Multiparty Distributed Point Functions

Aarushi Goel* Mingyuan Wang† Zhiheng Wang‡

Abstract

We present the first construction of multiparty distributed point functions based on one-way functions, where the share sizes remain sublinear in the domain size and grow *only polynomially* with the number of parties. In contrast, existing multiparty distributed point function constructions in Minicrypt have share sizes that grow *exponentially* with the number of parties.

Contents

1 Introduction **2**
- 1.1 Our Results . . . 2
- 1.2 Applications . . . 3
- 1.3 Related Works . . . 4

2 Technical Overview **4**
- 2.1 Multiparty DPFs from Special Combinatorial Design . . . 6
- 2.2 Deterministic Construction Requires Exponential Size . . . 8
- 2.3 Randomized Construction with Polynomial Size . . . 9

3 Preliminaries **12**

4 Multiparty Distributed Point Functions **14**
- 4.1 Multiparty DPFs from PRGs and a Special Combinatorial Design . . . 14
- 4.2 Lower Bound . . . 18
- 4.3 Randomized Special Combinatorial Design . . . 19
- 4.4 Privacy Amplification with Near-linear Stretch . . . 23
- 4.5 (t, n)-Threshold DPFs . . . 25

References **26**

*Purdue University aarushi@purdue.edu
†New York University, Shanghai mingyuan.wang@nyu.edu
‡Shanghai Jiao Tong University serpentg@sjtu.edu.cn

1 Introduction

A *point function*, denoted $f_{\alpha,\beta} : \mathcal{D} \to \mathcal{R}$, outputs 0 for all inputs $x \neq \alpha$ where $x, \alpha \in \mathcal{D}$, and outputs $\beta \in \mathcal{R}$ when $x = \alpha$. A *distributed point function* (DPF) [GI14, BGI15] enables the point function to be securely split into shares distributed among a group of mutually distrusting parties. Crucially, the size of these function shares is significantly smaller than the size of the domain $|\mathcal{D}|$. Given an input $x \in \mathcal{D}$, each party can independently compute an additive share of the function's output using its function share, without interacting with others.

Over the last few years, DPFs have proven to be immensely useful in a wide range of secure computation applications, including private information retrieval [GI14], private set intersection [TSS+20, DIL+20, GRS22, GRS23, GGM24], pseudorandom correlation generators [BCGI18, BCG+19, BCG+20], distributed oblivious RAM [Ds17, BKKO20], privacy-preserving machine learning [RTPB22, YJG+23, JGB+24], compressing OR proofs [BS24] and secure computation of functions with mixed-mode operations [BGI19, BCG+21, BDSS25], among others. Consequently, significant research efforts [GI14, BGI15, BGIK22a, BGIK22b, BDSS25, BGI19, BGH+25] have focused on designing efficient DPFs with share sizes that are sublinear in the domain size.

However, most advancements in both the design and applications of DPFs have been restricted to the two-party setting.[1] In the multiparty setting, existing DPF constructions in Minicrypt have share sizes that grow exponentially with the number of parties [BGI15] and are therefore only efficient for a constant number of parties.[2] This inefficiency has also hindered the exploration of applications of multiparty DPFs. As a result, the following problem, posed in [BGI15], has remained open for the past decade:

Do efficient multiparty DPFs exist in Minicrypt for any number of parties?

In this work, we answer this question affirmatively. We present the first such construction of multiparty DPFs with share sizes growing sublinearly in the domain size and only polynomially in the number of parties.

1.1 Our Results

We now elaborate upon our results in more detail.

Multiparty Distributed Point Functions. We construct multiparty DPFs based on one-way functions, where the share sizes are sublinear in the domain size and polynomial in the number of parties. This construction remains private against collusion of up to all but one of the parties. Our result is summarized informally as follows (See Corollary 1 for a formal statement).

(Informal) Theorem 1. *Let n be the total number of parties and $f_{\alpha,\beta} : \mathcal{D} \to \mathbb{Z}_{p^q}$ be a point function, where p is an arbitrary constant prime. Assuming the existence of one-way functions, for arbitrary constant $\varepsilon > 0$, there exists an $(n-1)$-private multiparty DPF where the size of each party's function share is $O(|\mathcal{D}|^{1/2+\varepsilon} \cdot n^3)$.*[3]

[1] In the two-party setting, the most efficient constructions of DPFs [BGI16] have share sizes that depend logarithmically on the size of the domain.

[2] The only other construction in Minicrypt that we are aware of is limited to at most four parties [BGIK22a] and allows only a single corrupt party.

[3] Throughout the remainder of this paper, we will disregard polynomial factors in the security parameter in all asymptotic analyses.

In contrast, the only other known construction of multiparty DPFs [BGI15] produces shares of size $O(\sqrt{|\mathcal{D}|} \cdot 2^{n-1})$. Similar to this construction and all existing two-party DPF constructions [GI14, BGI15, BGIK22a, BGIK22b, BDSS25, BGI19, BGH+25], given any input $x \in \mathcal{D}$, our multiparty DPF allows parties to independently use their respective function shares to obtain *additive shares* of the corresponding function output.

Lower Bound. Our multiparty DPF construction relies on a *randomized combinatorial design*, whereas the construction in [BGI15] can be viewed as being based on a *deterministic combinatorial design*. We show that any multiparty DPF based on a deterministic combinatorial design must have share sizes that grow exponentially with the number of parties. Thus, the use of a randomized design is *essential* to achieve share sizes that scale sub-exponentially in the number of parties.

(t, n)-Threshold DPFs. While distributed point functions are typically used to compute *additive shares* of a point function, one can also consider an alternative notion where the goal is to compute (t, n)-*threshold shares* instead. In particular, using such function shares, parties can locally compute (t, n)-threshold shares of the function output for any given input. We refer to such DPFs as (t, n)-threshold DPFs.

Using standard share conversion techniques [CDI05], our construction of multiparty DPFs can be extended to obtain (t, n)-threshold DPFs for certain thresholds that to the best of our knowledge were previously unattainable (refer to Corollary 2). Specifically, we show how to efficiently compute *replicated* and *Shamir* secret sharing [Sha79] of point functions for all $t < n$, where $\binom{n}{t} = \mathrm{poly}(\lambda)$, in other words where either $t = O(1)$ or $n - t = O(1)$. We note that a similar transformation of the multiparty DPF construction from [BGI15] would only yield an efficient construction for threshold DPFs when $t = O(1)$.

1.2 Applications

Our multiparty DPFs and (t, n)-threshold DPFs can be used to establish new feasibility results for various secure computation problems in the *multiparty* setting. We discuss several representative applications.

Multi-Server Private Information Retrieval. Private Information Retrieval (PIR) [CGKS95] enables a user to privately retrieve the i-th entry from a public database stored across multiple servers. As noted in [BGI15], a multiparty DPF enables multiserver PIR protocols with sublinear query length and constant answer length. Leveraging our multiparty DPF and threshold DPFs, we obtain the first efficient n-server, t-private PIR with these properties, for any n, t satisfying $\binom{n}{t} = \mathrm{poly}(\lambda)$.

MPC in the Random DPF Preprocessing Model. Recent works [BGI19, BCG+21] have demonstrated that access to shares of a *random* point function[4] can accelerate secure multiparty computation for functions such as equality testing, where parties hold secret shares of two inputs and need to determine if they are the same, or zero testing, where parties holding secret shares of a value wish to check whether it is zero. While these techniques were previously limited to the two-party setting, we show that they naturally extend to the multiparty setting using our multiparty and threshold DPFs.

In general, these ideas can be extended to show that, given shares of a random point function in a preprocessing phase, it is possible to efficiently compute shares of the required point function in the online phase, which can, in turn, be helpful for other tasks. This avoids the need to run an MPC to compute the DPF share generation algorithm in a distributed manner during the online phase.

Multiparty Distributed ORAM. Oblivious RAM [GO96] (ORAM) is a technique for hiding the memory access pattern of a RAM program. Distributed ORAM [LO13] extends ORAM to the multiparty setting,

[4] That is, a point function $f_{\alpha,\beta}$, where α and β are randomly sampled and secret shared among the parties along with the function share.

where both the memory and access locations are distributed among multiple parties in a privacy-preserving manner. DORAM is a fundamental building block for the secure computation of RAM-based programs. While it is theoretically possible to design (multiparty) DORAM by generically combining secure computation with existing ORAM constructions, such approaches are highly inefficient in practice. Starting from the seminal work of [Ds17], a long line of works [IKH+23, HNO+23, VHG23, BPRS23] have leveraged (two-party) DPFs as a foundational building block to design *concretely efficient* DORAM among two parties or three parties.[5] Our construction of multiparty DPF can be used as a drop-in replacement within this framework to enable multiparty DORAM protocols. It remains an exciting direction for future work to explore the concrete efficiency of such extensions.[6]

1.3 Related Works

DPF Constructions. The first two-party DPF construction based on one-way functions was proposed by Gilboa et al. [GI14]. Since then, several works [BGI15, BGI16, BGH+25] have focused on improving the efficiency of two-party DPFs. Boyle et al. [BGI15] also introduced a multiparty DPF, but the key size in their scheme grows exponentially with the number of parties.

Multiparty DPF beyond Minicrypt. In the pursuit of secure multiparty computation with sublinear communication, recent work [CK24] proposed DPF constructions for a fixed number of parties (e.g., $n = 8$) from (combinations of) public-key assumptions (e.g., DCR and LPN). More recent works [ARS24, CKY25] show how to extend these constructions to any number of parties.

On a separate note, Boyle et al. [BGIK22a] explored the notion of *information-theoretic* DPFs for three and four parties that can tolerate at most 1 corruption.

Variants of DPFs. Since key generation is the most computationally expensive step in DPFs, some recent works have explored ways to optimize it. For instance, Boyle et al. [BDSS25] designed two-party DPFs where key generation can be performed non-interactively in a distributed manner. Boyle et al. [BGIK22b] introduced programmable DPFs, where an input-independent offline key enables sharing multiple point functions.

FSS Constructions for Related Functions. While recent research has largely focused on optimizing two-party DPFs or exploring new applications, some works have drawn inspiration from DPF constructions [GI14, BGI15] to design function secret sharing (FSS) schemes for related functions, such as comparison and decision trees [BGI19, BCG+21].

2 Technical Overview

This section presents the main ideas behind our multiparty distributed point function construction. For the most part, in this paper, we are in the *dishonest majority* setting. In particular, this requires that the collusion of all but one party should not learn any information about the point function. Throughout this paper, we use N to denote the domain size, i.e., $\mathcal{D} = [N]$.

[5] The works on three-party DORAM still rely on two-party DPF and, hence, they can only allow one corruption.

[6] Note that the $\sqrt{|\mathcal{D}|}$-sized DPF shares do not pose an (asymptotic) bottleneck for this DORAM construction framework, as it inherently relies on the first-generation square-root ORAM [GO96] (i.e., each memory access incurs an asymptotic overhead of $O(\sqrt{|\mathcal{D}|})$).

Let $f_{\alpha,\beta} : [N] \to \mathbb{F}$ stand for a point function defined as

$$f_{\alpha,\beta}(x) = \begin{cases} \beta & x = \alpha \\ 0 & x \neq \alpha \end{cases}.$$

To distribute a point function among n parties, a trusted dealer takes $f_{\alpha,\beta}$ as input and outputs n succinct function shares $k_1, \ldots, k_n$, one for each of the n parties, such that each party can locally expand their respective private share k_i into a vector x_i of dimension-N such that

$$x_1 + \cdots + x_n = (0, \ldots, 0, \beta, 0, \ldots, 0).$$

Note that, this scheme is non-trivial only if the secret share size is $o(N)$. Existing works [BGI15] present constructions where the share size is $2^n \cdot \sqrt{N}$. Looking ahead, in this work, we present a construction with share size $\mathsf{poly}(n) \cdot \sqrt{N}$.

Before we develop our ideas, we first need to recall a simple *two-party* DPF construction with $\Theta\left(\sqrt{N}\right)$ share size from [GI14, BGI16].

