Structure-Aware Private Set Intersection, With Applications to Fuzzy Matching[*]

Gayathri Garimella Mike Rosulek Jaspal Singh

Oregon State University

Abstract

In two-party private set intersection (PSI), Alice holds a set X, Bob holds a set Y, and they learn (only) the contents of $X \cap Y$. We introduce **structure-aware PSI** protocols, which take advantage of situations where Alice's set X is publicly known to have a certain structure. The goal of structure-aware PSI is to have communication that scales with the *description size* of Alice's set, rather its *cardinality.*

We introduce a new generic paradigm for structure-aware PSI based on function secret-sharing (FSS). In short, if there exists compact FSS for a class of structured sets, then there exists a semi-honest PSI protocol that supports this class of input sets, with communication cost proportional only to the FSS share size. Several prior protocols for efficient (plain) PSI can be viewed as special cases of our new paradigm, with an implicit FSS for unstructured sets.

Our PSI protocol can be instantiated from a significantly weaker flavor of FSS, which has not been previously studied. We develop several improved FSS techniques that take advantage of these relaxed requirements, and which are in some cases exponentially better than existing FSS.

Finally, we explore in depth a natural application of structure-aware PSI. If Alice's set X is the union of many radius-δ balls in some metric space, then an intersection between X and Y corresponds to **fuzzy PSI**, in which the parties learn which of their points are within distance δ. In structure-aware PSI, the communication cost scales with the number of balls in Alice's set, rather than their total volume. Our techniques lead to efficient fuzzy PSI for ℓ_∞ and ℓ_1 metrics (and approximations of ℓ_2 metric) in high dimensions. We implemented this fuzzy PSI protocol for 2-dimensional ℓ_∞ metrics. For reasonable input sizes, our protocol requires 45–60% less time and 85% less communication than competing approaches that simply reduce the problem to plain PSI.

1 Introduction

Private Set Intersection (PSI) allows Alice with input A and Bob with input B to learn *only* the intersection $A \cap B$ of their sets and reveals no additional information. In this paper, we introduce **structure-aware private set intersection (PSI)** where one of the parties, say Alice, has an input set A with some publicly known structure and Bob's input B is a set of unstructured points. They want to jointly compute the set of Bob's inputs that lie within Alice's input structure. An immediate approach for this problem is to enumerate or expand Alice's structured input into an unstructured set A^* and employ existing efficient plain PSI protocols. However, this is impractical

[*]Authors partially supported by NSF award 2150726

because the cardinality of the expanded set can be prohibitively large, and PSI protocols have communication cost that scale with the cardinality of the input sets. We ask the question -

> *Can we make PSI protocols more efficient when there is publicly known structure in the parties' input sets?*

In particular, is there a PSI protocol whose cost scales with the *description size* of the structured input A rather than the *cardinality* of A^*? We answer the above question affirmatively with a new framework for structure-aware PSI based on function secret sharing (FSS). We relax the constraints of standard FSS to introduce and study the notion of weak FSS. As our main result we show that for any structured input A, our framework reduces PSI to the task of designing efficient and succinct weak FSS for Alice's structured set A.

We design several novel and efficient FSS schemes for the family of sets of union of balls assuming different levels of structure. A standard boolean FSS scheme for a collection of sets $\mathcal{S}$ can compute "succinct" shares k_0, k_1 of any set $S \in \mathcal{S}$ and for any input x, the individual shares k_0 or k_1 can be used to evaluate secret shares of the membership test of input $x \in S$. Formally, FSS schemes consist of algorithm Share which computes the secret-shares k_0, k_1 and algorithm Eval that can be used to compute shares of the membership test $((\mathsf{Eval}(k_0, x) \oplus \mathsf{Eval}(k_1, x)) = (x \in S)$.[1] PRG based FSS constructions are known for many interesting family of sets with membership functions that can be expressed as point functions, comparison functions, multi-dimensional intervals or decision trees [BGI15, BGI16, BCG+21].

Our work introduces weak boolean **FSS** that allows for "false positives" in the membership evaluation—*i.e.*, when $x \notin S$, the xor of the shares evaluate $((\mathsf{Eval}(k_0, x) \oplus \mathsf{Eval}(k_1, x))$ to true with at least some bounded probability p. We also make another useful relaxation (the details are in Section 3.2) which together enable more efficient constructions compared to standard boolean FSS.

Finally, we instantiate our PSI framework with an FSS for unions of balls to obtain **fuzzy private set intersection (PSI)**. Here, Alice has a set of points A and Bob has a set of points B and they would like to learn the pairs $(a, b) \in A \times B$ satisfying $d(a, b) \leq \delta$, where d is a distance metric and δ is a public threshold. They should learn nothing about A and B, beyond this set of close pairs. We can compute this using our structure-aware PSI protocol by assigning Alice's structured input A^* as the union of many **δ-balls** (a δ-ball of radius δ centered at a is $\{b \mid d(a, b) \leq \delta\}$) and Bob's input is his unstructured input set B.

1.1 Our Contributions

New weak FSS constructions. We introduce the notion of **weak FSS** (parameterized by p and k), which for a family of sets consists of algorithms Share and Eval defined as:

- If $(k_0, k_1) \leftarrow \mathsf{Share}(A)$, for a set A in the family, then each k_i individually looks pseudorandom.
- If $x \in A$ then $\Pr[\mathsf{Eval}(k_0, x) \oplus \mathsf{Eval}(k_1, x) = 0^k] = 1$, where the probability is over the sampling of $(k_0, k_1) \leftarrow \mathsf{Share}(A)$.
- If $x \notin A$ then $\Pr[\mathsf{Eval}(k_0, x) \oplus \mathsf{Eval}(k_1, x) \neq 0^k] \geq p$, where the probability is over the sampling of $(k_0, k_1) \leftarrow \mathsf{Share}(A)$.

We also propose several new techniques for (weak) FSS constructions offering two significant advantages. One, our weak FSS has significantly smaller share sizes than standard FSS. Second, we achieve significantly more efficient share-evaluation cost than existing FSS.

Consider our motivating example of fuzzy PSI where the structured set is the union of balls. Existing FSS techniques can be used for this kind of structure — however, the result is an FSS

[1] The original formulation of FSS by Boyle et al. [BGI15] is in terms of functions instead of sets, however in this work we are only interested in boolean set membership functions - hence we reframed the FSS definition

where evaluating the share on a single point requires time linear in the number of balls. This cost leads to a fuzzy PSI protocol with whose computation cost scales with the *product* of the two sets' cardinalities. Our new techniques provide FSS for a union of balls, where the share-evaluation cost is independent of the number of balls.

We specifically focus on the case where the structured set is a union of n balls of radius δ under the ℓ_∞ norm in d-dimensional space $\{0, 1, \ldots, 2^u - 1\}^d$. We use the concat technique (described in Subsection 4.2) to develop weak FSS for a d dimensional ℓ_∞ ball (which is a cross product of d intervals). This technique essentially combines the outputs of FSS for d 1-dimension intervals that make the input ball - making the output length $k = d$. We further employ the $\mathsf{spatial\text{-}hashing}$ technique (from Subsection 4.3) to design weak bFSS for union of balls. Using this technique, at a high level, we divide the input space into contiguous grid cells; construct FSS keys for each grid cell that intersect with any input ball; and then pack these FSS keys into an oblivious key value data structure. In our construction we ensure that for point in a grid cell not intersecting with any input ball, the FSS outputs is a random string - setting the false positive probability $p = 1 - 1/2^k$. Hence both relaxations in our bFSS definitions are key when designing bFSS for union of l_∞ balls.

An important theme in this work is that *more structure leads to more efficient FSS/PSI.* We consider three increasing levels of structure for such sets, which result in significantly better dependence on the dimension:

- The least amount of structure is when the balls are disjoint. Existing techniques give an FSS for this case whose share size depends on the dimension as $O(\min\{u, \delta\}^d)$. In our new FSS for disjoint balls, the dependence on the dimension is $O((4 \log \delta)^d)$.
- If the balls are spaced far apart — *i.e.*, no two centers are closer than distance 4δ — then we can achieve FSS whose dependence on the dimension is $O(2^d)$.
- Finally, if the balls are *globally-axis disjoint* — meaning that the projection of the balls onto every axis is disjoint — then we can achieve FSS whose dependence on the dimension is only $O(d)$.

Our techniques can also apply to the ℓ_1 metric (and to close approximations of the ℓ_2 metric), although the dependence on the dimension changes for the three different cases above.

Structure-aware PSI. We initiate the study of **structure-aware PSI,** which exploits known structure in Alice's input set to achieve better efficiency than a conventional, "general purpose PSI" protocol. Our primary measure of efficiency is the *communication cost* of PSI protocols. Conventional PSI protocols have communication cost $O((|A| + |B|)\kappa)$, where A and B are the input sets. A structure-aware PSI protocol should have communication cost $O((d + |B|)\kappa)$, where $d \ll |A|$. Ideally d is the *description size*, not cardinality, of A. Various different structures for A can be considered — not just the union of radius-δ balls, as in our motivating application for fuzzy PSI.

General protocol paradgim. Most efficient PSI protocols use the classic oblivious PRF (OPRF) paradigm of Freedman *et al.* [FIPR05]. In an OPRF, Alice has an input set A, and Bob has no input. Bob learns a PRF seed k while Alice learns $\{\mathsf{PRF}_k(a) \mid a \in A\}$. The parties can obtain a PSI protocol by having Bob send $\{\mathsf{PRF}_k(b) \mid b \in B\}$ to Alice. Our structure-aware PSI is an instance of the OPRF-to-PSI paradigm, but with a **structure-aware OPRF** that takes advantage of publicly known structure in the OPRF receiver's set A. We construct semi-honest structure-aware PSI/OPRF from any weak FSS construction for the receiver's set A. Our key theorem can be summarized as follows:

Main Theorem (informal). *If there is a weak FSS for a family $\mathcal{S}$ of sets, with shares of length*

σ, then there is a semi-honest structure-aware PSI protocol (where Alice's input $A \in \mathcal{S}$) with communication $O((\sigma + |B|)\kappa)$.

In particular, the reliance on Alice's set is reduced from $O(|A|\kappa)$ in a general-purpose PSI protocol to $O(\sigma\kappa)$. The problem of constructing structure-aware PSI therefore reduces to the simpler problem of constructing a weak FSS for the supported structure, with share size smaller than the set cardinality.

Our structure-aware PSI protocol is inspired by the IKNP OT extension protocol [IKNP03]. As such, it uses only cheap symmetric-key operations apart from a small number of base-OTs and FSS operations. Hence, if the underlying FSS is also based on symmetric-key operations, the resulting PSI protocol has high potential to be practically efficient.

Generalizing other PSI protocols. Our protocol generalizes several prior leading PSI/OPRF protocols [IKNP03, PRTY19, CM20], in the sense that these protocols are obtained by instantiating our protocol with an appropriate FSS. Since these protocols support PSI for arbitrary sets, we can interpret them as implicitly defining an FSS for arbitrary sets. These FSS schemes have share size proportional to the cardinality of the set. The real power of our paradigm is when it is used for structured sets, that have compact FSS shares that are significantly smaller than the cardinality of the set.

Fuzzy PSI implementation. We build a prototype implementation of fuzzy PSI using our new techniques. Our implementation supports 2-dimensional balls in the ℓ_∞ norm. When Alice has roughly 2700 balls of radius 30 (covering 10 million points), and Bob has a million points, our protocol requires roughly 41 seconds and 156 MB of communication. In contrast, the naïve approach (plain PSI with Alice's expanded set) requires 75 seconds and 1180 MB of communication using the efficient semi-honest plain PSI protocol of [KKRT16].

1.2 Related Work

Conventional PSI. Over the last decade, PSI techniques have matured and become truly practical. PSI is regularly used in practice to solve some of the problems listed above, at scale. There are quite a few protocol paradigms for PSI, including circuit-based [HEK12, PSWW18], oblivious polynomial evaluation via additively homomorphic encryption [KS05, DMRY11], key agreement [HFH99, DT10, JL10], bloom filters [DCW13, RR17a], to name a few.

Despite many interesting protocol approaches, modern PSI is practical and scalable **thanks almost entirely to the oblivious transfer (OT) paradigm.** Using modern OT extension [IKNP03] techniques, it is possible to generate many (millions) of OT instances extremely efficiently. These OTs are then used to carry out the comparisons necessary for PSI. With OT extension, the *marginal* cost each OT instance involves only cheap symmetric-key operations (*e.g.*, calls to AES). Thus, the OT-based PSI paradigm is the **only approach** in which each of the parties' items contributes just a small constant number of fast symmetric-key operations to the overall protocol cost. Pinkas, Schneider, and Zohner [PSZ14] were the first to propose basing PSI directly on OT; their approach was later refined in a series of works [PSSZ15, KKRT16, RR17b, PRTY19, PRTY20, CM20, GPR$^+$21, RS21]. The current leading OT-based 2-party PSI protocols are [KKRT16, CM20] in the semi-honest model and [PRTY20, RS21, GPR$^+$21] for malicious security.

Note that even recent progress on so-called *silent OT* [BCGI18, BCG$^+$19b, BCG$^+$19a, SGRR19, CRR21], which allows parties to generate essentially unlimited oblivious transfer instances with no communication, does **not** solve the problem of structure-aware PSI. Silent OT techniques generate only *random* OT correlations, which must be converted to chosen-input OT instances using communication [Bea95]. Conventional PSI protocols, even based on silent OT, require a number of OT primities (and hence communication) proportional to the cardinality of sets, and do not take advantage of sets with low description size.

PSI with Sublinear Communication. It is possible to construct a PSI protocol with communication sublinear in one of the parties' sets, using RSA accumulators [ADT11] or leveled fully homomorphic encryption [CLR17, CHLR18]. Both of these techniques are "heavy machinery" in the sense that they imply single-server PIR.

Other works have explored the use of an one-time offline phase for PSI [KLS$^+$17, RA18], especially in the context of private contact discovery [KRS$^+$19], where a large set remains relatively static. The use of an offline phase is out of scope in our work, as we measure total communication cost.

Fuzzy PSI. Fuzzy PSI was introduced by Freedman *et al.* [FNP04], who give a protocol for Hamming-distance (over tuples of strings). Later, Chmielewski and Hoepman [CH08] showed an attack against this protocol and proposed their own protocols, one of which was itself later attacked and improved upon by Ye *et al.* [YSPW10]. All of these protocols use the *oblivious polynomial evaluation* technique, in which the parties encode their input sets as roots of a polynomial and use additively homomorphic public-key encryption to manipulate these polynomials.

Indyk & Woodruff [IW06] describe a fuzzy PSI protocol for Hamming and ℓ_2 metrics, but their protocol requires generic MPC (*e.g.*, Yao's protocol) evaluation of a decryption circuit for a homomorphic encryption scheme, for every item. Bedő *et al.* [BCRT16] construct a fuzzy PSI protocol using homomorphic encryption. Doumen [Dou07] gives a fuzzy PSI protocol under a non-standard security model, where the goal is to bound the loss in entropy about the input sets caused by running the protocol.

Chakraborti *et al.* [CFR21] construct fuzzy PSI protocols (which they call *distance-aware PSI*) for Hamming distance, based on homomorphic encryption. Their protocol has false positives in the final result. They also describe a fuzzy PSI protocol for 1-dimensional integers — *i.e.*, points a and b are matched if $|a - b| \leq \delta$. This protocol is an elegant reduction to plain PSI, where parties can simply run plain PSI on sets that are larger by a factor of $O(\log \delta)$. Uzun *et al.* [UCK$^+$21] also recently proposed a fuzzy PSI protocol for Hamming distance, based on homomorphic encryption techniques..

