\alpha|$ to denote its length, and αi to represent the prefix of length i of α, where $i \in [|\alpha|]$. For two strings $\alpha, \beta \in \{0,1\}^*$, we use $\alpha \| \beta$ to denote string concatenation.

Ball of Diameter δ in ℓ_∞ Metric Space. Let $\boldsymbol{x} = (x_1, \ldots, x_d)$ be a point in a d-dimensional space. The ℓ_∞ norm of $\boldsymbol{x}$ is defined as $|\boldsymbol{x}|_\infty \overset{\text{def}}{=} \max_i |x_i|$. A **ball** consists of the set of points within some distance of a center point. The ball of diameter δ (or radius $\frac{\delta}{2}$) centered at x is defined as $\mathcal{B}(x, \frac{\delta}{2}) \overset{\text{def}}{=} \{y \mid d(x,y) \leq \frac{\delta}{2}\}$. Under the ℓ_∞ norm, a ball $\mathcal{B}(x, \frac{\delta}{2})$ is a polytope with $2d$ faces, *i.e.*, the intersection of $2d$ half-spaces. Namely, the ball $\mathcal{B}(\mathbf{0}, \frac{\delta}{2})$ centered at the origin is the intersection of all half-spaces of the form $\pm x_i \leq \frac{\delta}{2}$ for every i and every choice of sign.

Oblivious Key Value Stores. An oblivious key value stores (OKVS) (introduced in [38]) is a data structure that encodes a set of key value mappings, which effectively hides the underlying key set when the associated values are pseudorandom.

Definition 1 (Oblivious Key Value Stores (OKVS) [38]). *An OKVS consists of algorithms* Encode *and* Decode*, with an associated key space $\mathcal{K}$ and value space $\mathcal{V}$.*

- Encode *takes as input a set of $A \in \mathcal{K} \times \mathcal{V}$ of key-value pairs and outputs an object $\mathcal{KV}$ (or, with statistically small probability, an error indicator $\perp$).*
- Decode *takes as input an object $\mathcal{KV}$, any key $k \in \mathcal{K}$, and outputs a value v.*

An OKVS must satisfy the following properties:

- **Correctness:** *For all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys, and all $(k, v) \in A$:*

$$\Pr[\mathsf{Decode}(\mathsf{Encode}(A), k) = v] = 1$$

  Note that, we can invoke Decode$(\mathcal{KV}, k)$ *on any key k on a given object $\mathcal{KV}$.*
- **Obliviousness:** *For all distinct $\{k_1^0, \ldots, k_n^0\}$ and distinct $\{k_1^1, \ldots, k_n^1\}$, the output of $\mathcal{R}(k_1^0, \ldots, k_n^0)$ is computationally indistinguishable from that of $\mathcal{R}(k_1^1, \ldots, k_n^1)$, where:*

```
$\mathcal{R}(k_1, \ldots, k_n)$:
    for $i \in [n]$: $v_i \leftarrow \mathcal{V}$
    return Encode($\{(k_1, v_1), \ldots, (k_n, v_n)\}$)
```

The obliviousness property guarantees that if the values are pseudorandom, then any two outputs of experiment $\mathcal{R}$, encoding different sets of keys are computationally indistinguishable. Additionally, our construction requires an additional independence property, that is satisfied by all the constructions presented in [38].

Definition 2. *An OKVS satisfies the* ***independence property*** *if for all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct $\mathcal{K}$-values, and any k^* not appearing in the first component of any pair in A, the output of $\mathsf{Decode}(\mathsf{Encode}(A), k^*)$ is indistinguishable from random, over the randomness in Encode.*

Other Cryptographic Tools. We give definitions for other cryptographic tools including pseudorandom generators, Hamming correlation robust hash functions and secure two-party computation (2PC) and specific 2PC functionalities, including oblivious transfer (OT), PSI cardinality, and PSI with associated sum in the full version of the paper.

4 Incremental Boolean Function Secret Sharing

In this section, we define a weak multi-dimensional incremental Boolean Function Secret Sharing (ibFSS) scheme. Looking ahead, we will use this scheme as a building block to construct our structure-aware PSI protocol in Sect. 5. We then give a construction of ibFSS for d-dimensional ℓ_∞-balls in Sect. 6 and discuss our PSI protocol instantiated with it in Sect. 7.

A d-dimensional ibFSS scheme is defined for a family of d-dimensional sets $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$ where $\mathcal{U} = \{0,1\}^u$.[1] As a concrete example, one can think of $\mathcal{S}$ as the family of all d-dimensional ℓ_∞-balls. For each d-dimensional set in the family $\mathcal{S}$, namely $\vec{S} = (S_1 \times \cdots \times S_d) \in \mathcal{S}$ (in the example, $\vec{S}$ is a d-dimensional ℓ_∞-ball), the set $\vec{S}$ implicitly defines a Boolean function f on input space $(\{0,1\}^u)^d$ as follows. For all $\vec{x} = (x_1, \ldots, x_d)$ where each $x_i \in \{0,1\}^u$, we define $f(\vec{x}) = 0$ if $\vec{x} \in \vec{S}$, and $f(\vec{x}) = 1$ otherwise. In *incremental* Boolean FSS, we additionally define the function f evaluated on all prefixes of $\vec{x}$. Specifically, for all $\vec{x} = (x_1, \ldots, x_d)$ where each $x_i \in \bigcup_{\ell=1}^{u} \{0,1\}^\ell$ can be of arbitrary length (at most u bits), we define $f(\vec{x}) = 0$ if the prefix $\vec{x}$ *spans* a set that is entirely contained in $\vec{S}$ (in our formal definition below, $\mathsf{PreS}_{\vec{x}} \subseteq \vec{S}$), and $f(\vec{x}) = 1$ otherwise. An ibFSS scheme splits the function f into two functions f_0, f_1 (defined by keys k_0, k_1) such that each function f_b hides f, and that $f_0(\vec{x}) \oplus f_1(\vec{x}) = f(\vec{x})$ for all prefixes $\vec{x}$.

What we need for structure-aware PSI is a *weak* ibFSS, which relaxes the above definition in two ways. First, the function f outputs a t-bit string instead of a single bit. Second, for each prefix $\vec{x}$, it holds that $f_0(\vec{x}) \oplus f_1(\vec{x}) = 0^t$ if

[1] In the ibFSS definition, we consider all dimensions to have the same input domain for simplicity. The same definition and our constructions also work for dimensions with different input domains.

$\mathsf{PreS}_{\vec{x}} \subseteq \vec{S}$ and $f_0(\vec{x}) \oplus f_1(\vec{x}) \neq 0^t$ otherwise. This weak definition follows from prior work [39,40]. In fact, they further relaxed the definition to allow for false positives, which we don't need to consider in this work. Throughout the rest of the paper, we refer to ibFSS as the *weak* variant unless specified otherwise.

Definition 3. *[(u, d, t)-ibFSS: Syntax] A (two-party) d-dimensional weak incremental Boolean function secret sharing scheme (u, d, t)-ibFSS for a family of sets* $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$, *where* $\mathcal{U} = \{0,1\}^u$, *consists of a pair of algorithms* $(\mathsf{Share}, \mathsf{Eval})$ *with the following syntax:*

- $(k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, \hat{S})$*: The randomized share function takes as input the security parameter κ and (the description of) a d-dimensional set $\vec{S} = (S_1 \times \cdots \times S_d) \in \mathcal{S}$, and outputs two key shares.*
- $y_{idx} \leftarrow \mathsf{Eval}(idx, k_{idx}, \vec{x})$*: The deterministic evaluation function takes as input a party index $idx \in \{0,1\}$, the corresponding key share k_{idx}, and input $\vec{x} = (x_1, \ldots, x_d)$ where each $x_i \in \bigcup_{\ell=1}^{u}\{0,1\}^\ell$. It outputs a string $y_{idx} \in \{0,1\}^t$.*

Notation. For a string $x \in \bigcup_{\ell=1}^{u}\{0,1\}^\ell$ and $\mathcal{U} = \{0,1\}^u$, we define a prefix set $\mathsf{PreS}_x := \{s \mid s|x| = x, s \in \mathcal{U}\}$ as the set of all the strings in $\mathcal{U}$ with a prefix x. For a vector $\vec{x} = (x_1, \ldots, x_d)$, where each $x_i \in \bigcup_{\ell=1}^{u}\{0,1\}^\ell$, we define $\mathsf{PreS}_{\vec{x}} := \{\vec{s} = (s_1, \ldots, s_d) \mid (s_i)|x_i| = x_i \text{ for all } i \in [d], s_i \in \mathcal{U}\}$. Note that $\mathsf{PreS}_{\vec{x}} = \mathsf{PreS}_{x_1} \times \cdots \times \mathsf{PreS}_{x_d}$.

Definition 4. *[(u, d, t)-ibFSS: Security] A (two-party) (u, d, t)-ibFSS scheme $(\mathsf{Share}, \mathsf{Eval})$ for $\mathcal{S}$ is secure if it satisfies the following conditions:*

- ***Correctness for yes-instances:*** *For every d-dimensional set $\vec{S} = (S_1 \times \cdots \times S_d) \in \mathcal{S}$, every input $\vec{x} = (x_1, \ldots, x_d)$ for which $\mathsf{PreS}_{x_1} \subseteq S_1, \ldots, \mathsf{PreS}_{x_d} \subseteq S_d$ (or equivalently, $\mathsf{PreS}_{\vec{x}} \subseteq \vec{S}$), and security parameter κ,*

$$\Pr\left(y_0 \oplus y_1 = 0^t \;\middle|\; \begin{array}{l} (k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, \vec{S}) \\ y_0 \leftarrow \mathsf{Eval}(0, k_0, \vec{x}) \\ y_1 \leftarrow \mathsf{Eval}(1, k_1, \vec{x}) \end{array} \right) = 1.$$

- ***Correctness for no-instances:*** *For every d-dimensional set $\vec{S} = (S_1 \times \cdots \times S_d) \in \mathcal{S}$, every input $\vec{x} = (x_1, \ldots, x_d)$ for which $\mathsf{PreS}_{x_1} \not\subseteq S_1$ or ... or $\mathsf{PreS}_{x_d} \not\subseteq S_d$ (or equivalently, $\mathsf{PreS}_{\vec{x}} \not\subseteq \vec{S}$), and security parameter κ,*

$$\Pr\left(y_0 \oplus y_1 \neq 0^t \;\middle|\; \begin{array}{l} (k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, \vec{S}) \\ y_0 \leftarrow \mathsf{Eval}(0, k_0, \vec{x}) \\ y_1 \leftarrow \mathsf{Eval}(1, k_1, \vec{x}) \end{array} \right) = 1.$$

- ***Privacy:*** *There exists a PPT simulator Sim such that for all $\mathsf{idx} \in \{0,1\}$ and all $\vec{S} \in \mathcal{S}$, the following distributions are computationally indistinguishable in the security parameter κ:*

$$\boxed{\begin{array}{l} (k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, \vec{S}) \\ \text{return } k_{\mathsf{idx}} \end{array}} \cong_\kappa \mathsf{Sim}(1^\kappa, \mathsf{idx}).$$

We say the ibFSS scheme has pseudorandom keys property, if the output of the simulator is a random string of fixed length.

Decomposing a Set into Prefix Sets. In our structure-aware PSI protocol, we will decompose each structured set into a disjoint union of prefix sets. Specifically, for any single-dimensional set $S \in 2^{\mathcal{U}}$, there is a unique way to decompose it into a disjoint union of prefix sets, $S = \dot{\bigcup}_{j\in[w]} \mathsf{PreS}_{x^j}$ where $x^j \in \bigcup_{\ell=1}^{u} \{0,1\}^{\ell}$. Similarly, for any d-dimensional set $\vec{S} = (S_1 \times \cdots \times S_d) \in \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$, there is a unique way to decompose it into a disjoint union of prefix sets, $\vec{S} = \dot{\bigcup}_{j\in[w]} \mathsf{PreS}_{\overrightarrow{x^j}}$ where $\overrightarrow{x^j} = (x_1^j, \dots, x_d^j)$ and $x_i^j \in \bigcup_{\ell=1}^{u} \{0,1\}^{\ell}$.