A Paradigm for Two-party DPF. In this construction, the dealer starts by interpreting the index $\alpha \in [N]$ as a pair of indices $(i, j) \in [\sqrt{N}] \times [\sqrt{N}]$. To sample the DPF shares, the dealer performs the following: first, samples $\sqrt{N}$ fresh PRG seeds $s_1, \ldots, s_{\sqrt{N}}$ and gives it to the first party P_1. Each one of these seeds can be expanded (using a pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \mathbb{F}^{\sqrt{N}}$) to a $\sqrt{N}$-sized pseudorandom vector. Observe that the concatenation of these vectors

$$\mathsf{PRG}(s_1) \| \mathsf{PRG}(s_2) \| \cdots \| \mathsf{PRG}(s_{\sqrt{N}})$$

is a vector of length N. Similarly, the second party P_2 is also given $\sqrt{N}$ seeds $s'_1, \ldots, s'_{\sqrt{N}}$ such that

$$s'_i = \begin{cases} s_i & i \neq i^* \\ \text{random} & i = i^* \end{cases}.$$

Namely, P_2 receives the same set of seeds, except for s'_{i^*}, which is a freshly sampled random seed. The second party can also expand these seeds into:

$$\mathsf{PRG}(s'_1) \| \mathsf{PRG}(s'_2) \| \cdots \| \mathsf{PRG}(s'_m).$$

Note that the sum of these two vectors (assuming $\mathbb{F}$ has characteristics 2)[7] computed by P_1 and P_2 is

$$0^{\sqrt{N}} \| \cdots \| 0^{\sqrt{N}} \| \mathsf{PRG}(s_{i^*}) + \mathsf{PRG}(s'_{i^*}) \| 0^{\sqrt{N}} \| \cdots \| 0^{\sqrt{N}}.$$

However, to ensure correctness, parties must correct the sum of the i^*-th substring (computed as $\mathsf{PRG}(s_{i^*}) + \mathsf{PRG}(s'_{i^*})$) such that the overall sum corresponds to the intended vector

$$e_{j^*} = (\underbrace{0, \ldots, 0, \beta, 0, \ldots, 0}_{\sqrt{N}}).$$

[7] In other cases, we may simply let P_2 multiply its PRG expansion by "-1". So, this assumption is just for simplicity.

This can be achieved by having the dealer send the following *correction string* to both parties

$$w = \mathrm{PRG}(s_{i^*}) + \mathrm{PRG}(s'_{i^*}) + e_{j^*}.$$

For privacy of α, both i^* and j^* must be kept hidden from both parties. As a result, we cannot have the parties *only* shift the i^*-th substring by w. Instead, for each location i, the dealer computes an additive sharing (u_i, v_i) of the bit $\mathbb{1}_{i=i^*}$. To summmarize, this two-party DPF construction with shares of size $\sqrt{N}$ works as follows:

$$\mathsf{P}_1\text{'s share: } w, (s_1, u_1), \ldots, (s_{\sqrt{N}}, u_{\sqrt{N}}) \qquad \mathsf{P}_2\text{'s share: } w, (s'_1, v_1), \ldots, (s'_{\sqrt{N}}, v_{\sqrt{N}})$$
$$\mathsf{P}_1\text{'s Evaluation: } \mathrm{PRG}(s_1) + (u_1 \cdot w) \| \cdots \| \mathrm{PRG}(s_{\sqrt{N}}) + (u_{\sqrt{N}} \cdot w)$$
$$\mathsf{P}_2\text{'s Evaluation: } \mathrm{PRG}(s'_1) + (v_1 \cdot w) \| \cdots \| \mathrm{PRG}(s'_{\sqrt{N}}) + (v_{\sqrt{N}} \cdot w)$$

It is easy to verify that the above scheme is correct and yields shares of size $\Theta\left(\sqrt{N}\right)$.

For security, observe that except for w, all the other values sent to each party (as part of their shares) information-theoretically hides $f_{\alpha,\beta}$, while w only computationally hides $f_{\alpha,\beta}$. This is because, for either party, w is masked with a pseudorandom string, the seed corresponding to which *is only known to this party.*[8]

2.1 Multiparty DPFs from Special Combinatorial Design

In order to extend the above framework to the multiparty setting, we consider a special type of *combinatorial design* on bipartite graphs. We emphasize that, while similar ideas were implicitly employed by Boyle et al. [BGI15] in their multiparty DPF construction, we view this abstraction as an important conceptual contribution of our work. Looking ahead, this abstraction is crucial in the design of our construction.

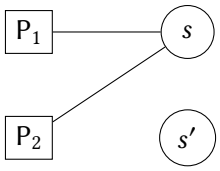
A General Perspective. To motivate this abstraction, let us view the above two-party construction from a more general perspective that includes it as a special case. Essentially, for every index i, the dealer samples an independent collection of ℓ random seeds $s_0, \ldots, s_{\ell-1}$ and each party P_j is given some *subsets* S_j of these seeds (i.e., $\{s_i\}_{i \in S_j}$). During expansion, party P_j evaluates the PRG on each seed, adds these pseudorandom strings and views this sum $\sum_{i \in S_j} \mathrm{PRG}(s_i)$. as its private share of the i-th $\sqrt{N}$-size substring (i.e., of the vector obtained before applying the correction string).

This *seed distribution process* should come in two modes, depending on if $i = i^*$ or not. For $i \neq i^*$, we want the parties' shares to sum up to the zero string. For this, the dealer must send each seed to *an even number of parties*. In the other case, where $i = i^*$, the shares of all parties should add up to e_{j^*} after applying the correction string.

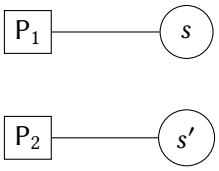
To ensure that no information about i^* is leaked, we require the following two properties to hold in the presence of an adversary who corrupts all but one party: (1) First, the correction string needs to appear pseudorandom to this adversary. This can be achieved if, for any party P, there exists a seed s that is *exclusively* sent to P. (2) Second, the two modes of distributing seeds should remain indistinguishable for this adversary.[9]

If we were to represent the above process of distributing seeds as a bipartite graph, the previously discussd two-party construction can be viewed as Figure 1. Graph G_0 (resp., G_1) on the left (resp., right) corresponds to the mode $i \neq i^*$ (resp., $i = i^*$).

[8] For example, if P_1 is corrupted, the seed s'_{i^*} guarantees that w is pseudorandom to P_1. On the other hand, if P_2 is corrupted,



(a) G_0 with only even degrees



(b) G_1 with a dedicated seed per party

Figure 1: Two-party scheme represented as a combinatorial design: $n = 2$ and $\ell = 2$.

While this may seem like a convoluted way to view a two-party DPF scheme, the advantage is that this framework can be naturally extended to any number of parties n, as long as we have two bipartite graphs – each with n left vertices – that satisfy all the required properties. We refer to such a pair of graphs as a *special combinatorial design*.

Special Combinatorial Design. Let us formally summarize what we mean by a special combinatorial design (see Definition 5 for a technical definition). It comprises of two bipartite graphs G_0 and G_1, where both have n vertices on the left and ℓ vertices on the right. They should satisfy the following three properties.

1. **Correctness.** The degree of each right vertex in G_0 must be *even*.
2. **Pseudorandomness.** In G_1, for *every* left vertex, there exists a right vertex that is *only incident to this particular left vertex*.
3. **Indistinguishability.**[10] The subgraphs of G_0 and G_1 induced by any proper subset of the left vertices are *indistinguishable*. When the two subgraphs are perfectly indistinguishable, we call such schemes *perfect*.

For intuition, G_0 and G_1 represent the seed distribution process for the $i \neq i^*$ and $i = i^*$ case, respectively. The correctness property guarantees the correctness when $i \neq i^*$ (i.e., the sum of the evaluations of the DPF shares is a 0 string), and the pseudorandom property and indistinguishability properties jointly guarantee the privacy of the (multiparty) DPF scheme. This implication is formally presented in Section 4.1.

(Perfect) Special Combinatorial Design from Existing Constructions. Given such a framework, the first natural question is whether such special combinatorial designs exist for any number of parties n. Fortunately, [BGI15] gives an elegant construction. We include their construction below as it is closely related to our lower-bound proof later.

For any n, set $\ell = 2^{n-1}$. Label all the right vertices in G_0 as n-bit binary strings with an *even* Hamming weight, e.g., $(1, 1, 0, 0, \ldots, 0)$.[11] Conversely, label all the right vertices in G_1 as n-bit binary strings with an *odd* Hamming weight. Let the graph be defined by the labeling of each right vertex as its incident vector. For example, if s_0 is labeled with the vector $(1, 0, 1, 0, \ldots, 0)$, s_0 is connected to P_1 and P_3.

The reader may verify that this construction satisfies all properties of our special combinatorial design (refer to [BGI15] for a detailed proof). In particular, Figure 2 pictorially presents this construction for $n = 3$.

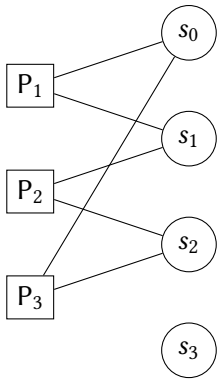
While the above solution is elegant, we note that it requires exponentially many vertices on the right — namely, $\ell = 2^{n-1}$. Observe that the *total number of edges* in the special combinatorial design determines

the seed s_{i^*} guarantees that w is pseudorandom to P_2.

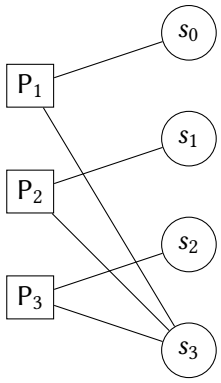
[9] Intuitively, the first (resp., second) property ensures that parties cannot infer any information about i^* from the correction string (resp., the collection of seeds) they receive.

[10] In Definition 5, we refer to this as the privacy property.

[11] Note that there are exactly 2^{n-1} such strings.



(a) G_0 with only even degrees



(b) G_1 with a dedicated seed per party

Figure 2: A perfect combinatorial design for $n = 3$ with $\ell = 4$ [BGI15]. The incident vector for (the right vertices of) G_0 consists of all possible 3-bit strings with even Hamming weights; while the incident vector for (the right vertices of) G_1 consists of all possible 3-bit strings with odd Hamming weights.

the *share size* of the corresponding multiparty DPF. Consequently, their construction only gives a multiparty DPF with share size exponential in the number of parties. This naturally leads to the following key technical question.

Can we construct such a special combinatorial design with smaller size?
In particular, polynomial *size?*

We answer this question in both directions. First, we show that such a construction cannot exist if one restricts to *deterministic* constructions. In particular, our lower bound implies that the construction of [BGI15] is *optimal*. On the other hand, we manage to give a *randomized* construction where $\ell = \mathrm{poly}(n)$, which leads to a multiparty DPF with $\mathrm{poly}(n) \cdot \sqrt{N}$ share size. On the downside, the indistinguishability of our construction only holds with inverse polynomial probability. Therefore, we need a privacy amplification technique borrowed from [BGIK22b]. We elaborate on these points next.

2.2 Deterministic Construction Requires Exponential Size

We first prove an exponential lower bound for any deterministic special combinatorial designs. In particular, we show that $\ell \geqslant 2^{n-1}$, which implies that the construction of [BGI15] is optimal. The proof is formally presented as Theorem 2 in Section 4.3. We present the main idea here.

Suppose we have a deterministic special combinatorial design. We are going to label every right vertex (in both G_0 and G_1) by its n-bit incidence vector.[12] For example, if this vertex is only connected with the left vertices P_1 and P_3. We are going to label it with $(1, 0, 1, 0, 0, ..., 0)$.[13] In a nutshell, we are going to prove that, for every possible vector v in $\{0,1\}^n$, there exists a right vertex with the label v. This suffices for the proof as it shows that G_0 and G_1 combine for $\geqslant 2^n$ many right vertices. Our proof follows an inductive structure based on the Hamming weight of the vector to alternatively prove the following two claims.

- For an odd integer wt, if G_1 contains all the vectors of Hamming weight wt, then G_0 must contain all the vectors of Hamming weight wt + 1.

[12] In this informal discussion, we interchangeably use the term vector and right vertex.
[13] Note that the labeling is *not unique*. Two balls might have the same label.

- For an even integer wt, if G_0 contains all the vectors of Hamming weight wt, then G_1 must contain all the vectors of Hamming weight wt + 1.

In this overview, we show why the first two steps (i.e., wt = 2 and wt = 3) are correct. The rest are analogous. First, the base case wt = 1 of the induction is correct. Namely, G_1 does contain all the vectors with Hamming weight 1 as mandated by the "pseudorandomness property".