There are several works studying so-called *fuzzy matching* or *fuzzy handshake* protocols [AKB07, WG14, WXL$^+$18]. These protocols reveal to the participants whether the intersection of their sets has cardinality above some threshold (*i.e.*, whether $|A \cap B| \geq t$; see also [GS19, ZCL21]). Such a functionality can be used for applications like fingerprint matching [SSNO12] and ride-sharing [HOS17]. For these works, the "fuzziness" is with respect to the entire sets A and B, measured by *exact* matches between items of A and B. In our setting, fuzziness refers to individual items of A and B that are similar but not necessarily equal.

Applications of Fuzzy PSI. Pal *et al.* [PIRC21] use fuzzy PSI in the context of compromised credentials checking, to check whether a user's password is *similar* to passwords that have been leaked online. They use the "naïve reduction" of fuzzy PSI to plain PSI, but set sizes in their application are small enough that this approach is practical.

2 Preliminaries

Semi-honest security. We rely on the *Universal Composability* (UC) framework from [Can01] to show that our 2-party protocols are secure against passive adversaries. Parties P_0 and P_1 with inputs x_0 and x_1 run protocol Π to learn the output of a function $f(x_0, x_1)$; P_i's view $\mathsf{View}(P_i, 1^\kappa, x_0, x_1)$ during an honest execution consists of her private input x_i, privately chosen randomness and the transcript of the protocol.

We say that protocol Π securely realizes a functionality f if there exists a simulator Sim for both parties and for all possible inputs x_0, x_1 such that:

$$\mathsf{Sim}(P_i, x_i, f(x_0, x_1), 1^\kappa) \cong \mathsf{View}(P_i, x_0, x_1, 1^\kappa)$$

the views from the simulation and honest execution are computationally indistinguishable in the security parameter κ.

2.1 Hamming Correlation Robustness

Our protocol is based on IKNP OT-extension [IKNP03]. That protocol requires a hash function with a certain security property:

Definition 1 ([IKNP03])**.** *Let $H : \{0,1\}^* \times \{0,1\}^\kappa \to \{0,1\}^v$ be a function and define the related function $F_s(t, x) = H(t; x \oplus s)$, where $s \in \{0,1\}^\kappa$. We say that H is **correlation robust** if F_s is indistinguishable from a random function, against distinguishers that never query with repeated t-values. Intuitively, values of the form $H(t_i; x_i \oplus s)$ look jointly pseudorandom, even with known t_i, x_i values and a common s.*

All protocols in the "IKNP family" require a hash function with a similar kind of security property; *e.g.*, [KK13, KKRT16]. The specific property we use is defined below:

Definition 2. *Let $H : \{0,1\}^* \times (\{0,1\}^k)^\ell \to \{0,1\}^v$ be a function and define the related function $F_s(t, x, \Delta) = H(t; x \oplus s \odot \Delta)$, where $s \in \{0,1\}^\ell$; x, Δ are vectors of length ℓ with components in $\{0,1\}^k$; and $\odot$ is componentwise multiplication (of a bit times a string). We say that H is **Hamming correlation robust** if F_s is indistinguishable from a random function, against distinguishers that never query with repeated t-values and always query with Δ having at least κ nonzero components.*

Intuitively, values of the form $H(t_i; x_i \oplus s \odot \Delta_i)$ look jointly pseudorandom, even with known t_i, x_i, Δ_i values and a common s, provided that each Δ_i has high Hamming weight.

This definition generalizes the one from [KKRT16] in that x and Δ are bit strings (vectors of bits) in [KKRT16], whereas in our protocol x and Δ can be vectors with components from $\{0,1\}^k$.

3 Building Blocks

3.1 2PC Ideal Functionalities

Oblivious transfer is a special case of secure two-party computation, in which a sender has a pair of input strings m_0, m_1, a receiver has input $b \in \{0,1\}$, and the receiver learns output m_b. The sender learns nothing about b, and the receiver learns nothing about m_{1-b}.

Our structure-aware PSI protocol requires the parties to perform a small number of oblivious transfers. We use $\mathcal{F}_{\mathsf{ot}}$ to denote an ideal functionality providing an instance of oblivious transfer.

3.2 Function Secret Sharing

A 2 party FSS scheme for a class of functions $\mathcal{F}$ allows a dealer to distribute a function $f \in \mathcal{F}$ into two shares (f_1, f_2), where each share individually hides the function f. Furthermore, $f(x) = f_1(x) \oplus f_2(x)$ for all inputs x. The main efficiency measure of FSS is the size of the function shares f_1, f_2. FSS was first introduced by Boyle, Gilboa, and Ishai [BGI15], who also described efficient FSS schemes for point functions and other function classes. Their original formulation of FSS can be found in Appendix D.1.

Boolean Function Secret Sharing In this work we will specifically be interested in secret sharing *indicator functions* for a family of sets. The indicator function evaluates to 0 when the input belongs to the set, and otherwise it evaluates to 1. We relax the definition of FSS for indicator functions to allow for one-sided false positive error and output length being greater than a bit. We call our definition (p, k)-**Boolean Function Secret Sharing** ((p, k)-bFSS), where p is the false positive error probability and k is the output length. Let $\mathcal{S} \subseteq 2^{\mathcal{U}}$ be a family of sets for some universe of points $\mathcal{U}$. Then we formally define the syntax and security of this relaxed FSS primitive as follows:

Definition 3 (bFSS syntax)**.** *A 2-party (p, k)-bFSS scheme for a family of sets $\mathcal{S} \subseteq 2^{\mathcal{U}}$ with input domain $\mathcal{U}$ consists of a pair of algorithms (Share,Eval) with the following syntax:*

- $(k_0, k_1) \leftarrow \mathit{Share}(1^\kappa, \hat{S})$*: The randomized share function takes as input the security parameter κ and (the description of) a set $S \in \mathcal{S}$ as input. It outputs two shares.*
- $y_{\mathit{idx}} \leftarrow \mathit{Eval}(1^\kappa, \mathit{idx}, k_{\mathit{idx}}, x)$*: The deterministic evaluation function takes as input the security parameter, party index $\mathit{idx} \in \{0, 1\}$, the corresponding share k_{idx} and input $x \in \mathcal{U}$, and it outputs a string $y_{\mathit{idx}} \in \{0, 1\}^k$.*

Usually the security parameter 1^κ is not written as an explicit function argument.

Definition 4 (bFSS security)**.** *A 2-party (p, k)-bFSS scheme $(\mathit{Share}, \mathit{Eval})$ for $\mathcal{S} \subseteq 2^{\mathcal{U}}$ is* ***secure*** *if is satisfies the following conditions:*

- ***Correctness for yes-instances:*** *For every set $S \in \mathcal{S}$, $x \in S$, and security parameter κ:*

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

- ***Bounded false positive rate:*** *For every set $S \in \mathcal{S}$, $\underline{x \in \mathcal{U} \setminus S}$, and security parameter κ:*

$$\Pr\left(y_0 \oplus y_1 \neq 0^k \;\middle|\; \begin{array}{l} (k_0, k_1) \leftarrow \mathit{Share}(1^\kappa, S) \\ y_0 \leftarrow \mathit{Eval}(1^\kappa, 0, k_0, x) \\ y_1 \leftarrow \mathit{Eval}(1^\kappa, 1, k_1, x) \end{array} \right) \geq p$$

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

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

  In other words, each individual share leaks nothing about S. We further say that the bFSS *has* ***pseudorandom keys*** *if the output of Sim is a random string of some fixed length.*

Definition 5 (Strong bFSS)**.** *A* ***strong*** bFSS *is a $(1, 1)$-bFSS.*

Strong bFSS corresponds to the original FSS definition of [BGI15] — *i.e.*, no false positives — when restricted to sharing indicator functions of sets.

Definition 6 (PRF property)**.** *A (p,k)-bFSS scheme (Share, Eval) for a family of sets $\mathcal{S} \subseteq 2^{\mathcal{U}}$ with input domain $\mathcal{U}$ and key space $\mathcal{K}$ is said to satisfy the PRF property if for any idx $\in \{0,1\}$, $x \in \mathcal{U}$, Expt$(1^\kappa, \mathsf{idx}, x)$ is indistinguishable from a uniform random string, where Expt is defined as:*

```
Expt($1^\kappa$, idx, $x$):
    $k \leftarrow \mathcal{K}$
    return Eval($1^\kappa$, idx, $k$, $x$)
```

3.3 Oblivious Key Value Store

An **oblivious key value store (OKVS)** [GPR+21] is a data structure that encodes a set of key-value mappings while hiding the set of keys used.

Definition 7 ([GPR+21])**.** *An **oblivious key-value store (OKVS)** consists of algorithms Encode and Decode, with an associated space $\mathcal{K}$ of keys and space $\mathcal{V}$ of values. An OKVS must satisfy the following properties:*

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

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

  One may call Decode with any $k \in \mathcal{K}$, and indeed, someone who holds Encode(A) may not know whether a particular k was included in A.
- **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 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($\{(k_1, v_1), \ldots, (k_n, v_n)\}$)
```

Garimella *et al.* [GPR+21] define the properties of an OKVS and construct an efficient one based on 3-way cuckoo hashing. If values from $\in \mathcal{V}$ require v bits to write, then their construction encodes n key-value pairs with roughly $1.35nv$ bits — close to the optimal length nv.

A special class of OKVS is **boolean OKVS**, where the OKVS data structure itself is a vector of strings $D = (d_1, \ldots, d_n)$ and Eval is defined as $\mathsf{Eval}(D, x) = \bigoplus_{i \in \pi(x)} d_i = \langle \pi(x), D \rangle$ for some function π. The function π specifies which positions of the data structure to probe.

Our construction also requires the following additional property of OKVS, that we prove is satisfied by all existing OKVS constructions in Appendix A :

Definition 8. *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 Decode$(\mathsf{Encode}(A), k^*)$ is indistinguishable from random, over the randomness in Encode.*

4 bFSS Constructions

Our high-level goal is efficient bFSS schemes for collections of ℓ_∞ balls in d dimensions. This kind of geometric structure can be viewed hierarchically: a *union* of balls, where each ball is a *Cartesian product* of *1-dimensional intervals.* Our final bFSS constructions reflect this hierarchy. At each level of the hierarchy there may be different choices of bFSS construction. A visual overview of the possibilities is provided in Figure 1.

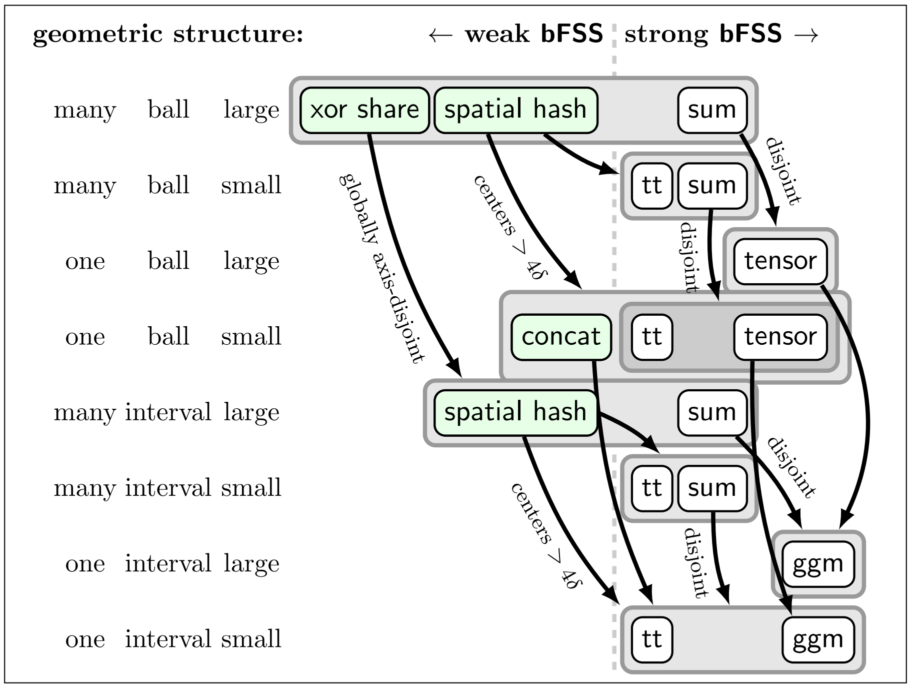


Figure 1: Map of bFSS constructions for geometric structures. Large/small refer to whether the structure exists in a large or small geometric domain. Edges in this map represent reductions. For example, the leftmost edge means: *The "xor share" construction reduces the problem of bFSS for "many ball large" structures to the problem of bFSS for "many interval large" structures, provided that the input satisfies the "globally axis-disjoint" condition.* All weak bFSS constructions are new in this work.

4.1 Existing Schemes

In this section we recall bFSS constructions from previous works that are relevant for the geometric structures that we consider. All prior work considers only **strong**, *i.e.*, $(1, 1)$-bFSS. All of which satisfy the PRF property:

Theorem 9. *Any strong bFSS F for a collection of sets $\mathcal{S}$ in the universe $\mathcal{U}$ with pseudo-random keys satisfies the PRF property (Definition 6).*

A proof of this theorem can be found in Appendix F.

The simplest of all bFSS schemes is a trivial sharing of a **truth table**. We denote this construction as tt. The two parties hold additive secret shares of the truth table for the set's membership function. This bFSS construction is viable only for sets that exist in a relatively small universe of items.

What we call the sum construction is a simple method to construct bFSS for a disjoint union, from an bFSS for each term in the union. The method works only for strong, *i.e.*, $(1, 1)$-bFSS.

Theorem 10 (sum). *If F_1 is a $(1, 1)$-bFSS for $\mathcal{S}_1$ with share size σ_1, and F_2 is a $(1, 1)$-bFSS for $\mathcal{S}_2$ with share size σ_2, then $\mathsf{sum}[F_1, F_2]$ is a $(1, 1)$-bFSS for $\{S_1 \cup S_2 \mid S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2, S_1 \cap S_2 = \emptyset\}$ with share size $\sigma_1 + \sigma_2$.*

A proof of this can be found in Appendix E.2. Boyle *et al.* [BGI15] construct an bFSS scheme for intervals, which we call ggm (it composes PRGs together into a GGM-style tree). Let $\mathcal{D} = \{0, 1, \ldots, 2^u - 1\}$ and define $\mathsf{INT}_u = \{[a, b] \mid a, b \in \mathcal{D}\}$ — *i.e.*, the set of all 1-dimensional intervals.

Theorem 11 (ggm [BGI15]). *There exists a $(1, 1)$-bFSS for INT_u satisfying the PRF property with pseudomrandom keys of share size $O(\kappa u)$ and Eval cost containing $O(u)$ PRG calls.*

Boyle *et al.* [BGI15] also introduce a technique, which we call tensor, for decision trees and cross-products of bFSS schemes. This technique can be combined with the ggm construction to realize bFSS for d-dimensional intervals. Define $\mathsf{multi\text{-}INT}_{u,d} = \{I_1 \times I_2 \times \cdots \times I_d \mid I_1, I_2, \ldots, I_d \in \mathsf{INT}u\}$ — *i.e.*, the set of all d-dimensional intervals. Note that an ℓ_∞ ball is a d-dimensional interval with sides of equal length.

Theorem 12 (tensor [BGI15]). *There exists a $(1, 1)$-bFSS for $\mathsf{multi\text{-}INT}_{u,d}$ with pseudorandom keys of share size $O(\kappa u^d)$ and Eval cost dominated by $O(u^d)$ PRG calls.*

Define $\mathsf{union\text{-}dint}_{u,d,n}$ to be the family of sets consisting of the union of at most n disjoint d-dimensional intervals in the domain $\{0, 1, \ldots, 2^u - 1\}^d$. From Theorem 10 and Theorem 12 we obtain a bFSS for this collection of sets:

Theorem 13. *There exists a $(1, 1)$-bFSS for $\mathsf{union\text{-}dint}_{u,d,n}$ with pseudorandom keys of share size $O(\kappa n u^d)$ and Eval cost dominated by $O(nu^d)$ PRG calls.*

4.2 New concat Technique for Cross Products

We now describe our new bFSS techniques. A common theme in all of these constructions is that they take advantage of the bFSS generalization to construct (p, k)-bFSS for $p < 1$ and/or $k > 1$.

concat denotes our simple new approach for cartesian product of several bFSS. We construct an bFSS for a product $S_1 \times S_2$ by simply concatenating outputs of an bFSS for S_1 and an bFSS for S_2. This gives us a secure (p, k)-bFSS construction for the cross product of sets with $p = \min_i p_i$ and $k = k_1 + k_2$, assuming we start from a (p_1, k_1)-bFSS and (p_2, k_2)-bFSS. The construction is