For instance, consider a single-dimensional interval $S = \{x \mid x < \alpha,\ x \in \{0,1\}^u\}$, where $\alpha = \overline{\alpha^{(1)}\alpha^{(2)}\dots\alpha^{(u)}} \in \{0,1\}^u$ (bit representation of α). The set S can be decomposed into a disjoint union of at most u prefix sets, namely $S = \dot{\bigcup}_{j\in[u]:\alpha^{(j)}=1} \mathsf{PreS}_{\overline{\alpha^{(1)}\dots\alpha^{(j-1)}0}}$. Similarly, a d-dimensional interval can be decomposed into a disjoint union of at most u^d prefix sets.

5 sa-PSI with Alice Learning Output

In this section, we present our new sa-PSI protocols where Alice holds a structured set and Bob holds a set of unstructured points. We present the protocol where Alice's input is a union of (overlapping) structures and Bob's input contains multiple points in Sect. 5.1 and our new spatial hashing techniques in Sect. 5.2. In both protocols, only Alice learns the output.

5.1 Multiple (Overlapping) Balls with Multiple Points

We present in Fig. 2 a general framework for two-party, semi-honest secure sa-PSI protocol, where Alice's input is a union of (overlapping) structures in a d-dimensional, well-defined metric space. We summarize the key features in our framework below.

- **Incremental Boolean FSS** is a new tool, that allows Alice and Bob to evaluate on the prefixes of their inputs. Bob evaluates his key share on *every prefix* of his input. Alice identifies a set of *critical prefixes*, such that, a match with any critical prefix indicates that the element is in the intersection.
- **Intersection Search** is a method for Alice to recursively search for Bob's input in her set. For every critical prefix in Alice's input, she checks if there is a matching FSS evaluation in Bob's set Y. She then appends bit 0 to the prefix and checks if the FSS evaluation belongs to set Y. If not, she is guaranteed to find prefix append with bit 1 in set Y. This process continues until she figures out the matching input.

- **OR Observation** is a new method for sa-PSI when Alice's input is a union of N_A structures. In our protocol, we break away from the ibFSS abstraction for the union of structures. Instead, Alice computes an independent ibFSS secret sharing *for each of her structures* and compares with *Bob's entire input* set S_B. An immediate advantage is that the sets can be overlapping since they are handled independently. Furthermore, prior known FSS constructions for union of structures are not as efficient as FSS for individual structures, leading to an advantage for our approach (for the set families considered in [39,40]). We discuss the comparison in more detail in Sect. 7.

Parameters: A family of sets $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$ where $\mathcal{U} = \{0,1\}^u$. Number of Alice's structured sets N_A and size of Bob's set N_B.

Functionality:
1. Receive input $S_A = \bigcup_{i \in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$ (or a concise representation of S_A) from Alice.
2. Receive input $S_B \subseteq \mathcal{U}^d$ of size N_B from Bob.
3. **[output]** Send $S_A \cap S_B$ to Alice.

Fig. 1. Ideal functionality $\mathcal{F}_{\textsf{sa-PSI}}$ for structure-aware PSI, where Alice learns the output.

Algorithm 1. Recursive Intersection Search

```
1: global variable: intersection set $I$
2: procedure IntSearch($\{k^{(\eta)}\}_{\eta \in [\kappa]}, u, \mathsf{index}, \vec{x}, Y$)
3:     Parse $\vec{x} = (x_1, \ldots, x_d)$
4:     Compute $h := \mathsf{H}\left(\vec{x} \| \mathsf{index}; \mathsf{Eval}(k^{(1)}, \vec{x}) \| \ldots \| \mathsf{Eval}(k^{(\kappa)}, \vec{x})\right)$
5:     if $h \notin Y$ then
6:         return
7:     else if $|x_i| = u$ for all $i \in [d]$ then
8:         $I := I \cup \{\vec{x}\}$
9:         return
10:    Let $i \in [d]$ be the smallest index such that $|x_i| \leq u - 1$
11:    for $b = 0$ to $1$ do
12:        $\vec{x'} := (x_1, \ldots, x_{i-1}, x_i \| b, x_{i+1}, \ldots, x_d)$
13:        IntSearch($\{k^{(\eta)}\}_{\eta \in [\kappa]}, u, \mathsf{index}, \vec{x'}, Y$)
```

Theorem 1. *Given a two-party (u, d, t)-ibFSS scheme for a family of sets $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$ where $\mathcal{U} = \{0,1\}^u$ and every set in $\mathcal{S}$ is a disjoint union of at most w prefix sets, and a Hamming-correlation robust hash $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda + \log|N_A| + \log|N_B| + \log w + d \log u}$, the protocol in Fig. 2 realizes $\mathcal{F}_{\textsf{sa-PSI}}$ (Fig. 1) in the $\mathcal{F}_{\textsf{OT}}$-hybrid model in the presence of semi-honest adversaries.*

Parameters:

- computational security parameter κ and statistical security parameter λ
- family of sets $\mathcal{S}$ with corresponding (u,d,t)-ibFSS scheme $(\mathsf{Share}, \mathsf{Eval})$
- oblivious transfer functionality $\mathcal{F}_{\mathsf{OT}}$
- hamming-correlation robust hash H with output length $\lambda + \log|N_A| + \log|N_B| + \log w + d\log u$

Inputs:

- Alice has N_A structured sets $S_A = \bigcup_{i\in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$.
- Bob has an unstructured set $S_B \subseteq \mathcal{U}^d$ of size N_B.

Protocol:

1. Alice initializes the intersection as $I := \emptyset$.
2. Bob chooses a random string $s \leftarrow \{0,1\}^\kappa$.
3. For each $i \in [N_A]$, the parties do the following (in parallel for each set):
   (a) Alice generates κ independent ibFSS sharings of her structured set $S_A^{(i)}$. For each $\eta \in [\kappa]$, compute $(k_0^{(\eta)}, k_1^{(\eta)}) \leftarrow \mathsf{Share}(1^\kappa, S_A^{(i)})$.
   (b) The parties invoke κ (parallel) instances of oblivious transfer using $\mathcal{F}_{\mathsf{OT}}$. In the η-th instance:
      - Alice is the sender with input $(k_0^{(\eta)}, k_1^{(\eta)})$;
      - Bob is the receiver with choice bit $s[\eta]$. He obtains output $k_*^{(\eta)} = k_{s[\eta]}^{(\eta)}$.
   (c) Bob does the following:
      i. Initialize an empty set $Y := \emptyset$.
      ii. For each element $\vec{y} \in S_B$:
         - Write $\vec{y} = (y_1, \dots, y_d)$, where $y_j \in \mathcal{U}$ for all $j \in [d]$.
         - Compute

$$Y' = \left\{ \mathsf{H}\left(\vec{y'} \| i; \mathsf{Eval}(k_*^{(1)}, \vec{y'}) \| \dots \| \mathsf{Eval}(k_*^{(\kappa)}, \vec{y'})\right) \middle| \begin{array}{l} \ell_1 \in [u], \dots, \ell_d \in [u] \\ \vec{y'} := ((y_1)_{[1:\ell_1]}, \dots, (y_d)_{[1:\ell_d]}) \end{array} \right\}$$

         - Update $Y := Y \cup Y'$.
      iii. Pad Y with dummy random strings such that $|Y| = N_B \cdot u^d$ and send it to Alice.
   (d) Alice does the following:
      i. Write $S_A^{(i)} = \dot{\bigcup}_{j\in[w']} \mathsf{PreS}_{\vec{x^j}}$ as disjoint union of w' prefix sets (w is the upper bound for w').
      ii. For each $j \in [w']$, run $\textsc{IntSearch}\left(\{k_0^{(\eta)}\}_{\eta\in[\kappa]}, u, i, \vec{x^j}, Y\right)$, with I being a global variable.
4. **[output]** Alice outputs the intersection I.

Fig. 2. Protocol realizing $\mathcal{F}_{\mathsf{sa\text{-}PSI}}$ (Fig. 1) using ibFSS in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model.

Proof. **Bob is Corrupt.** In the protocol, Bob learns only one of the key shares from OT for each of the κ independent ibFSS sharings. We can simulate Bob's view by invoking the ibFSS simulator (Definition 4) that takes as input the security parameter κ and (public information) a description of Alice's set family and nothing else about her private input.

Alice is Corrupt. First, we define the following useful functions, where each function's output $\{\{0,1\}^t\}^\kappa$ is a vector of length κ, where each component (string length t) is the result of ibFSS evaluation:

$$\begin{aligned}
E_*(\vec{y}) &= \left(\mathsf{Eval}(k_*^{(1)}, \vec{y}) \| \ldots \| \mathsf{Eval}(k_*^{(\kappa)}, \vec{y})\right) \\
E_0(\vec{y}) &= \left(\mathsf{Eval}(k_0^{(1)}, \vec{y}) \| \ldots \| \mathsf{Eval}(k_0^{(\kappa)}, \vec{y})\right) \\
\Delta(\vec{y}) &= \left(\mathsf{Eval}(k_0^{(1)}, \vec{y}) \oplus \mathsf{Eval}(k_1^{(1)}, \vec{y}) \| \ldots \| \mathsf{Eval}(k_0^{(\kappa)}, \vec{y}) \oplus \mathsf{Eval}(k_1^{(\kappa)}, \vec{y})\right)
\end{aligned}$$

The goal is to simulate (through a sequence of hybrids) Alice's view from the honest protocol execution independent of Bob's input.

Hybrid 0. In the protocol execution, Alice's view consists of N_A sets of values from Bob, which can effectively be viewed as a single of set of values Y. We use the fact that

$$\mathsf{Eval}(k_*^{(\eta)}, \vec{y'}) = \mathsf{Eval}(k_{s_i}^{(\eta)}, \vec{y'}) = \mathsf{Eval}(k_0^{(\eta)}, \vec{y'}) \oplus s \odot (\mathsf{Eval}(k_0^{(\eta)}, \vec{y'}) \oplus \mathsf{Eval}(k_1^{(\eta)}, \vec{y'}))$$

where $\odot$ denotes component wise multiplication (of single bit with t bit string) to rewrite the message Y as shown below:

$$\begin{aligned}
Y &= \left\{ \mathsf{H}\left(\vec{y'} \| i; \mathsf{Eval}(k_*^{(1)}, \vec{y'}) \| \ldots \| \mathsf{Eval}(k_*^{(\kappa)}, \vec{y'})\right) \,\middle|\, \begin{array}{l} i \in [N_A], \vec{y} = (y_1, \ldots, y_d) \in S_B, \\ \ell_1 \in [u], \ldots, \ell_d \in [u], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\} \\
&= \left\{ \mathsf{H}\left(\vec{y'} \| i; E_*(\vec{y'}\right) \,\middle|\, \begin{array}{l} i \in [N_A], \vec{y} = (y_1, \ldots, y_d) \in S_B, \\ \ell_1 \in [u], \ldots, \ell_d \in [u], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\} \\
&= \left\{ \mathsf{H}\left(\vec{y'} \| i; E_0(\vec{y'}) \oplus s \odot \Delta(\vec{y'})\right) \,\middle|\, \begin{array}{l} i \in [N_A], \vec{y} = (y_1, \ldots, y_d) \in S_B, \\ \ell_1 \in [u], \ldots, \ell_d \in [u], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\} \\
&= Y_1 \,\dot\cup\, Y_2 \,\dot\cup\, Y_3 \dot\cup\, Y_4,
\end{aligned}$$

where we expand the terms as follows:

$$Y_1 = \left\{ \mathsf{H}\left(\vec{y'} \| i; E_0(\vec{y'})\right) \,\middle|\, \begin{array}{l} i \in [N_A], S_A^{(i)} = \dot\bigcup_{j \in [w']} \mathsf{PreS}_{\vec{x^j}}, \overrightarrow{x^j} = (x_1^j, \ldots, x_d^j), \\ \vec{y} = (y_1, \ldots, y_d) \in \mathsf{PreS}_{\vec{x^j}} \cap S_B, \\ \ell_1 \in [|x_1^j| : u], \ldots, \ell_d \in [|x_d^j| : u], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\}$$

$$Y_2 = \left\{ \mathsf{H}\left(\vec{y'} \| i; E_0(\vec{y'}) \oplus s \odot \Delta(\vec{y'})\right) \,\middle|\, \begin{array}{l} i \in [N_A], S_A^{(i)} = \dot\bigcup_{j \in [w']} \mathsf{PreS}_{\vec{x^j}}, \overrightarrow{x^j} = (x_1^j, \ldots, x_d^j), \\ \vec{y} = (y_1, \ldots, y_d) \in \mathsf{PreS}_{\vec{x^j}} \cap S_B, \\ (\ell_1, \ldots, \ell_d) \in [u]^d, \text{where } \ell_1 \in [|x_1^j - 1|] \vee \ldots \vee \ell_d \in [|x_d^j - 1|], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\}$$

$$Y_3 = \left\{ \mathsf{H}\left(\vec{y'}\|i; E_0(\vec{y'}) \oplus s \odot \Delta(\vec{y'})\right) \middle| \begin{array}{l} i \in [N_A], \vec{y} = (y_1, \ldots, y_d) \in S_B \setminus \vec{S}_A^{(i)}, \\ \ell_1 \in [u], \ldots, \ell_d \in [u], \\ \vec{y'} := ((y_1)\ell_1, \ldots, (y_d)\ell_d) \end{array} \right\}$$

and Y_4 consists of dummy pseudorandom strings with length equal to the hash outputs, such that $|Y| = N_B \cdot u^d$. Recall that, Bob sends hash evaluations on all prefix combinations of each of his inputs for every input $S_A^{(i)} = \dot{\bigcup}_{j \in [w']} \mathsf{PreS}_{\vec{x^j}}$ in Alice's private input. For Bob's protocol message Y we get the following cases for every set $S_A^{(i)}$:

1. If Bob's input $\vec{y} \in \mathsf{PreS}_{\vec{x^j}}$ and input prefix $\vec{y'} \in \mathsf{PreS}_{\vec{x^j}}$: set Y_1 includes hash evaluation on prefix $\mathsf{H}(\vec{y'}\|i; E_*(\vec{y'}))$. The ibFSS scheme correctness guarantees $\Delta(\vec{y'}) = 0^t$ when evaluated on an input $\vec{y'}$ in some prefix set $\vec{y'} \in \mathsf{PreS}_{\vec{x^j}}$ that comprises one of Alice's sets. Thus, we can simplify $\mathsf{H}(\vec{y'}\|i; E_0(\vec{y'}) \oplus s \odot \Delta(\vec{y'})) = \mathsf{H}(\vec{y'}\|i; E_0(\vec{y'}))$.
2. If Bob's input $\vec{y} \in \mathsf{PreS}_{\vec{x^j}}$ and input prefix $\vec{y'} \notin \mathsf{PreS}_{\vec{x^j}}$: set Y_2 includes hash evaluation on prefix $\mathsf{H}(\vec{y'}\|i; E_*(\vec{y'}))$. The ibFSS scheme correctness guarantees $\Delta(\vec{y'})$ consists of κ components, where every component (string of length t) is non-zero, when evaluated on an input $\vec{y'}$ outside the prefix set.
3. If Bob's input $\vec{y} \notin S_A^{(i)}$ and input prefix $\vec{y'}$: set Y_3 includes hash evaluation on prefix $\mathsf{H}(\vec{y'}\|i; E_*(\vec{y'}))$. Again, the ibFSS scheme correctness guarantees $\Delta(\vec{y'})$ consists of κ components, where every component (string of length t) is non-zero, when evaluated on an input $\vec{y'}$ outside every prefix set of $S_A^{(i)}$.

Hybrid 1. In this hybrid, we replace all the hash outputs in Y_2 and Y_3 with uniform strings of length $\lambda + \log|N_A| + \log|N_B| + \log w + d\log u$ as shown below:

$$Y = Y_1 \,\dot{\cup}\, \{h_1, \ldots, h_{|Y_2|}\} \,\dot{\cup}\, \{h'_1, \ldots, h_{|Y_3|}\} \,\dot{\cup}\, Y_4$$

where each $h_i, h'_i \leftarrow \{0,1\}^{\lambda + \log|N_A| + \log|N_B| + \log w + d\log u}$ are sampled uniformly. This hybrid is indistinguishable from previous hybrid by the hamming correlation robust property which states that hash values of the form $H(\vec{y'}\|i; t_i \oplus s \odot \Delta(\vec{y'}))$ are jointly pseudorandom, if they are never queried on repeated $\vec{y'}\|i$ values and $\Delta(\vec{y'})$ has at least κ non-zero components. As previously discussed all the hash outputs in Y_2 and Y_3 are computed on unique prefix concatenate with structure index values and have non-zero $\Delta(\vec{y'})$ evaluation with κ non-zero components.

Simulator. Hybrid 1 defines a valid simulation in the ideal world computed just using Alice's input S_A and output $S_A \cap S_B$. Note that, the output can be used to determine the cardinality of sets Y_2, Y_3 and Y_4.

Correctness. Only Alice learns the output and our goal is to show that the protocol satisfies correctness. It suffices to consider Alice's simulated view since it is indistinguishable from her view in the honest execution of the protocol.

For every input $\vec{x} \in S_B \cap \mathsf{PreS}_{\vec{x^j}}$ in the intersection (over all choices of $i \in [N_A], S_A^{(i)} = \dot{\bigcup}_{j\in[w']}\mathsf{PreS}_{\vec{x^j}}, \overrightarrow{x^j} = (x_1^j, \dots, x_d^j)$), ibFSS scheme guarantees $\Delta(\vec{x'}) = 0^t$ and the simulator includes the following hash values in the message to Alice:

$$\left\{ \mathsf{H}\left(\vec{x'}\|i; E_0(\vec{x'})\right) \middle| \begin{array}{l} \ell_1 \in [|x_1^j| : u], \dots, \ell_d \in [|x_d^j| : u] \\ \vec{x'} := ((x_1)\ell_1, \dots, (x_d)\ell_d) \end{array} \right\}$$

By the collision resistant hash property, Alice can uniquely compute and recognize hash values from Bob's message and will correctly include every such element $\vec{x}$ in the output.

What is the probability that Alice wrongly includes an element $\vec{x} \in S_A \setminus S_B$ in her output? This value is upper bounded by the probability that Alice finds the hash output of prefix of $\vec{x}$ in Bob's message, that is, $\mathsf{H}\left(\vec{x'}\|i; E_0(\vec{x'})\right) \in Y$, where $\vec{x} \in \mathsf{PreS}_{\vec{x'}}$. By union bound, the probability that Alice finds a value matching a specific prefix $\mathsf{PreS}_{\vec{x'}}$ is $|Y'| \cdot 2^{-\lambda - \log|N_A| - \log|N_B| - \log w - d\log u} = 2^{-\lambda - \log|N_A| - \log w}$. Again, by the union bound over total number of prefixes, the probability that Alice includes a wrong element $\vec{x}$ in her output is less than $w \cdot |N_A| \cdot 2^{-\lambda - \log|N_A| - \log w} = 2^{-\lambda}$, which is negligible.

5.2 Spatial Hashing

In the above construction, both parties compute ibFSS over the entire universe. Improving upon this construction framework, Garimella et al. [39] introduced a spatial hashing technique to reduce the input domain for better efficiency. In this section, we present a new way of looking at spatial hashing, which relaxes the restrictions on the structured sets and improves both computation and communication costs.

Overview of Our Spatial Hashing Technique. We start with the same setting as prior work [39,40], where Alice holds a union of d-dimensional ℓ_∞-balls of diameter δ, namely every ball can be written as $[x_1, x_1+\delta) \times \cdots \times [x_d, x_d+\delta)$. We denote the ball as $\mathsf{Ball}_{\vec{x}}$ for its "left-bottom corner" $\vec{x} = (x_1, \dots, x_d)$. We partition the entire universe $(\{0,1\}^u)^d$ into contiguous grid cells of side length δ. For each vertex in the partitioned space, namely $\vec{o} = (o_1, o_2, \dots, o_d)$ where each o_i is a multiple of δ, we define a *mini-universe* with *origin* $\vec{o}$ and diameter 2δ. That is, $\mathsf{Univ}_{\vec{o}} := [o_1, o_1 + 2\delta) \times \dots [o_d, o_d + 2\delta)$. If all of Alice's balls are disjoint and have the same diameter δ, then each $\mathsf{Ball}_{\vec{x}}$ corresponds to a distinct grid cell, which is where $\vec{x}$ is located at. Let $\vec{o}$ be the "left-bottom corner" of that grid cell, then it is guaranteed that $\mathsf{Ball}_{\vec{x}} \subseteq \mathsf{Univ}_{\vec{o}}$. We can construct an ibFSS for $\mathsf{Ball}_{\vec{x}}$ in the mini-universe $\mathsf{Univ}_{\vec{o}}$ with diameter 2δ.

Recall that each ball corresponds to a unique origin $\vec{o}$, which we refer to as *active origins* with *active mini-universes*. We prepare an ibFSS key for each ball $\mathsf{Ball}_{\vec{x}}$ in its corresponding mini-universe $\mathsf{Univ}_{\vec{o}}$. Then we pack them into an oblivious-key value storage (OKVS) data structure [38], where the origin is treated as the OKVS key, and the corresponding ibFSS key is treated as its value.

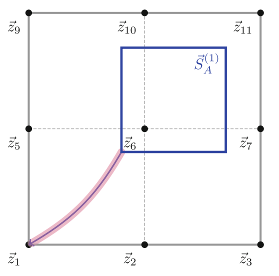
To check if a point $\vec{y}$ in Bob's set is contained in any of Alice's input balls, we only need to check all the mini-universes that contains $\vec{y}$, namely $\vec{y} \in \mathsf{Univ}_{\vec{o}}$ for origins $\vec{o}$ (there are 2^d of them). Bob can probe the OKVS data structure to obtain the ibFSS key for each relevant origin $\vec{o}$, and evaluate ibFSS on $\vec{y}$ in the corresponding mini-universe $\mathsf{Univ}_{\vec{o}}$. Thanks to our new OR observation, Bob can simply send ibFSS evaluations for all the structures within any relevant origin $\vec{o}$ back to Alice, without having to perform a sum transformation.

Illustrative Examples. We give a few examples in Fig. 3 in 2-dimensional space to illustrate our new spacial hashing techniques. A structured set $\vec{S}$ can be assigned to an origin $\vec{o}$ if $\vec{S}$ is completely contained in the mini-universe with origin $\vec{o}$ and diameter 2δ, namely $\vec{S} \subseteq \mathsf{Univ}_{\vec{o}}$. For instance, Fig. 3a shows a mini-universe with origin $\vec{z}_1$, and $\vec{S}_A^{(1)} \subseteq \mathsf{Univ}_{\vec{z}_1}$. Therefore, the structured set $\vec{S}_A^{(3)}$ in Fig. 3b can be assigned to any of the four origins $\vec{z}_2, \vec{z}_3, \vec{z}_6, \vec{z}_7$. In Fig. 3c, we consider 3 structured sets in Alice's set. $\vec{S}_A^{(1)}$ can be assigned to $\vec{z}_1$; $\vec{S}_A^{(2)}$ can be assigned to origins $\vec{z}_2$ or $\vec{z}_6$; $\vec{S}_A^{(3)}$ can be assigned to any of the four origins $\vec{z}_2, \vec{z}_3, \vec{z}_6, \vec{z}_7$. In this example, we assign $\vec{S}_A^{(1)}, \vec{S}_A^{(2)}, \vec{S}_A^{(3)}$ to $\vec{z}_1, \vec{z}_6, \vec{z}_3$, respectively (highlighted in the figure). Alice generates an ibFSS key share for each structured set and packs them into an OKVS, as shown in Fig. 3e. When Bob evaluates at point $\vec{y}$ (orange point in the figure), he needs to consider all the mini-universes that contain $\vec{y}$, namely $\mathsf{Univ}_{\vec{z}_1}, \mathsf{Univ}_{\vec{z}_2}, \mathsf{Univ}_{\vec{z}_5}, \mathsf{Univ}_{\vec{z}_6}$. In our new spatial hashing framework, Alice's structured sets can overlap as long as every $\vec{S}_A^{(i)}$ can be assigned to a distinct origin $\vec{o}^{(i)}$ such that $\vec{S}_A^{(i)} \subseteq \mathsf{Univ}_{\vec{o}^{(i)}}$. In another example in Fig. 3d, we cannot find such a mapping.

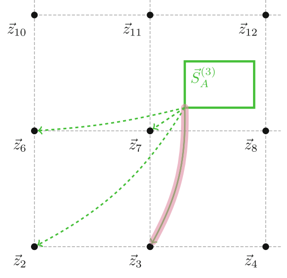
What if There is no One-to-One Mapping? When it is impossible to find a balls-to-origins mapping in spatial hashing (e.g., Fig. 3d), one potential solution is to increase the number of structures that can be mapped to each universe, and then perform multi-ball-multi-point $\mathsf{sa\text{-}PSI}$ for each universe. Another potential solution is to have smaller grid cells and a larger universe size, so that Alice can find a unique mapping. Both solutions would incur higher communication and computation costs. The protocol can be adapted depending on the specific application.