Now that we know G_1 contains all vectors with wt = 1. Take $(1, 0, 0, ..., 0)$ as an example. Suppose P_2 is the only honest party and consider the subgraph induced by $\{1, 3, 4, \ldots, n\}$. Then, for G_1, this subgraph contains a vector $(1, \perp, 0, 0, \ldots, 0)$.[14] Now, for G_0, we should see the same vector $(1, \perp, 0, 0, \ldots, 0)$ in the induced subgraph (otherwise, the design is completely distinguishable). The question is, what is the original vector that resulted in $(1, \perp, 0, 0, \ldots, 0)$? It could potentially be

$$(1, 0, 0, 0, \ldots, 0) \qquad \text{or} \qquad (1, 1, 0, 0, \ldots, 0).$$

However, by the "correctness property", G_0 should only contain vertices with an even degree. Hence, the original vector can only be $(1, 1, 0, 0, \ldots, 0)$, which must be contained by G_0. In this manner, one can prove that G_0 must contain all the vectors with wt = 2.

Now that we know G_0 contains all vectors with wt = 2, we are going to prove that G_1 contains all vectors with wt = 3. Take the vector $(1, 1, 0, 0, ..., 0)$ in G_0 as an example. Suppose P_3 is honest and consider the subgraph induced by $\{1, 2, 4, 5, \ldots, n\}$. Following a similar argument, we know that $(1, 1, \perp, 0, 0, ..., 0)$ should be contained in both induced subgraphs. However, the above proof strategy runs into an issue because G_1 is *not required* to contain only odd-degree vertices. We cannot directly argue that $(1, 1, 1, 0, 0, ..., 0)$ must be contained in G_1. Yet, for deterministic constructions, we can arrive at this conclusion by a more careful analysis.[15]

We prove this by contradiction. Suppose we see m many $(1, 1, \perp, 0, 0, ..., 0)$ vertices in G_1, and all of them are due to $(1, 1, 0, 0, 0, ..., 0)$, i.e., G_1 contains m vertices with the label $(1, 1, 0, 0, ..., 0)$. Now, remember that G_1 also contains at least one vertex with the label $(1, 0, 0, 0, ..., 0)$.[16] This means that there are $\geqslant m + 1$ instances of $(1, \perp, 0, 0, ..., 0)$ for G_1 (when we look at the induced subgraph from $\{1, 3, 4, \ldots, n\}$). By our argument before, in order to match this view, G_0 must have $m + 1$ vertices with labeling $(1, 1, 0, ..., 0)$, which means that G_0 will have $m + 1$ instances of $(1, 1, \perp, 0, ..., 0)$ in the induced subgraph — rendering G_0 and G_1 completely distinguishable when P_3 is the only honest party.

By alternatively invoking these two claims, we can indeed prove that all possible vectors are contained in either G_0 or G_1 and any deterministic design must have exponential size.

2.3 Randomized Construction with Polynomial Size

To bypass the lower bound, we resort to randomized constructions. Our construction is pictorially presented in Figure 3.

The G_0 graph is simply a 2-regular bipartite graph where the two neighbors of each right vertex $s_1, \ldots, s_\ell$ are randomly chosen with replacement (i.e., the two neighbors could be the same). For G_1, we additionally plant t more vertices $s'_1, \ldots, s'_t$ such that they only have only one random neighbor.[17]

[14] Here, $\perp$ is a placeholder for the removed second (left) vertex.

[15] The analysis fails for randomized constructions completely.

[16] For randomized constructions, the number of instances for $(1, 0, 0, \ldots, 0)$ and $(1, 1, 0 \ldots, 0)$ could be *highly correlated*. Thus, a counting argument will not work.

[17] One may consider other ways (e.g., a deterministic way) to plant these additional vertices. For instance, one may simply add n vertices, each connected to one distinct party P_i. Our analysis will also apply. We opt for this construction as we find the statistical analysis simpler in this way.

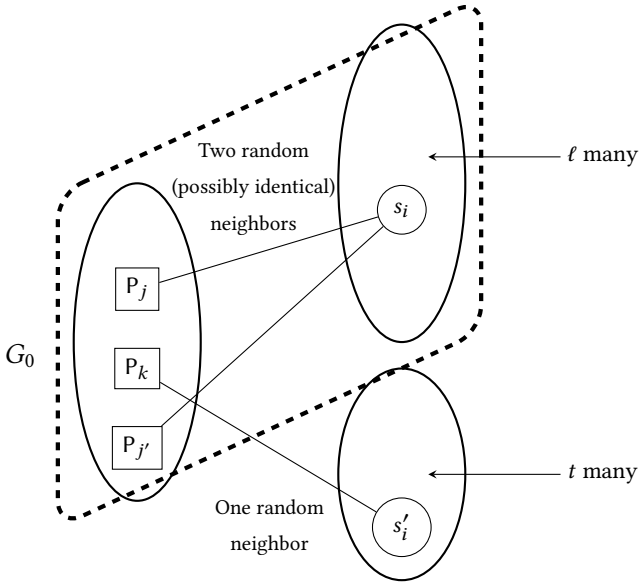


Figure 3: Our randomized construction: G_0 consists of only the highlighted part, while G_1 consists of the entire graph. We emphasize that the two random neighbors for s_i's are sampled *with replacement* (i.e., they could be identical).

To argue that this is a special combinatorial design, we need to verify the three properties. First, the correctness property is always guaranteed by the construction. Second, the pseudorandom property is satisfied as long as the additional vertices s'_i that we plant will cover all parties. This will hold with overwhelming probability as long as t is sufficiently large (e.g., $t = n \cdot \lambda$).

Finally, the indistinguishability is the most non-trivial to see. In a nutshell, the reason why G_0 and G_1 are hard to distinguish is because the adversary only sees a subgraph of G_0 and G_1. For those ℓ seeds that have degree-2, there is a significant chance ($\approx 1/n$) that the adversary sees this vertex has only degree-1 (or even degree-0).[18] Consequently, these degree-1 vertices in the adversary's view play a pivotal role in masking the additional t degree-1 vertices that we plant for G_1.

Technically, we can think of this randomized construction as a *balls and bins* experiment. There are n bins in total and the adversary gets to see the configuration of $n - 1$ out of these n bins. In one case, there are ℓ balls that are dropped twice independently and uniformly randomly into these bins. In the other case, there are t additional balls that are only dropped only once into a random bin. We are interested in the statistical distance between the adversary's views in these two cases.

We first observe that, to argue the statistical distance between these cases, it suffices to consider the (joint) distribution $(\mathcal{S}, \mathcal{T})$, where $\mathcal{S}$ (resp., $\mathcal{T}$) is the random variable denoting the *number of balls* that the adversary sees once (resp., twice). This is because, conditioned on $(\mathcal{S}, \mathcal{T})$, the view of the adversary (i.e., the particular configuration of the balls in each bin) is identically distributed in either case (i.e., uniformly randomly distributed).[19] Therefore, for measuring the statistical distance, it suffices to only consider the distribution $(\mathcal{S}, \mathcal{T})$.

Now that we only focus on $(\mathcal{S}, \mathcal{T})$, the problem simplifies greatly. Note that, in case one, conditioned on any fixed $\mathcal{T} = \tau$, $\mathcal{S}$ is simply a sum of $(\ell - \tau)$ many Bernoulli distributions with probability $\approx (n -$

[18] Such events happen when (at least) one of the random neighbors is the honest party.

[19] This is the benefit of planting the t additional s'_i's at random parties compared to deterministically planting a seed per party.

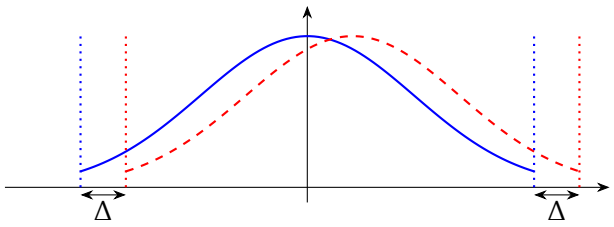


Figure 4: A pictorial view of the distribution shift: the blue distribution represents the number of degree-1 balls the adversary sees in G_0. The dashed red distribution represents the number of degree-1 balls the adversary sees in G_1, and is the "shifted" by some (random) amount Δ from the blue distribution.

$1)/n$.[20] On the other hand, in case two, $\mathcal{S}$ is distributed as the same Bernoulli distribution, but *shifted* (due to the planted degree-1 vertices) by an amount that roughly equals $t \cdot (n-1)/n$. More precisely, the shift amount (i.e., the additional degree-1 balls in the adversary's view) is also a sum of independent Bernoulli distributions (since each planted ball could have degree-0 with probability $1/n$ in the adversary's view). Figure 4 gives an informal view of the distributions of $\mathcal{S}$ in case one and two. We are interested in the statistical distance between these two distributions. It turns out that this problem is well-studied in mathematics, and we imported several lemmas to bound the SD (refer to Lemma 1 and Lemma 2). This proof is formally presented as Theorem 3 in Section 4.3.

Let us also give an informal justification for the asymptotics of the statistical distance. We know the expectation of τ is roughly $\ell \cdot \frac{n-1}{n}$. Hence, $\mathcal{S}$ in case one is roughly the sum of ℓ/n many Bernoulli distributions with probability $(n-1)/n$. By the Central Limit Theorem, this distribution is going to be close to a Gaussian distribution with standard deviation $\sigma = \Theta\left(\sqrt{\ell/n^2}\right)$.[21] If one approximates a Gaussian distribution as a uniform distribution[22] over the support of size $\Theta(\sigma)$, then any shift Δ will result in a statistical distance of $\Theta(\Delta/\sigma)$. Since we are going to shift this distribution by $\Theta(n)$ in expectation, the statistical distance will asymptotically be of the order $n/\sqrt{\ell/n^2} = n^2/\sqrt{\ell}$.

To summarize, this randomized construction gives a special combinatorial design with $\ell + \Theta(n) \approx \ell$ number of right vertices and $\Theta\left(n^2/\sqrt{\ell}\right)$ indistinguishability guarantee. In terms of the multiparty DPF, this will imply a multiparty DPF construction with share size $\Theta\left(\ell \cdot \sqrt{N}\right)$ and $\Theta\left(n^2/\sqrt{\ell}\right)$ privacy. In particular, setting $\ell = n^4 \cdot \mathrm{poly}(\lambda)$ will result in an inverse-polynomial error.

Privacy Amplification. Our randomized special combinatorial design will already give us a multiparty distributed point function with share size $\mathrm{poly}(n) \cdot \sqrt{N}$. However, it has one downside that the privacy guarantee is not negligible, but an inverse polynomial. To obtain a scheme with negligible error, we borrow a result from [BGIK22b], which gives a generic way to amplify the security of (potentially multiparty) DPF constructions.

We have to use their technique with some extra care with the following reason. On a high level, their amplification employs a locally decodable code (with some locality parameter T) to reduce one instance of DPF in the message domain to T instances of DPFs in the codeword domain. When we invoke their

[20] This is the reason why, when we sample the two neighbors, it sampled *with replacement* (i.e., the two neighbors could be the same). Note that, if we sample *without replacement*, conditioned on $\mathcal{T} = \tau$, $\mathcal{S}$ is *fixed to be* $\ell - \tau$.

[21] In our technical proof, we use translated Poisson distributions to approximate these sums of (heavily-)biased Bernoulli distributions (refer to Lemma 1).

[22] This is certainly not true, but just for the purpose of developing intuition.

techniques, we need to be careful since we do not have polylog(N) share size to begin with. Consequently, if the locally decodable code has a quadratic blowup (e.g., encode an N-bit message as an N^2-bit codeword), the amplified scheme would have a linear share size (e.g., $\sqrt{N^2}$) — rendering the overall scheme trivial. In Section 4.4, we carefully argue that we can invoke their result using a locally decodable code with near-linear codeword length. This will eventually give us our final result — multiparty DPF with share size poly$(n) \cdot \sqrt{N}$ and negligible error.

Open Problems. This concludes the overview of our techniques. We end this section with a few fascinating open problems.