```
$\underline{\mathsf{Share}_n^{\mathcal{S}}(1^\kappa, S_1 \times S_2)}$:
    Initialize $k_0, k_1$ as empty associated arrays
    $(k_0[0], k_1[0]) \leftarrow \mathsf{Share}^{\mathcal{S}_1}(1^\kappa, S_1)$
    $(k_0[1], k_1[1]) \leftarrow \mathsf{Share}^{\mathcal{S}_2}(1^\kappa, S_2)$
    return $(k_0, k_1)$

$\underline{\mathsf{Eval}_n^{\mathcal{S}}(1^\kappa, \mathsf{idx}, \mathsf{FSSkey}, x)}$:
    return $\mathsf{Eval}^{\mathcal{S}_1}(1^\kappa, \mathsf{idx}, \mathsf{FSSkey}[0], x) ||$
    $\mathsf{Eval}^{\mathcal{S}_2}(1^\kappa, \mathsf{idx}, \mathsf{FSSkey}[1], x)$
```

Figure 2: bFSS for cross product $S_1 \times S_2$ given bFSS for S_1 and S_2

described formally in Figure 2. Similar to the disjoint union construction, the share size and the Eval complexity of this construction is the sum of share size and Eval cost for individual bFSS respectively.

Theorem 14. *If F_1 is a (p_1, k_1)-bFSS for $\mathcal{S}_1$ with share size σ_1, and F_2 is a (p_2, k_2)-bFSS for $\mathcal{S}_2$ with share size σ_2, then concat$[F_1, F_2]$ is a $(\min\{p_1, p_2\}, k_1 + k_2)$-bFSS for $\{S_1 \times S_2 \mid S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2\}$ with share size $\sigma_1 + \sigma_2$.*

Since the output of concat is simply the concatenation of the output of the two Eval's, we get the following property as well:

Theorem 15. *If bFSS F_1 and F_2 satisfy the PRF property, then concat$[F_1, F_2]$ satisfies the PRF property.*

A single ℓ_∞ ball in d dimensions can be represented as the cross product of d intervals along each of the dimension. Hence using the general bFSS construction rule in Subsection 4.2 (the concat technique) and the strong bFSS for a single interval we get a $(1, d)$-bFSS for a single ℓ_∞ ball.

Theorem 16. *There exists a $(1, d)$-bFSS for $\mathcal{S} =$ multi-INT$_{u,d}$ in $\mathcal{U} = \{0, 1, \ldots, 2^u - 1\}^d$ satisfying the PRF property with pseudo-random keys of size $O(\kappa u d)$ and Eval cost $O(ud)$ PRG calls.*

4.3 New Spatial Hashing Technique

Here we describe our proposed approach for constructing bFSS keys for geometric objects. The spatial-hashing approach reduces the problem of bFSS for a geometric structure in a large domain to an easier problem of bFSS in a smaller domain.

Intuition : Partition the space $\{0, \ldots, 2^u - 1\}^d$ into regular grid cells. Call a grid cell "active" if it intersects with the input set that is being shared. We will build an bFSS that gives correct output in all active grid cells, while ensuring that the bFSS output in inactive grid cells is random. Because the inactive grid cells contain only points outside of the set, this approach can produce only false positives in the bFSS output (with bounded probability), which suffices for a (p, k)-bFSS with $p < 1$.

We require a component bFSS (let's call it GridFSS) which supports the possible structures that can exist in a single grid cell. To hide the identities of the active cells, we use an OKVS data structure to map grid cell identifiers to shares of GridFSS. This ensures two properties:

1. Decoding the OKVS at an active grid cell outputs the corresponding correct GridFSS share
2. Decoding the OKVS at a non-active grid cell outputs a uniformly random string, independent of the other values in the OKVS.

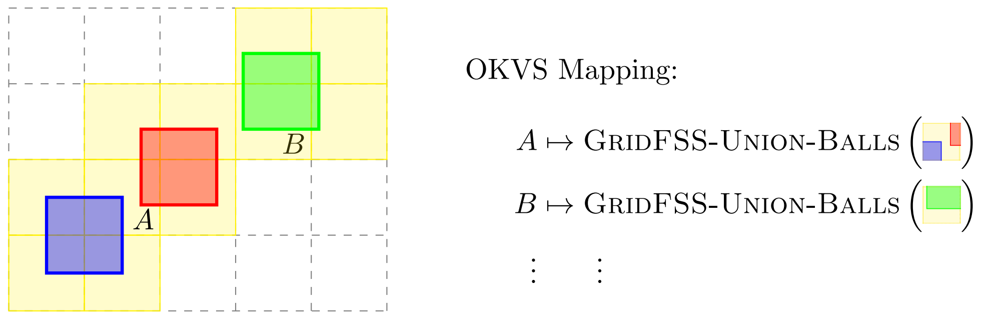


Figure 3: Spatial hashing technique applied to an example of 3 disjoint ℓ_∞ balls in 2 dimensions. Active grid cells are shaded yellow. Contents of each active grid cell are shared using a component bFSS for union of balls. bFSS shares for each active grid cell are encoded into an OKVS (mapping grid cell ID to bFSS share).

Hence we define our overall bFSS shares to each be an OKVS data structure that maps grid cells to GridFSS shares. To evaluate this share at a point $\boldsymbol{x}$, identify that point's grid cell, query the OKVS at that grid cell, interpret the OKVS output as a share in GridFSS, and evaluate that share at $\boldsymbol{x}$. The first property above ensures the correctness of the FSS when the input point is in an active grid cell. The second property ensures that outside the active grid cell the output of Eval is random.

This spatial-hashing technique is also illustrated on an example input geometry in Figure 3.

Spatial hashing reduces the problem of bFSS in a large universe to bFSS in a small grid cell. The benefit of this is that bFSS in small grid cells can be extremely efficient. Specifically, there are many *asymptotically* efficient bFSS for various structures, but when the universe is small enough, they are concretely inferior to the trivial truth table (tt) construction. The reader should think of grid cells as small enough so that the tt approach has the smallest concrete share size. In practice, the threshold for this is grid cells with side length of a few hundred.

We partition the space $\mathcal{D} = [0, 2^u - 1]^d$ into *grid cells*, which are d-dimensional cubes of side length 2δ. We can uniquely *label* each grid cell by the point contained in it that is closest to the origin. Further we define a function cell_δ (parameterized by the grid size) that maps any point in the domain $\mathcal{D}$ to its unique grid cell label. Hence the function $\mathsf{cell}_\delta^{-1}$ maps a grid cell label to the set of points contained in it. Formally we define these maps as:

Definition 17. *For any vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathcal{D}$, we define the function that maps a point to its grid cell label as: $\mathsf{cell}_\delta(\boldsymbol{x}) = \lfloor \boldsymbol{x}/2\delta \rfloor = (\lfloor x_1/2\delta \rfloor, \ldots, \lfloor x_d/2\delta \rfloor)$. We also define $\mathsf{cell}_\delta^{-1}(\boldsymbol{x}) = [x_1, x_1 + 2\delta) \times [x_2, x_2 + 2\delta) \times \ldots \times [x_d, x_d + 2\delta)$, which maps any grid cell label to the set of points contained in that grid cell.*

Definition 18. *Define $G(\delta, u, d)$ = set of all grid cells = $\{\mathsf{cell}_\delta^{-1}(\boldsymbol{x}) \mid x_1, \ldots, x_d \in \{2k\delta | k \in [0, 2^{u-1}/\delta]\}\}$*

Definition 19. *Let $\mathcal{S}$ be a family of sets over universe $\mathcal{U} = \{0, \ldots, 2^u - 1\}^d$. The set of all active grid cells is $\mathsf{ActiveCells}(\delta, S) = \{C \in G(\delta, u, d) \mid C \cap S \neq \emptyset\}$. Define $\mathsf{MaxActiveCellCount}(\delta, \mathcal{S}) = \max_{S \in \mathcal{S}} |\mathsf{ActiveCells}(\delta, S)|$.*

We propose a bFSS for the input structure S, given a bFSS GridFSS that supports the contents of each active grid cell. To take advantage of the fact that grid cells are very small, we normalize all grid cells to the origin with the following function which translates a grid cell to the origin:

```
Given a $(p,k)$-bFSS $F$ for $\mathcal{S}_\delta$ and an OKVS $(\mathsf{Encode}, \mathsf{Decode})$
$\underline{\mathsf{Share}^{\mathcal{S}}(1^\kappa, S)}$:

    $\mathsf{DummyCells} \leftarrow \emptyset$
    $\mathsf{GridKey}_0, \mathsf{GridKey}_1$ - associative arrays initialized empty
    for each $C(\boldsymbol{x}) \in \mathsf{ActiveCells}(\delta, S)$:
        $(\mathsf{GridKey}_0[\boldsymbol{x}], \mathsf{GridKey}_1[\boldsymbol{x}]) \leftarrow \mathsf{GridFSS.Share}(1^\kappa, \mathsf{ShiftOrigin}(S \cap \mathsf{cell}_\delta^{-1}(-\boldsymbol{x}))$
    do $\mathsf{MaxActiveCellCount}(\delta, \mathcal{S}) - |\mathsf{ActiveCells}(\delta, S)|$ times:
        Pick any $C'(\boldsymbol{y}) \in G(\delta, u, d) \setminus (\mathsf{ActiveCells}(\delta, S) \cup \mathsf{DummyCells})$
        $\mathsf{DummyCells} \leftarrow \mathsf{DummyCells} \cup \{\boldsymbol{y}\}$
        $(\mathsf{GridKey}_0[\boldsymbol{y}], \mathsf{GridKey}_1[\boldsymbol{y}]) \leftarrow \mathsf{GridFSS.Share}(1^\kappa, \emptyset)$
    $k_0 \leftarrow \mathsf{OKVS.Encode}(\mathsf{GridKey}_0)$
    $k_1 \leftarrow \mathsf{OKVS.Encode}(\mathsf{GridKey}_1)$
    return $(k_0, k_1)$