Extensions and Improvements Over Prior Work. First, we improve both computation and communication costs of spatial hashing. In prior works [39,40], Bob needs to perform an expensive sum transformation on the bFSS outputs. This restricts their bFSS to *strong* bFSS that outputs a single bit, for which both computation and communication costs grow exponentially in the dimension d. In contract, we get rid of the sum transformation, hence allowing for *weak* bFSS that outputs a bit string, with both computation and communication costs linear in d.

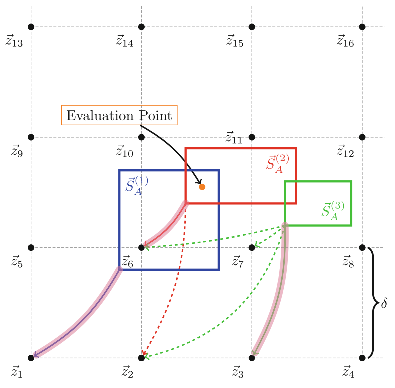
Second, when constructing and evaluating our ibFSS in a mini-universe $\mathsf{Univ}_{\vec{o}}$, we *shift* all the points by the origin $\vec{o}$. Specifically, for $\vec{x} = (x_1, \ldots, x_d)$ and $\vec{o} = (o_1, \ldots, o_d)$, we consider $\vec{x}.\mathsf{Shift}(\vec{o}) := (x_1 - o_1, \ldots, x_d - o_d)$ in the mini-universe with diameter 2δ. When Bob computes the hash value, we include the



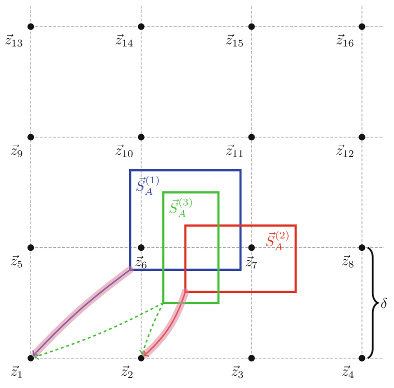
(a) $\vec{S}_A^{(1)} \subseteq \mathsf{Univ}_{\vec{z}_1}$



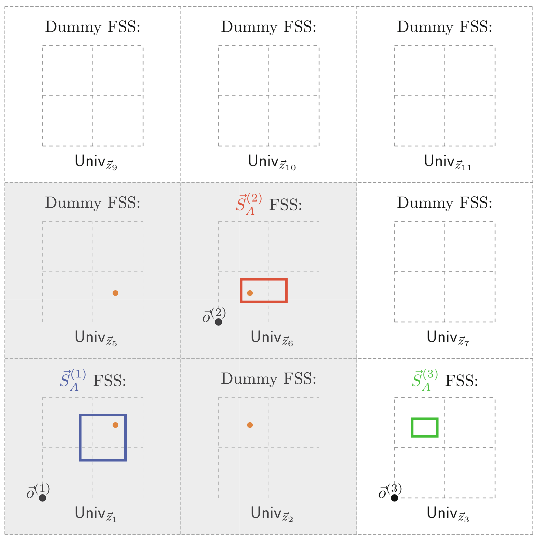
(b) $\vec{S}_A^{(3)}$ can be mapped to any of the four origins $\vec{z}_2, \vec{z}_3, \vec{z}_6, \vec{z}_7$.



(c) Mapping structured sets to distinct origins (highlighted arrows).



(d) Impossible to find a mapping.


Dummy FSS:
$\mathsf{Univ}_{\vec{z}_9}$
Dummy FSS:
$\mathsf{Univ}_{\vec{z}_{10}}$
Dummy FSS:
$\mathsf{Univ}_{\vec{z}_{11}}$
Dummy FSS:
$\mathsf{Univ}_{\vec{z}_5}$
$\vec{S}_A^{(2)}$ FSS:
$\vec{o}^{(2)}$
$\mathsf{Univ}_{\vec{z}_6}$
Dummy FSS:
$\mathsf{Univ}_{\vec{z}_7}$
$\vec{S}_A^{(1)}$ FSS:
$\vec{o}^{(1)}$
$\mathsf{Univ}_{\vec{z}_1}$
Dummy FSS:
$\mathsf{Univ}_{\vec{z}_2}$
$\vec{S}_A^{(3)}$ FSS:
$\vec{o}^{(3)}$
$\mathsf{Univ}_{\vec{z}_3}$

(e) Bob's evaluation on all mini-universes that contain $\vec{y}$.

Fig. 3. Our new spatial hashing techniques in 2-dimensional space.

Parameters:

- computational security parameter κ and statistical security parameter λ
- set family $\mathcal{S}$ with corresponding $((\log(2\delta), d, t)$-ibFSS scheme $(\mathsf{Share}, \mathsf{Eval})$ (Definition 3)
- oblivious key-value store scheme $(\mathsf{Encode}, \mathsf{Decode})$ (Definition 1, Definition 2)
- oblivious transfer ideal functionality $\mathcal{F}_{\mathsf{OT}}$
- ℓ OT instances such that $\Pr[\mathsf{Binomial}(1 - 2^{-t}, \ell) < \kappa] < 2^{-\lambda - |N_B| - d(1+\log\log(2\delta))}$
- hamming-correlation robust hash H with output length $\lambda + \log|N_A| + \log w + \log|N_B| + d(1 + \log\log(2\delta))$

Inputs:

- Alice has N_A structured sets $S_A = \bigcup_{i \in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$.
- Bob has an unstructured set $S_B \subseteq \mathcal{U}^d$ of size N_B.

Notation:

- For $\vec{o} = (o_1, \ldots, o_d) \in \mathcal{U}^d$, $\mathsf{Univ}_{\vec{o}} := [o_1, o_1 + 2\delta) \times \cdots \times [o_d, o_d + 2\delta)$.
- For $\vec{x} = (x_1, \ldots, x_d), \vec{o} = (o_1, \ldots, o_d) \in \mathcal{U}^d$, $\vec{x}.\mathsf{Shift}(\vec{o}) := (x_1 - o_1, \ldots, x_d - o_d)$.
- For $S \subseteq \mathcal{U}^d$ and $\vec{o} = (o_1, \ldots, o_d) \in \mathcal{U}^d$, $S.\mathsf{Shift}(\vec{o}) := \{\vec{x}.\mathsf{Shift}(\vec{o}) | \vec{x} \in S\}$.

Protocol:

1. For each $i \in [N_A]$, Alice identifies a distinct origin $\vec{o}^{(i)}$ such that $\vec{S}_A^{(i)} \subseteq \mathsf{Univ}_{\vec{o}^{(i)}}$.
2. For each $\eta \in [\ell]$, Alice does the following:
   (a) For each $i \in [N_A]$ compute $(k_0^{(i,\eta)}, k_1^{(i,\eta)}) \leftarrow \mathsf{Share}(1^\kappa, \vec{S}_A^{(i)}.\mathsf{Shift}(\vec{o}^{(i)}))$.
   (b) For $b \in \{0, 1\}$ encode $\mathcal{KV}_b^{(\eta)} \leftarrow \mathsf{Encode}(\{(\vec{o}^{(1)}, k_b^{(1,\eta)}), \ldots, (\vec{o}^{(N_A)}, k_b^{(N_A,\eta)})\})$.
3. Bob chooses a random string $s \leftarrow \{0, 1\}^\ell$.
4. The parties invoke ℓ (parallel) instances of $\mathcal{F}_{\mathsf{OT}}$. In the η-th instance:
   - Alice is the sender with input $(\mathcal{KV}_0^{(\eta)}, \mathcal{KV}_1^{(\eta)})$;
   - Bob is the receiver with choice bit $s[\eta]$. He obtains output $\mathcal{KV}_*^{(\eta)} = \mathcal{KV}_{s[\eta]}^{(\eta)}$.
5. Bob initializes a set $Y := \emptyset$ and does the following:
   (a) For each element $\vec{y} \in S_B$, and for each origin $\vec{z}$ where $\vec{y} \in \mathsf{Univ}_{\vec{z}}$:
      i. Write $\vec{y}.\mathsf{Shift}(\vec{z}) = (y_1, \ldots, y_d)$, where $y_j \in [0, 2\delta)$ for all $j \in [d]$.
      ii. For each $\eta \in [\ell]$, compute $k_*^{(\eta)} := \mathsf{Decode}(\mathcal{KV}_*^{(\eta)}, \vec{z})$.
      iii. Compute
$$Y' = \left\{\mathsf{H}\left(\vec{y'} \| \vec{z}; \mathsf{Eval}(k_*^{(1)}, \vec{y'}) \| \ldots \| \mathsf{Eval}(k_*^{(\ell)}, \vec{y'})\right) \middle| \begin{array}{l} \ell_1 \in [\log(2\delta)], \ldots, \ell_d \in [\log(2\delta)] \\ \vec{y'} := ((y_1)_{[1:\ell_1]}, \ldots, (y_d)_{[1:\ell_d]}) \end{array}\right\}$$
      iv. Update $Y := Y \cup Y'$.
   (b) Pad Y with dummy random strings such that $|Y| = N_B \cdot (2 \cdot \log(2\delta))^d$ and send it to Alice.
6. Alice initializes set $I := \emptyset$. For each set $i \in [N_A]$, Alice does the following:
   (a) Write $\vec{S}_A^{(i)}.\mathsf{Shift}(\vec{o}^{(i)}) = \dot{\bigcup}_{j \in [w']} \mathsf{PreS}_{\vec{x^j}}$ as disjoint union of w' prefix sets (w is the upper bound for w').
   (b) For each $j \in [w']$, run $\textsc{IntSearch}\left(\{k_0^{(i,\eta)}\}_{\eta \in [\ell]}, \log(2\delta), \vec{o}^{(i)}, \vec{x^j}, Y\right)$, with I being a global variable.
7. **[output]** Alice outputs the intersection I.

Fig. 4. Protocol realizing $\mathcal{F}_{\mathsf{sa\text{-}PSI}}$ (Fig. 1) using spatial hashing techniques.

origin $\vec{o}$ as an index to avoid collisions. This shifting approach eliminates the need for point functions in prior work [40]. On a minor note, we also improve on the domain of the ibFSS (size of *mini-universe*) from 3δ [40] to 2δ.

Finally, we can further relax the restrictions on Alice's input balls. Instead of requiring all the balls to be disjoint and have the same diameter, the only requirement we need for our protocol to work is that *every structured set* $\vec{S}_A^{(i)}$ *can be assigned to a distinct origin* $\vec{o}^{(i)}$ *such that* $\vec{S}_A^{(i)} \subseteq \mathsf{Univ}_{\vec{o}^{(i)}}$. In particular, these balls can have *different diameters* and *overlap* with each other. In fact, they don't even have to be ℓ_∞-balls with the same diameter in each dimension; our construction works for any structured set with an ibFSS scheme. We state the theorem below and defer its proof to the full version of the paper.

Theorem 2. *Given a two-party* $(\log(2\delta), d, t)$*-ibFSS scheme with pseudorandom keys for a family of sets* $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$ *where* $\mathcal{U} = 2\delta$ *and every set in* $\mathcal{S}$ *is disjoint union of at most* w *prefix sets, an oblivious key-value store scheme* $(\mathsf{Encode}, \mathsf{Decode})$*, and a Hamming-correlation robust hash* $\mathsf{H}: \{0,1\}^* \to \{0,1\}^{\lambda+\log|N_A|+\log w+\log|N_B|+d(1+\log\log(2\delta))}$*, the protocol in Fig. 4 realizes* $\mathcal{F}_{\mathsf{sa\text{-}PSI}}$ *(Fig. 1) in the* $\mathcal{F}_{\mathsf{OT}}$*-hybrid model secure in the presence of semi-honest adversaries.*

6 ibFSS for d-Dimensional ℓ_∞-Balls

In this section, we present a construction an ibFSS for d-dimensional balls with ℓ_∞-norm. Looking ahead, we will instantiate our structure-aware PSI protocols with it and analyze our computation and communication costs in Sect. 7. What we need for d-dimensional ℓ_∞-balls is a weak incremental Boolean Distributed Comparison Function (u, d, t)-ibDCF. It is an instance of (u, d, t)-ibFSS for the specific set family of d-dimensional ℓ_∞-balls. We formalize the definition in Sect. 6.1 and give a construction in Sect. 6.2.