- Our lower bound only rules out deterministic special combinatorial designs of polynomial size. Do there exist randomized special combinatorial designs with negligible error and polynomial size? This would give a much more efficient construction as one does not need to employ the privacy amplification techniques of [BGIK22b].
- In this work, we consider the statistical distance for the indistinguishability in the special combinatorial design. Is there a way to leverage *computational assumption* to make the construction more efficient? Several recent works [ABI+23, BJRZ24] have considered a similar problem on distinguishing random graphs from a random graph with a planted minor efficiently.
- So far, all the multiparty DPF functions can only achieve share size square root in the database size. Is there any way that we can break this barrier to obtain a scheme with share size $o\left(\sqrt{N}\right)$?

3 Preliminaries

In this section, we establish the notation and recall some preliminaries.

Notation. We use λ to denote the security parameter and $[N]$ to denote the set $\{1, 2, \ldots, N\}$. The operators $\bigoplus$ or $\oplus$ denote bitwise exclusive-or, while $||$ denotes string concatenation. The unit vector e_j refers to the vector whose j-th entry is 1 and all other entries are 0. For a binary string x, we use $|x|$ to denote its length. For an array or vector v, $v[i]$ refers to its i-th entry. The inner product of two vectors x and y is denoted by $\langle x, y \rangle$.

We write $x \xleftarrow{\$} \mathcal{X}$ to denote that x is sampled according to the probability distribution $\mathcal{X}$. When $\mathcal{X}$ is a finite set, this indicates that x is sampled uniformly from that set. We use Bern(p) to denote the Bernoulli distribution with success probability p, defined as: That is,

$$\text{Bern}(p) = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}.$$

Definition 1 (Statistical Distance)**.** *For any two discrete distributions $\mathcal{X}_1$ and $\mathcal{X}_2$ over the support $\mathcal{S}$, their* statistical distance *is defined as*

$$\text{SD}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \frac{1}{2} \cdot \sum_{s \in \mathcal{S}} |\Pr[\mathcal{X}_1 = s] - \Pr[\mathcal{X}_2 = s]|.$$

Definition 2 (Chernoff Bound [Che52])**.** *Let $X_1, X_2, \ldots, X_n$ be independent random binary variables, and let*

$X := \sum_{i=1}^{n} X_i$. *Let the expectation of* X *be denoted as* $\mathbb{E}(X) = \mu$ *and* $\delta > 0$. *The Chernoff's bound states that:*

$$\Pr\left(X \geqslant (1+\delta)\mu\right) \leqslant \exp\left(-\frac{\delta^2\mu}{2+\delta}\right),$$
$$\Pr(X \leqslant (1-\delta)\mu) \leqslant \exp\left(-\frac{\delta^2\mu}{2}\right)$$

Definition 3 (Poisson Distribution)**.** *Let* $\ell > 0$. *A discrete random variable* X *is distributed according to the Poisson distribution with parameter* ℓ*, if and only if its probability mass function is defined as follows:*

$$\Pr(X = k) = \begin{cases} \frac{\ell^k}{e^\ell \cdot k!} & \text{if } k \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}.$$

Definition 4 (Multiparty DPF [BGI16])**.** *Let* $f_{\alpha,\beta} : \mathcal{D} \to \mathcal{R}$ *be a point function. For* $n \in \mathbb{N}$ *and* $t < n$*, an* n*-party,* t*-secure distributed point function with respect to share reconstruction algorithm* Recon*, is a pair of PPT algorithms* (Gen, Eval) *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, \alpha, \beta)$*: On input the security parameter* 1^λ *and function parameters* $\alpha \in \mathcal{D}, \beta \in \mathcal{R}$*, the share generation algorithm outputs* n *shares,* $(k_0, \ldots, k_{n-1})$.
- $\mathsf{Eval}(i, k_i, x)$*: On input a party index* i*, share* k_i*, and input string* $x \in \mathcal{D}$*, the evaluation algorithm outputs a value* $y_i \in \mathcal{R}$*, corresponding to this party's share of* $f_{\alpha,\beta}(x)$.

These two algorithms satisfy the following:

- $(1-\gamma)$***-Correctness.*** *For all* $x, \alpha \in \mathcal{D}$ *and* $\beta \in \mathcal{R}$*, it holds that,*

$$\Pr\left[\begin{array}{c}(k_0, k_1, \ldots, k_{n-1}) \leftarrow \mathsf{Gen}(1^\lambda, \alpha, \beta); \\ \mathsf{Recon}\left(\mathsf{Eval}(0, k_0, x), \ldots, \mathsf{Eval}(n-1, k_{n-1}, x)\right) = f_{\alpha,\beta}(x)\end{array}\right] \geqslant 1 - \gamma(\lambda).$$

- (t, v)***-Security.*** *Consider the following indistinguishability challenge experiment for any* t*-size subset* $T \subset \{0, 1, \ldots, n-1\}$ *of corrupted parties:*
  1. *The adversary outputs* $(\alpha_0, \beta_0, \alpha_1, \beta_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$*, where* $\alpha_0, \alpha_1 \in \mathcal{D}$ *and* $\beta_0, \beta_1 \in \mathcal{R}$.
  2. *The challenger samples a bit* $b \leftarrow \{0, 1\}$ *and computes* $(k_0, k_1, \ldots, k_{n-1}) \leftarrow \mathsf{Gen}(1^\lambda, \alpha_b, \beta_b)$.
  3. *The adversary outputs a guess* $b' \leftarrow \mathcal{A}((k_i)_{i \in T}, \mathsf{state})$*, given the keys for the corrupted* T.

  Denote by $\mathbf{Adv}(1^\lambda, \mathcal{A}) := \left|\Pr(b' = b) - \frac{1}{2}\right|$*, the advantage of* $\mathcal{A}$ *in guessing* b *in the above experiment, where probability is taken over the randomness of the challenger and of* $\mathcal{A}$*. We say that the scheme is* (t, v)*-secure if for all non-uniform PPT adversaries* $\mathcal{A}$*, it holds that* $\mathbf{Adv}(1^\lambda, \mathcal{A}) \leqslant v(\lambda)$.

A few remarks are in order.

1. We do not require the function v to necessarily be negligible, as we deal with both negligible and inverse polynomial functions.
2. The indistinguishability based security notion defined here is equivalent to a simulation-based one, where one must be able to simulate the distribution of corrupt parties' shares without knowledge of the shared point function [BGI16].
3. While the above definition is more general, in this work, we will focus on point functions of the form $f_{\alpha,\beta} : \mathcal{D} \to \mathbb{F}_{p^q}$, where p is an arbitrary constant prime.

4. While the Eval algorithm is formally defined as taking a single input $x \in \mathcal{D}$ and returning shares of the corresponding output, as in prior works [BGIK22b], our construction evaluates the entire domain in a single pass. Consequently, our DPF is only applicable to point functions over domains of polynomial size, a setting commonly referred to as small-domain DPF in the literature [BGIK22b].

5. **Maximal Threshold $t = (n-1)$.** For most of this paper, we focus on the maximal corruption threshold $t = n - 1$. In this case, the reconstruction algorithm Recon is simply a sum over elements in $\mathcal{R}$. When referring to security in this setting, we use the term ν-security and omit the explicit mention of $t = n - 1$ for simplicity.

6. **Threshold $t < (n-1)$.** In Section 4.5, we consider a variant with a smaller corruption threshold, referred to as the (t, n)-threshold DPF. Here, privacy is preserved against any coalition of up to t corrupt parties, and reconstruction can be performed using any subset of $t + 1$ evaluated shares via the Recon algorithm.

4 Multiparty Distributed Point Functions

In this section, we present our construction of a multiparty distributed point function (DPF). In Section 4.1, we show that the multiparty DPF of Gilboa et al. [GI14] can be viewed as reducing the problem of designing a multiparty DPF to constructing a *combinatorial design* with specific properties, which we formalize. In Section 4.2, we prove that the implicit design used in the construction of [BGI15] is optimal, by showing that any such *deterministic* combinatorial design must be exponential in the number of parties. Section 4.3 presents a *randomized* combinatorial design satisfying the required properties, yielding a scheme with inverse polynomial security. In Section 4.4, we apply security amplification techniques from [BGIK22b] to achieve negligible security loss. Finally, in Section 4.5, we extend our construction to support threshold DPFs for certain thresholds.

4.1 Multiparty DPFs from PRGs and a Special Combinatorial Design

As discussed in Section 2, the [GI14] paradigm for designing multiparty DPFs can be viewed as a clever combination of PRGs and a *special combinatorial design*. We start by formalizing the properties needed from such a special combinatorial design.

Definition 5 (Special Combinatorial Design)**.** *Let $\lambda, n \in \mathbb{N}$ and ℓ be a polynomial in λ, n. A* special combinatorial design *consists of two (possibly random) bipartite graphs G_0 and G_1, each with n vertices on the left-hand side and $\ell = \ell(\lambda, n)$ [23] vertices on the right-hand side, satisfying the following properties:*

1. *$(1-\delta)$-**Correctness.** The probability that each right vertex in G_0 has an even degree is at least $1 - \delta$.*
2. *$(1-\rho)$-**Pseudorandomness.** With probability at least $(1-\rho)$, for each left vertex in G_1, there exists a right vertex to which it is exclusively connected.*
3. *ε-**Privacy.** The statistical distance between the subgraphs obtained by removing the i-th (for any $i \in [n]$) left vertices from G_0 and G_1 is at most ε.*

Here δ, ρ and ε are assumed to be functions of λ, n.[24]

[23] We note that for simplicity here we are assuming that both G_0 and G_1 have the same number of right vertices. While this is the case for the deterministic special combinatorial design that is implicit in [BGI15], in our randomized construction we will have different number of right vertices in the two graphs.

[24] We sometimes abuse notation and use $\ell, \delta, \rho, \varepsilon$ instead of $\ell(\lambda, n), \delta(\lambda, n), \rho(\lambda, n), \varepsilon(\lambda, n)$.

We now describe how to construct a multiparty DPF using a PRG and a special combinatorial design that satisfies the properties outlined above. For simplicity, we assume throughout this subsection that the point function is of the form $f_{\alpha,\beta} : \mathcal{D} \to \mathbb{Z}_2$, where $|\mathcal{D}| = N$. At the end of the subsection, we explain how the construction can be extended to support arbitrary fields as the function's range. We start by presenting the main ideas and include a formal description in Algorithms 1, 2 and 3.

Share Generation. As discussed earlier, we reinterpret the N-length output vector of $f_{\alpha,\beta}$ as an $m \times \mu$ matrix M, where $m, \mu = O(\sqrt{N})$. Let the coordinate corresponding to α in this matrix be (j, k). Let $\mathrm{PRG} : \{0,1\}^\lambda \to \{0,1\}^{|\mathbb{F}| \cdot \mu}$ be a pseudorandom generator. Informally, each party receives a set of PRG seeds – one per row of the matrix. For all rows except the j-th, seeds are sampled and distributed according to the bipartite graph G_0; for the j-th row, seeds follow the distribution defined by G_1 from the special combinatorial design.

More precisely, sampling and distributing seeds according to a graph G_b works as follows: for each right vertex in G_b, sample a random seed. Each left vertex corresponds to a party, and each party i receives the seeds associated with the right vertices connected to its corresponding left vertex. A formal description of this is included in Algorithm 3.

In addition to these seeds, each party also receives a share of the zero-bit for all rows other than the j-th, and a share of the one-bit for the j-th row in M. We refer to these as shares of *control* bits. Finally, all parties receive a correction word $\mathbf{cw}$, computed by XORing the j-th row of matrix M with the outputs of the PRG evaluated on the ℓ seeds associated with graph G_1. A formal description of this step is provided in Algorithm 1.

Evaluation. For each row of the matrix M, each party computes its output share of the point function by multiplying the corresponding bit share with the correction word and XORing the result with PRG evaluations on all the seeds obtained for that row. The output share is the concatenation of these strings across all rows.

We note that our DPF evaluation process differs slightly from the original construction in [GI14]. This deviation arises from the need to apply security amplification techniques (see Section 4.4 for details). Rather than compressing the final output, we retain the full domain evaluation as the result. Formal details are provided in Algorithm 2.