$\underline{\mathsf{Eval}^{\mathcal{S}}(1^\kappa, \mathsf{idx}, k, \boldsymbol{x})}$:
    $\mathsf{GridKey} \leftarrow \mathsf{OKVS.Decode}(k, \mathsf{cell}(\boldsymbol{x}))$
    return $\mathsf{GridFSS.Eval}(1^\kappa, \mathsf{idx}, \mathsf{GridKey}, \boldsymbol{x})$
```

Figure 4: $\mathsf{spatial\text{-}hashing}_{\delta,d}$ construction for the collection of sets $\mathcal{S}$ with grid size δ in domain $\mathcal{U} = \{0, 1, \ldots, 2^u - 1\}^d$

Definition 20. *For any grid cell $C \in G(\delta, u, d)$ and any $\boldsymbol{x} \in \mathcal{U}$ we can define the function $\mathit{ShiftOrigin}(C, \boldsymbol{x}) = \{\boldsymbol{y} - \boldsymbol{x} \mid \boldsymbol{y} \in C\}$*

The spatial-hashing technique is presented formally in Figure 4 and it gives us an bFSS with the following parameters (formal proof is provided in Appendix F):

Theorem 21. *Let $\mathcal{S}$ be a family of sets over universe $\mathcal{U} = \{0, \ldots, 2^u - 1\}^d$. Let δ be an arbitrary integer representing the grid size. Define $\mathcal{S}_\delta = \{\mathit{ShiftOrigin}(S \cap \mathit{cell}_\delta^{-1}(\boldsymbol{x}), \boldsymbol{x}) \mid S \in \mathcal{S}, C(\boldsymbol{x}) \in G(\delta, u, d)\}$.*

If GridFSS is a (p,k)-bFSS for $\mathcal{S}_\delta$ with pseudo-random keys and satisfying the PRF property with share size σ, then spatial-hashing$_{\delta,d}$[GridFSS] is a $(\min\{1 - 2^{-k}, p\}, k)$-bFSS for $\mathcal{S}$ with share size $O(\mathit{MaxActiveCellCount}(\delta, \mathcal{S}) \cdot \sigma)$

We next employ this spatial-hashing technique to develop efficient bFSS constructions for union of ℓ_∞-balls.

4.3.1 bFSS for union of disjoint ℓ_∞ balls

We define the collection of sets $\mathsf{union\text{-}disj}_{u,d,\delta,n}$ to contain sets, each containing at most n disjoint ℓ_∞ balls of radius δ in the $\mathcal{U} = \{0, 1, \ldots, 2^u - 1\}^d$. If we use the spatial hashing technique for the same grid size parameter δ then each active grid cell would intersect at most 2^d input balls and MaxActiveCellCount would be $n2^d$.

Lemma 22. *Any radius-δ ℓ_∞ ball in $\mathcal{U}$ intersects with at most 2^d disjoint ℓ_∞-balls.*

Proof. An ℓ_∞ ball with radius δ is a d dimensional cube of side length 2δ. Hence the intersection of any two overlapping ℓ_∞ δ radius balls contains at least one vertex of both the balls. The lemma follow from the fact that a d dimensional cube has at max 2^d vertices and that the input set S contain all disjoint ℓ_∞ balls. □

Lemma 23. *For any $S \in$ union-disj$_{u,d,\delta,n}$ and any grid cell $C \in G(\delta, u, d)$, the cell $\mathsf{cell}_\delta^{-1}(C)$ intersects with at max 2^d balls in S.*

Proof. $\mathsf{cell}_\delta^{-1}(C)$ is itself a ℓ_∞ ball of radius δ. Hence this follows from the previous lemma. □

Lemma 24. *For $\mathcal{S} =$ union-disj$_{u,d,\delta,n}$, MaxActiveCellCount$(\delta, \mathcal{S}) = n2^d$*

To construct bFSS for union-disj$_{u,d,\delta,n}$ we need a component GridFSS that can support the union of at most 2^d balls in a single grid cell. We can use the bFSS for union-dint with $\mathcal{U} = \{0, 1, \ldots, 2\delta - 1\}^d$, which gives us the following theorem:

Theorem 25. *There exists a (0.5,1) − bFSS for the collection of sets union-disj$_{u,d,\delta,n}$ in $\mathcal{U} = \{0, 1, \ldots, 2^u - 1\}^d$, with key size $O(n\kappa(4 \log \delta)^d)$ bits and evaluation cost dominated by $O((2 \log \delta)^d)$ calls to a PRG.*

4.3.2 Union of ℓ_∞ balls with pairwise distance greater than 4δ.

We can have a more efficient bFSS when the input balls are known to be sufficiently far apart. Suppose the set of balls have pairwise distance greater than 4δ. This collection of sets union-4delta$_{u,d,\delta,n}$ is parameterized by u (defines the universe $\mathcal{U} = \{0, 2, \ldots, 2^u - 1\}^d$), number of dimensions d, radius of balls δ and the number of balls n.

Lemma 26. *Any grid cell intersects with at most 1 ℓ_∞ balls from S, for any $S \in$ union-4delta$_{u,d,\delta,n}$.*

Proof. Suppose that an ℓ_∞ ball centered at c_0 intersects two balls in S centered at c_1 and c_2 respectively. Then we have:

$$\begin{aligned} & d_\infty(c_0, c_1) \leq 2\delta \text{ and } d_\infty(c_0, c_2) \leq 2\delta \\ \Longrightarrow\ & d_\infty(c_1, c_2) \leq d_\infty(c_0, c_1) + d_\infty(c_0, c_1) \leq 4\delta \qquad \text{(By triangle inequality)} \end{aligned}$$

This contradicts the assumption that all balls have centers greater than 4δ apart. □

If we apply spatial hashing to such a set of balls, we get that each active grid cell will intersect with at most 1 input ball. Hence, we can apply the spatial hashing construction using a simpler GridFSS— namely, we can use the bFSS for a single d-dimensional interval (multi-INT$_{2\delta,d}$). This saves a factor of $O(2^d/d)$ from the overall share size.

Theorem 27. *There exists a $(1 - 1/2^d, d)$-bFSS for the collection of sets union-4delta$_{u,d,\delta,n}$ in $\mathcal{U} = \{0, \ldots, 2^u - 1\}^d$, with key size $O(nd\kappa(2)^d \log \delta)$ and evaluation cost dominated by $O(d \log \delta)$ calls to a PRG.*

4.3.3 ℓ_1 balls.

ℓ_∞ balls are the simplest objects we support, but ℓ_1 balls can also be supported. We sketch here the main differences between ℓ_1 and ℓ_∞ balls. Regardless of the type of objects being shared in a bFSS, the spatial hashing technique requires the *grid cells* to be ℓ_∞ balls, as they tile the space.

There are three factors to consider when changing the type of balls in our constructions:

(1) Each ℓ_1 ball can intersect at most 2^d grid cells, because the worst case (as with ℓ_∞ balls) is that the ℓ_1 ball covers a corner of the grid cell and extends into all 2^d cells incident to that corner. Hence the maximum number of active grid cells is $n2^d$ — the same as for ℓ_∞ balls.

(2) When considering bFSS for the union of disjoint ℓ_1-balls, we must have an upper bound on the number of balls that can intersect a single grid cell. We can compute such a bound via a

simple volume argument. The volume of a d-dimensional ℓ_1 ball of radius 1 is $2^d/d!$.Any ℓ_1 ball that intersects with a grid cell (with sidelength 2) is contained completely in a grid cell of sidelength 4, and that grid cell has volume 4^d. Hence there can be at most $4^d/(2^d/d!) = d!2^d$ balls. This is a factor $d!$ larger than the case of ℓ_∞ balls.

(3) Within a grid cell, we require an bFSS for either one (in the case of distant balls) or many (in the case of merely disjoint balls) balls in the grid cell. bFSS for a single ℓ_1 ball is more expensive than for ℓ_∞. This is because an ℓ_∞ ball is the intersection of d pairs of parallel half-planes, while an ℓ_1 ball is the intersection of 2^{d-1} paris of parallel half-planes. So in the component GridFSS, the dependence on dimension increases from d to 2^d when considering ℓ_1 balls.

4.4 xor-share technique

The bFSS construction for union of ℓ_∞ disjoint balls does not scale well with the number of dimensions, as its share size is proportional to 2^d. This stems from the fact that each input ball can intersect with 2^d grid cells in the worst case.

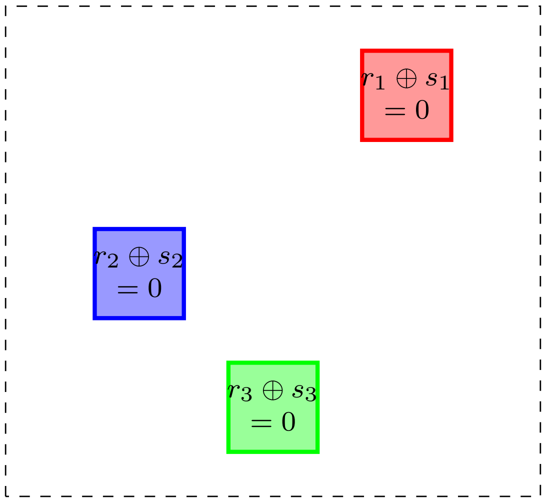
To improve the dependence on the dimension, consider the following approach. For each input ball i, generate an additive sharing of 0: $R[i,1] \oplus \cdots \oplus R[i,d] = 0$, where each share is assigned to one of the d dimensions. Now imagine projecting all balls onto the jth dimension — the result is a union of intervals. Suppose we had a bFSS for the union of 1-dimensional intervals, which would output $R[i,j]$ whenever a point is in the projection of the ith ball, and would output a random bit whenever the point is outside of all ball-projections. Then for every point $\boldsymbol{x}$, we could evaluate one bFSS for each dimension (the jth bFSS for 1-dimensional intervals, evaluated at x_j), and xor the results.

If $\boldsymbol{x}$ is contained in input ball i, then the result yields $R[i,1] \oplus \cdots \oplus R[i,d] = 0$. If $\boldsymbol{x}$ is "hit" by the projection of ball i in all but the last dimension, say, then the resulting xor contains $R[i,1]\oplus\cdots\oplus R[i,d-1]$ which is uniformly random – even if $\boldsymbol{x}$ is "hit" by a different ball in dimension d. If $\boldsymbol{x}$ is not "hit" by any ball in the jth dimension, then its corresponding xor likewise gives a random result. No matter what, if $\boldsymbol{x}$ is not in any ball, then its xor is random, and this leads to a $(p,1)-$ bFSS for $p = 1/2$. Note that the total share size for this bFSS is only d times larger than a bFSS for a union of 1-dimensional intervals. In other words, the dependence on dimension is no longer exponential.

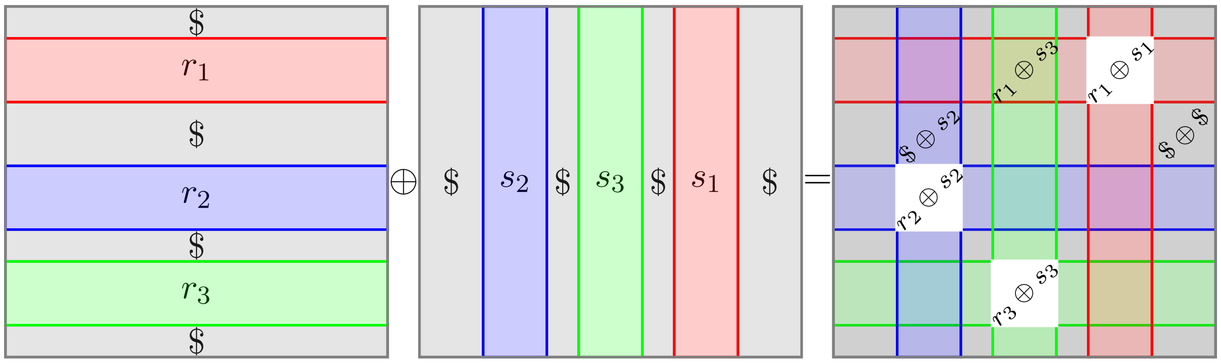
However, there is one problem with this approach: What should we do if balls i and i' overlap when projected onto dimension j? In that case we would expect the jth component bFSS to evaluate to both $R[i,j]$ and $R[i',j]$ on some points. Hence, this general approach only works when the input set of ℓ_∞ balls are *globally axis disjoint*, meaning that the balls have disjoint projections onto each dimension. We define the collection of sets union-glob-disj$_{u,d,\delta,n}$, to contain globally axis disjoint n ℓ_∞ d dimentional balls of radius δ in $\mathcal{U} = \{0,1,\ldots,2^u-1\}^d$. This xor-share technique is illustrated in Figure 5 and described formally in Figure 6.

Let $\pi_i(x_1,\ldots,x_d) = x_i$ denote the projection of a point along dimension i. We extend this function to sets as: $\pi_i(S) = \{\pi_i(\boldsymbol{x}) \mid \boldsymbol{x} \in S\}$, and note that the projection of an ℓ_∞ ball onto any dimension is a 1-dimensional interval.

We can use our spatial hashing technique, but with one important caveat. Usually spatial hashing simply performs an bFSS share of the intersection of the input set with each active grid cell. However, our standard approach for spatial hashing causes the bFSS to always output 0 for intervals in the input set. In this construction we need the bFSS to sometimes output 0 and sometimes output 1 for these intervals, since they encode a particular secret share. To account for this we modify the spatial-hashing construction to take as input a set S and a set of grid cells $acvCells$, such that ActiveCells $\subset acvCells$ and the spatial-hashing construction encodes GridFSS



(a) Original geometric structure - globally axis disjoint



(b) Representing the original structure as the xor of two 1-dimensional structures.

Figure 5: An illustration of our xor-share technique, applied to 3 globally axis-disjoint balls in 2 dimensions

keys into an OKVS for each cell in *acvCells*. We use this generalized spatial-hashing construction when presenting this construction in Figure 6. Proof for the following theorem can be found in Its proof can be found in Appendix F.

Theorem 28. *The construction in Figure 6 is a $(0.5, 1)$-bFSS for collection of sets union-glob-disj$_{u,d,\delta,n}$ in $\mathcal{U} = \{0, \ldots, 2^u - 1\}^d$, with key size $O(nd \log \delta)$ bits and the evaluation cost being dominated by $O(d \log \delta)$ calls to a PRG.*

5 Structure-aware PSI from bFSS

In this section we present our protocol for **structure-aware PSI**. This is a variant of PSI in which the receiver's (Alice's) input set has a known structure. The functionality details are given in Figure 7.

Protocol Intuition. As a warmup, suppose we have a strong bFSS (*i.e.*, a $(1, 1)$-bFSS). Let Alice generate κ independent sharings of her structured set A, as $(k_0^{(i)}, k_1^{(i)}) \leftarrow \mathsf{Share}(A)$. Bob chooses a

```
Parameters:
    • $(1,1)$-bFSS $F$ for the collection of sets $\mathsf{union\text{-}dint}_{\lceil \log \delta \rceil,1,2}$
    • An OKVS $(\mathsf{Encode}, \mathsf{Decode})$
$\mathsf{Share}(1^\kappa, S)$:
  for $i = 1$ to $n$:    // additive sharing of zero for each ball
    $R[i,1], \ldots, R[i,d] \leftarrow_\$ \{0,1\}$, subject to $\oplus_{j=1}^{d} R[i,j] = 0$
  $S', acvCells \leftarrow \emptyset$
  for $j = 1$ to $d$:
    For each $C = \mathsf{cell}^{-1}(x) \in G(\delta, u, 1)$ such that $C \cap \pi_j(S) \neq \emptyset$
      Sample $r \leftarrow_\$ \{0,1\}$
      $acvCells \leftarrow acvCells \cup \{x\}$
      $S' \leftarrow S' \cup \{y \in C \mid$ if for some $i \in [1,n], y \in \pi_j(S_i)$ and $R[i,j] = 0\}$
      $S' \leftarrow S' \cup \{y \in C \mid y \notin \pi_j(S)$ and $r = 0\}$
    $(k_0[j], k_i[j]) \leftarrow \mathsf{spatial\text{-}hashing}_{\delta,1}[F](S', acvCells)$
  return $(k_0[1,\ldots,d], k_1[1,\ldots,d])$