6.1 ibDCF Definition

We consider (u, d, t)-ibFSS for the set family of d-dimensional intervals $\vec{S}_{\mathsf{INT}} = (S_1^{(\alpha_1,\beta_1)} \times \cdots \times S_d^{(\alpha_d,\beta_d)})$ represented by interval tuples $((\alpha_1, \beta_1), \ldots, (\alpha_d, \beta_d))$, where $\alpha_i \in \{0,1\}^u$ is a partition point and $\beta_i \in \{0,1\}$ is an indicator of left or right interval. Specifically, for every $i \in [d]$, the set is either a left interval $S_i^{(\alpha_i,0)} = \{x \mid x < \alpha_i,\ x \in \{0,1\}^u\}$ or a right interval $S_i^{(\alpha_i,1)} = \{x \mid x > \alpha_i,\ x \in \{0,1\}^u\}$. We define the syntax below and the security requirement follows from (u, d, t)-ibFSS (Definition 4).

Definition 5. *[(u, d, t)-ibDCF: Syntax] A (two-party) d-dimensional weak incremental Boolean distributed comparison function scheme (u, d, t)-ibDCF for the family of d-dimensional interval sets over the universe $(\{0,1\}^u)^d$ consists of a pair of algorithms* $(\mathsf{Share}, \mathsf{Eval})$ *with the following syntax:*

- $(k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, ((\alpha_1, \beta_1), \ldots, (\alpha_d, \beta_d))$*: The randomized share function takes as input the security parameter* κ *and the description of a* d*-dimensional interval, namely a set of partition points and bits (one for each dimension), where* $\alpha_i \in \{0,1\}^u$ *and* $\beta_i \in \{0,1\}$*, and outputs two key shares.*
- $y_{\mathsf{idx}} \leftarrow \mathsf{Eval}(\mathsf{idx}, k_{\mathsf{idx}}, \vec{x})$*: The deterministic evaluation function takes as input a party index* $\mathsf{idx} \in \{0,1\}$*, the corresponding key share* k_{idx}*, and input* $\vec{x} = (x_1, \ldots, x_d)$ *where each* $x_i \in \bigcup_{\ell=1}^{u} \{0,1\}^\ell$*. It outputs a string* $y_{\mathsf{idx}} \in \{0,1\}^t$*.*

Handling Two-Sided Intervals. While the above definition only considers *single-sided* intervals in each dimension, we can use it to handle *two-sided* intervals in a generic manner. Specifically, a two-dimensional $(u, 2, t)$-ibDCF for the set family $\vec{S}_{2\text{-INT}} = (S_1^{(L,0)} \times S_2^{(R,1)}) \in 2^{\mathcal{U}} \times 2^{\mathcal{U}}$ can be used as a single-dimensional ibDCF for a two-sided interval (L, R) if evaluating on $\vec{x} = (x, x)$ with the *same* input in both dimensions.

Similarly, a $2d$-dimensional $(u, 2d, t)$-ibDCF for the set family $\vec{S}_{2d\text{-INT}} = (S_1^{(L_1,0)} \times S_2^{(R_1,1)} \times \cdots \times S_{2d-1}^{(L_d,0)} \times S_{2d}^{(R_d,1)}) \in \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{(2d)}$ can be used as an ibDCF for a d-dimensional ℓ_∞-ball $(L_1, R_1) \times \cdots \times (L_d, R_d)$ if evaluating on $\vec{x} = (x_1, x_1, \ldots, x_d, x_d)$ with the *same* input along each of the d dimensions of the ball.

6.2 ibDCF Constructions

We first give a construction for a single-dimensional $(u, 1, 1)$-ibDCF for left-sided intervals in Fig. 6. Our construction builds upon the optimized Distributed Point Function (DPF) construction from [19] with the addition of an extra variable. For a left-sided interval $\{x \mid x < \alpha,\ x \in \{0,1\}^u\}$, we consider the DPF construction for the point function $f(x) = 1$ if $x = \alpha$ and $f(x) = 0$ otherwise. At a high level, in the GGM-tree based DPF construction [19], each node in the binary tree consists of a PRG seed and an additional output bit. They focus on the *critical path* along α and keep the invariant that the output bit is a secret share of 1 *on* the critical path and secret share of 0 *off* the critical path.

In our construction, we use their output bit (t) as an *indicator bit* of whether we are on or off the critical path, and add another bit (y) as the output bit. We leverage the indicator bit and keep the following invariants: (1) if we are on the critical path, then the output bit is a secret share of 1; (2) if we deviate to the right of the critical path (namely $\alpha^{(i)} = 0$), then the output bit remains a secret share of 1; (3) if we deviate to the left of the critical path (namely $\alpha^{(i)} = 1$), then the output bit becomes a secret share of 0; and (4) once we deviate from the critical path, the output remains a secret share of the same bit for the rest of the subtree (1 if deviating to the right and 0 if deviating to the left). The additional output bit y is highlighted in blue in the construction (Fig. 6).[2] We

[2] Note that similar ideas were presented in [7] in to construct DCF from iDPF, where there are also evaluation values at each node (v_i) in their construction (Algorithm 7).

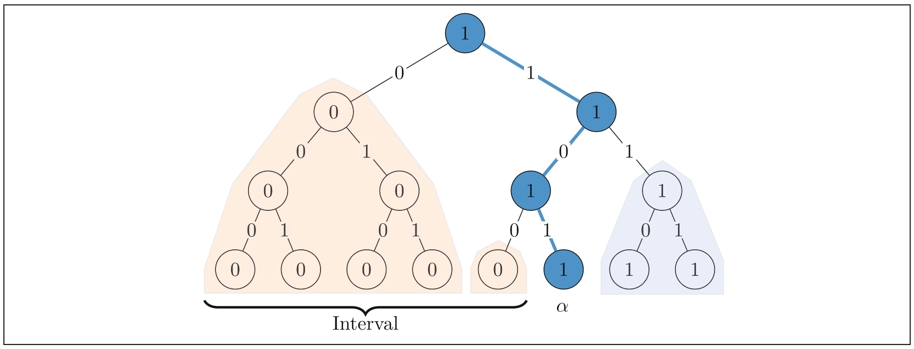


Fig. 5. Incremental DCF for a single-dimensional left-sided interval $\{x \mid x < \alpha,\ x \in \{0,1\}^u\}$. (Color figure online)

illustrate these invariants in Fig. 5. Specifically, for all the nodes highlighted in light orange, the parties obtain secret shares of 0; for all the nodes highlighted in blue and light blue, the parties obtain secret shares of 1.

Note that we present a construction for left-sided intervals. It is straightforward to construct an analogous scheme for right-sided intervals, and we omit the details here. We state the theorem below and defer its security proof to the full version of the paper.

Theorem 3. *Given a pseudorandom generator $G : \{0,1\}^\kappa \rightarrow \{0,1\}^{2(\kappa+2)}$, the construction* (Share, Eval) *in Fig. 6 is a secure $(u,1,1)$-ibDCF scheme (Definition 5) for the family of single-dimensional one-sided intervals, with key size $|k_0| = |k_1| = u \cdot (\kappa + 4) + \kappa$ bits and output size $|y_0| = |y_1| = 1$.*

Next we present our construction for a d-dimensional (u,d,d)-ibDCF scheme in Fig. 7, where we adapt the concat technique from prior work [39]. We state the theorem below and defer its proof to the full version of the paper.

Theorem 4. *Given a single-dimensional $(u,1,1)$-ibDCF scheme, the construction (Share*, Eval*) in Fig. 7 is a secure (u,d,d)-ibDCF scheme (Definition 5) for the family of d-dimensional intervals, with key size $|k_0| = |k_1| = d \cdot (u \cdot (\kappa+4)+\kappa)$ bits and output size $|y_0| = |y_1| = d$.*

7 Instantiation and Evaluation

We instantiate our sa-PSI protocols with d-dimensional balls with ℓ_∞-norm using the ibDCF presented in Sect. 6, and analyze the computation and communication

However, outputting these values doesn't immediately work for us because we need the critical path to output a secret share of 1 (instead of 0 in their construction). Another critical difference is that v_i in [7] remains unchanged when $x_i = 1$, whereas our construction generates fresh randomness for each node, resulting in fresh secret shares at each node and achieving slightly stronger security guarantees that are essential in our sa-PSI protocols.

```
$(u,1,1)$-ibDCF (Share, Eval):

Parameters: Let $G : \{0,1\}^\kappa \to \{0,1\}^{2(\kappa+2)}$ be a pseudorandom generator.

Share$(1^\kappa, (\alpha, \beta = 0))$:
  Let $\alpha = \overline{\alpha^{(1)}\alpha^{(2)} \ldots \alpha^{(u)}} \in \{0,1\}^u$ be the bit decomposition.
  Sample $s_0^{(0)} \leftarrow \{0,1\}^\kappa$ and $s_1^{(0)} \leftarrow \{0,1\}^\kappa$ uniformly at random.
  Assign $t_0^{(0)} = 0$ and $t_1^{(0)} = 1$.
  for $i = 1$ to $u$ do
    $s_0^L \| t_0^L \| y_0^L \;\|\; s_0^R \| t_0^R \| y_0^R \leftarrow G(s_0^{(i-1)})$ and $s_1^L \| t_1^L \| y_1^L \;\|\; s_1^R \| t_1^R \| y_1^R \leftarrow G(s_1^{(i-1)})$
    if $\alpha^{(i)} = 0$ then
      $s_{CW} = s_0^R \oplus s_1^R$
      $t_{CW}^L = t_0^L \oplus t_1^L \oplus 1$ and $t_{CW}^R = t_0^R \oplus t_1^R$
      $y_{CW}^L = y_0^L \oplus y_1^L$ and $y_{CW}^R = y_0^R \oplus y_1^R$
      $CW^{(i)} = s_{CW} \| t_{CW}^L \| t_{CW}^R \| y_{CW}^L \| y_{CW}^R$
      $s_b^{(i)} = s_b^L \oplus t_b^{(i-1)} \cdot s_{CW}$ for $b = 0,1$
      $t_b^{(i)} = t_b^L \oplus t_b^{(i-1)} \cdot t_{CW}^L$ for $b = 0,1$
    else
      $s_{CW} = s_0^L \oplus s_1^L$
      $t_{CW}^L = t_0^L \oplus t_1^L$ and $t_{CW}^R = t_0^R \oplus t_1^R \oplus 1$
      $y_{CW}^L = y_0^L \oplus y_1^L \oplus 1$ and $y_{CW}^R = y_0^R \oplus y_1^R$
      $CW^{(i)} = s_{CW} \| t_{CW}^L \| t_{CW}^R \| y_{CW}^L \| y_{CW}^R$
      $s_b^{(i)} = s_b^R \oplus t_b^{(i-1)} \cdot s_{CW}$ for $b = 0,1$
      $t_b^{(i)} = t_b^R \oplus t_b^{(i-1)} \cdot t_{CW}^R$ for $b = 0,1$
  Let $k_b = s_b^{(0)} \| CW^{(1)} \| CW^{(2)} \| \ldots \| CW^{(u)}$
  return $(k_0, k_1)$

Eval$(1^\kappa, b, k_b, x)$:
  Parse $k_b = s^{(0)} \| CW^{(1)} \| \ldots \| CW^{(u)}$ and $x = x^{(1)}, x^{(2)}, \ldots x^{(\ell)}$
  Assign $t^{(0)} = b$. Assign $y^{(0)} = b$.
  for $i = 1$ to $\ell$ do
    Parse $CW^{(i)} = s_{CW} \| t_{CW}^L \| t_{CW}^R \| y_{CW}^L \| y_{CW}^R$
    $\tau^i = G(s^{(i-1)}) \oplus (t^{(i-1)} \cdot [s_{CW} \| t_{CW}^L \| y_{CW}^L \| s_{CW} \| t_{CW}^R \| y_{CW}^R])$
    Parse $\tau^i = s^L \| t^L \| y^L \;\|\; s^R \| t^R \| y^R \in \{0,1\}^{2(\kappa+2)}$
    Update $y^L = y^L \oplus y^{(i-1)}$ and $y^R = y^R \oplus y^{(i-1)}$
    if $x_i = 0$ then $s^{(i)} = s^L, t^{(i)} = t^L, y^{(i)} = y^L$
    else $s^{(i)} = s^R, t^{(i)} = t^R, y^{(i)} = y^R$
  return $y^{(\ell)}$
```

Fig. 6. Construction of a single-dimensional Incremental Boolean Distributed Comparison Function $(u,1,1)$-ibDCF for left-sided intervals.

improvements over prior work [39,40]. Since our protocol is semi-honest secure, we mainly compare with the semi-honest work [39]. Note that [40] follows the same construction framework as [39] for the most part while focusing on achieving malicious security, except that [40] also introduces a new spatial hashing technique that improves upon the semi-honest construction. Hence we compare our work with both spatial hashing constructions.

```
$(u, d, d)$-ibDCF $(\mathsf{Share}^*, \mathsf{Eval}^*)$:
Parameters: Let $(\mathsf{Share}, \mathsf{Eval})$ be a single-dimensional $(u, 1, 1)$-ibDCF scheme.

$\mathsf{Share}^*(1^\kappa, ((\alpha_1, \beta_1), \ldots, (\alpha_d, \beta_d)))$:
    for $i = 1$ to $d$ do
        $(k_0^i, k_1^i) \leftarrow \mathsf{Share}(1^\kappa, (\alpha_i, \beta_i))$
    $k_0 = (k_0^1, \ldots, k_0^d)$ and $k_1 = (k_1^1, \ldots, k_1^d)$
    return $(k_0, k_1)$

$\mathsf{Eval}^*(b, k_b = (k_b^1, \ldots, k_b^d), \vec{x} = (x_1, \ldots, x_d))$:
    return $(\mathsf{Eval}(b, k_b^1, x_1) \parallel \ldots \parallel \mathsf{Eval}(b, k_b^d, x_d))$
```

Fig. 7. Construction of Incremental Boolean Distributed Comparison Function (u, d, d)-ibDCF for d-dimensional intervals with output length d.