Algorithm 1 $\mathrm{Gen}(1^\lambda, \alpha, \beta)$

Let $\mathrm{PRG} : \{0,1\}^\lambda \to \{0,1\}^{|\mathbb{F}| \cdot \mu}$ be a pseudorandom generator.
Interpret α as (j, k).
Compute $(S_1^i, \ldots, S_m^i)_{i \in [n]} \longleftarrow \mathrm{Share}(j, n, m)$ using Algorithm 3.
Sample $n \cdot m$ random bits $(t_1^i, \ldots, t_m^i)_{i \in [n]}$ such that, $\bigoplus_{i=1}^n t_g^i$ is 0 when $g \neq j$, and is 1 otherwise.
Let $\mathbf{cw} \leftarrow \beta \cdot \mathbf{e}_k$ and $i \leftarrow 0$.
for $i < n$ **do**
    Parse S_j^i as a sequence of ℓ_i PRG seeds $(r_{j,1}^i, \ldots, r_{j,\ell_i}^i)$.
    $\mathbf{cw} := \mathbf{cw} \bigoplus_{b=1}^{\ell_i} \mathrm{PRG}(r_{j,b}^i)$.
    $i := i + 1$.
end for
For each $i \in [n]$, let $K_i \leftarrow \left(S_1^i, t_1^i, \ldots, S_m^i, t_m^i, \mathbf{cw}\right)$. Send K_i to party i.

Algorithm 2 Eval(i, k_i, x)

Let PRG : $\{0,1\}^\lambda \to \{0,1\}^{\mu \cdot |\mathbb{F}|}$ be a pseudorandom generator.
Parse K_i as $\left(S_1^i, t_1^i, \ldots, S_m^i, t_m^i, \mathbf{cw}\right)$ and each S_g^i as a sequence of ℓ_i PRG seeds $(r_{g,1}^i, \ldots, r_{g,\ell_i}^i)$
For $g \in [m]$, compute $\mathbf{y}_g^i = t_g^i \cdot \mathbf{cw} \bigoplus_{b=1}^{\ell_i} \mathrm{PRG}(r_{g,b}^i)$
Output $(\mathbf{y}_1^i \parallel, \ldots, \parallel \mathbf{y}_m^i)$.

Algorithm 3 Share(j, n, m)

Let PRG : $\{0,1\}^\lambda \to \{0,1\}^{\mu \cdot |\mathbb{F}|}$ be a pseudorandom generator.
for $g \in [m]$ **do**
    For each $i \in [n]$, let $S_g^i = \emptyset$.
    if $g \neq j$ **then**
        Sample a graph G_0 as per Definition 5. Let this graph be denoted as $G_{0,g}$.
        for each right vertex $b \in [\ell]$ in $G_{0,g}$ **do**
            Sample a PRG seed $r_b^{0,g} \xleftarrow{\$} \{0,1\}^\lambda$.
            for $i \in [n]$ **do**
                If the b-th right vertex is connected to the i-th left vertex, then update $S_g^i \leftarrow S_g^i \cup \{r_b^{0,g}\}$
            end for
        end for
    else
        Sample a graph G_1 as per Definition 5. Let this graph be denoted as $G_{1,j}$.
        for each right vertex $b \in [\ell]$ in $G_{1,j}$ **do**
            Sample a PRG seed $r_b^{1,g} \xleftarrow{\$} \{0,1\}^\lambda$.
            for $i \in [n]$ **do**
                If the b-th right vertex is connected to the i-th left vertex, then update $S_j^i \leftarrow S_j^i \cup \{r_b^{1,g}\}$
            end for
        end for
    end if
end for
return $(S_1^1, \ldots, S_m^1), \ldots, (S_1^n, \ldots, S_m^n)$

Theorem 1. *Let λ be the security parameter, n the number of parties, and let $f_{\alpha,\beta} : \mathcal{D} \to \mathbb{Z}_2$ be a point function with $|\mathcal{D}| = N$. Assume $m \cdot \mu = N$, where $m, \mu = O(\sqrt{N})$. Given a pseudorandom generator $\mathrm{PRG} : \{0,1\}^\lambda \to \{0,1\}^{|\mathbb{F}| \cdot \mu}$ and a special combinatorial design (see Definition 5) with $(1-\delta)$-correctness, $(1-\rho)$-pseudorandomness, and ε-privacy, Algorithms 1 and 2 yield a multiparty DPF with:*

- *$O(\varepsilon + \rho)$ computational security,*
- *$O(1 - \sqrt{N} \cdot \delta)$ correctness, and*
- *$O(\ell \cdot \sqrt{N})$ per-party share size.*

Proof. We prove each of the above proerties:

- **Security**: Let the advantage of any computationally bounded adversary in breaking the pseudorandom generator PRG be $\mathsf{negl}(\lambda)$. Let $\mathbf{K}^{\alpha_1}$ and $\mathbf{K}^{\alpha_2}$ denote the set of all shares generated by $\mathsf{Gen}(1^\lambda, \alpha_1, \beta)$ and $\mathsf{Gen}(1^\lambda, \alpha_2, \beta)$, respectively. Our goal is to bound the maximum advantage of any computationally bounded adversary in distinguishing between the shares of corrupt parties in $\mathbf{K}^{\alpha_1}$ and $\mathbf{K}^{\alpha_2}$.

  This advantage can be upper bounded by the sum of three components:

  1. The probability of breaking the PRG, which is $\mathsf{negl}(\lambda)$.
  2. The probability that the PRG seed is leaked due to failure of the psudorandomness property in Definition 5, which is at most $1 - (1-p)^2$.
  3. The statistical distance between the distributions of PRG seeds given to the corrupt parties in the two configurations, which is at most 2ε. To see this, let (j_1, k_1) and (j_2, k_2) be the positions in the $m \times \mu$ matrix corresponding to α_1 and α_2, respectively. Let

  $$((G^{\alpha_1}_{0,g})_{g\in[m],\, g\neq j_1}, G^{\alpha_1}_{1,j_1}) \quad \text{and} \quad ((G^{\alpha_2}_{0,g})_{g\in[m],\, g\neq j_2}, G^{\alpha_2}_{1,j_2})$$

  be the sets of bipartite graphs sampled by $\mathsf{Gen}(1^\lambda, \alpha_1, \beta)$ and $\mathsf{Gen}(1^\lambda, \alpha_2, \beta)$, respectively. Observe that $(G^{\alpha_1}_{0,g})_{g\in[m],\, g\notin\{j_1,j_2\}}$ and $(G^{\alpha_2}_{0,g})_{g\in[m],\, g\notin\{j_1,j_2\}}$ are identically distributed. Thus,

  $$\begin{aligned}&\mathsf{SD}\left(((G^{\alpha_1}_{0,g})_{g\neq j_1}, G^{\alpha_1}_{1,j_1}),\ ((G^{\alpha_2}_{0,g})_{g\neq j_2}, G^{\alpha_2}_{1,j_2})\right)\\ \leqslant\ &\mathsf{SD}(G^{\alpha_1}_{1,j_1}, G^{\alpha_2}_{0,j_1}) + \mathsf{SD}(G^{\alpha_1}_{0,j_2}, G^{\alpha_2}_{1,j_2}) \leqslant 2\varepsilon.\end{aligned}$$

  Therefore, the advantage is $\leqslant \mathsf{negl}(\lambda) + \left(1 - (1-\rho)^2\right) + 2\varepsilon = O(\varepsilon + \rho)$

  We note that it suffices to analyze the difference in the PRG seeds included in the DPF shares. The shares of the control bits can be ignored in this analysis. This is because, by triangle inequality, when $g \neq j$, it follows that:

  $$\begin{aligned}&\mathsf{SD}\left((S^i_g, t^i_g)_{i\in C}, (S^i_j, t^i_j)_{i\in C}\right)\\ \leqslant\ &\mathsf{SD}\left((S^i_g)_{i\in C}, (S^i_j)_{i\in C}\right) + \mathsf{SD}\left((t^i_g)_{i\in C}, (t^i_j)_{i\in C}\right),\end{aligned}$$

  where C is the subset of parties corrupted by the adversary. Since t's are additive shares of 0 or 1, $\mathsf{SD}\left((t^i_g)_{i\in C}, (t^i_j)_{i\in C}\right)$ is 0.

- **Correctness**: The correctness of our construction follows analogously to the approach in [BGI15] and is straightforward to verify. When Algorithm 3 is instantiated with a combinatorial design satisfying $(1-\delta)$-correctness, there are $m-1$ instances of G_0 used in total. Therefore, by a union bound, the overall probability of correctness in the resulting multiparty DPF is at least $(1-\delta)^{m-1} = O(1 - \sqrt{N}\cdot\delta)$, since $m = O(\sqrt{N})$.

- **Efficiency**: Observe that Algorithm 1 samples a total of $m \cdot \ell$ balls PRG seeds. Therefore, the per-party share size is bounded by $O(\ell \cdot \sqrt{N})$.

□

Point Functions over Arbitrary Fields. Thus far, for simplicity, we have focused on point functions with outputs in $\mathbb{Z}_2$. To support an arbitrary field $\mathbb{F}$ as the range, only minor modifications are needed: replace all $\bigoplus$ operations with addition over $\mathbb{F}$, and augment each PRG seed with an additional bit to indicate its sign. Specifically, given a seed x, the output becomes $(-1)^b \cdot \mathsf{PRG}(x)$, where $b \in \{0,1\}$ is the sign bit.

4.2 Lower Bound

Before presenting our construction of a special combinatorial design, we first show that any deterministic combinatorial design must require an exponential (in the number of parties) number of right vertices in both G_0 and G_1. As discussed in Section 2, the multiparty DPF of [BGI15] can be interpreted as an instantiation of our template from the previous section, using a deterministic special combinatorial design. The design implicit in their construction indeed requires an exponential number of right vertices in both G_0 and G_1, resulting in a multiparty DPF with share sizes that grow exponentially with the number of parties. Our lower bound formally establishes that this is inherent, thereby showing that their construction is *optimal*.

Theorem 2. *Let $n \in \mathbb{N}$ be the number of left vertices in both G_0 and G_1 in a special combinatorial design (see Definition 5). If the design is deterministic, then both G_0 and G_1 must have $\Omega(2^n)$ right vertices.*

Proof. Let us begin by introducing some notation. We represent the incidence pattern of each right vertex in G_0 and G_1 using binary vectors of length n. For example, $\mathbf{b} = (1, 0, 1, 0, \ldots, 0)$ indicates that the right vertex is connected to the first and third left vertices in the corresponding graph. The Hamming weight of such a vector, denoted $\mathrm{wt}(\mathbf{b})$, is the number of 1s in the vector.

We claim that for every n-bit binary vector with even Hamming weight, there must exist a corresponding right vertex in G_0. Similarly, for every n-bit binary vector with odd Hamming weight, there must exist a corresponding right vertex in G_1. We prove this claim by induction on the Hamming weight.

- **Base Case:** By the *pseudorandomness* property of the special combinatorial design (see Definition 5), we know that for every n-bit binary vector with $\mathrm{wt} = 1$, there exists a corresponding right vertex in G_1.
- **Inductive Hypothesis:** Suppose that for all $i \in [n-1]$, every n-bit vector with $\mathrm{wt} = i$ corresponds to some right vertex in G_1 if i is odd, and in G_0 if i is even.
- **Inductive Step:** We consider the cases of odd and even i separately:
  - i **is odd:** Suppose the above claim holds for all vectors with $\mathrm{wt} \leqslant i$, where i is odd. Consider an arbitrary right vertex in G_1 corresponding to a vector b_1^i with $\mathrm{wt}(b_1^i) = i$. Choose any coordinate $j \in [n]$ such that the j-th entry of b_1^i is 0, and assume the j-th party is honest.

    By the *privacy* property of the special combinatorial design (see Definition 5), the subgraph observed by the corrupt parties must be indistinguishable from that in the alternate configuration (i.e., G_0). Therefore, there must exist a right vertext corresponding to some vector b_0^x in G_0 such that it is identical to b_1^i except possibly at coordinate j.

    Since i is odd and the *correctness* property of a special combinatorial design (see Definition 5) requires right vertices in G_0 to have even degree, we conclude $\mathrm{wt}(b_0^x) = i + 1$. Because both the choice of j and the vector b_1^i were arbitrary, this implies that a right vertex corresponding to every vector with Hamming weight $i + 1$ must appear in G_0.
  - i **is even:** If i is an even number, then by the induction hypothesis, all vectors with Hamming weight i have an associated right vertex in G_0. Consider an arbitrary right vertex in G_1 corresponding to a vector b_0^i with $\mathrm{wt}(b_0^i) = i$. Choose any coordinate $j \in [n]$ such that the j-th entry of b_0^i is 0, and assume the j-th party is honest.