$\mathsf{Eval}(1^\kappa, \mathsf{idx}, k, \boldsymbol{x})$:
  Parse $k$ as an array of $d$ OKVS $(k[1], k[2], \ldots, k[d])$
  for $j = 1$ to $d$:
    $\mathsf{GridKey}[j] \leftarrow \mathsf{OKVS.Decode}(k[j], \mathsf{cell}_\delta(x_j))$
    $y[j] \leftarrow F.\mathsf{Eval}(1^\kappa, \mathsf{idx}, \mathsf{GridKey}[j], x_j)$
  return $\oplus_{j=1}^{d} y_j$
```

Figure 6: A $(0.5, 1)$-bFSS for the collection of sets $\mathsf{union\text{-}glob\text{-}disj}_{n,d,\delta}$ in the $\mathcal{U} = \{0, 1, \ldots, 2^u - 1\}^d$

random string $s \leftarrow \{0,1\}^\kappa$ and, using the bits of s as choice bits to κ instances of oblivious transfer, he learns one share from each of the different sharings: $k_*^{(i)} = k_{s_i}^{(i)}$.

Supppose Bob defines the function

$$F(x) = \mathsf{H}\Big(\mathsf{Eval}(k_*^{(1)}, x), \mathsf{Eval}(k_*^{(2)}, x), \cdots, \mathsf{Eval}(k_*^{(\kappa)}, x)\Big)$$

Bob can compute $F(x)$ for all x, but which values of $F(x)$ can Alice compute? Note that she knows all the FSS shares but does not know Bob's OT choice bits s, which determine the choice of shares used to define F.

If $x \in A$, then the correctness of the bFSS establishes that $\mathsf{Eval}(k_0^{(i)}, x) = \mathsf{Eval}(k_1^{(i)}, x)$. In this case, Bob's OT choice doesn't make a difference — both shares produce the same output. Therefore, Alice can compute $F(x)$ as

$$F(x) = \mathsf{H}\Big(\mathsf{Eval}(k_0^{(1)}, x), \mathsf{Eval}(k_0^{(2)}, x), \cdots, \mathsf{Eval}(k_0^{(\kappa)}, x)\Big) \qquad \text{(for } x \in A)$$

However, if $x \in A$ then $\mathsf{Eval}(k_0^{(i)}, x) \neq \mathsf{Eval}(k_1^{(i)}, x)$ by the properties of a strong bFSS. Intuitively, Bob's OT choice bits make a significant difference on $F(x)$. Alice would have to guess which of $(k_0^{(i)}, k_1^{(i)})$ was chosen by Bob, for *every* i simultaneously, if she is to compute $F(x)$. Equivalently, we can write $\mathsf{Eval}(k_*^{(i)}, x) = \mathsf{Eval}(k_0^{(i)}, x) \oplus s_i$, and hence:

$$F(x) = \mathsf{H}\Big(\Big[\mathsf{Eval}(k_0^{(1)}, x), \mathsf{Eval}(k_0^{(2)}, x), \cdots, \mathsf{Eval}(k_0^{(\kappa)}, x)\Big] \oplus s\Big) \qquad \text{(for } x \notin A)$$

Values of this form are indistinguishable from random when H is a correlation-robust hash (Definition 1) and s is uniform & secret.

In summary, Alice can learn $F(x)$ for $x \in A$, while $F(x)$ looks random for $x \notin A$. We have created an oblivious PRF (OPRF) that Alice can evaluate on a structured set A. We can then obtain a PSI protocol in the usual way [FIPR05], by having Bob send $\{F(b) \mid b \in B\}$ to Alice.

Details. The formal protocol description is given in Figure 8. The warmup protocol above considers only $(1,1)$-bFSS. In the general case, suppose we use a (p,k)-bFSS. The important parameter here is the false-positive probability p. When $x \notin A$, we don't always have $\mathsf{Eval}(k_0^{(i)}, x) \neq \mathsf{Eval}(k_1^{(i)}, x)$ — we have it only with probability at least p.

To adjust for false positives from the bFSS, we simply increase the number of oblivious transfers (and independent bFSS sharings). We need enough bFSS instances to guarantee the following property with overwhelming probability: For all $x \notin A$, at least κ of the bFSS instances correctly satisfy $\mathsf{Eval}(k_0^{(i)}, x) \neq \mathsf{Eval}(k_1^{(i)}, x)$. These κ instances are enough to make $F(x)$ pseudorandom, if the underlying hash function H is now *Hamming* correlation robust (Definition 2).

We emphasize that even though the underlying bFSS may have false positives (with bounded probability), the resulting PSI protocol accounts for this fact and computes the intersection without error. In Appendix C we prove the following:

Theorem 29. *The protocol in Figure 8 securely realizes $\mathcal{F}_{\mathsf{saPSI}}$ (Figure 7) against semi-honest adversaries, when $(\mathsf{Share}, \mathsf{Eval})$ is a secure (p,k)-bFSS and H is Hamming-correlation robust (Definition 2).*

5.1 Costs

Suppose our protocol is instantiated with a certain bFSS whose total share size is σ_{bFSS}, where $t^{\mathsf{Share}}_{\mathsf{bFSS}}$ is the time to share a set (a set from the family $\mathcal{S}$), and where $t^{\mathsf{Eval}}_{\mathsf{bFSS}}$ is the time to evaluate a share on a single input.

The communication cost of the structure-aware PSI protocol is therefore:

- From Alice to Bob: ℓ instances of OTs, in which she transfers a pair of bFSS shares — hence, $\ell \cdot \sigma_{\mathsf{bFSS}}$ bits.
- From Bob to Alice: an output of H for each of Bob's items — hence, $|B| \cdot (\lambda + \log|A| + \log|B|)$ bits.[2]

The computation cost of the protocol is:

- For Alice: she generates ℓ independent bFSS sharings of her set A, so computation $O(\ell \cdot t^{\mathsf{Share}}_{\mathsf{bFSS}})$. She also evaluates each sharing on each of her items in A, but we assume that in most bFSS this can be done as a side-effect of running the Share algorithm. We ignore the insignificant cost of the ℓ OTs.
- For Bob: he evaluates ℓ independent bFSS sharings on each of his items in B, so computation $O(\ell \cdot t^{\mathsf{Eval}}_{\mathsf{bFSS}} \cdot |B|)$.

[2]The presence of a $|B|\log|B|$ term here is deceptive. The protocol would be equally secure if the output of H were κ bits, in which case the length of Bob's message would be $|B|\kappa$ bits. What we have written here is an optimization, observing that shorter output of H is possible, namely $\lambda + \log|A| + \log|B|$ bits. Every PSI protocol that is based on the OPRF paradigm has communication cost of this kind — in order to achieve correctness error bounded by $2^{-\lambda}$, the OPRF outputs that Bob sends to Alice must have length at least $\lambda + \log|A| + \log|B|$.

Parameters: a family of subsets $\mathcal{S} \subseteq 2^{\mathcal{U}}$, over a universe $\mathcal{U}$ of items. A bound n on the cardinality of the unstructured set.

Functionality:

1. Receive input $A \in \mathcal{S}$ (or a concise representation of A) from Alice.
2. Receive input $B \subseteq \mathcal{U}$ of cardinality at most n from Bob.
3. Give output $A \cap B$ to Alice.

Figure 7: Ideal functionality $\mathcal{F}_{\mathsf{saPSI}}$ for structure-aware PSI.

Parameters:

- computational security parameter κ and statistical security parameter λ
- family of sets $\mathcal{S}$ with corresponding (p, k)-bFSS scheme $(\mathsf{Share}, \mathsf{Eval})$
- Hamming-correlation robust hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda + \log|A| + \log|B|}$
- oblivious transfer functionality $\mathcal{F}_{\mathsf{ot}}$
- length ℓ, chosen so that $\Pr[\mathsf{Binomial}(p, \ell) < \kappa] < 1/2^{\lambda + \log|B|}$

Inputs: Alice has (structured) set $A \in \mathcal{S}$ and Bob has (unstructured) set B.

Protocol:

1. Bob chooses a random string $s \leftarrow \{0,1\}^\ell$.
2. Alice generates ℓ independent FSS sharings of her input A: for $i \in [\ell]$ do $(k_0^{(i)}, k_1^{(i)}) \leftarrow \mathsf{Share}(1^\kappa, A)$.
3. The parties invoke ℓ (parallel) instances of oblivious transfer using $\mathcal{F}_{\mathsf{ot}}$. In the ith instance:
   - Alice is the sender with input $(k_0^{(i)}, k_1^{(i)})$.
   - Bob is the receiver with choice bit s_i. He obtains output $k_*^{(i)} = k_{s_i}^{(i)}$.
4. Bob computes the set
$$\widetilde{B} = \left\{ \mathsf{H}\big(b;\ \mathsf{Eval}(k_*^{(1)}, b), \mathsf{Eval}(k_*^{(2)}, b), \cdots, \mathsf{Eval}(k_*^{(\ell)}, b)\big) \;\middle|\; b \in B \right\}$$
   and sends it (randomly shuffled) to Alice.
5. Alice outputs:
$$\left\{ a \in A \;\middle|\; \mathsf{H}\big(a;\ \mathsf{Eval}(k_0^{(1)}, a), \mathsf{Eval}(k_0^{(2)}, a), \cdots, \mathsf{Eval}(k_0^{(\ell)}, a)\big) \in \widetilde{B} \right\}$$

Figure 8: PSI protocol for structured input A and unstructured input B using bFSS.

5.2 Other Protocols as Instances of Our Framework

The PSI protocols of Pinkas *et al.* [PRTY19] (sparse OT) and of Chase-Miao [CM20] are actually instances of our framework, meaning that we can identify an bFSS construction that yields each of those protocols when using that bFSS construction in our protocol framework. Since these PSI protocols support arbitrary sets, their underlying bFSS constructions are for *unstructured*, arbitrary sets — the only "structure" being the cardinality of the sets. If the classic IKNP protocol [IKNP03] protocol is used for PSI (over a small universe of items) in a natural way, it also can be viewed as an instance of our protocol using a trivial bFSS (secret-shared truth table). In Section B we describe these three protocols as specific instances of our approach.

5.3 bFSS Performance

In Figure 11 we summarize the asymptotic costs of different "recipes" to build bFSS for a union of balls, under various assumptions about those balls. We also show a selection of concrete share sizes for certain parameters (ball radius and dimension), in Figure 10.

6 Fuzzy PSI Application and Performance

We demonstrate the practicality of our structure-aware PSI protocol and bFSS constructions by exploring in-depth our original motivating example: fuzzy PSI.

Our protocol requires one party to share the same set many times in a bFSS, and the other party to evaluate all bFSS shares on the same point (many times). In Appendix G we describe how some of our bFSS constructions can be optimized for such batch operations.

6.1 Protocol Selection

Other approaches to Fuzzy PSI, and Parameter Choices. We compare our fuzzy PSI protocol to other competing approaches. First, there is the simple approach (mentioned in the introduction) where Alice expands her set into all points within distance δ, and the parties use a plain PSI protocol to compute the intersection. This approach can be practical when the volume of each ball is very small (*i.e.*, when δ and the dimension d are very small), meaning that Alice's expanded set is only a modest factor larger than the number of balls.

Another approach for fuzzy PSI is to use generic MPC. The parties evaluate a circuit that tests whether a point is inside a ball, for each combination of Alice's balls and Bob's points. Testing whether a point is within a ball is a simple distance calculation with a small circuit. This approach does not depend on the radius δ of the balls, and it can be practical when the number of balls is very small.

We are not aware of a generic-MPC-based approach for fuzzy PSI that uses more sophisticated geometric algorithms (than comparing every ball to every point), which is likely to be practical for fuzzy PSI. Other approaches for circuit-based (plain) PSI have been proposed, including sorting-network-based techniques [HEK12] and hashing techniques [PSWW18]. It is not clear how to apply these ideas to geometric fuzzy PSI. Other approaches to fuzzy PSI may include running a standard range query algorithm inside of an oblivious RAM-based MPC protocol. It remains to be seen whether this approach could be practical.

These alternate approaches to fuzzy PSI scale poorly **when Alice holds a large number of large balls;** hence, this is the range of parameters where our new approach is most attractive, and where we focus much attention in this section. However, the balls cannot be too large in volume, since Alice's computation (not communication) scales with the total volume of all balls. A reasonable range of parameters is to let Alice have on the order of 3000-7000 balls, each with volume on the order of 1600-3600.

Our choice of bFSS. Our bFSS constructions support three different assumptions about the set of input balls.

construction	(p,k)-bFSS	share size	eval cost
disjoint balls:			
spatial hash ∘ sum ∘ tensor ∘ ggm	$(0.5, 1)$	$O(n(4\log\delta)^d\kappa)$	$O((2\log\delta)^d)$
sum ∘ tensor ∘ ggm	$(1, 1)$	$O(n\kappa u^d)$	$O(\boldsymbol{n}u^d)$
ball centers $> 4\delta$ apart:			
spatial hash ∘ concat ∘ ggm	$(1-1/2^d, d)$	$O(nd2^d\kappa\log\delta)$	$O(d\log\delta)$
spatial hash ∘ concat ∘ tt	$(1-1/2^d, d)$	$O(nd2^d\delta)$	0
globally axis-disjoint balls:			
xor share ∘ spatial hash ∘ ggm	$(0.5, 1)$	$O(nd\kappa\log\delta)$	$O(d\log\delta)$
xor share ∘ spatial hash ∘ tt	$(0.5, 1)$	$O(nd\delta)$	0
any arrangement of balls:			
bFSS for unstructured sets	$(0.5, 1)$	$O(n(2\delta)^d)$	0

Figure 9: Asymptotic size of bFSS share for n balls (ℓ_∞ norm) of radius δ in d dimensions, over u-bit integers. Evaluation time for evaluating a share on a single point, measured in number of PRG calls. Keep in mind that each ball consists of $(2\delta)^d$ points.

	δ: 32	32	1024	1024
bFSS construction	d: 2	10	2	10
disjoint balls:				
spatial hash ∘ sum ∘ tensor ∘ ggm	24.3	3.42	0.080	0.00004
sum ∘ tensor ∘ ggm [16 bit ints]	**8.0**	**0.5**	**0.008**	**5E-7**
ball centers $> 4\delta$ apart:				
spatial hash ∘ concat ∘ ggm	0.68	0.001	**0.003**	**6E-9**
spatial hash ∘ concat ∘ tt	**0.17**	**0.0003**	0.005	1E-8
globally axis-disjoint balls:				
xor share ∘ spatial hash ∘ ggm	0.34	0.0002	**0.002**	**8E-10**
xor share ∘ spatial hash ∘ tt	**0.084**	**0.00004**	0.003	1E-9
any arrangement of balls:				
bFSS for unstructured sets (amortized)	1.0	1.0	1.0	1.0

Figure 10: Concrete size of bFSS share for n balls (ℓ_∞ norm) of radius δ in d dimensions, reported in **bits per point.** bFSS for unstructured sets refers to the polynomial-based bFSS implicit in [PRTY19](Figure 13) , which achieves 1 bit per point only when generating many sharings of the same set.

Our construction for arbitrary disjoint balls has reasonable asymptotic cost, but the asymptotic behavior is achieved only for much larger balls than we consider (*e.g.*, δ on the order of 10,000). For δ of medium size, this construction has concrete cost worse than an bFSS for arbitrary sets, which exploits no structure. In other words, for realistic parameters, this bFSS does not lead to fuzzy

PSI that beats the "trivial PSI" approach where Alice simply expands her set — despite being the asymptotically best bFSS for this class of sets to date.

Our construction for the union of balls whose centers have pairwise distance at least 4δ is concretely efficient for medium size δ and dimension d. In our implementation and in this section, **we mostly focus on this bFSS construction.**

Our construction for globally axis-disjoint balls (Theorem 28) is even more efficient. However, we appreciate that the assumption of global axis-disjointness is somewhat artificial. We see that construction more as an exploration of how much structure must be assumed to remove the exponential dependence on dimension.

6.2 Performance Comparison

The main benefit of our fuzzy PSI approach is its low communication. In this section we compare the communication costs of different fuzzy PSI protocols.

As a representative example we consider the case where Alice has a structured set A with 10 million total points, and Bob has an unstructured set B of 1.2 million points. We hold the total cardinality of Alice's set constant and consider two different ways that her points could be arranged into balls:

- 6250 balls of radius $\delta = 20$, in 2 dimensions[3]
- 2778 balls of radius $\delta = 30$, in 2 dimensions

Our protocol. Our protocol is instantiated with computational security parameter $\kappa = 128$ and statistical security parameter $\lambda = 40$. With $d = 2$, our choice of bFSS is a $(0.75, 2)$-bFSS, and we use $\ell = 280$ base OTs so that $\Pr[\mathsf{Binomial}(0.75, \ell) < \kappa]$ is negligible (as specified in Figure 8).

Ignoring the fixed cost of ℓ base OTs, the communication cost of our protocol is as follows. Alice sends ℓ bFSS shares encoding her set, while Bob sends $|B|$ OPRF outputs, each of length $\lambda + \log(|A| \cdot |B|)$. Using the same calculations as in Figure 10, the cost of a single FSS share for $\delta = 20$ is 0.27 bits per item, and for $\delta = 30$ it is 0.18 bits per item. Accounting for all parameters, we obtain communication cost (in bits):[4]

$$
\begin{aligned}
(\delta = 20) \quad & 280 \cdot 0.27|A| + 84|B| = 76|A| + 84|B| \text{ bits} = 108\mathsf{MB} \\
(\delta = 30) \quad & 280 \cdot 0.18|A| + 84|B| = 50|A| + 84|B| \text{ bits} = 75\mathsf{MB}
\end{aligned}
$$

PSI based on Silent OT. As discussed earlier, one approach for fuzzy PSI is for Alice to simply enumerates all points with δ of her set and perform plain PSI on the result. A particularly appealing PSI protocol for this purpose is the VOLE-PSI protocol of Rindal & Schoppmann [RS21], because it has the lowest communication cost (to the best of our knowledge). In the VOLE-PSI protocol, Alice and Bob start by using the silent OT technique [BCGI18, BCG+19b, BCG+19a, SGRR19, CRR21] to obtain an instance of pseudorandom vector OLE. This step requires communication that is sublinear in the size of the parties' sets, and we ignore it for the sake of simplicity.

However, a pseudorandom vector-OLE requires additional communication, so it can be derandomized for the parties' chosen PSI inputs. This derandomization takes the form of an OKVS that Alice sends to Bob, which is proportional to the size of her input set. Alice's total communication is $\rho \cdot \kappa$ bits per item, where ρ is the expansion factor of the particular OKVS (*e.g.*, $\rho = 1.35$ in

[3]Our prototype implementation currently supports only 2 dimensions.

[4]Our actual implementation sends twice this amount of data because we do not optimize the base OTs for the case where one of the OT messages is random, as with bFSS shares.

[GPR+21]). Bob sends an OPRF output of length $\lambda + \log(|A| \cdot |B|)$ bits for each item in his set, similar to our PSI protocol. The total communication cost is therefore:

$$173 \cdot |A| + 84 \cdot |B| \text{ bits} = \mathsf{229MB}$$