7.1 Single Ball with Single Point

We start our comparison with the setting where Alice holds a single ball with ℓ_∞-norm as the distance metric in the d-dimensional universe $(\{0,1\}^u)^d$, and Bob holds a single point. The computation cost of the protocol can be broken down into three parts: (1) Alice generates all the FSS keys, (2) Bob evaluates all the FSS on his inputs and computes hash values, and (3) Alice identifies the set intersection from the hash values received from Bob. Our unit of computation cost is one PRG or hash operation.

The communication cost can be broken down into two parts: (1) oblivious transfer where Alice is the OT sender with FSS keys and Bob is the OT receiver with random choice bits, and (2) the hash values sent from Bob to Alice. The computation and communication costs are summarized in Table 1. Note that all the computation only involves efficient symmetric-key operations such as PRGs (which can be instantiated with AES) and hash functions (which can be instantiated with e.g., SHA256). There is also computation cost from OT on both parties, but we can leverage the efficient OT extension [46], and the computation cost is dominated by the FSS key generation and evaluation, so we omit this cost from the table.

Cells in green indicate that our protocol achieves better complexity, and those in grey indicate that our protocol requires higher overhead. Our main improvement over prior work is Alice's computation cost to identify the set intersection. In particular, prior work requires Alice to compute hash values (i.e., perform FSS evaluations) on every element in her set, which could potentially be exponentially large, namely $|S_A|$ is upper bounded by $2^{u \cdot d}$. In our work, by leveraging our new ibFSS construction for d-dimensional ℓ_∞ balls, Alice only needs to compute hash values for up to u^d prefixes. In case there is a match, she can perform an efficient search for Bob's element in time $u \cdot d \cdot \ell_{\mathsf{OT}}$. To enable Alice to perform such a search, we require Bob to compute hash values for all combinations of his prefixes, hence introducing a computation and communica-

Table 1. Summary of computation and communication costs for the setting where Alice holds a single ℓ_∞ ball in the d-dimensional universe $(\{0,1\}^u)^d$ and Bob holds a single point in the universe. $h_{\mathsf{out}} = \mathcal{O}(\lambda)$ is the output length of the final hash function. $\ell_{\mathsf{OT}} = \mathcal{O}(\kappa)$ is the number of OTs. $|S_A|$ is upper bounded by $\mathbf{2^{u \cdot d}}$.

Comp. & Comm. Costs			[39]	Ours
Comp.	Alice	FSS	$\mathcal{O}(u \cdot d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(u \cdot d \cdot \ell_{\mathsf{OT}})$
		Intersection	$\mathcal{O}(\min(\boldsymbol{\lvert S_A \rvert} \cdot u, 2^u) \cdot d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(\boldsymbol{u^d} + u \cdot d \cdot \ell_{\mathsf{OT}})$
	Bob's Eval		$\mathcal{O}(u \cdot d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(\boldsymbol{u^d} + u \cdot d \cdot \ell_{\mathsf{OT}})$
Comm. (bits)	OT		$\mathcal{O}(\kappa \cdot u \cdot d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(\kappa \cdot u \cdot d \cdot \ell_{\mathsf{OT}})$
	Bob's Hashes		$\mathcal{O}(h_{\mathsf{out}})$	$\mathcal{O}(\boldsymbol{u^d} \cdot h_{\mathsf{out}})$

tion overhead of u^d on Bob's side. Text in ***red*** highlights our difference from prior work that attributes to our new ibFSS technique.

7.2 Multiple (Overlapping) Balls with Multiple Points

Next, we consider the setting where Alice holds N_A balls with ℓ_∞-norm as the distance metric in the d-dimensional universe $(\{0,1\}^u)^d$, and Bob holds N_B points in the universe. The computation and communication costs can be broken down in a similar way as above, which is summarized in Table 2.

Table 2. Summary of computation and communication costs for the setting where Alice holds N_A number of ℓ_∞ balls in the d-dimensional universe $(\{0,1\}^u)^d$ and Bob holds N_B points in the universe. $h_{\mathsf{out}} = \mathcal{O}(\lambda)$ is the output length of the final hash function. $\ell_{\mathsf{OT}} = \mathcal{O}(\kappa)$ is the number of OTs. $|S_A|$ is upper bounded by $\mathbf{2^{u \cdot d}}$. Prior work [39,40] only allows Alice to hold ***disjoint*** balls while our protocol also supports ***overlapping*** balls.

Comp. & Comm. Costs			[39,40]	Ours
Comp.	Alice	FSS	$\mathcal{O}(N_A \cdot \boldsymbol{u^d} \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_A \cdot \boldsymbol{u \cdot d} \cdot \ell_{\mathsf{OT}})$
		Intersection	$\mathcal{O}(\boldsymbol{\lvert S_A \rvert} \cdot N_A \cdot \boldsymbol{u^d} \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_A \cdot \boldsymbol{u^d} + N_B \cdot \boldsymbol{u \cdot d} \cdot \ell_{\mathsf{OT}})$
	Bob's Eval		$\mathcal{O}(N_A \cdot N_B \cdot \boldsymbol{u^d} \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_A \cdot N_B \cdot (\boldsymbol{u^d} + \boldsymbol{u \cdot d} \cdot \ell_{\mathsf{OT}}))$
Comm. (bits)	OT		$\mathcal{O}(\kappa \cdot N_A \cdot \boldsymbol{u^d} \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(\kappa \cdot N_A \cdot \boldsymbol{u \cdot d} \cdot \ell_{\mathsf{OT}})$
	Bob's Hashes		$\mathcal{O}(N_B \cdot h_{\mathsf{out}})$	$\mathcal{O}(N_B \cdot \boldsymbol{N_A} \cdot \boldsymbol{u^d} \cdot h_{\mathsf{out}})$

Besides the improvement from ibFSS (which is highlighted in ***red***), we have several new contributions in this setting. The way that prior work deals with multiple balls is via a sum transformation to combine multiple FSS into a single one that achieves the *OR* functionality. Since their sum transformation only works for FSS that outputs a single bit, they have to use *strong FSS* for d-dimensional balls [18], incurring an overhead of u^d in both computation and communication for every ball and every FSS evaluation. By utilizing our new observation for the *OR* functionality in PSI, we no longer need the sum transformation, allowing for the usage of *weak FSS* that outputs multiple bits for each ball. This reduces the exponential overhead (u^d) down to linear in the dimension ($u \cdot d$). Moreover, the sum transformation in prior work only works for *disjoint* balls. We eliminate this restriction in our protocol and allow for overlapping balls as well. We highlight our difference from prior work in ***blue*** in the table.

7.3 Spatial Hashing

In this section, we analyze the multi-ball-multi-point protocols using the spatial hashing technique [39]. In particular, Alice holds N_A number ℓ_∞-balls of diameter δ in the d-dimensional universe $(\{0,1\}^u)^d$, and Bob holds N_B points. Note that prior work [39,40] additionally considers more restricted settings where the balls are not only disjoint, but are guaranteed to be far apart ($> 4\delta$ [39] or $> 8\delta$ [40] in distance), which we do not compare with. In fact, our protocol works for a more relaxed setting where Alice's balls do *not* have the same diameter, and they can even overlap. The computation and communication costs can be broken down in a similar way as above, which is summarized in Table 3. All three constructions require an OKVS [14,38,61], typically with linear overhead (and small constants) in the number of keys, so we ignore this overhead.

Table 3. Summary of computation and communication costs for spatial hashing. $h_{\mathsf{out}} = \mathcal{O}(\lambda)$ is the output length of the final hash function. $\ell_{\mathsf{OT}} = \mathcal{O}(\kappa)$ is the number of OTs. $|S_A|$ is upper bounded by $\mathbf{2}^{u \cdot d}$. Prior work [39,40] only allows Alice to hold ***disjoint*** balls with ***fixed diameter*** δ while our protocol also supports ***overlapping*** balls with ***different diameters***.

Comp. & Comm. Costs			[39]	[40]	Ours
Comp.	Alice	FSS	$\mathcal{O}(N_A \cdot (\mathbf{4}\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_A \cdot \boldsymbol{u} \cdot \boldsymbol{d} \cdot (\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_A \cdot \mathbf{log}\,\boldsymbol{\delta} \cdot \boldsymbol{d} \cdot \ell_{\mathsf{OT}})$
		Intersection	$\mathcal{O}(\lvert \boldsymbol{S_A} \rvert \cdot (\mathbf{2}\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(\lvert \boldsymbol{S_A} \rvert \cdot \boldsymbol{u} \cdot \boldsymbol{d} \cdot (\mathbf{2}\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_A \cdot (\mathbf{log}\,\boldsymbol{\delta})^d + N_B \cdot \mathbf{log}\,\boldsymbol{\delta} \cdot \boldsymbol{d} \cdot \ell_{\mathsf{OT}})$
	Bob's Eval		$\mathcal{O}(N_B \cdot (2\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_B \cdot \boldsymbol{u} \cdot \boldsymbol{d} \cdot (2\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(N_B \cdot 2^d \cdot ((\mathbf{log}\,\boldsymbol{\delta})^d + \mathbf{log}\,\boldsymbol{\delta} \cdot \boldsymbol{d} \cdot \ell_{\mathsf{OT}})$
Comm. (bits)	OT		$\mathcal{O}(\kappa \cdot N_A \cdot (\mathbf{4}\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$O(\kappa \cdot N_A \cdot \boldsymbol{u} \cdot \boldsymbol{d} \cdot (\mathbf{log}\,\boldsymbol{\delta})^d \cdot \ell_{\mathsf{OT}})$	$\mathcal{O}(\kappa \cdot N_A \cdot \mathbf{log}\,\boldsymbol{\delta} \cdot \boldsymbol{d} \cdot \ell_{\mathsf{OT}})$
	Bob's Hashes		$\mathcal{O}(N_B \cdot h_{\mathsf{out}})$	$\mathcal{O}(N_B \cdot h_{\mathsf{out}})$	$\mathcal{O}(N_B \cdot (\mathbf{2}\,\mathbf{log}\,\boldsymbol{\delta})^d \cdot h_{\mathsf{out}})$

In addition to the improvements from ibFSS (highlighted in ***red***) and those from the new OR observation (highlighted in ***blue***), we introduce new techniques to improve upon spatial hashing, which may be of independent interest. The spatial hashing technique is introduced in [39] and improved in [40]. In [39], they consider all the *active grids* that overlap with at least one ball. Each active grid may contain up to 2^d union of disjoint balls, so this problem is reduced to the multi-ball-multi-point setting. Therefore, they need to use the expensive sum transformation on strong FSS, inheriting the $(\log \delta)^d$ overhead and introducing

Table 4. Suppose Alice has $N_A = 300$ balls with fixed diameter $\delta = 32$ and Bob has a collection of $N_B = 10^6$ points in $d = \{2, 3, 5\}$ dimensional space over $\mathcal{U} = \{0, 1\}^u, u = 32$. Let computational security parameter $\kappa = 128$ and statistical security parameter $\lambda = 40$. We estimate the computation and communication cost of our new, spatial hashing technique and compare against the previous best construction [40]. Our unit of computation cost is one PRG or hash operation. K, M, B, T, Q stand for thousand, million, billion, trillion, quadrillion respectively in computation units. For example, 1K means 1000 AES/SHA256 calls.