By the *privacy* property of the special combinatorial design, there must exist a right vertex corresponding to vector b_1^x in G_1 such that it is identical to b_0^i, except possibly at coordinate j. However, at this point, we cannot directly conclude that $\mathrm{wt}(b_1^x) = i+1$. To show that a right vertex corresponding to such a vector with Hamming weight $i+1$ must exist in G_1, we proceed by contradiction.

Suppose all such vectors b_1^x in G_1 matching b_0^i at all but possibly one position (specifically, the j-th position) have Hamming weight different from $i+1$ and let there be m such vectors. Let $S = \{j' \in [n] \mid b_0^i[j'] = 1\}$, and choose any $j' \in S$ as the honest party. By the induction hypothesis, there exists a right vertex corresponding to vector b_1^{i-1} in G_1 with Hamming weight $i-1$ such that it is identical to b_0^i in all but (possibly) the j'-th location.

Now, both b_1^x and b_1^{i-1} match b_0^i on all but one coordinate, but have different Hamming weights. In particular, we now have $\geqslant m+1$ vectors of the form b_1^x that match b_0^i in all but possibly the j'-th location and only m vectors of the form b_1^x that match b_0^i in all but possibly the j-th location.

But this clearly violates the *privacy* property of the special combinatorial design, since the corrupt parties would see two inconsistent matches for b_0^i. Therefore, our assumption must be false, and such a vector with Hamming weight $i+1$ must exist in G_1. Since the choice of b_0^i and the honest party is arbitrary, it follows that every vector with Hamming weight $i+1$ appears in G_1.

Putting everything together, we have proven the claim by induction. Finally, note that the number of binary vectors of Hamming weight i is $\binom{n}{i}$. Therefore, the number of right vertices in each of G_0 and G_1 is at least:

$$\sum_{\substack{0 \leqslant i \leqslant n \\ i \text{ even}}} \binom{n}{i} = \sum_{\substack{1 \leqslant i \leqslant n \\ i \text{ odd}}} \binom{n}{i} = 2^{n-1}.$$

□

4.3 Randomized Special Combinatorial Design

To circumvent the above lower bound, we now present a new *randomized* construction of a special combinatorial design.

Our Construction. We propose to define the two graphs as follows:

1. **Graph G_0:** This graph has n left vertices and M right vertices. For each of the M right vertices, we randomly sample (with repetition) two left vertices[25] and add edges between the right vertex and the selected left vertices.

2. **Graph G_1:** This graph has n left vertices and $M+t$ right vertices. The first M vertices are connected to two (posssibly same) randomly picked left vertices, similarly to G_1. For each of the remaining t right vertices, we sample one random left vertex and introduce an edge between them.

We now show that for appropriate values of M and t, this is a construction of a special combinatorial design.

Theorem 3. *Let $\lambda, n \in \mathbb{N}$. For parameters $M = \mathsf{poly}(\lambda, n)$ and $t = \mathsf{poly}(\lambda, n)$, the above construction yields a special combinatorial design (see Definition 5) satisfying the following properties:*

[25] Recall from Section 2 that it is possible for the same left vertex to be selected twice, in which case we add two edges between this pair of vertices.

1. 1-***Correctness***.
2. $\left(1-\exp\left(-\Omega\left(\frac{t}{n}\right)\right)\right)$-***Pseudorandomness***.
3. $O\left(t\cdot\sqrt{\frac{n^2}{M}}\right)$-***Privacy***.

In particular, setting $t=\Omega(n\cdot\lambda)$ and $M=\Omega(n^2\cdot t^2\cdot\lambda^2)$ yields a special combinatorial design with negligible pseudorandomness error and inverse polynomial (in λ) privacy.

Before further analysis, we import two theorems and related definitions that are useful for our proof.

Definition 6 (Translated Poisson Distribution [Röl06])**.** *An integer random variable Y is distributed according to the translated Poisson distribution with parameter σ^2 and μ if and only if Y can be written in the form of $Y=Z+\lfloor\mu-\sigma^2\rfloor$. Z is distributed according to the Poisson distribution with parameter $\sigma^2+\{\mu-\sigma^2\}$ where $\{\mu-\sigma^2\}$ is the fractional part of $\mu-\sigma^2$. We denote this distribution with $\mathsf{TP}(\mu,\sigma^2)$.*

Lemma 1 (Equation (3.4) of [Röl06])**.** *Let $J_1,J_2\cdots J_n$ be independent random variables distributed according to Bernoulli distributions $\mathsf{Bern}(p_1),\mathsf{Bern}(p_2)\cdots\mathsf{Bern}(p_n)$. Denote $\mu=\sum_{i=1}^n p_i$ and $\sigma^2=\sum_{i=1}^n p_i(1-p_i)$. Then we have*

$$\mathsf{SD}\left(\sum_{i=1}^n J_i,\mathsf{TP}(\mu,\sigma^2)\right)\leqslant\frac{\sqrt{\sum_{i=1}^n p_i^3(1-p_i)}+2}{\sigma^2}.$$

Lemma 2 (Lemma 2.1 of [BLU06])**.** *Let $\mu_1,\mu_2\in\mathbb{R}$ and $\sigma_1^2,\sigma_2^2\in\mathbb{R}_+$. If $\lfloor\mu_1-\sigma_1^2\rfloor\leqslant\lfloor\mu_2-\sigma_2^2\rfloor$, we have*

$$\mathsf{SD}(\mathsf{TP}(\mu_1,\sigma_1^2),\mathsf{TP}(\mu_2,\sigma_2^2))\leqslant\frac{|\mu_1-\mu_2|}{\sigma_1}+\frac{|\sigma_1^2-\sigma_2^2|+1}{\sigma_1^2}.$$

Now we are ready to give a proof of Theorem 3.

Proof of Theorem 3. We prove each property of the special combinatorial design in order.

- 1-**Correctness**: In our special combinatorial design, since each right vertex in G_0 has exactly two edges, this means the δ in Property 1 of Definition 5 is always zero.
- $\left(1-\exp\left(-\Omega\left(\frac{t}{n}\right)\right)\right)$-**Pseudorandomness**: We argue that the term ρ in Definition 5 is $\exp\left(-\Omega\left(\frac{t}{n}\right)\right)$. Let us define E as the following event: *For any left vertex in G_1, there exists at least one right vertex connected exclusively to it* and Bad_i as the following event: *The i-th left vertex is not connected to any of the additional degree-1 right vertices in G_1*. By definition, we have

$$\begin{aligned}\rho=\Pr(\neg E)&=\Pr(\mathsf{Bad}_1\cup\mathsf{Bad}_2\cup\ldots\cup\mathsf{Bad}_n)\\&\leqslant\sum_{i=1}^n\Pr(\mathsf{Bad}_i)=n\cdot\left(\frac{n-1}{n}\right)^t,\end{aligned}$$

  where the inequality holds by union bound. We conclude the proof by noting that

$$n\cdot\left(\frac{n-1}{n}\right)^t=n\cdot\left(1-\frac{1}{n}\right)^{n\cdot\frac{t}{n}}=\exp\left(-\Omega\left(\frac{t}{n}\right)\right).$$

- $O\left(t \cdot \sqrt{\frac{n^2}{M}}\right)$ – **Privacy**: For simplicity, we will interchangeably use *balls* and *bins* to denote the right and left nodes, respectively. For privacy, we want to argue that given any proper subset of the left vertices, the distribution of the corresponding induced (random) subgraphs of G_0 and G_1 are statistically close. Due to symmetry and without loss of generality, we will assume that the last left vertex (last bin) is not in the adversary's view and focus on the subgraphs induced by the first $n-1$ left vertices (first $n-1$ bins).

  We introduce some notations first. Let C (resp., C') denote the configuration of the balls (i.e., the histogram representing how many balls are in each bin) in G_0 (resp., G_1). Therefore, we are interested in $\mathrm{SD}(C, C')$. Note that C can be represented as a joint distribution of $(\mathcal{S}, \mathcal{D}, \mathcal{H}_s, \mathcal{H}_d)$.[26] Here, $\mathcal{S}$ and $\mathcal{D}$ are the random variables denoting *the number of* balls tossed once and twice, respectively. $\mathcal{H}_s$ and $\mathcal{H}_d$ are the random variables denoting the *configuration* of those balls tossed once and twice. Similarly, for G_1, we define $(\mathcal{S}', \mathcal{D}', \mathcal{H}'_s, \mathcal{H}'_d)$.

  We first note that

$$\mathrm{SD}\left((\mathcal{S}, \mathcal{D}, \mathcal{H}_s, \mathcal{H}_d), (\mathcal{S}', \mathcal{D}', \mathcal{H}'_s, \mathcal{H}'_d)\right) = \mathrm{SD}\left((\mathcal{S}, \mathcal{D}), (\mathcal{S}', \mathcal{D}')\right).$$

  This is because, conditioned on the numbers of single-tossed balls and twice-tossed balls, the configurations of the balls are *identically distributed* in G_0 and G_1 (i.e., the bin each ball goes to is independently randomly chosen). Therefore, in the rest of the analysis, we simply focus on the joint distributions $(\mathcal{S}, \mathcal{D})$ and $(\mathcal{S}', \mathcal{D}')$.

  Next, we note that the distributions of $\mathcal{D}$ and $\mathcal{D}'$ are identical.[27] Therefore, we may write

$$\mathrm{SD}\left((\mathcal{S}, \mathcal{D}), (\mathcal{S}', \mathcal{D}')\right) = \mathop{\mathrm{E}}_{d}\left[\mathrm{SD}\left(\left(\mathcal{S}|\mathcal{D}=d\right), \left(\mathcal{S}'|\mathcal{D}'=d\right)\right)\right].$$

  To upper bound $\mathrm{SD}\left(\left(\mathcal{S}|\mathcal{D}=d\right), \left(\mathcal{S}'|\mathcal{D}'=d\right)\right)$, we will approximate both distributions by translated Poisson distributions and rely triangle inequality to argue their closeness. Note that, before conditioning on $\mathcal{D}=d$, each of those M balls have probability

$$\left(\frac{n-1}{n}\right)^2, \quad \frac{2(n-1)}{n^2}, \text{ and } \frac{1}{n^2}$$

  to be a twice-tossed, single-tossed, or zero-times-tossed ball. After conditioning on $\mathcal{D}=d$, we know that the remaining $M-d$ balls have *conditional* probability

$$\frac{2(n-1)}{2n-1} \text{ and } \frac{1}{2n-1}$$

  to be a single-tossed or zero-times-tossed ball. Therefore, we may write

$$\mathcal{S} = \sum_{i=1}^{M-d} \mathcal{J}_i + \sum_{i=M-d+1}^{M-d+t} \mathcal{J}'_i$$
$$\mathcal{S}' = \sum_{i=1}^{M-d} \mathcal{J}_i + \sum_{i=M-d+1}^{M-d+t} \mathcal{J}''_i$$

[26] There could be balls that are tossed *zero times* into the first $n-1$ bins. Those balls do not affect the distribution of the configuration (i.e., subgraph).