Here the sizes of the balls makes no difference – only the total number of points in Alice's set, which we are holding constant. We can see that the dominant factor in the communication cost is the coefficient on $|A|$, which our protocol improves significantly (from 173 to 119 or 80). Also, the PSI sender pays less (overhead of 84 bits) than the receiver (overhead of 173 bits) per items and since Alice's set size if the dominant cost, we can get better communication if Alice is PSI sender. However, our fuzzy protocol only allows for the party with the larger set (structured) to learn the output, so we present the silent OT based PSI cost assuming that Alice is the PSI receiver.

KKRT. Another possibility for solving fuzzy PSI using plain PSI is the KKRT protocol [KKRT16], which is the fastest PSI protocol in the LAN setting. Bob's communication would be same as in our protocol except that he now needs to send 3 hash outputs instead of a single hash output per item. Alice would send ℓ bits per hash table item, where ℓ is the parameter our protocol would use for a $(p = 0.5, k) - \mathsf{bFSS}$ ($\ell \approx 440$ for these input sizes), hash table has expansion factor $\rho = 1.35$ to number of items.[5] For the parameters considered above, this leads to total communication:

$$594 \cdot |A| + 252 \cdot |B| \text{ bits} = \mathsf{780MB}$$

Again, we reiterate that in our protocol we assume the receiver to have the larger input set and acknowledge that we apply this same restriction to KKRT to make our comparison.

6.3 Implementation

General protocol implementation. We implemented our main structure-aware fuzzy PSI protocol (Figure 8) in C++ and our choice of parameters, input sizes and bFSS is as described in the preceding section. Our protocol Figure 8 involves cryptographic components like (1) base oblivious transfers (2) hamming correlation hash function (3) encryption/decryption functionalities and a general communication framework. For the hamming-correlation robust hash function we use SHA256. For base OTs and general framework we use the `libOTe` library of Rindal [Rin]. We implement the bFSS recipe $\mathsf{spatial\ hash} \circ \mathsf{concat} \circ \mathsf{tt}$, for balls of pairwise distance $> 4\delta$, and use the OKVS implementation from [NTY21][6] for spatial hashing. Our implementation will be made available on Github upon publication.

We ran all our experiments on a single Intel Xeon processor at 2.30 GHz with 256 GB RAM over a single thread of execution. We emulate the different network settings using Linux tc command and use the same command to measure the communication cost of the protocol. We emulate a WAN-like setting assuming an average latency of 80 ms and range of bandwidth settings. For restricted network with bandwidth (including internet-like settings) $\{\mathsf{50Mbps}, \mathsf{100Mbps}, \mathsf{150Mbps}, \mathsf{200Mbps}, \mathsf{250Mbps}\}$, our fuzzy PSI protocol has superior performance to KKRT for $\delta = \{20, 30\}$ owing largely to our communication efficiency. The vole-based PSI implementation is not publicly available to make a comparison of our run time. As discussed earlier, we believe that our communication efficiency will yield better run time on restricted bandwidths.

[5]The implementation of KKRT that we used has a large expansion factor, which accounts for the difference between this estimate and the actual communication that we measured.

[6]https://github.com/asu-crypto/mPSI.

protocol	50 Mbps time (s)	100 Mbps time (s)	150 Mbps time (s)	200 Mbps time (s)	250 Mbps time (s)	comm (MB)
KKRT [KKRT16]	218.409	131.057	101.312	81.511	75.092	1184.224
ours, $\delta = 20$ (6250 balls)	**115.49**	**87.593**	**78.404**	**73.358**	**72.27**	**319.314**
ours, $\delta = 30$ (2778 balls)	**61.675**	**48.904**	**43.573**	**42.107**	**41.198**	**156.878**

Figure 11: Run time (in seconds) and communication (in MB) comparison of our fuzzy PSI protocol with input set sizes $|A| = 10M$ and $|B| = 1.2M$ where A consists of 2-dimensional ℓ_∞ balls We instantiate our protocol in Figure 8 with bFSS recipe $\mathsf{spatial\ hash} \circ \mathsf{concat} \circ \mathsf{tt}$, for balls with centers $> 4\delta$ apart. We simulated a network with latency 80ms and the given bandwidth cap.

7 Limitation and Open Problems

The proposed OPRF-PSI framework is designed in the semi-honest model, and the malicious setting is left for future work.

In our structure-aware PSI protocol, only one input set is structured and it does not attempt to take advantage of any structure in Bob's set B. The sender in our protocol sends OPRF values for each element in its set - hence a natural question is whether we can exploit some structure in Bob's input to send this set of OPRF values more efficiently.

Alice's computation cost is still $O(|A|)$, despite A having a more concise representation. This is because she must enumerate OPRF outputs for each $a \in A$, in order to recognize them in Bob's PSI message. In order for Alice's computation to scale with the description size of A, she would need a way to efficiently "recognize" OPRF outputs that she is entitled to learn, but without explicitly enumerating them. We leave it to future work to explore how to make this possible. For now, supporting exponentially large A remains an important open problem.

Our techniques can be used to get weak bFSS for union of balls in ℓ_2 metric space by approximating each ball by a polyhedron with sufficiently good isoperimetric quotient (which is a measure of how close a shape is to a sphere). However, its still unknown if we can get symmteric key based weak bFSS for an **exact** ℓ_2 ball, which is more suited for fuzzy PSI related applications. Given a weak bFSS for a single ℓ_2 ball, we can use our $\mathsf{spatial\text{-}hashing}$ technique to get an efficient weak bFSS for a union of balls in ℓ_2 metric space.

References

[ADT11] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) size matters: Size-hiding private set intersection. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 156–173. Springer, Heidelberg, March 2011.

[AKB07] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *NDSS*, volume 7, pages 43–54, 2007.

[BCG$^+$19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

[BCG$^+$19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019.

[BCG$^+$21] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 871–900. Springer, Heidelberg, October 2021.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[BCRT16] Justin Bedő, Thomas Conway, Kim Ramchen, and Vanessa Teague. Privately matching k-mers. Cryptology ePrint Archive, Report 2016/781, 2016. https://eprint.iacr.org/2016/781.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, Heidelberg, August 1995.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 337–367. Springer, Heidelberg, April 2015.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CFR21] Anrin Chakraborti, Giulia Fanti, and Michael K. Reiter. Distance-aware private set intersection, 2021.

[CH08] Lukasz Chmielewski and Jaap-Henk Hoepman. Fuzzy private matching. In *2008 Third International Conference on Availability, Reliability and Security*, pages 327–334. IEEE, 2008.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018.

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Heidelberg, August 2020.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Heidelberg.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 789–800. ACM Press, November 2013.

[DMRY11] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Secure efficient multiparty computing of multivariate polynomials and applications. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 130–146. Springer, Heidelberg, June 2011.

[Dou07] J.M. Doumen. Non-interactive fuzzy private matching. WorkingPaper TR-CTIT-07-45, Centre for Telematics and Information Technology (CTIT), Netherlands, June 2007.

[DT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, Heidelberg, January 2010.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 303–324. Springer, Heidelberg, February 2005.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

[GPR$^+$21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 395–425, Virtual Event, August 2021. Springer, Heidelberg.

[GS19] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 3–29. Springer, Heidelberg, August 2019.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

[HFH99] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *ACM CONFERENCE ON ELECTRONIC COMMERCE*. ACM, 1999.

[HOS17] Per A. Hallgren, Claudio Orlandi, and Andrei Sabelfeld. PrivatePool: Privacy-preserving ridesharing. In Boris Köpf and Steve Chong, editors, *CSF 2017 Computer Security Foundations Symposium*, pages 276–291. IEEE Computer Society Press, 2017.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[IW06] Piotr Indyk and David P. Woodruff. Polylogarithmic private approximations and efficient matching. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 245–264. Springer, Heidelberg, March 2006.

[JL10] Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 418–435. Springer, Heidelberg, September 2010.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 54–70. Springer, Heidelberg, August 2013.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[KLS$^+$17] Ágnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. Private set intersection for unequal set sizes with mobile applications. *PoPETs*, 2017(4):177–197, October 2017.

[KRS$^+$19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1447–1464. USENIX Association, August 2019.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005.

[NTY21] Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection. Cryptology ePrint Archive, Report 2021/1221, 2021. `https://eprint.iacr.org/2021/1221`.

[PIRC21] Bijeeta Pal, Mazharul Islam, Thomas Ristenpart, and Rahul Chatterjee. Might i get pwned: A second generation password breach alerting service, 2021.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 515–530. USENIX Association, August 2015.

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 125–157. Springer, Heidelberg, April / May 2018.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 797–812, 2014.

[RA18] Amanda C. Davi Resende and Diego F. Aranha. Faster unbalanced private set intersection. In Sarah Meiklejohn and Kazue Sako, editors, *FC 2018*, volume 10957 of *LNCS*, pages 203–221. Springer, Heidelberg, February / March 2018.

[Rin] Peter Rindal. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe.

[RR17a] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 235–259. Springer, Heidelberg, April / May 2017.

[RR17b] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017.

[RS21] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Heidelberg, October 2021.

[SGRR19] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1055–1072. ACM Press, November 2019.

[SSNO12] Siamak F Shahandashti, Reihaneh Safavi-Naini, and Philip Ogunbona. Private fingerprint matching. In *Australasian Conference on Information Security and Privacy*, pages 426–433. Springer, 2012.

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

[WG14] Yamin Wen and Zheng Gong. Private mutual authentications with fuzzy matching. *International Journal of High Performance Systems Architecture*, 5(1):3–12, 2014.

[WXL+18] Xu An Wang, Fatos Xhafa, Xiaoshuang Luo, Shuaiwei Zhang, and Yong Ding. A privacy-preserving fuzzy interest matching protocol for friends finding in social networks. *Soft Computing*, 22(8):2517–2526, 2018.

[YSPW10] Qingsong Ye, Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. Efficient fuzzy matching and intersection on private datasets. In Donghoon Lee and Seokhie Hong, editors, *ICISC 09*, volume 5984 of *LNCS*, pages 211–228. Springer, Heidelberg, December 2010.

[ZCL21] En Zhang, Jian Chang, and Yu Li. Efficient threshold private set intersection. *IEEE Access*, 9:6560–6570, 2021.

A OKVS Preliminaries

Lemma 30. *If* $(\mathsf{Encode}, \mathsf{Decode})$ *is a boolean OKVS, then it satisfies the independence property of Definition 8.*

Proof. In a boolean OKVS, the output $D = \mathsf{Encode}(A)$ is a vector satisfying the set of linear constraints $\langle \pi(k), D\rangle = v$ for all $(k, v) \in A$. Correctness of the OKVS requires the set of $\pi(k)$ vectors to be linearly independent. Obliviousness of a boolean OKVS requires that Encode selects a *random* solution from the subspace of all solutions to this set of linear constraints,

With overwhelming probability, the additional vector $\pi(k^*)$ is also linearly independent of the set of $\pi(k)$ values. But for a random solution to the equation $M \times \boldsymbol{x} = \boldsymbol{t}$, and a vector $\boldsymbol{v}$ which is not in the rowspace of M, the value $\langle \boldsymbol{v}, \boldsymbol{x}\rangle$ is uniformly distributed (independent of $\boldsymbol{t}$. $\square$

B Other Protocols as Instances of Our Framework

In this section we describe how three existing (standard) PSI protocols can be obtained by instantiating our structure-aware PSI framework with a suitable (weak) bFSS. Since these PSI protocols support arbitrary sets, these bFSS constructions support unstructured arbitrary sets — the only restriction being either the domain or the cardinality of the sets.

IKNP-based PSI. Suppose parties have inputs from a small domain $\mathcal{U} = [n]$, where n is polynomial in the security parameter. Suppose the parties instantiate n instances of oblivious transfer on random payloads. In the ith instance, the sender has input strings $(r_0^{(i)}, r_1^{(i)})$, and the receiver with choice bit c_i gets output $r_{c_i}^{(i)}$. The sender can define a function $F(i) = r_0^{(i)}$, which clearly the receiver learns whenever $c_i = 0$. The function F is then a random function defined on the domain $[n]$, and the protocol is an OPRF where the receiver can learn the output of F on any subset of its choice. Using this OPRF, the parties can obtain PSI in the usual way.

When the parties instantiate the many instances of OT using IKNP OT extension [IKNP03], we obtain a special case of our protocol. The underlying bFSS is the trivial one — an additive secret sharing of the set's truth table (inverted to give result 0 when the item is in the set, as required for our bFSS definition). This is a strong $(1, 1)$-bFSS. The details are given in Figure 12. One party's share is a PRG key while one is a string of length n. In IKNP, the longer share k_1 is transfered via OT by sending a seed s over OT and then sending $\mathsf{G}(s) \oplus k_1$ in the clear, where G is a PRG. With these simplifications, this IKNP-based PSI protocol collapses exactly to our protocol instantiated with the bFSS.

Sparse-OT. Pinkas *et al.* [PRTY19] proposed a PSI protocol based on a technique called sparse OT extension. Their protocol is essentially our bFSS-based protocol, instantiated with the $(0.5, 1)$-bFSS shown in Figure 13. This bFSS supports arbitrary subsets of a large field $\mathbb{F}$, of cardinality n. One party's share is a PRF seed, and the others is a polynomial over $\mathbb{F}$ of degree less than n. The polynomial is chosen so that it agrees with the PRF share evaluation on points in the set. The security of the PRF ensures that for points not in the set, the two evaluations are an additive sharing of a random bit, making the construction a $(0.5, 1)$-bFSS.

The sparse OT protocol deviates from our framework in one small aspect. We defined the bFSS scheme so that one party's share is a polynomial with coefficients in $\mathbb{F}$, but that polynomial is only used for a single bit of output. In sparse OT as well as our structure-aware PSI framework, the receiver generates many independent sharings of the same set. Hence, the sparse OT protocol uses a single polynomial over $\mathbb{F}$ and interpret each of its output bits as an independent bFSS.

Chase-Miao PSI. Chase & Miao [CM20] proposed an elegant extension of the IKNP protocol. Their protocol can be obtained by instantiating our structure-aware PSI framework with an bFSS that is an *additive sharing of a bloom filter encoding* of the set. Notably, a bloom filter has only one-sided error (false positives only), making it a good fit for our bFSS definition. Only a constant bound on false-positive probability is needed, which can be achieved by a *bloom filter with just a single hash function* and size $O(n)$ to encode a set of cardinality n. As above, the bloom filter is inverted to give result 0 when the item is in the set, and one party's share can be compressed to a PRG seed. Details are given in Figure 14.

C Proofs for Structure-Aware PSI Protocol

Theorem 29. *The protocol in Figure 8 securely realizes $\mathcal{F}_{\mathsf{saPSI}}$ (Figure 7) against semi-honest adversaries, when (Share, Eval) is a secure (p, k)-bFSS and H is Hamming-correlation robust (Definition 2).*

Proof. (Case of corrupt Bob) Bob's view consists only of his OT outputs, the $k_*^{(i)}$ values. Each $k_*^{(i)}$ is one share of an independent bFSS sharing of Alice's input set. By the security of bFSS, Bob's view is indistinguishable from a collection of *simulated* bFSS shares, simulated with no information about Alice's input (apart from the fact that her set is in the family $\mathcal{S}$).

(Case of corrupt Alice) Alice's view consists only of Bob's protocol message $\widetilde{B}$. We prove security by showing a sequence of hybrids that define how $\widetilde{B}$ is generated. The simulator will be defined in the final hybrid.

Hybrid 0. In this hybrid, $\widetilde{B}$ is generated as in the honest protocol (using Bob's private input B). To help in the proof, we rewrite how $\widetilde{B}$ is generated, in an equivalent way.

Fix $k_j^{(i)}$ and $k_*^{(i)}$ as in the protocol, and define the following functions:

$$\begin{aligned}
\boldsymbol{E}_*(b) &= \big(\mathsf{Eval}(k_*^{(1)}, b), \quad \ldots, \mathsf{Eval}(k_*^{(\ell)}, b) \big)\\
\boldsymbol{E}_0(b) &= \big(\mathsf{Eval}(k_0^{(1)}, b), \quad \ldots, \mathsf{Eval}(k_0^{(\ell)}, b) \big)\\
\boldsymbol{\Delta}(b) &= \big(\mathsf{Eval}(k_1^{(1)}, b) \oplus \mathsf{Eval}(k_0^{(1)}, b), \quad \ldots, \mathsf{Eval}(k_1^{(\ell)}, b) \oplus \mathsf{Eval}(k_0^{(\ell)}, b) \big)
\end{aligned}$$