Comp. & Comm. Costs			$d = 2$		$d = 3$		$d = 5$	
			[40]	Ours	[40]	Ours	[40]	Ours
Comp. (PRGs, SHA256)	Alice	FSS	86M	840K	516M	1.3M	13.8B	2.1M
		Intersection	352B	2.8B	135T	4.2B	1478Q	7.0B
	Bob's Eval		1.2T	11.3B	13.8T	34.6B	1468T	324B
Comm.	OT		1.3GB	12.5MB	7.7GB	18.8MB	205GB	31.3MB
	Bob's Hashes		4.77MB	477MB	4.77MB	4.77GB	4.77MB	477GB

a new 2^d overhead as the maximum number of balls per grid. Additionally, each ball may overlap with up to 2^d grids, hence the total number of active grids is upper bounded by $N_A \cdot 2^d$, introducing another 2^d overhead in FSS key generation. The follow-up work [40] improves the spatial hashing technique by mapping every ball into a unique grid, hence reducing the number of active grids. However, for each point in Bob's set, he needs to evaluate FSS for all the adjacent grids and perform the sum transformation, which still requires strong FSS and the $(\log \delta)^d$ overhead. Moreover, they apply a distributed point function (DPF) to reduce the domain from the universe to the grid, introducing another overhead of $u \cdot d$.

In our work, we also map every ball into a unique grid, but Bob does not need to perform the sum transformation on adjacent cells. Instead, we can leverage our new OR observation and achieve a $\log \delta \cdot d$ overhead similar to the multi-ball-multi-point protocol. Additionally, we eliminate the usage of DPF by shifting both the balls and points to their corresponding mini-universe, saving another factor of $u \cdot d$. Finally, we relax our restriction on Alice's balls in that they can overlap and have different diameters. The only remaining restriction is that every ball can be mapped to a distinct grid that fully contains this ball. We highlight our difference from prior work in ***brown*** in the table.

To illustrate our improvement, we estimate the concrete computational and communication costs for specific settings in Table 4 comparing our protocol with the previous best spatial hashing technique [40]. Alice's computation cost is improved by $125 - 211\text{M}\times$, and Bob's computation cost is improved by $106 - 4,531\times$. In term of communication, we achieve lower communication cost in OT $(104 - 6,550\times)$ while incurring higher communication overhead in Bob's hashes $(100 - 100\text{K}\times)$. The overhead can outweigh the OT savings in certain scenarios.

For instance, when $d = 5$, the total communication cost of [40] is 205GB while ours is 477GB. However, the total computation cost of [40] is 1478Q while ours is 331B, showing a 4.5 million times improvement.

Concretely, we can estimate for the ride sharing application setting where Alice (passengers) has $N_A = 300$ balls with fixed diameter $\delta = 32$ wants to be matched with Bob (drivers) $N_B = 300$ in a two-dimensional space over $\mathcal{U} = \{0,1\}^u$ for $u = 32$. In our protocol (respectively, in prior work [40]), the computation cost of Alice generating FSS is 840K (86M), Alice computing intersection is 848K (352B), Bob evaluating FSS is 3.3M (344M); the communication cost of OT is 12.8MB (1.28GB), Bob's hashes is 146KB (1.46KB). Our protocol outperforms prior work by orders of magnitudes in both computation and communication. .

8 Extending Functionality

In this section, we present a construction for sa-PSI where Bob learns the output (Sect. 8.1). With a slight modification, we construct protocols that allow Alice (or Bob) to learn structure-aware PSI cardinality or PSI with associated sum (Sect. 8.2). One difference from prior constructions is that we only allow Alice's structured set to be a union of *disjoint* sets.

8.1 sa-PSI with Bob Learning Output

Construction Overview. Following the syntax from spatial hashing, both parties prepare their inputs. In Step 1, Alice assigns each of her structures to a distinct origin and prepares a set X of all prefix values along with the origin as an identifier. In Step 2, Bob computes a set Y for each input $\vec{y}$, which consists of all prefixes associated with every origin that contains the input. If $\vec{y} \in S_A \cap S_B$, then there is exactly one matching value (corresponding to Alice's prefix and origin), namely $|X \cap Y| = 1$, which must be accounted for while computing PSI cardinality between X and Y. If $\vec{y} \notin S_A \cap S_B$, then there is no matching value, hence $|X \cap Y| = 0$. Alice and Bob can use a $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$ ideal functionality, for each set Y, for Bob to learn $|X \cap Y|$. If Bob learns that the cardinality is 1 then he includes the input $\vec{y}$ in intersections, if cardinality is 0 then he does not include it in the intersection. We define the ideal functionality for sa-PSI where Bob learns the output in Fig. 8. Note that a difference from Fig. 1 is that Alice's structured sets are *disjoint*. We present our protocol in Fig. 9, state the theorem below and defer its security proof to the full version of the paper.

Theorem 5. *The protocol in Fig. 9 securely realizes $\mathcal{F}_{\mathsf{sa\text{-}PSI}}$ (Fig. 8) in the $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$-hybrid model in the presence of semi-honest adversaries.*

Parameters: A family of sets $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$ where $\mathcal{U} = \{0,1\}^u$. Number of Alice's structured sets N_A and size of Bob's set N_B.

Functionality:

1. Receive input $S_A = \dot{\bigcup}_{i \in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$ and S_A is a *disjoint* union of structured sets (or a concise representation of S_A) from Alice.
2. Receive input $S_B \subseteq \mathcal{U}^d$ of size N_B from Bob.
3. [**output**] Send $S_A \cap S_B$ to Bob.

Fig. 8. Ideal functionality $\mathcal{F}_{\mathsf{sa\text{-}PSI}}$ for structure-aware PSI, where Bob learns the output.

Parameters:

– private set intersection cardinality ideal functionality $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$

Inputs:

– Alice has N_A structured sets $S_A = \dot{\bigcup}_{i \in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$ and the structured sets in S_A are *disjoint*.

– Bob has an unstructured set $S_B \subseteq \mathcal{U}^d$ of size N_B.

Protocol:

```
1. Alice initializes $X := \emptyset$ and does the following:
   (a) Identify a distinct origin $\vec{o}^{(i)}$ for each $i \in [N_A]$ such that $\vec{S}_A^{(i)} \subseteq \mathsf{Univ}_{\vec{o}^{(i)}}$.
   (b) For each $i \in [N_A]$:
       i. Write $\vec{S}_A^{(i)}.\mathsf{Shift}(\vec{o}^{(i)}) = \dot{\bigcup}_{j \in [w']} \mathsf{PreS}_{\vec{x^j}}$ as disjoint union of $w'$ prefix sets ($w$ is the upper bound for $w'$).
       ii. Update set $X := X \cup \{(\overrightarrow{x^j} \| \vec{o}^{(i)}) \mid j \in [w']\}$.
   (c) Pad $X$ with dummy random strings such that $|X| = N_A \cdot w$.
2. Bob initializes $I := \emptyset$ and does the following:
   (a) For each element $\vec{y} \in S_B$, and for each origin $\vec{z}$ where $\vec{y} \in \mathsf{Univ}_{\vec{z}}$:
       i. Write $\vec{y}.\mathsf{Shift}(\vec{z}) = (y_1, \ldots, y_d)$, where $y_j \in [0, 2\delta)$ for all $j \in [d]$.
       ii. Compute
           $$Y := \left\{ (\vec{y'} \| \vec{z}) \middle| \begin{array}{l} \ell_1 \in [\log(2\delta)], \ldots, \ell_d \in [\log(2\delta)] \\ \vec{y'} := ((y_1)_{[1:\ell_1]}, \ldots, (y_d)_{[1:\ell_d]}) \end{array} \right\}$$
   (b) Bob sends set $Y$ and Alice sends $X$ to ideal functionality $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$.
   (c) Bob receives output $\mathsf{cnt}$ from $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$. Update set $I = I \cup \{\vec{y}\}$ if $\mathsf{cnt} = 1$.
3. [output] Bob outputs the intersection $I$.
```

Fig. 9. Protocol realizing $\mathcal{F}_{\mathsf{sa\text{-}PSI}}$ (Fig. 8), where Bob learns the output $S_A \cap S_B$.

Cost Analysis. For the set family $\mathcal{S}$ as disjoint union of N_A balls with diameter $\leq \delta$ in ℓ_∞ norm, Alice and Bob run N_B instances of $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$, between sets of size $n_1 = N_A \cdot w = N_A \cdot (\log(2\delta))^d$ and $n_2 = (2\log(2\delta))^d$. If we instantiate $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$ with a semi-honest construction with linear computation and communication in set sizes [22,37,42,45,49,57,64], then we achieve $\mathsf{sa\text{-}PSI}$ for Bob to learn output with communication and computation costs $O(N_B \cdot (n_1 + n_2))$, which is $\ll |S_A|$.

Parameters: a family of sets $\mathcal{S} \subseteq \underbrace{2^{\mathcal{U}} \times \cdots \times 2^{\mathcal{U}}}_{d}$ where $\mathcal{U} = \{0,1\}^u$. Number of Alice's structured sets N_A and size of Bob's set N_B.

Functionality:
1. Receive input $S_A = \dot{\bigcup}_{i \in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$ and S_A is a *disjoint* union of structured sets (or a concise representation of S_A) from Alice.
2. Receive input $S_B \subseteq \mathcal{U}^d$ of size N_B from Bob.
3. [**output**] Send output $|S_A \cap S_B|$ to Alice (or Bob).

Fig. 10. Ideal functionality $\mathcal{F}_{\textsf{sa-PSI-CA}}$ for structure-aware PSI cardinality.

8.2 Structure-Aware PSI Cardinality/Sum

Construction Overview. The above principle can be extended to allow Alice (or Bob) to learn structure-aware PSI cardinality $|S_A \cap S_B|$, except that we accumulate over all the Y sets computed for each of Bob's input. It could be the case that two different inputs have overlapping prefixes within the same origin (for example, points which are "close" to each other). In our protocol, Bob computes a multi-set with an associated $\textsf{cnt}$, indicating the number of inputs that any given prefix is associated with. After this, Alice and Bob can use $\mathcal{F}_{\textsf{PSI-Sum}}$, to learn the cardinality. We define the ideal functionality in Fig. 10 and present our construction in Fig. 11. We state the theorem below and defer its proof to the full version of the paper.

Theorem 6. *The protocol in Fig. 11 securely realizes $\mathcal{F}_{\textsf{sa-PSI-CA}}$ (Fig. 10) in the $\mathcal{F}_{\textsf{PSI-Sum}}$-hybrid model, secure in the presence of semi-honest adversaries.*

Cost Analysis. For the set family $\mathcal{S}$ as disjoint union of N_A balls of diameter $\leq \delta$ in ℓ_∞ norm, Alice and Bob run an instance of $\mathcal{F}_{\textsf{PSI-Sum}}$ between sets of size $n_1 = N_A \cdot w = N_A \cdot (\log(2\delta))^d$ and $n_2 = |S_B| \cdot (2 \cdot \log(2\delta))^d$. If we realize $\mathcal{F}_{\textsf{PSI-Sum}}$ with a semi-honest construction with linear computation and communication in set sizes [22,42,49,57,64], then we achieve $\textsf{sa-PSI-CA}$ with communication and computation costs of $O(n_1 + n_2)$, which is $\ll |S_A|$. Note that we require the protocol to only reveal the associated sum, while hiding the intersection cardinality.

Structure-Aware PSI-Sum. The construction in Fig. 11 can be extended to realize structure-aware PSI with associated sum, when Bob's inputs have associated values. The only change is that the associated value $\textsf{cnt}$, for any prefix $\vec{y'} \| \vec{z}$ in origin $\vec{z}$ in set Y_v, is the sum of Bob's associated values for all inputs that share the prefix $\vec{y'}$.

Notation: We denote a set of elements as X and a set where every element has an associated value as $X_v = \{(x, v) \mid x \in X, \mathsf{val}(x) = v\}$.

Parameters:

- private set intersection with associated sum ideal functionality $\mathcal{F}_{\mathsf{PSI\text{-}Sum}}$

Inputs:

- Alice has N_A structured sets $S_A = \dot{\bigcup}_{i \in N_A} \vec{S}_A^{(i)}$, where $\vec{S}_A^{(i)} \in \mathcal{S}$ and sets in S_A are non-overlapping.
- Bob has an unstructured set $S_B \subseteq \mathcal{U}^d$ of size N_B.