[27] The only difference between $\mathcal{D}$ and $\mathcal{D}'$ is due to the additional t balls. However, they are only tossed once and, thus, they have no effect on the distribution of the number of balls tossed twice.

where

$$\mathcal{J}_i = \mathsf{Bern}\left(\frac{2(n-1)}{2n-1}\right)$$
$$\mathcal{J}_i' = \mathsf{Bern}(0)$$
$$\mathcal{J}_i'' = \mathsf{Bern}\left(\frac{n-1}{n}\right)$$

Now, we invoke Lemma 1 (adopting its notation for p_i), we have

$$\mathsf{SD}\left(\mathcal{S}, \mathsf{TP}(\mu_1, \sigma_1^2)\right) \leqslant \frac{\sqrt{\sum_{i=1}^{M-d+t} p_i^3(1-p_i)} + 2}{\sigma_1^2} = \frac{\frac{2(n-1)}{2n-1} \cdot \sigma_1 + 2}{\sigma_1^2},$$

where

$$\sigma_1^2 = (M-d) \cdot \frac{2(n-1)}{(2n-1)^2} \qquad \text{and} \qquad \mu_1 = (M-d) \cdot \frac{2(n-1)}{2n-1}.$$

Note that the above bound is $O\left(\sqrt{\frac{n}{M-d}}\right)$. Similarly, we also have

$$\mathsf{SD}\left(\mathcal{S}', \mathsf{TP}(\mu_2, \sigma_2^2)\right) \leqslant \frac{\sqrt{\sum_{i=1}^{M-d+t} p_i^3(1-p_i)} + 2}{\sigma_2^2} = \frac{\sqrt{\frac{(2(n-1))^3}{(2n-1)^4} \cdot (M-d) + \frac{(n-1)^3}{n^4} \cdot t + 2}}{\sigma_2^2},$$

where

$$\sigma_2^2 = (M-d) \cdot \frac{2(n-1)}{(2n-1)^2} + t \cdot \frac{n-1}{n^2} \qquad \text{and} \qquad \mu_2 = (M-d) \cdot \frac{2(n-1)}{2n-1} + t \cdot \frac{n-1}{n}.$$

Again, this bound is $O\left(\sqrt{\frac{n}{M-d}}\right)$. Now, apply Lemma 2, we have

$$\mathsf{SD}\left(\mathsf{TP}(\mu_1, \sigma_1^2), \mathsf{TP}(\mu_2, \sigma_2^2)\right) \leqslant \frac{t \cdot \frac{n-1}{n}}{\sigma_1} + \frac{1}{\sigma_1^2},$$

which is $O\left(t \cdot \sqrt{\frac{n}{M-d}}\right)$. This gives

$$\begin{aligned}
&\mathsf{SD}((\mathcal{S} \mid \mathcal{D} = d), (\mathcal{S}' \mid \mathcal{D}' = d)) \\
&\leqslant \mathsf{SD}\left(\left(\mathcal{S}|\mathcal{D} = d\right), \mathsf{TP}(\mu_1, \sigma_1^2)\right) + \mathsf{SD}\left(\mathsf{TP}(\mu_1, \sigma_1^2), \mathsf{TP}(\mu_2, \sigma_2^2)\right) + \mathsf{SD}\left(\mathsf{TP}(\mu_2, \sigma_2^2), \left(\mathcal{S}'|\mathcal{D}' = d\right)\right) \\
&= O\left(t\sqrt{\frac{n}{M-d}}\right).
\end{aligned}$$

Now we go back to

$$\mathop{\mathrm{E}}_{d}\left[\mathsf{SD}\left(\left(\mathcal{S}|\mathcal{D} = d\right), \left(\mathcal{S}'|\mathcal{D}' = d\right)\right)\right].$$

Let us use $\mathcal{D}_i$ as the indicator of the event that the i-th ball (among the M balls) is tossed twice into the remaining bins. Note that $\mathcal{D} = \sum_{i=1}^{M} \mathcal{D}_i$. Define $\mu := \mathrm{E}[\mathcal{D}] = M \cdot (\frac{n-1}{n})^2$. By Chernoff's bounds, we have

$$\Pr\left(\sum_{i=1}^{M} \mathcal{D}_i \geqslant (1+\delta) \cdot \mu\right) \leqslant \exp\left(-\frac{\delta^2 \mu}{2+\delta}\right).$$

Setting $\delta = \frac{1}{n-1}$, we get

$$\Pr\left[M - \sum_{i=1}^{M} \mathcal{D}_i \leqslant \frac{M}{n}\right] = \Pr\left[\sum_{i=1}^{M} \mathcal{D}_i \geqslant \frac{n-1}{n} \cdot M\right] \leqslant \exp\left(-\frac{M}{2n^2 + \frac{n^2}{n-1}}\right).$$

Finally, we conclude the proof by noting that

$$\begin{aligned}
&\mathop{\mathrm{E}}_{d}\left[\mathrm{SD}(\ (\mathcal{S} \mid \mathcal{D} = d), (\mathcal{S}' \mid \mathcal{D}' = d))\right] \\
&= \sum_{d \geqslant M \cdot \frac{n-1}{n}} \Pr(\mathcal{D} = d) \cdot \mathrm{SD}\left((\mathcal{S} \mid \mathcal{D} = d), (\mathcal{S}' \mid \mathcal{D}' = d)\right) \\
&\qquad + \sum_{d \leqslant M \cdot \frac{n-1}{n}} \Pr(\mathcal{D} = d) \cdot \mathrm{SD}\left((\mathcal{S} \mid \mathcal{D} = d), (\mathcal{S}' \mid \mathcal{D}' = d)\right) \\
&\qquad \text{Note that the bound } \sqrt{\frac{n}{M-d}} \text{ is monotonically decreasing in } d \\
&\leqslant \Pr\left(\mathcal{D} \geqslant M \cdot \frac{n-1}{n}\right) + \left(1 - \Pr\left(\mathcal{D} \geqslant M \cdot \frac{n-1}{n}\right)\right) \cdot O\left(t\sqrt{\frac{n}{M - \frac{M(n-1)}{n}}}\right) \\
&\leqslant \exp\left(-\frac{M}{2n^2 + \frac{n^2}{n-1}}\right) + O\left(t\sqrt{\frac{n^2}{M}}\right),
\end{aligned}$$

which is $O\left(t\sqrt{\frac{n^2}{M}}\right)$ when M is sufficiently big, e.g., $\geqslant n^2 \cdot t^2$. Note that when $M < n^2 \cdot t^2$, the bound is trivial (i.e., bigger than one). Therefore, the bound $O\left(t\sqrt{\frac{n^2}{M}}\right)$ hold regardless.

□

4.4 Privacy Amplification with Near-linear Stretch

Our randomized special combinatorial design only achieves inverse polynomial (in λ) privacy. Using this to instantiate Theorem 1 yields a multiparty DPF with inverse polynomial loss in security. To obtain a multiparty DPF with negligible security error, we borrow an amplification technique from [BGIK22b, Section 3.1]. We present the main ideas of their technique here and refer the readers to their paper for further details.

Overview of the Amplification Technique from [BGIK22b]. Their idea is to first view the point function $f_{\alpha,\beta}$ with domain size N as a unit vector e_α of the same length.[28] Let x be the input on which we wish to evaluate this point function. Observe that the vector $\left(f_{1,\beta}(x), f_{2,\beta}(x), \ldots, f_{\alpha,\beta}(x), \ldots, f_{N,\beta}(x)\right)$ is identical to $\beta \cdot e_x = f_{\alpha,\beta}(x)$.

We can now use an instance of Reed-Muller codes (RM-codes) as a q-query, locally decodable code C to encode this vector as $C(e_x)$. Given this encoding, let $\Delta_1, \Delta_2, \ldots, \Delta_q$ be the set of queries to retrieve the

[28] This is why we defined Eval in Algorithm 2 such that it evaluates the entire domain in one go. See the third remark of Definition 4.

α-th entry of the encoded vector. The α-th entry in vector $\beta \cdot e_x$ can be computed as

$$\beta \cdot e_x[\alpha] = \langle \beta \cdot e_x, e_\alpha \rangle = \beta \sum_{g=1}^{q} C(e_x)[\Delta_g] = \beta \sum_{g=1}^{q} \left\langle C(e_x), e_{\Delta_g} \right\rangle$$

Each e_{Δ_g} (essentially a point function) can now be shared using our multiparty DPF with inverse polynomial privacy error. In effect, this reduces the problem of constructing a DPF for $f_{\alpha,\beta}$ with strong privacy guarantees to the problem of constructing DPFs for each e_{Δ_g} – each of which individually may have weaker privacy. Although the DPF for each e_{Δ_g} may incur non-negligible privacy leakage, the overall scheme achieves negligible leakage. This is analogous to using a not-quite-perfect additive secret sharing scheme for the original DPF, where the aggregate leakage remains negligible due to the properties of the Reed-Muller (RM) code (specifically, Property 3 in Lemma 3).

We note that the RM-code decoding in our construction involves evaluating over a curve. For further details, we refer the reader to [Yek10].

Lemma 3 (Lemma 2 from [BGIK22b]). *Fix integers $\sigma, N, r, w > 0$, such that $N \leqslant \binom{r+w}{w}$ and let p be a prime. There exists a deterministic mapping[29] $C : \mathbb{Z}_p^N \to \mathbb{Z}_p^L$ and a randomized mapping $d : [N] \to [L]^q$, $L, q \in \mathbb{N}$, such that for every $z \in \mathbb{Z}_p^N$ and $\alpha \in [N]$ it holds that*

$$\Pr\left(\Delta \leftarrow d(\alpha) : \sum_{g=1}^{q} C(z)[\Delta_g] = z_\alpha\right) = 1,$$

where the probability is taken over the internal randomness of d. Moreover, the following properties hold:

1. *$q = O(\sigma^2 r)$ and $L = O(p^{\omega+1}\sigma^{\omega+1}r^{\omega+1})$.*
2. *C and d can be computed efficiently.*
3. *For every $\alpha \in [N]$, the random variables $\Delta_1, \ldots, \Delta_q$ are σ-wise independent. Namely, the marginal distribution of any σ coordinates is independent of α.*

C will encode a vector of size N to another vector of size $L > N$. A randomized mapping d determines the set of q queried symbols of the codeword given a target index $\alpha \in [N]$ of the "message" vector. The decoding procedure is simply the sum of the queried symbols $\sum_{g=1}^{q} C(z)[\Delta_g]$. For simplicity, we let p be 2.

Our Parameters. When using the RM codes, we can actually set L to be less than N^2 (otherwise the overall construction becomes trivial). These Δ's will then correspond to evaluations of a random σ-degree polynomial (similar to Shamir's secret sharing). We argue here that the parameters can be set to meet our demands, and we refer the readers to [BGIK22b] for details on security proofs.

Formally, we have the following theorem.

Theorem 4. *Let $\sigma, w, r, N \in \mathbb{N}$ and let $\varepsilon > 0$ be a real number such that $N \leqslant \binom{r+w}{w}$. Assume the prime p is constant, and let $q = (\sigma+1)(\sigma r+1)$. Suppose there exists an n-party DPF with share size $O(\mathrm{poly}(n) \cdot \sqrt{L})$ and privacy error $O(1/q)$, for point functions with L-sized domain and outputs in $\mathbb{Z}_p$. Then using Reed-Muller codes and following the transformation of [BGIK22b, Theorem 6], we obtain an n-party DPF with share size $O(\mathrm{poly}(n) \cdot N^{1/2+\varepsilon})$ and privacy error $O(2^{-\Omega(\sigma)} + \mathsf{negl}(\lambda))$, for point functions with N-sized domain and outputs in $\mathbb{Z}_p$.*

[29] Basically, C is the RM encoding when one treats the message as a w-variate r-degree multivariate polynomial.

Proof. We employ Reed–Muller (RM) codes that encode messages of length N into codewords of length L. The point function is split into q smaller point functions, so the overall share size becomes $O(q \cdot \mathrm{poly}(n) \cdot \sqrt{L})$. We begin by setting the RM-code parameter $r = \mathrm{poly}(\log N)$, which implies $q = \mathrm{poly}(\log N)$ (since $q = (\sigma+1)(\sigma r+1)$). Hence, the dominant contributor to the share size is L. Our goal is to choose r and w such that the constraint $N \leqslant \binom{r+w}{w}$ holds, and L remains near-linear in N. From Lemma 3, we have

$$L = O(p^{\omega+1} \cdot \sigma^{\omega+1} \cdot r^{\omega+1}).$$

We fix $\sigma = \log^2(\lambda)$ to ensure negligible privacy error in the final scheme (we refer the reader to [BGIK22b] to details). To target a final share size of $N^{1+\varepsilon}$ for any $\varepsilon > 0$, we set:

$$w = \frac{\delta \cdot \log N}{\log \log N} \qquad \text{and} \qquad r = (\log N)^{1+1/\delta},$$

for a sufficiently small constant $\delta > 0$. To verify that $N \leqslant \binom{r+w}{w}$, observe:

$$\binom{r+w}{w} \geqslant \frac{r^w}{w!} = \frac{N^{1+\delta}}{w!},$$

and using the fact that $w! \leqslant w^w = N^\delta$, we get:

$$\binom{r+w}{w} \geqslant \frac{N^{1+\delta}}{N^\delta} = N.$$

To estimate L, recall:

$$L = O(p^{\omega+1} \cdot \sigma^{\omega+1} \cdot r^{\omega+1}) = N^{o(1)} \cdot N^{\delta \cdot \frac{\log\log\lambda}{\log\log N}} \cdot N^{1+\delta},$$

which is $\leqslant N^{1+\varepsilon}$ for a suitably small choice of δ.

Thus, the resulting scheme achieves domain size N, negligible privacy loss, and share size $O(\mathrm{poly}(n) \cdot N^{1/2+\varepsilon})$, completing the proof.