```
Parameters:
$\mathcal{U} = [n] = \{1, \ldots, n\}$
$\mathcal{S} = 2^{\mathcal{U}}$
PRG $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^n$

$\underline{\mathsf{Share}(S \subseteq [n]):}$
  $T = 1^n$
  for $x \in S$:
    $T[x] = 0$
  $k_0 \leftarrow \{0,1\}^\kappa$
  $k_1 = \mathsf{G}(k_0) \oplus T$
  return $(k_0, k_1)$

$\underline{\mathsf{Eval}(0, k_0, x):}$
  return $\mathsf{G}(k_0)[x]$

$\underline{\mathsf{Eval}(1, k_1, x):}$
  return $k_1[x]$
```

Figure 12: Trivial $(1, 1)$-bFSS implicit in IKNP-based PSI.

```
Parameters:
$\mathcal{U} = \mathbb{F}$, a field
$\mathcal{S} = \{S \subseteq \mathbb{F} : |S| = n\}$
PRF $\mathsf{F} : \{0,1\}^\kappa \times \mathbb{F} \to \{0,1\}$

$\underline{\mathsf{Share}(S \subseteq \mathbb{F}):}$
  $k_0 \leftarrow \{0,1\}^\kappa$
  $k_1$ = interpolate polynomial
      passing through points
      $\Big\{(x, \mathsf{F}(k_0, x)) \;\Big|\; x \in S\Big\}$
  return $(k_0, k_1)$

$\underline{\mathsf{Eval}(0, k_0, x):}$
  return $\mathsf{F}(k_0, x)$

$\underline{\mathsf{Eval}(1, k_1, x):}$
  return $k_1(x)$
```

Figure 13: $(0.5, 1)$-bFSS implicit in sparse-OT PSI [PRTY19].

```
Parameters:
$\mathcal{U} = \{0,1\}^*$
$\mathcal{S} = \{S \subseteq \{0,1\}^* : |S| = n\}$
bloom filter length $m$
random function $h : \{0,1\}^* \to [m]$
PRG $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^m$

$\underline{\mathsf{Share}(S \subseteq \{0,1\}^*):}$
  $B = 1^m$
  for $x \in S$:
    $B[h(x)] = 0$
  $k_0 \leftarrow \{0,1\}^\kappa$
  $k_1 = \mathsf{G}(k_0) \oplus B$
  return $(k_0, k_1)$

$\underline{\mathsf{Eval}(0, k_0, x):}$
  return $\mathsf{G}(k_0)[h(x)]$

$\underline{\mathsf{Eval}(1, k_1, x):}$
  return $k_1[h(x)]$
```

Figure 14: $(p, 1)$-bFSS implicit in Chase-Miao PSI [CM20], where $1-p$ is the false positive probability of a single-hash-function bloom filter of size m encoding n items. Hence $p = (1 - \frac{1}{m})^n \approx e^{-n/m}$.

Each function's output is a vector of length ℓ whose entries are from $\{0,1\}^k$ (where k here is the output length of Eval). Using this notation, Bob's protocol message can be written as:

$$\begin{aligned}
\widetilde{B} &\overset{\text{def}}{=} \Big\{\mathsf{H}\big(b;\; \mathsf{Eval}(k_*^{(1)}, b), \mathsf{Eval}(k_*^{(2)}, b), \cdots, \mathsf{Eval}(k_*^{(\ell)}, b)\,\big) \;\Big|\; b \in B\Big\} \\
&= \Big\{\mathsf{H}\big(b; \boldsymbol{E}_*(b)\big) \;\Big|\; b \in B\Big\} \\
&= \Big\{\mathsf{H}\big(b; \boldsymbol{E}_0(b) \oplus s \odot \boldsymbol{\Delta}(b)\big) \;\Big|\; b \in B\Big\} && (1) \\
&= \Big\{\mathsf{H}\big(b; \boldsymbol{E}_0(b)\big) \;\Big|\; b \in A \cap B\Big\} \cup \Big\{\mathsf{H}\big(b; \boldsymbol{E}_0(b) \oplus s \odot \boldsymbol{\Delta}(b)\big) \;\Big|\; b \in B \setminus A\Big\} && (2)
\end{aligned}$$

In step (1) the $\odot$ refers to componentwise multiplication (a single bit times a k-bit string). Step (1) follows by rewriting

$$\mathsf{Eval}(k_*^{(i)}, b) = \mathsf{Eval}(k_{s_i}^{(i)}, b) = \mathsf{Eval}(k_0^{(i)}, b) \oplus s_i\Big(\mathsf{Eval}(k_1^{(i)}, b) \oplus \mathsf{Eval}(k_0^{(i)}, b)\Big)$$

in each component. Step (2) follows by observing that if $b \in A$ then $\boldsymbol{\Delta}(b) = \mathbf{0}$ by correctness of the bFSS scheme.

Hybrid 1. This hybrid is identical to the previous one, except we simply abort (*i.e.*, set $\widetilde{B} = \bot$) if there exists any $b \in B \setminus A$ whose $\mathbf{\Delta}(b)$ has fewer than κ nonzero components.

From the correctness of the (p, k)-bFSS, each component of $\mathbf{\Delta}(b)$ is nonzero with independent probability at least p. Since $\mathbf{\Delta}(b)$ has length ℓ, the probability of an abort is distributed exactly as the event that $\mathsf{Binomial}(p, \ell) < \kappa$. Recall that ℓ is chosen so that this event happens with probability at most $2^{-(\lambda + \log|B|)}$. By a union bound over the choices of $b \in B \setminus A$, the probability of an abort in this hybrid is at most $2^{-\lambda}$. Since the hybrids are identical except for this artificial abort, the hybrids are indistinguishable if $2^{-\lambda}$ is negligible.

Hybrid 2: In this hybrid, $\widetilde{B}$ is generated as:

$$\widetilde{B} = \Big\{ \mathsf{H}\big(b; \boldsymbol{E}_b(b)\big) \;\Big|\; b \in A \cap B \Big\} \cup \Big\{ h_1, \ldots, h_{|B \setminus A|} \Big\} \tag{3}$$

where each h_i is uniform in $\{0,1\}^{\lambda + \log|A| + \log|B|}$. The hybrid differs from Hybrid 1 by replacing $\mathsf{H}(b; \boldsymbol{E}_0(b) \oplus s \odot \mathbf{\Delta}(b))$ with a uniformly random value, for each $b \in B \setminus A$.

Conditioned on the event that Hybrid 1 doesn't abort, each $\mathbf{\Delta}(b)$ in these expressions is nonzero in at least κ positions. This matches the situation described in the definition of Hamming correlation robustness Definition 2 — namely, we have expressions of the form $H(b_i; e_i \oplus s \odot \Delta_i)$ where each Δ_i is nonzero in at least κ positions and the b_i's are distinct. By Definition 2, such values are indistinguishable from random, so the hybrids are indistinguishable.

Simulator. Note that Hybrid 2 defines a valid simulation in the ideal world. In other words, $\widetilde{B}$ can be generated as in (3) given only the information available to the simulator: A and $A \cap B$ and $|B|$ (from which $|B \setminus A|$ can be easily deduced). □

Lemma 31. *The protocol in Figure 8 is correct.*

Proof. Since Alice is the only party with output, and since the simulation for a corrupt (semi-honest) Alice is indistinguishable from her real view, it suffices to show correctness when Alice sees the *simulated* view.

Suppose $a \in A \cap B$, then $\mathbf{\Delta}(a) = \mathbf{0}$ and the simulator (on behalf of Bob) will include

$$\mathsf{H}(a; \boldsymbol{E}_*(a)) = \mathsf{H}(a; \boldsymbol{E}_0(a) \oplus s \odot \mathbf{\Delta}(a)) = \mathsf{H}(a; \boldsymbol{E}_0(a))$$

in the set $\widetilde{B}$. Later, Alice will check whether $\mathsf{H}(a; \boldsymbol{E}_0(a)) \in \widetilde{B}$, and, finding it there, she will correctly include a in her output.

For $a \in A \setminus B$, Alice will wrongly include a in her output if by chance $\mathsf{H}(a; \boldsymbol{E}_0(a)) \in \widetilde{B}$. The elements of $\widetilde{B}$ are either outputs of H (with first argument other than a, since $a \notin A \cap B$) or uniformly random values chosen by the simulator. In either case, the probability that a particular $\mathsf{H}(a; \boldsymbol{E}_0(a))$ equals a particular element of $\widetilde{B}$ is at most $2^{-\lambda - \log|A| - \log|B|}$, since these are strings of length $\lambda + \log|A| + \log|B|$. By a union bound over all choices of $a \in A$ and all values in $\widetilde{B}$ (of which there are $|B|$), the total probability of Alice including an erroneous value in her output is at most $2^{-\lambda}$. □

D Function Secret Sharing

D.1 Original formuation of FSS

Definition 32 (FSS syntax)**.** *A two party function secret sharing scheme for a class of functions $\mathcal{F}$ with input domain $\{0,1\}^n$ (where $n \in \mathcal{N}$) and co-domain $\{0,1\}^m$ consists of a pair of algorithms (Share, Eval) and a security parameter κ with the following syntax:*

- $(k_0.k_1) \leftarrow \mathsf{Share}(1^\kappa, \hat{f})$ - *The randomized share function takes as input the security parameter κ and the function description $\hat{f}$ for some function $f \in \mathcal{F}$, and it outputs two keys, representing shares of the function f.*
- $y_p \leftarrow \mathsf{Eval}(1^\kappa, \mathsf{idx}, k_{\mathsf{idx}}, x)$ - *The deterministic evaluation function takes as input the security parameter, party index $\mathsf{idx} \in \{0,1\}$, the corresponding FSS key k_{idx} and the input $x \in \{0,1\}^n$, and it outputs $y_{\mathsf{idx}} \in \{0,1\}^m$.*

Definition 33 (FSS security)**.** *A 2 party FSS scheme ($\mathsf{Share}, \mathsf{Eval}$) for the class of functions $\mathcal{F}_n$ is termed secure if is satisfied the following conditions:*

- *Correctness: For every function $f \in \mathcal{F}_n$, $x \in \{0,1\}^n$ and security parameter κ we have:*

$$Pr\left(y_0 \oplus y_1 = f(x) \;\middle|\; \begin{array}{l} y_{\mathsf{idx}} \leftarrow \mathsf{Eval}(1^\kappa, \mathsf{idx}, k_{\mathsf{idx}}, x) \text{ for } \mathsf{idx} \in \{0,1\} \\ (k_0, k_1) \leftarrow \mathsf{Share}(1^\kappa, \hat{f}) \text{ for } f \in \mathcal{F} \end{array} \right) = 1$$

- *Privacy: For every function $f \in \mathcal{F}_n$, $x \in \{0,1\}^n$, $\mathsf{idx} \in \{0,1\}$ and security parameter κ there exists a simulator Sim:*

$$k_{\mathsf{idx}} \cong_\kappa \{\mathsf{Sim}(1^\kappa, \mathsf{idx})\}$$

  where $\cong_\kappa$ denotes computational indistinguishability with respect to security parameter κ.

E General FSS construction rules

E.1 Complement

Given a $(1,1)$-bFSS for a collection of sets $\mathcal{S}$ we can get an efficient bFSS for the compliment of sets contained in $\mathcal{S}$ by just complementing the output of Eval for one of the parties. For this collection of sets $\overline{\mathcal{S}} = \{\overline{S} | S \in \mathcal{S}\}$ the bFSS construction is formally presented in Figure 15.

E.2 Disjoint Union (sum)

For any n, let the sets $S_1, S_2, \ldots, S_n$ be disjoint sets over some universe $\mathcal{U}$. Define $S = \cup_i S_i$ Then we can construct the indicator function for the compliment of $\bar{S}$ in terms of indicator functions for sets S_i as follows

$$I_{\bar{S}} = I_{\bar{S}_1} \oplus I_{\bar{S}_2} \oplus \ldots \oplus I_{\bar{S}_n}$$

This is true since for each $x \in S$, there exists a unique $i \in [1, \ldots, n]$ such that $x \in S_i$. Which implies the $I_{\bar{S}_j}(x) = 0$ for all $j \neq i$ and $I_{\bar{S}_i}(x) = 1$. And for $x \notin S$, we have $I_{\bar{S}_j}(x) = 0$ for all j - this proves the correctness of the indicator function for $\bar{S}$.

Hence given strong bFSS for disjoint sets $S_1, S_2, \ldots, S_n$ we can construct a strong bFSS for their union by simply adding the bFSS output for the compliment of these sets and then complimenting the output of one party's Eval. Hence this technique is also referred to as the sum construction. The protocol is formally described in Figure 16. The key size for the disjoint union bFSS is the sum of the key size of individual sets. Similarly the Eval complexity would be the sum of Eval cost for each of the n sets.

E.3 Single bit bFSS output

Given a (p,k)-bFSS for a collection of sets $\mathcal{S}$ we can get a $(p/2,1)$-bFSS for the same collection by just applying a pair-wise independent hash function with output space $\{0,1\}$ on the bFSS output. A formal description of the scheme is presented in Figure 17.