Protocol:

1. Alice initializes $X := \emptyset$ and does the following:
   (a) Identify a distinct origin $\vec{o}^{(i)}$ for each $i \in [N_A]$ such that $\vec{S}_A^{(i)} \subseteq \mathsf{Univ}_{\vec{o}^{(i)}}$.
   (b) For each $i \in [N_A]$:
      i. Write $\vec{S}_A^{(i)}.\mathsf{Shift}(\vec{o}^{(i)}) = \dot{\bigcup}_{j \in [w']} \mathsf{PreS}_{\vec{x^j}}$ as disjoint union of w' prefix sets (w is the upper bound for w').
      ii. Update set $X := X \cup \{(\overrightarrow{x^j} \| \vec{o}^{(i)}) \mid j \in [w']\}$.
   (c) Pad X with dummy random strings such that $|X| = N_A \cdot w$.
2. Bob initializes a set with associated value $Y_v := \emptyset$ and does the following:
   (a) For each element $\vec{y} \in S_B$, and for each origin $\vec{z}$ where $\vec{y} \in \mathsf{Univ}_{\vec{z}}$:
      i. Write $\vec{y}.\mathsf{Shift}(\vec{z}) = (y_1, \ldots, y_d)$, where $y_j \in [0, 2\delta)$ for all $j \in [d]$.
      ii. Compute
      $$Y' = \left\{ (\vec{y'} \| \vec{z}) \,\middle|\, \begin{array}{l} \ell_1 \in [\log(2\delta)], \ldots, \ell_d \in [\log(2\delta)] \\ \vec{y'} := ((y_1)_{[1:\ell_1]}, \ldots, (y_d)_{[1:\ell_d]}) \end{array} \right\}$$
      iii. For each $(y' \| \vec{z}) \in Y'$: if $(y' \| \vec{z}) \notin Y$ then $Y_v = Y_v \cup \{(y' \| \vec{z}, 1)\}$, else for $(y' \| \vec{z}, \mathsf{cnt}) \in Y_v$, update $\mathsf{cnt} = \mathsf{cnt} + 1$.
   (b) Pad Y_v with dummy random strings and values such that $|Y_v| = N_B \cdot (2 \cdot \log(2\delta))^d$.
3. Alice sends X and Bob sends Y_v to $\mathcal{F}_{\mathsf{PSI\text{-}Sum}}$.
   - **Alice learns output**: functionality $\mathcal{F}_{\mathsf{PSI\text{-}Sum}}$ returns output to Alice.
   - **Bob learns output**: functionality $\mathcal{F}_{\mathsf{PSI\text{-}Sum}}$ returns output to Bob.

Fig. 11. Protocol realizing $\mathcal{F}_{\mathsf{sa\text{-}PSI\text{-}CA}}$ (Fig. 10) in the $\mathcal{F}_{\mathsf{PSI\text{-}Sum}}$-hybrid model.

Acknowledgments. This project is supported in part by the NSF CNS Award 2247352, Brown DSI Seed Grant, Meta Research Award, Google Research Scholar Award, and Amazon Research Award.

References

1. Password Monitor: Safeguarding passwords in Microsoft Edge. https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/
2. Password Monitoring – Apple Platform Security. https://support.apple.com/en-al/guide/security/sec78e79fc3b/web

3. Privacy-Preserving Contact Tracing. https://covid19.apple.com/contacttracing
4. Private Intersection-Sum Protocols with Applications to Attributing Aggregate Ad Conversions. https://research.google/pubs/pub51026/
5. Protect your accounts from data breaches with Password Checkup. https://security.googleblog.com/2019/02/protect-your-accounts-from-data.html
6. Technology preview: Private contact discovery for Signal. https://signal.org/blog/private-contact-discovery/
7. Agarwal, A., Peceny, S., Raykova, M., Schoppmann, P., Seth, K.: Communication efficient secure logistic regression. Cryptology ePrint Archive, Report 2022/866 (2022). https://eprint.iacr.org/2022/866
8. Alamati, N., Branco, P., Döttling, N., Garg, S., Hajiabadi, M., Pu, S.: Laconic private set intersection and applications. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 94–125. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90456-2_4
9. Ali, A.,et al.: Communication-computation trade-offs in PIR. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021, pp. 1811–1828. USENIX Association (2021)
10. Aranha, D.F., Lin, C., Orlandi, C., Simkin, M.: Laconic private set-intersection from pairings. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 111–124. ACM Press (2022)
11. Ateniese, G., De Cristofaro, E., Tsudik, G.: (If) size matters: size-hiding private set intersection. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 156–173. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_10
12. Ateniese, G., Kirsch, J., Blanton, M.: Secret handshakes with dynamic and fuzzy matching. In: NDSS, vol. 7, pp. 43–54 (2007)
13. Berke, A., Bakker, M., Vepakomma, P., Raskar, R., Larson, K., Pentland, A.: Assessing disease exposure risk with location histories and protecting privacy: a cryptographic approach in response to a global pandemic. CoRR, abs/2003.14412 (2020)
14. 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.) 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, 9–11 August 2023. USENIX Association (2023)
15. Boneh, D., Boyle, E., Corrigan-Gibbs, H., Gilboa, N., Ishai, Y.: Lightweight techniques for private heavy hitters. In: 2021 IEEE Symposium on Security and Privacy, pp. 762–776. IEEE Computer Society Press (2021)
16. Boyle, E., et al.: Function secret sharing for mixed-mode and fixed-point secure computation. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 871–900. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_30
17. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_12
18. 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, pp. 1292–1303. ACM Press (2016)
19. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: improvements and extensions. Cryptology ePrint Archive, Report 2018/707 (2018). https://eprint.iacr.org/2018/707

20. Boyle, E., Gilboa, N., Ishai, Y., Kolobov, V.I.: Programmable distributed point functions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 121–151. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-15985-5_5
21. Chakraborti, A., Fanti, G., Reiter, M.K.: Distance-aware private set intersection (2021)
22. Chandran, N., Gupta, D., Shah, A.: Circuit-PSI with linear complexity via relaxed batch OPPRF. PoPETs **2022**(1), 353–372 (2022)
23. Chase, M., Miao, P.: Private set intersection in the internet setting from lightweight oblivious PRF. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 34–63. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_2
24. 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, pp. 1223–1237. ACM Press (2018)
25. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 1243–1255. ACM Press (2017)
26. Chongchitmate, W., Ishai, Y., Lu, S., Ostrovsky, R.: PSI from ring-OLE. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 531–545. ACM Press (2022)
27. Cong, K., et al.: Labeled PSI from homomorphic encryption with reduced computation and communication. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021, pp. 1135–1150. ACM Press (2021)
28. Couteau, G., Rindal, P., Raghuraman, S.: Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 502–534. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_17
29. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Secure efficient multiparty computing of multivariate polynomials and applications. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 130–146. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21554-4_8
30. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14577-3_13
31. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 789–800. ACM Press (2013)
32. Döttling, N., Kolonelos, D., Lai, R.W., Lin, C., Malavolta, G., Rahimi, A.: Efficient laconic cryptography from learning with errors. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 417–446. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-30620-4_14
33. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
34. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_1
35. Gao, J., Wong, T., Selim, B., Wang, C.: VOMA: a privacy-preserving matching mechanism design for community ride-sharing. IEEE Trans. Intell. Transp. Syst. **23**(12), 23963–23975 (2022)

36. Garg, S., Hajiabadi, M., Miao, P., Murphy, A.: Laconic branching programs from the Diffie-Hellman assumption. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14603, pp. 323–355. Springer, Heidelberg (2024). https://doi.org/10.1007/978-3-031-57725-3_11
37. Garimella, G., Mohassel, P., Rosulek, M., Sadeghian, S., Singh, J.: Private set operations from oblivious switching. In: Garay, J.A. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 591–617. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75248-4_21
38. 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.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 395–425. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84245-1_14
39. Garimella, G., Rosulek, M., Singh, J.: Structure-aware private set intersection, with applications to fuzzy matching. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 323–352. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-15802-5_12
40. Garimella, G., Rosulek, M., Singh, J.: Malicious secure, structure-aware private set intersection. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 577–610. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-38557-5_19
41. Ghosh, S., Simkin, M.: The communication complexity of threshold private set intersection. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 3–29. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_1
42. Han, K., Moon, D., Son, Y.: Improved circuit-based PSI via equality preserving compression. Cryptology ePrint Archive, Report 2021/1440 (2021). https://eprint.iacr.org/2021/1440
43. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS 2012. The Internet Society (2012)
44. Huberman, B.A., Franklin, M., Hogg, T.: Enhancing privacy and trust in electronic communities. In: Feldman, S.I., Wellman, M.P. (eds.) Proceedings of the First ACM Conference on Electronic Commerce (EC 1999), Denver, CO, USA, 3–5 November 1999, pp. 78–86. ACM (1999)
45. Ion, M., et al.: On deploying secure computing: private intersection-sum-with-cardinality. In: IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, 7–11 September 2020, pp. 370–389. IEEE (2020)
46. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
47. Jarecki, S., Liu, X.: Fast secure computation of set intersection. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 418–435. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_26
48. Kales, D., Rechberger, C., Schneider, T., Senker, M., Weinert, C.: Mobile private contact discovery at scale. In: Heninger, N., Traynor, P. (eds.) USENIX Security 2019, pp. 1447–1464. USENIX Association (2019)
49. Karakoç, F., Küpçü, A.: Linear complexity private set intersection for secure two-party protocols. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 2020. LNCS, vol. 12579, pp. 409–429. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_20

50. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_15
51. 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, pp. 818–829. ACM Press (2016)
52. 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.) CRYPTO 2020. LNCS, vol. 12172, pp. 3–33. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_1
53. Orrù, M., Orsini, E., Scholl, P.: Actively secure 1-out-of-N OT extension with application to private set intersection. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 381–396. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_22
54. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: SpOT-light: lightweight private set intersection from sparse OT extension. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 401–431. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_13
55. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: PSI from PaXoS: fast, malicious private set intersection. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 739–767. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_25
56. Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: private set intersection using permutation-based hashing. In: Jung, J., Holz, T. (eds.) USENIX Security 2015, pp. 515–530. USENIX Association (2015)
57. Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based PSI with linear communication. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 122–153. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_5
58. Pinkas, B., Schneider, T., Weinert, C., Wieder, U.: Efficient circuit-based PSI via cuckoo hashing. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 125–157. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_5
59. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: Fu, K., Jung, J. (eds.) USENIX Security 2014, pp. 797–812. USENIX Association (2014)
60. Rabin, M.O.: How to exchange secrets with oblivious transfer. Cryptology ePrint Archive (2005)
61. 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, pp. 2505–2517. ACM Press (2022)
62. Ramezanian, S., Akman, G., Damir, M.T., Niemi, V.: Lightweight privacy-preserving ride-sharing protocols for autonomous cars. In: Brücher, B., Krauß, C., Fritz, M., Hof, H.-J., Wasenmüller, O. (eds.) Computer Science in Cars Symposium, CSCS 2022, Ingolstadt, Germany, 8 December 2022, pp. 11:1–11:11. ACM (2022)
63. Rindal, P., Rosulek, M.: Improved private set intersection against malicious adversaries. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 235–259. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_9

64. Rindal, P., Schoppmann, P.: VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 901–930. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_31
65. Trieu, N., Shehata, K., Saxena, P., Shokri, R., Song, D.: Epione: lightweight contact tracing with strong privacy. IEEE Data Eng. Bull. **43**(2), 95–107 (2020)
66. 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, pp. 911–928. USENIX Association (2021)
67. van Baarsen, A., Pu, S.: Fuzzy private set intersection with large hyperballs. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part V. LNCS, vol. 14655, pp. 340–369. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-58740-5_12
68. Wang, X.A., Xhafa, F., Luo, X., Zhang, S., Ding, Y.: A privacy-preserving fuzzy interest matching protocol for friends finding in social networks. Soft Comput. **22**(8), 2517–2526 (2018)
69. Wen, Y., Gong, Z.: Private mutual authentications with fuzzy matching. Int. J. High Perform. Syst. Archit. **5**(1), 3–12 (2014)
70. Zhang, E., Chang, J., Li, Yu.: Efficient threshold private set intersection. IEEE Access **9**, 6560–6570 (2021)