□

Putting Everything Together. Let us summarize our multiparty DPF construction with the following corollary.

Corollary 1. *Let $\lambda, N, n \in \mathbb{N}$. Let $f_{\alpha,\beta} : \mathcal{D} \to \mathbb{Z}_{p^q}$ be a point function, $|\mathcal{D}| = N$, p be an arbitrary constant prime and ε be an arbitrary constant. Assuming the existence of one-way functions, there exists a construction of a n-party $\mathsf{negl}(\lambda)$-secure distributed point function, where the total share size is $O\left(n^4 \cdot N^{1/2+\varepsilon}\right)$ and each per-party share size is $O\left(n^3 \cdot N^{1/2+\varepsilon}\right)$ in expectation.*

4.5 (t, n)-Threshold DPFs

In this section, we consider a threshold variant of DPFs, where privacy is guaranteed against at most t corruptions and any subset of $t+1$ shares can be used for reconstructing the output. In the following theorem we show a simple transformation from any multiparty DPF to a (t, n)-threshold DPF using standard share conversion techniques from [CDI05].

Theorem 5. *Let $n \in \mathbb{N}$, $f_{\alpha,\beta} : \mathcal{D} \to \mathcal{R}$ be a point function and $(\mathsf{Gen}, \mathsf{Eval})$ be an n-party $(n-1, v)$-secure DPF with total share size s. There exists a $\left(t, n, \binom{n}{t+1} \cdot v\right)$-secure threshold DPF with total share size $s \cdot \binom{n}{t+1}$.*

Proof. Given any $(n-1, n, v)$-secure multiparty DPF $(\mathsf{Gen}, \mathsf{Eval})$, one can construct a simple (t, n)-threshold DPF $(\mathsf{Gen}', \mathsf{Eval}', \mathsf{Recon}')$ with respect to replicated secret sharing as follows:

- $\mathsf{Gen}'(1^\lambda, \alpha, \beta)$: Let $\mathcal{S}$ be the set of all subsets of $0, \ldots, n-1$ of size $t+1$. For each $S \in \mathcal{S}$, compute $(k^S_{S[0]}, \ldots, k^S_{S[n-1]}) \leftarrow \mathsf{Gen}(1^\lambda, \alpha, \beta)$. For each $b \in 0, \ldots, n-1$, the party b receives share $K_b = \{k^S_b \mid S \in \mathcal{S}, b \in S\}$.
- $\mathsf{Eval}'(b, K_b, x)$: Party b parses $K_b = \{k^S_b \mid S \in \mathcal{S}, b \in S\}$. For each $S \in \mathcal{S}$, where $b \in S$, compute $y^S_b \leftarrow \mathsf{Eval}(b, k^S_b, x)$. Output $y_b = \{y^S_b \mid S \in \mathcal{S}, b \in S\}$.
- $\mathsf{Recon}(S, y_{S[0]}, \ldots, y_{S[n-1]})$: For each $i \in \{0, \ldots, n-1\}$, parse $y_{S[i]} = \{y^{S'}_{S[i]} \mid S' \in \mathcal{S}, S[i] \in S'\}$. Output $\sum_{i \in \{0,\ldots,n-1\}} Y^S_{S[i]}$.

Correctness follows from the fact that any subset of $t+1$ parties will have access to all $t+1$ shares associated with one of the underlying multiparty DPF instances, allowing them to reconstruct the output of the point function for any given input. Security follows from the security of each underlying multiparty DPF instance, as no subset of $\leqslant t$ parties will have access to all the keys associated with any of the underlying instances. □

Instantiating the above theorem with the multiparty DPF from Corollary 1, we get the following Corollary:

Corollary 2. *Let $\lambda, N, n \in \mathbb{N}$. Let $f_{\alpha,\beta} : \mathcal{D} \to \mathbb{Z}_{p^q}$ be a point function, $|\mathcal{D}| = N$, p be an arbitrary constant prime and ε be an arbitrary constant, and $t < n$ be such that $\binom{n}{t+1} = \mathsf{poly}(\lambda)$. Assuming the existence of one-way functions, we have an explicit construction of a $(t, n, \mathsf{negl}(\lambda))$-secure distributed point function, where the total share size is $O\big(t^4 \cdot N^{1/2+\varepsilon} \cdot \binom{n}{t+1}\big)$.*

We note that, before our work, one may instantiate Theorem 5 with the multiparty DPFs from [BGI15], yields an efficient (t, n)-threshold DPF only when $t = O(1)$.

Shamir Variant. Cramer et al. [CDI05] proposed a generic method for parties to locally transform their shares associated with a replicated secret sharing scheme into Shamir shares. This approach can be applied to the transformation in Theorem 5 to obtain Shamir shares of the output of a point function.

Acknowledgements

We would like to thank Yuval Ishai for helpful discussions on the applications of multiparty distributed point functions.

References

[ABI+23] Damiano Abram, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Varun Narayanan. Cryptography from planted graphs: Security with logarithmic-size messages. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part I*, volume 14369 of *Lecture Notes in Computer Science*, pages 286–315, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-48615-9_11. 12

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 301–330, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58751-1_11. 4

[BCG+19] 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, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-26954-8_16. 2

[BCG+20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 387–416, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-56880-1_14. 2

[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, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 871–900, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-77886-6_30. 2, 3, 4

[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: 25th Conference on Computer and Communications Security*, pages 896–912, Toronto, ON, Canada, October 15–19, 2018. ACM Press. doi:10.1145/3243734.3243868. 2

[BDSS25] Elette Boyle, Lalita Devadas, and Sacha Servan-Schreiber. Non-interactive distributed point functions. Cryptology ePrint Archive, Paper 2025/095, 2025. URL: https://eprint.iacr.org/2025/095. 2, 3, 4

[BGH+25] Elette Boyle, Niv Gilboa, Matan Hamilis, Yuval Ishai, and Yaxin Tu. Improved Constructions for Distributed Multi-Point Functions . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 44–44, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00044, doi:10.1109/SP61157.2025.00044. 2, 3, 4

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46803-6_12. 2, 3, 4, 5, 6, 7, 8, 14, 17, 18, 26

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. My-

ers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1292–1303, Vienna, Austria, October 24–28, 2016. ACM Press. doi:10.1145/2976749.2978429. 2, 4, 5, 13

[BGI19] Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 341–371, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-36030-6_14. 2, 3, 4

[BGIK22a] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Information-theoretic distributed point functions. In Dana Dachman-Soled, editor, *ITC 2022: 3rd Conference on Information-Theoretic Cryptography*, volume 230 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:14, Cambridge, MA, USA, July 5–7, 2022. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITC.2022.17. 2, 3, 4

[BGIK22b] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Programmable distributed point functions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 121–151, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15985-5_5. 2, 3, 4, 8, 11, 12, 14, 23, 24, 25

[BJRZ24] Andrej Bogdanov, Chris Jones, Alon Rosen, and Ilias Zadik. Low-degree security of the planted random subgraph problem. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part II*, volume 15365 of *Lecture Notes in Computer Science*, pages 255–275, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-78017-2_9. 12

[BKKO20] Paul Bunn, Jonathan Katz, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient 3-party distributed ORAM. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 215–232, Amalfi, Italy, September 14–16, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-57990-6_11. 2

[BLU06] Andrew D. Barbour, Torgny Lindvall, and Göteborgs Universitet. Translated poisson approximation for markov chains. *Journal of Theoretical Probability*, 19:609–630, 2006. URL: https://api.semanticscholar.org/CorpusID:15055580. 20

[BPRS23] Lennart Braun, Mahak Pancholi, Rahul Rachuri, and Mark Simkin. Ramen: Souper fast three-party computation for RAM programs. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 3284–3297, Copenhagen, Denmark, November 26–30, 2023. ACM Press. doi:10.1145/3576915.3623115. 4

[BS24] Katharina Boudgoust and Mark Simkin. The power of NAPs: - compressing OR-proofs via collision-resistant hashing. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part I*, volume 15364 of *Lecture Notes in Computer Science*, pages 35–66, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-78011-0_2. 2

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362, Cambridge, MA, USA, February 10–12, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-30576-7_19. 3, 25, 26

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. doi:10.1109/SFCS.1995.492461. 3

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952. URL: https://api.semanticscholar.org/CorpusID:122118814. 12

[CK24] Geoffroy Couteau and Naman Kumar. 10-party sublinear secure computation from standard assumptions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 39–73, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68400-5_2. 4

[CKY25] Geoffroy Couteau, Naman Kumar, and Xiaxi Ye. Multiparty homomorphic secret sharing and more from LPN and MQ. Cryptology ePrint Archive, Paper 2025/966, 2025. URL: https://eprint.iacr.org/2025/966. 4

[DIL+20] Samuel Dittmer, Yuval Ishai, Steve Lu, Rafail Ostrovsky, Mohamed Elsabagh, Nikolaos Kiourtis, Brian Schulte, and Angelos Stavrou. Function secret sharing for PSI-CA: with applications to private contact tracing. *IACR Cryptol. ePrint Arch.*, page 1599, 2020. URL: https://eprint.iacr.org/2020/1599. 2

[Ds17] Jack Doerner and abhi shelat. Scaling ORAM for secure computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 523–535, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. doi:10.1145/3133956.3133967. 2, 4

[GGM24] Gayathri Garimella, Benjamin Goff, and Peihan Miao. Computation efficient structure-aware PSI from incremental function secret sharing. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VIII*, volume 14927 of *Lecture Notes in Computer Science*, pages 309–345, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68397-8_10. 2

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_35. 2, 3, 4, 5, 14, 15

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996. doi:10.1145/233551.233553. 3, 4

[GRS22] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 323–352, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15802-5_12. 2

[GRS23] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Malicious secure, structure-aware private set intersection. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 577–610, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38557-5_19. 2

[HNO+23] Brett Hemenway, Daniel Noble, Rafail Ostrovsky, Matan Shtepel, and Jacob Zhang. DORAM revisited: Maliciously secure RAM-MPC with logarithmic overhead. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part I*, volume 14369 of *Lecture Notes in Computer Science*, pages 441–470, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-48615-9_16. 4

[IKH+23] Atsunori Ichikawa, Ilan Komargodski, Koki Hamada, Ryo Kikuchi, and Dai Ikarashi. 3-party secure computation for RAMs: Optimal and concretely efficient. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part I*, volume 14369 of *Lecture Notes in Computer Science*, pages 471–502, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-48615-9_17. 4

[JGB+24] Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. Orca: FSS-based secure training and inference with GPUs. In *2024 IEEE Symposium on Security and Privacy*, pages 597–616, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press. doi:10.1109/SP54263.2024.00063. 2

[LO13] Steve Lu and Rafail Ostrovsky. Distributed oblivious RAM for secure two-party computation. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 377–396, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-36594-2_22. 3

[Röl06] Adrian Röllin. Translated poisson approximation using exchangeable pair couplings. 2006. URL: https://api.semanticscholar.org/CorpusID:207793987. 20

[RTPB22] Théo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis R. Bach. Ariann: Low-interaction privacy-preserving deep learning via function secret sharing. *Proc. Priv. Enhancing Technol.*, 2022(1):291–316, 2022. URL: https://doi.org/10.2478/popets-2022-0015, doi:10.2478/POPETS-2022-0015. 2

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. doi:10.1145/359168.359176. 3

[TSS+20] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *IEEE Data Eng. Bull.*, 43(2):95–107, 2020. URL: http://sites.computer.org/debull/A20june/p95.pdf. 2

[VHG23] Adithya Vadapalli, Ryan Henry, and Ian Goldberg. Duoram: A bandwidth-efficient distributed ORAM for 2- and 3-party computation. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023: 32nd USENIX Security Symposium*, pages 3907–3924, Anaheim, CA, USA, August 9–11, 2023. USENIX Association. 4

[Yek10] Sergey Yekhanin. Locally decodable codes, January 2010. URL: https://www.microsoft.com/en-us/research/publication/locally-decodable-codes/. 24

[YJG+23] Peng Yang, Zoe Lin Jiang, Shiqi Gao, Hongxiao Wang, Jun Zhou, Yangyiye Jin, Siu-Ming Yiu, and Junbin Fang. FssNN: Communication-efficient secure neural network training via function secret sharing. Cryptology ePrint Archive, Paper 2023/073, 2023. URL: https://eprint.iacr.org/2023/073. 2