```
$\underline{\mathsf{Share}^{\bar{\mathcal{S}}}(1^\kappa, S)}$:
    return $\mathsf{Share}^{\mathcal{S}}(1^\kappa, \bar{S})$
$\underline{\mathsf{Eval}^{\bar{\mathcal{S}}}(1^\kappa,\mathsf{idx},\mathsf{FSSkey},x)}$:
    return $\mathsf{idx} \oplus \mathsf{Eval}^{\mathcal{S}}(1^\kappa,\mathsf{idx},\mathsf{FSSkey},x)$
```

Figure 15: $(1, 1)$-bFSS for input set S given a $(1, 1)$-bFSS for the its compliment $\bar{S}$

```
$\underline{\mathsf{Share}_n^{\mathcal{S}}(1^\kappa, \cup_{i=1}^n S_i))}$:
    Initialize $k_0, k_1$ as empty associated arrays
    for $i \in [1, n]$ :
        $(k_0[i], k_1[i]) \leftarrow \mathsf{Share}^{\bar{\mathcal{S}}_i}(1^\kappa, \bar{S}_i)$
    return $(k_0, k_1)$
$\underline{\mathsf{Eval}_n^{\mathcal{S}}(1^\kappa,\mathsf{idx},\mathsf{FSSkey},x)}$:
    $y \leftarrow 0$
    for $i \in [1, n]$ :
        $y \leftarrow y \oplus \mathsf{Eval}^{\bar{\mathcal{S}}_i}(1^\kappa,\mathsf{idx},\mathsf{FSSkey}[i],x)$
    return $y \oplus \mathsf{idx}$
```

Figure 16: $(1, 1)$-bFSS for union of disjoint sets $\{S_i | i \in [1, n]\}$ given a $(1, 1)$-bFSS for each individual set

Theorem 34. $(\mathsf{Share}_1, \mathsf{Eval}_1)$ *is a* $(p/2, 1)$*-bFSS for the collection of sets* $\mathcal{S}$ *given* $(\mathsf{Share}, \mathsf{Eval})$ *is a* (p, k)*-bFSS for the same collection of sets*

Proof. Privacy follows directly, so here we focus on just the bFSS correctness proof:

- For any $x \in S$: we have $\mathsf{Eval}^{\mathcal{S}}(1^\kappa,0,\mathsf{FSSkey},x) = \mathsf{Eval}^{\mathcal{S}}(1^\kappa,1,\mathsf{FSSkey},x)$ for any key FSSkey. Hence the outputs of Eval_1) scheme will match on all inputs in S as well
- For $x \notin S$: With probability at least p, $u = \mathsf{Eval}^{\mathcal{S}}(1^\kappa,0,\mathsf{FSSkey},x) \neq \mathsf{Eval}^{\mathcal{S}}(1^\kappa,1,\mathsf{FSSkey},x) = v$. By the property of the pairwise independent hash function, we have $Pr(H(u) = H(v)) = 1/2$. Hence the outputs of Eval_1 for the two parties will disagree with at least probability $p/2$.

□

F Proofs for bFSS constructions

Theorem 9. *Any strong bFSS* F *for a collection of sets* $\mathcal{S}$ *in the universe* $\mathcal{U}$ *with pseudo-random keys satisfies the PRF property (Definition 6).*

```
Let $H$ be a pairwise independent hash function without space $\{0, 1\}$.
$\underline{\mathsf{Share}_1^{\mathcal{S}}(1^\kappa, S)}$:
    return $\mathsf{Share}^{\mathcal{S}}(1^\kappa, S)$
$\underline{\mathsf{Eval}_1^{\mathcal{S}}(1^\kappa,\mathsf{idx},\mathsf{FSSkey},x)}$:
    return $H(\mathsf{Eval}^{\mathcal{S}}(1^\kappa,\mathsf{idx},\mathsf{FSSkey},x))$
```

Figure 17: $(p/2, 1)$-bFSS for input set S given a (p, k)-bFSS for the same set S

Proof Sketch. Lets assume for some $x \in \mathcal{U}$, $\mathsf{Eval}(0, k, x)$ is not a uniformly bit, where key k is sampled uniformly random from the key space. Then $\mathsf{Eval}(0, k_0, x)$ is also not uniformly random when k_0 comes from a legitimate share (i.e. $(k_0, k_1) \leftarrow F.Share(S)$ and $S \in \mathcal{S}$), and its bias is public knowledge. Then $\mathsf{Eval}(1, k_1, x)$ is also not indistinguishable from a random bit, and by itself is biased towards the indicator function output which checks if $x \in S$? For e.g. if $x \notin S$, then $\mathsf{Eval}(0, k_0, x)$ and $\mathsf{Eval}(1, k_1, x)$ have the same distributions, while otherwise their probabilities for output bits 0 and 1 are reversed. Hence one can distinguish a share of S from a share of $S' \in \mathcal{S}$ such that precisely one of S or S' contain x. □

Theorem 21. *Let $\mathcal{S}$ be a family of sets over universe $\mathcal{U} = \{0, \ldots, 2^u - 1\}^d$. Let δ be an arbitrary integer representing the grid size. Define $\mathcal{S}_\delta = \{\mathit{ShiftOrigin}(S \cap \mathit{cell}_\delta^{-1}(\boldsymbol{x}), -\boldsymbol{x}) \mid S \in \mathcal{S}, C(\boldsymbol{x}) \in G(\delta, u, d)\}$.*

If GridFSS is a (p, k)-bFSS for $\mathcal{S}_\delta$ with pseudo-random keys and satisfying the PRF property with share size σ, then spatial-hashing$_{\delta,d}[F]$ is a $(\min\{1 - 2^{-k}, p\}, k)$-bFSS for $\mathcal{S}$ with share size $O(\mathit{MaxActiveCellCount}(\delta, \mathcal{S}) \cdot \sigma)$

Proof. Correctness:

Case 1 : if $\boldsymbol{x} \in S$: In this case $\mathsf{Decode}(k_0, \mathsf{cell}(\boldsymbol{x}))$ and $\mathsf{Decode}(k_1, \mathsf{cell}(\boldsymbol{x}))$ output two GridFSS keys respectively for the grid cell $\mathsf{cell}(\boldsymbol{x})$, which were inserted in the OKVS in the Share algorithm. By correctness of GridFSS we have $\mathsf{Decode}(k_0, \mathsf{cell}(\boldsymbol{x})) \oplus \mathsf{Decode}(k_1, \mathsf{cell}(\boldsymbol{x})) = 0^k$.

Case 2.1 : if $\boldsymbol{x} \notin S$ and $\mathsf{cell}(\boldsymbol{x}) \in$ ActiveCells: Similar to the previous case, $\mathsf{Decode}(k_0, \mathsf{cell}(\boldsymbol{x}))$ and $\mathsf{Decode}(k_1, \mathsf{cell}(\boldsymbol{x}))$ output two GridFSS keys respectively for the grid cell $\mathsf{cell}()$. GridFSS is a (p, k)-bFSS, hence $\mathsf{Decode}(k_0, \mathsf{cell}(\boldsymbol{x})) \oplus \mathsf{Decode}(k_1, \mathsf{cell}(\boldsymbol{x})) \neq 0^k$ with probability at least p.

Case 2.2 : if $\boldsymbol{x} \notin S$ and $\boldsymbol{x} \notin$ ActiveCells:
Let $C = \mathsf{cell}(\boldsymbol{x})$, $k_0^* \leftarrow \mathsf{Decode}(k_0, C)$ and $k_1^* \leftarrow \mathsf{Decode}(k_1, C)$.
By the independence property of OKVS, we have k_0^*, k_1^* are pair of random and independent strings. Next we show $Pr(F.\mathsf{Eval}(k_0^*, 0, \boldsymbol{x}) \oplus F.\mathsf{Eval}(k_1^*, 1, \boldsymbol{x}) = 0) = 1/2$. By the PRF property of the GridFSS F we have $F.\mathsf{Eval}(k_0^*, 0, *)$ is a pseudorandom string. Correctness follows.

Security: The OKVS values encoded in both k_0, k_1 are indistinguishable from random uniform strings of length σ - since the keys of GridFSS are pseudo-random. Let $\mathcal{R}$ be the function defined OKVS Definition 7. Then $k_0 \leftarrow \mathcal{R}(u_1, u_2, \ldots, u_m)$ where u_i are picked based on the structure of the input set S, and $m =$ MaxActiveCellCount. Then by the obliviousness property of the OKVS, we have k_0 is indistinguishable from $\mathcal{R}(1, 2, \ldots, m)$ - which would be the output of our simulator for $\mathsf{idx} = 0$. Similarly one can show that the same output will suffice for the case of $\mathsf{idx} = 1$ as well. □

Theorem 28. *The (Share, Eval) Figure 6 is a (0.5, 1)-bFSS for collection of sets union-glob-disj$_{u,d,\delta,n}$ in $\mathcal{U} = \{0, \ldots, 2^u - 1\}^d$, with key size $O(nd \log \delta)$ bits and the evaluation cost being dominated by $O(d \log \delta)$ calls to a PRG.*

Proof. Correctness: If S is the secret shared union of disjoint balls and $\boldsymbol{x} \in \{0, 1, \ldots, 2^u - 1\}^d$, we have:

Case 1 : if $\boldsymbol{x} \in S$: In this case $\mathsf{Decode}(k_0[j], x_j)$ and $\mathsf{Decode}(k_1[j], x_j)$ output the two GridFSS $(1, d)$-bFSS keys respectively for the dimension j. If $\boldsymbol{x} \in S_i$, by correctness of GridFSS we have $F.\mathsf{Eval}(0, \mathsf{Decode}(k_0[j], x_j), x_j) \oplus F.\mathsf{Eval}(1, \mathsf{Decode}(k_0[j], x_j).x_j) = R[i, j]$. Hence the xor of the Eval outputs $\oplus_{j=1}^{d} R[i, j] = 0$.

Case 2 : if $\boldsymbol{x} \notin S$ and for each $j = 1, \ldots, d$, $\pi_j(\boldsymbol{x}) \in \pi_j(S)$: For $j = 1, \ldots, d$: Let $k_j \in \{1, 2, \ldots, n\}$ such that $x_j \in \pi_j(S_{k_j})$. Hence in this case we have $F.\mathsf{Eval}(0, F.\mathsf{Eval}(0, \mathsf{Decode}(k_0[j], x_j), x_j)$ $\oplus F.\mathsf{Eval}(1, \mathsf{Decode}(k_0[j], x_j), x_j) = R[k_j, j]$. Making the xor of two Eval outputs equal $\oplus_{j=1}^{d} R[k_j, j] =$ $(\oplus_{k_j = k_1} R[k_1, j]) \oplus (\oplus_{k_j \neq k_1}^{d} R[k_j, j])$. Since not all these k_j are the same for $j = 1, \ldots, d$ we have

$\oplus_{j=1}^{d} R[k_j, j]$ is a random bit since the component $(\oplus_{k_j=k_1} R[k_1, j])$ is a random independent bit.

Case 3 : if $\boldsymbol{x} \notin S$ and for some $j \in \{1, \ldots, d\}$, $\pi_j(\boldsymbol{x}) \notin \pi_j(S)$:

- if x_j is in an active grid cell along dimension j: we have $\mathsf{Decode}(k_0[j], x_j) \oplus \mathsf{Decode}(k_1[j], x_j)$ is a random bit, where this bit was independently sampled. Hence the xor of the Eval outputs in this case is a random bit.
- if x_j is not in an grid cell along dimension j: We have $k_0[j], k_1[j]$ are a pair of random independent string (by the OKVS independence property). Further by the PRF property we have $F.\mathsf{Eval}(0, \mathsf{Decode}(k_0[j], x_j), x_j) \oplus F.\mathsf{Eval}(1, \mathsf{Decode}(k_1[j], x_j), x_j)$ is a random bit - making the xor of the Eval outputs to be 0 with probability 1/2.

Complexity Analysis Along each dimension, we apply the spatial-hashing trick. The number of active cells per dimension are less than or equal to $2n$. Each component bFSS key is the trivial union of 2 1d intervals in domain $\{1, 2, \ldots, 2\delta\}$ - which makes the component key size $O(\kappa \log \delta)$. Hence the key size of the collection of sets union-glob-disj is $O(\kappa d n \log \delta)$. ☐

G Batch Optimization for bFSS

We first focus on our spatial-hash construction that uses oblivious key-value stores (OKVS). Our suggested OKVS instantiation is a *binary OKVS* in the terminology of [GPR+21], meaning that the OKVS structure S is a vector of slots $S = (S_1, \ldots, S_n)$, and decoding involves xor'ing a subset of those slots. More precisely, $\mathsf{Decode}(S, x) = \bigoplus_{i \in \pi(x)} S_i$, where we call π the *probe* function. In an OKVS construction, the random choice of π guarantees correctness with high probability, but it has no effect on the obliviousness property. That means it does no harm to use the same π function across many bFSS sharings of the same set — the fact that false positives have bounded probability does not come from the choice of π. Not only does reusing π save some cost, but it enables further optimizations that we explain below.

Some of our bFSS constructions have the property that, for every fixed x, the function $\mathsf{Eval}(\cdot, x)$ is a $GF(2)$-linear function of the input share.

- $\mathsf{tt.Eval}(\cdot, x)$ returns a single bit of the share.
- $\mathsf{concat.Eval}(\cdot, x)$ selects several subsets of the share bits, calls a component Eval on each subset of bits, and concatenates the results.
- $\mathsf{spatial\text{-}hash.Eval}(\cdot, x)$ evaluates an OKVS and calls a component Eval on the result. As mentioned above, when the OKVS is a binary OKVS, evaluation involves only an xor of slots from from input share. When the same probe-function is used for many bFSS instances, and is understood from context, then the choice of which bits to access truly depends only on x and not at all on the bFSS share.
- $\mathsf{xor\text{-}share.Eval}(\cdot, x)$ selects several subsets of the share bits, calls a component Eval on each of those subsets of bits, and xors the results.

Any *composition* of these four bFSS building blocks is also $GF(2)$-linear. I.e., each output bit of Eval is an xor of some of the bits of the input share. Concretely, you can think of our bFSS recipes xor share $\circ$ spatial hash $\circ$ tt and spatial hash $\circ$ concat $\circ$ tt as suitable candidates to apply the optimization we describe below.

Suppose we run our structure-aware PSI protocol with a bFSS that is $GF(2)$-linear. The PSI receiver generates ℓ separate sharings of her set. Each share is transferred via OT, and we consider a share as a column of a matrix. Evaluating a share at x involves applying a linear function to that share, illustrated in Figure 18(a).

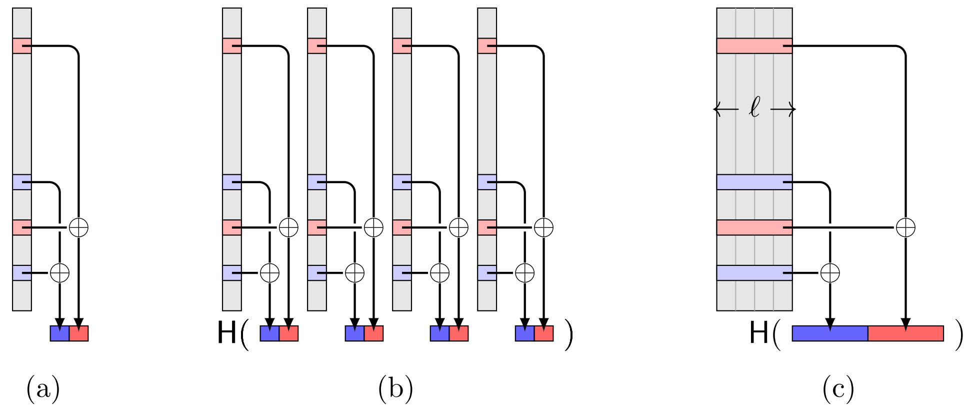


Figure 18: Illustration of our optimization for batch evaluation of many bFSS. Each column represents a bFSS share. Fig (a) illustrates $\mathsf{Eval}(k, x)$, a $GF(2)$-linear function for every fixed x. Fig (b) illustrates what the structure-aware PSI protocol specifies — evaluate on ℓ different shares, concate the results, and compute H. Fig (c) illustrates our optimization which applies the linear function to entire rows of the share matrix — *i.e.*, $GF(2^\ell)$ elements.

To compute an OPRF output for input x, a party evaluates *the same linear function* on ℓ different shares (columns), concatenates the result, and computes its hash. This is illustrated in Figure 18(b). Computing and concatenating xor's of individual bits is awkward in practice. Hence, we propose to take advantage of the fact that the same linear function is used for all columns. Our optimization is to extend the $GF(2)$ linear function to a $GF(2^\ell)$ linear function. The linear function can be applied to the entire matrix, operating on ℓ-bit blocks (the rows of the matrix) instead of individual bits. This is illustrated in Figure 18(c). As a result of this optimization, the input bits to H have been permuted in a fixed pattern, relative to what the protocol specifies. But it is easy to see that the property that we require of H (correlation-robustness) is invariant under fixed permutations of the input bits, so there is no effect on security if both parties use this optimization (*i.e.*, they agree on the same way to reorder bits to H).

With this optimization, we combine all of the similar bit-xor's into a single xor of ℓ-bit blocks, which is significantly faster in practice. We emphasize that any existing or future constructions that are a composition of $GF(2)$-linear operations are amenable to this optimization.